



Quantum GIS

Guida all'uso, installazione
e programmazione

Versione 1.0.0 '*Kore*'

Preambolo

Questo documento costituisce la traduzione italiana dell'originale guida all'uso, installazione e programmazione del programma Quantum GIS. Il software e l'hardware citato in questo documento è in molti casi marchio registrato e quindi soggetto a restrizioni legali. Quantum GIS è soggetto alla GNU General Public License. Maggiori informazioni alla homepage di Quantum GIS <http://qgis.osgeo.org>.

Dettagli, dati, risultati ecc. ecc. in questo documento sono stati scritti e verificati con la miglior diligenza dagli autori e dagli editori. Non si escludono, tuttavia, errori inerenti il contenuto.

Di conseguenza, nessun dato è da ritenere adatto ad alcuno scopo specifico e tantomeno garantito. Gli autori e gli editori non si assumono alcuna responsabilità per eventuali danni e per le loro conseguenze. Sono comunque ben accette le segnalazioni di possibili errori.

Questo documento è stato formattato con \LaTeX . È disponibile come codice sorgente \LaTeX mediante [subversion](#) e online come documento PDF all'indirizzo <http://qgis.osgeo.org/documentation/manuals.html>. Versioni tradotte di questo documento possono essere scaricate anche nell'area documentazione del progetto QGIS. Per ulteriori informazioni sul come contribuire a questo documento e alla sua traduzione, si prega di visitare questo link: <http://wiki.qgis.org/qgiswiki/DocumentationWritersCorner>

Collegamenti in questo documento

Questo documento contiene link interni ed esterni. Cliccando su un link interno ci si muove all'interno del documento stesso, mentre cliccando su un link esterno si aprirà un indirizzo internet. Nel formato PDF, i collegamenti interni sono mostrati in colore blu, mentre quelli esterni sono mostrati in colore rosso e gestiti dal browser di sistema. In formato HTML il browser mostra e gestisce in maniera identica entrambi i tipi di collegamento.

Autori ed editori della guida all'uso, installazione e programmazione:

Tara Athan	Radim Blazek	Godofredo Contreras
Otto Dassau	Martin Dobias	Jürgen E. Fischer
Stephan Holl	Marco Hugentobler	Magnus Homann
Lars Luthman	Gavin Macaulay	Werner Macho
Tyler Mitchell	Brendan Morely	Gary E. Sherman
Tim Sutton	David Willis	

Con i ringraziamenti a Tisham Dhar per aver preparato la documentazione iniziale dell'ambiente msys (MS Windows), a Tom Elwertowski e William Kyngesburye per l'aiuto alla sezione di installazione su MAC OSX e a Carlos Dávila, Paolo Cavallini e Christian Gunning per le revisioni. Se avessimo

dimenticato di menzionare qualche collaboratore, lo preghiamo di accettare le nostre scuse per la svista.

Copyright © 2004 - 2009 Quantum GIS Development Team

Internet: <http://qgis.osgeo.org>

Indice

Titolo	i
Preambolo	ii
Indice	iv
Elenco delle figure	xiii
Elenco delle tabelle	xv
Elenco dei suggerimenti (Tips)	xvi
1 Premessa	1
1.1 Caratteristiche	1
1.2 Convenzioni	5
2 Introduzione al GIS	8
2.1 Come mai così tante novità?	9
2.1.1 Dati raster	9
2.1.2 Dati vettoriali	10
3 Come iniziare	11
3.1 Installazione	11
3.2 Dati campione	11
3.3 Sessione di esempio	12
4 Panoramica sulle caratteristiche	15
4.1 Avvio e chiusura di QGIS	15
4.1.1 Opzioni da linea di comando	15
4.2 QGIS GUI	17
4.2.1 Barra Menù	18
4.2.2 Barre degli strumenti	20
4.2.3 Legenda	20
4.2.4 Il visualizzatore di mappa	22
4.2.5 Panoramica della mappa	23
4.2.6 Barra di stato	23
4.3 Visualizzazione	23
4.3.1 Visualizzazione in funzione della scala	24
4.3.2 Controllo della resa a video della mappa	24
4.4 Misurare	26
4.4.1 Misurare lunghezze ed aree	26
4.5 Progetti	26

4.6	Output	27
4.7	Opzioni dell'interfaccia grafica (GUI)	27
4.8	Segnalibri geospaziali	29
4.8.1	Creazione di un segnalibro	29
4.8.2	Uso e gestione dei segnalibri	30
4.8.3	Zoommare a un segnalibro	30
4.8.4	Cancellare un segnalibro	30
5	Lavorare con i dati vettoriali	31
5.1	ESRI Shapefiles	31
5.1.1	Caricare uno shapefile	31
5.1.2	Ottimizzare le prestazioni	33
5.1.3	Caricare uno strato MapInfo	34
5.1.4	Caricare una coverage ArcInfo	34
5.2	Gli strati PostGIS	34
5.2.1	Creare una connessione	35
5.2.2	Caricare un layer PostGIS	36
5.2.3	Alcuni dettagli sui livelli PostgreSQL	36
5.2.4	Importazione di dati in PostgreSQL	37
5.2.5	Migliorare le prestazioni	38
5.3	La finestra delle proprietà dei vettori	39
5.3.1	Linguetta Generale	40
5.3.2	Linguetta simbologia	40
5.3.3	Linguetta Metadati	42
5.3.4	Linguetta Etichette	42
5.3.5	Linguetta Azioni	44
5.3.6	La linguetta Attributi	49
5.4	Editing	50
5.4.1	Settare la tolleranza dello snapping e il raggio di ricerca degli elementi	50
5.4.2	Editing topologico	51
5.4.3	Modifica di un layer esistente	52
5.4.4	Creare un nuovo layer	59
5.5	Costruttore di interrogazioni (query builder)	61
5.6	Selezione mediante interrogazione	62
6	Lavorare con i dati raster	63
6.1	Cosa sono i dati raster?	63
6.2	Caricati dati raster in QGIS	64
6.3	Finestra delle proprietà raster	64
6.3.1	Linguetta Simbologia	66
6.3.2	Linguetta Trasparenza	67
6.3.3	Mappa colore	68

6.3.4	Linguetta Generale	68
6.3.5	Linguetta Metadata	69
6.3.6	Linguetta Piramidi	69
6.3.7	Linguetta Istogramma	69
7	Lavorare con i dati OGC	71
7.1	Cosa sono i dati OGC	71
7.2	Client WMS	71
7.2.1	Panoramica sul servizio WMS	71
7.2.2	Scegliere un server WMS	72
7.2.3	Caricare livelli WMS	73
7.2.4	Uso dello strumento di identificazione	75
7.2.5	Visualizzazione delle proprietà del server	75
7.2.6	Limitazioni del client WMS	77
7.3	Client WFS	77
7.3.1	Caricare un layer WFS	78
8	Lavorare con le proiezioni	80
8.1	Panoramica del supporto alle proiezioni	80
8.2	Specificare una proiezione	80
8.3	Definire la proiezione al volo (OTF)	81
8.4	Proiezione definita dall'utente	83
9	Integrazione con GRASS GIS	85
9.1	Avviare il plugin GRASS	85
9.2	Caricare livelli raster e vettoriali GRASS	86
9.3	LOCATION e MAPSET in GRASS	87
9.3.1	Creare una nuova LOCATION GRASS	87
9.3.2	Aggiungere un nuovo MAPSET	89
9.4	Importare dati nelle LOCATION GRASS	90
9.5	Il modello di dato vettoriale in GRASS	91
9.6	Creazione di un nuovo layer vettoriale GRASS	92
9.7	Digitalizzazione e modifica di layer vettoriali in GRASS	93
9.8	Lo strumento Region di GRASS	97
9.9	La barra strumenti GRASS	97
9.9.1	Lavorare con i moduli GRASS	98
9.9.2	Lavorare con il browser delle LOCATION GRASS	100
9.9.3	Personalizzazione degli strumenti GRASS	101
10	Compositore di stampe	102
10.1	Uso del compositore di stampe	103
10.1.1	Aggiungere una vista mappa al layout nel compositore di stampe	104
10.1.2	Aggiunta di altri elementi al compositore di stampa	105

10.1.3 Strumenti per l'esplorazione del layout di stampa	106
10.1.4 Creazione di file in uscita	107
11 Plugins di QGIS	109
11.1 Gestire i Plugins	109
11.1.1 Abilitare un Plugin Core	109
11.1.2 Caricamento di un plugin esterno di QGIS	110
11.1.3 Uso dell'Installatore di Plugins Python	110
11.2 Data Providers	113
12 Using QGIS Core Plugins	114
12.1 Plugin di Cattura delle Coordinate	115
12.2 Plugins Decorativi	117
12.2.1 Plugin etichetta di Copyright	117
12.2.2 Plugin Freccia Nord	118
12.2.3 Plugin Barra di Scala	118
12.3 Plugin Testo Delimitato	120
12.4 Plugin Convertitore Dxf2Shp	123
12.5 Plugin Georeferenziatore	124
12.6 Quick Print Plugin	128
12.7 Plugin GPS	130
12.7.1 Cos'è un GPS?	130
12.7.2 Caricare i dati GPS da un file	130
12.7.3 GPSTransfer	130
12.7.4 Importazione di dati GPS	131
12.7.5 Scaricare dati GPS da un dispositivo	131
12.7.6 Caricare dati GPS In un dispositivo	132
12.7.7 Definizione di un nuovo dispositivo	133
12.8 Plugin Creatore di Griglia	135
12.9 Interpolation Plugin	137
12.10 Plugin MapServer Export	140
12.10.1 Creare il file Progetto	140
12.10.2 Creazione del Map File	141
12.10.3 Testare il File Mappa	143
12.11 Plugin Convertitore Layer OGR	145
13 Uso dei Plugin esterni QGIS Python	147
14 Scrivere un plugin QGIS in C++	148
14.1 Perché C++ e circa le licenze	148
14.2 Programmare un plugin QGIS C++ in 4 passi	148
14.3 Ulteriori informazioni	166

15 Scrivere un Plugin QGIS in Python	167
15.1 Perchè Python e circa la licenza	167
15.2 Cosa è necessario installare per iniziare	167
15.3 Programmare un semplice plugin PyQGIS in quattro passaggi	168
15.4 Fare il commit del plugin nell'archivio	171
15.5 Ulteriori informazioni	171
16 Creare applicazioni C++	173
16.1 Creare un semplice widget mappa	173
16.2 Lavorare con QgsMapCanvas	176
17 Creare applicazioni PyQGIS	181
17.1 Progettare il GUI	181
17.2 Creare la Finestra Principale	182
17.3 Conclusione	187
17.4 Lanciare l'applicazione	188
18 Aiuto e Supporto	190
18.1 Mailinglist	190
18.2 IRC	191
18.3 Tracciatore di malfunzionamenti (Bug Tracker)	191
18.4 Blog	192
18.5 Wiki	192
A Formati supportati	193
A.1 Formati OGR supportati	193
A.2 Formati raster GDAL supportati	193
B Moduli degli strumenti GRASS	196
B.1 Moduli negli strumenti GRASS per l'importazione e l'esportazione di dati	196
B.2 Moduli negli strumenti GRASS per la conversione tra tipi di dato	196
B.3 Moduli negli strumenti GRASS per la definizione della regione e l'impostazione della proiezione	196
B.4 Moduli negli strumenti GRASS per le operazioni su dati raster	200
B.5 Moduli negli strumenti GRASS per le operazioni su dati vettoriali	206
B.6 Moduli negli strumenti GRASS per le operazioni su immagini	210
B.7 Moduli negli strumenti GRASS per la gestione dei database	211
B.8 Moduli negli strumenti GRASS per il 3D	212
B.9 Moduli negli strumenti GRASS per l'aiuto in linea	212
C Guida all'Installazione	213
C.1 Note generali di compilazione	213
C.2 Panoramica delle dipendenze richieste per la compilazione	213

D	Compilazione in ambiente windows usando msys	214
D.1	MSYS:	214
D.2	Qt4.3	214
D.3	Flex and Bison	215
D.4	Materiale Python: (opzionale)	215
D.4.1	Scaricare e installare Python - uso dell'installatore di Windows	215
D.4.2	Scaricare i sorgenti SIP and PyQt4	216
D.4.3	Compilare SIP	216
D.4.4	Compilare PyQt	216
D.4.5	Note finali python	216
D.5	Subversioni:	216
D.6	CMake:	216
D.7	QGIS:	217
D.8	Compilare:	217
D.9	Configurazione	218
D.10	Compilazione and installazione	218
D.11	Eseguire qgis.exe dalla directory dove è installato (CMAKE_INSTALL_PREFIX)	218
D.12	Creare il pacchetto d'installazione: (opzionale)	219
E	Compilare su Mac OSX usando frameworks e cmake (QGIS > 0.8)	219
E.1	Installare XCODE	219
E.2	Installare Qt4 from .dmg	219
E.3	Installare i frameworks di sviluppo per le dipendenze QGIS	220
E.3.1	Dipendenze aggiuntive: GSL	220
E.3.2	Dipendenze aggiuntive: Expat	221
E.3.3	Dipendenze aggiuntive: SIP	221
E.3.4	Dipendenze aggiuntive: PyQt	222
E.3.5	Dipendenze aggiuntive: Bison	223
E.4	Installare CMAKE per OSX	223
E.5	Installare subversioni per OSX	224
E.6	Check out QGIS da SVN	224
E.7	Configurare la versione compilata	225
E.8	Compilazione	226
F	Compilare sotto GNU/Linux	227
F.1	Compilare QGIS con Qt4.x	227
F.2	Preparare apt	227
F.3	Installare Qt4	227
F.4	Installare dipendenze software aggiuntive richieste da QGIS	228
F.5	Passaggi specifici GRASS	229
F.6	Impostare ccache (Opzionale)	229
F.7	Preparare il proprio ambiente di sviluppo	229




F.8	Check out del codice sorgente QGIS	229
F.9	Cominciare a compilare	230
F.10	Compilare i pacchetti Debian	231
F.11	Eseguire QGIS	232
G	Creation of MSYS environment for compilation of Quantum GIS	232
G.1	Initial setup	232
G.1.1	MSYS	232
G.1.2	MinGW	232
G.1.3	Flex and Bison	232
G.2	Installing dependencies	233
G.2.1	Getting ready	233
G.2.2	GDAL level one	233
G.2.3	GRASS	235
G.2.4	GDAL level two	236
G.2.5	GEOS	237
G.2.6	SQLITE	237
G.2.7	GSL	237
G.2.8	EXPAT	238
G.2.9	POSTGRES	238
G.3	Cleanup	238
H	Building with MS Visual Studio	239
H.1	Setup Visual Studio	239
H.1.1	Express Edition	239
H.1.2	All Editions	239
H.2	Download/Install Dependencies	240
H.2.1	Flex and Bison	240
H.2.2	To include PostgreSQL support in Qt	240
H.2.3	Qt	241
H.2.4	Proj.4	241
H.2.5	GSL	242
H.2.6	GEOS	242
H.2.7	GDAL	243
H.2.8	PostGIS	243
H.2.9	Expat	243
H.2.10	CMake	243
H.3	Building QGIS with CMAKE	244
I	Building under Windows using MSVC Express	244
I.1	System preparation	245
I.2	Install the libraries archive	245
I.3	Install Visual Studio Express 2005	245

I.4	Install Microsoft Platform SDK2	246
I.5	Edit your vsvars	249
I.6	Environment Variables	250
I.7	Building Qt4.3.2	251
I.7.1	Compile Qt	251
I.7.2	Configure Visual C++ to use Qt	251
I.8	Install Python	252
I.9	Install SIP	253
I.10	Install PyQt4	253
I.11	Install CMake	253
I.12	Install Subversion	253
I.13	Initial SVN Check out	254
I.14	Create Makefiles using cmakeSetup.exe	254
I.15	Running and packaging	255
J	QGIS Coding Standards	256
J.1	Classes	256
J.1.1	Names	256
J.1.2	Members	256
J.1.3	Accessor Functions	257
J.1.4	Functions	257
J.2	Qt Designer	257
J.2.1	Generated Classes	257
J.2.2	Dialogs	257
J.3	C++ Files	258
J.3.1	Names	258
J.3.2	Standard Header and License	258
J.3.3	CVS Keyword	258
J.4	Variable Names	259
J.5	Enumerated Types	259
J.6	Global Constants	259
J.7	Editing	259
J.7.1	Tabs	259
J.7.2	Indentation	260
J.7.3	Braces	260
J.8	API Compatibility	260
J.9	Coding Style	261
J.9.1	Where-ever Possible Generalize Code	261
J.9.2	Prefer Having Constants First in Predicates	261
J.9.3	Whitespace Can Be Your Friend	261
J.9.4	Add Trailing Identifying Comments	262
J.9.5	Use Braces Even for Single Line Statements	262

J.9.6	Book recommendations	263
K	SVN Access	263
K.1	Accessing the Repository	263
K.2	Anonymous Access	263
K.3	QGIS documentation sources	264
K.4	Documentation	264
K.5	Development in branches	264
K.5.1	Purpose	264
K.5.2	Procedure	265
K.5.3	Creating a branch	265
K.5.4	Merge regularly from trunk to branch	265
K.6	Submitting Patches	266
K.6.1	Patch file naming	266
K.6.2	Create your patch in the top level QGIS source dir	266
K.6.3	Including non version controlled files in your patch	267
K.6.4	Getting your patch noticed	267
K.6.5	Due Diligence	267
K.7	Obtaining SVN Write Access	267
K.7.1	Procedure once you have access	267
L	Unit Testing	269
L.1	The QGIS testing framework - an overview	269
L.2	Creating a unit test	270
L.3	Adding your unit test to CMakeLists.txt	276
L.4	Building your unit test	278
L.5	Run your tests	278
M	HIG (Human Interface Guidelines)	280
N	GNU General Public License	281
N.1	Quantum GIS Qt exception for GPL	286
	Cited literature	287
	Index	288

Elenco delle figure

1	Una sessione di esempio in QGIS 	14
2	Interfaccia grafica di QGIS GUI con dati campione dell'Alaska 	17
3	Strumento di misura in azione 	26
4	Finestra di dialogo Apri un layer vettoriale supportato da OGR 	32
5	QGIS con lo shapefile Alaska caricato 	33
6	Finestra delle proprietà di un vettoriale 	40
7	Opzioni per la simbologia dei layer 	41
8	Selezione di un elemento e scelta dell'azione 	48
9	Modifica delle opzioni di snapping per singoli layers 	51
10	Finestra di inserimento degli attributi per un elemento di nuova digitalizzazione 	55
11	Finestra di creazione di un nuovo layer vettoriale 	60
12	La finestra del costruttore di query 	61
13	Finestra delle proprietà dei livelli raster 	65
14	Finestra per l'aggiunta di un server WMS, nella quale vengono mostrati gli strati disponibili 	74
15	Aggiunta di un layer WFS 	79
16	Linguetta CRS nella finestra opzioni di QGIS 	81
17	Finestra di dialogo per l'impostazione della proiezione 	82
18	Finestra del CRS personalizzato 	84
19	Organizzazione dei dati GRASS nella LOCATION campione alaska (adattato da from Neteler & Mitasova 2008 (2))	88
20	Creazione di una nuova LOCATION GRASS o di un nuovo MAPSET in QGIS 	89
21	Barra strumenti di digitalizzazione GRASS 	93
22	Linguetta Categoria negli strumenti per la digitalizzazione in GRASS 	95
23	Linguetta Preferenze negli strumenti per la digitalizzazione in GRASS 	95
24	Linguetta Simbologia negli strumenti per la digitalizzazione in GRASS 	96
25	Linguetta Tabella negli strumenti per la digitalizzazione in GRASS 	96
26	Strumenti di GRASS e Lista Moduli con possibilità di ricerca 	98
27	Finestre degli strumenti GRASS 	99
28	Browser delle LOCATION GRASS 	100
29	Compositore di stampe 	103
30	Contenuti della linguetta Oggetto nel compositore di stampe 	104
31	Personalizzazione di immagini ed etichette nel layout di stampa 	106
32	Personalizzazione della legenda e della barra di scala nel compositore di stampe 	107
33	Compositore di stampe con layout completo di vista mappa, legenda, barra di scala e testo 	108
34	Plugin Manager 	110
35	Installazione di Plugin Esterni Python 	111
36	Plugin di cattura delle Coordinate 	115
37	Plugin Etichetta di Copyright 	117

38	Plugin Freccia Nord 	118
39	Plugin Barra di Scala 	119
40	finestra di dialogo Aggiungi layer di testo delimitato 	121
41	Plugin Convertitore Dxf2Shape 	123
42	Selezionare un'immagine da georeferenziare 	124
43	Organizzare le finestre di plugin con la mappa qgis 	125
44	Aggiungere punti all'immagine raster 	126
45	Mappa georeferenziata con la mappa roads della location spearfish60 sovrapposta 	127
46	Finestra di dialogo Quick Print 	128
47	Risultato ottenuto con Quick Print come DIN A4 PDF 	129
48	La finestra di dialogo <i>Strumenti GPS</i> 	131
49	Finestra di dialogo dello strumento di selezione del file da importare 	132
50	Lo strumento di download 	133
51	Creare uno strato griglia 	135
52	Plugin di interpolazione 	137
53	Interpolazione di dati elevp usando il metodo IDW 	139
54	predisporre i layers vettoriali e raster in un file progetto QGIS 	140
55	Finestra di dialogo MapServer Export 	142
56	Test PNG creato con shp2img con tutti i layer MapServer Export 	143
57	Plugin Convertitore Layer OGR 	145
58	Semplice applicazione C++ 	177
59	Applicazione QMainWindow con un menu, una barra degli strumenti e un'area canvas 	179

Elenco delle tabelle

1	Parametri di connessione al geodatabase PostGIS	35
2	Parametri del collegamento WMS	72
3	Esempi di indirizzi di server WMS pubblici	73
4	Strumenti per la digitalizzazione in GRASS	94
5	Strumenti del compositore di stampe	102
6	QGIS Core Plugins	114
7	Attuali Plugin QGIS esterni moderati	147
8	Strumenti GRASS: Moduli per l'importazione di dati	197
9	Strumenti GRASS: Moduli per per l'esportazione di dati	198
10	Strumenti GRASS: moduli per la conversione tra tipi di dato	198
11	Strumenti GRASS: moduli per la gestione della regione e della proiezione	199
12	Strumenti GRASS: moduli per le operazioni su dati raster	200
13	Strumenti GRASS: moduli per la gestione del colore dei dati raster	201
14	Strumenti GRASS: moduli per le operazioni di processamento spaziale di dati raster	202
15	Strumenti GRASS: moduli per la gestione di superfici	203
16	Strumenti GRASS: moduli per la modifica delle categorie e per le etichette di dati raster	204
17	Strumenti GRASS: moduli per la modellazione idrologica	204
18	Strumenti GRASS: moduli per la generazione di report e l'analisi statistica	205
19	Strumenti GRASS: moduli per la modifica di dati vettoriali	207
20	Strumenti GRASS: moduli per la gestione del collegamento ai database	208
21	Strumenti GRASS: moduli per la modifica delle categorie di un vettoriale	208
22	Strumenti GRASS: moduli per lavorare con vettoriali di punti	208
23	Strumenti GRASS: Moduli per l'analisi spaziale di vettoriali e per l'analisi di reti	209
24	Strumenti GRASS: moduli per l'aggiornamenti di vettoriali sulla base di altre mappe	209
25	Strumenti GRASS: moduli per i report e le statistiche su un vettoriale	209
26	Strumenti GRASS: moduli per l'analisi di immagini	210
27	Strumenti GRASS: moduli per i database	211
28	Strumenti GRASS: Visualizzazione 3D	212
29	Strumenti GRASS: Reference Manual	212

QGIS Tips

1	DOCUMENTAZIONE AGGIORNATA	1
2	ESEMPIO DI UTILIZZO DELLE OPZIONI DA RIGA DI COMANDO	16
3	RIPRISTINARE BARRE DEGLI STRUMENTI	20
4	ZOOMARE RAPIDAMENTE CON LA ROTELLA DEL MOUSE	22
5	MUOVERE LA MAPPA CON I TASTI FRECCIA E LA BARRA SPAZIATRICE	23
6	CALCOLARE LA SCALA CORRETTA DELLA MAPPA	24
7	COLORI DEL LAYER	32
8	SETTAGGIO E SICUREZZA DELLE IMPOSTAZIONI UTENTE IN QGIS	35
9	LIVELLI POSTGIS	36
10	ESPORTARE DATI DA POSTGIS	37
11	IMPORTARE SHAPEFILES CONTENENTI PAROLE RISERVATE A POSTGRESQL	38
12	INTEGRITÀ DEL DATO	52
13	MODIFICA DI ATTRIBUTI	52
14	SALVATAGGIO AD INTERVALLI REGOLARI	52
15	MODIFICHE CONCORRENTI	53
16	INGRANDIRE PRIMA DELLA MODIFICA	53
17	MARKER DEI VERTICI	54
18	TIPOLOGIE DI ATTRIBUTO	55
19	CONGRUENZA DEGLI ELEMENTI INCOLLATI	58
20	SUPPORTO ALLA CANCELLAZIONE DI ELEMENTI	59
21	CAMBIARE LA DEFINIZIONE DI UNO STRATO	62
22	VEDERE UNA SINGOLA BANDA DI UN RASTER MULTIBANDA	67
23	OTTENERE LE STATISTICHE DEL RASTER	70
24	A PROPOSITO DI INDIRIZZI DEI SERVER WMS	73
25	CODIFICA IMMAGINE	73
26	ORDINE DEGLI STRATI WMS	74
27	TRASPARENZA DEI LAYER WMS	74
28	LE PROIEZIONI WMS	75
29	ACCESSO A LIVELLI OGC CON PASSWORD	77
30	TROVARE SERVERS WMS E WFS	78
31	FINESTRA DELLE PROPRIETÀ DELLA PROIEZIONE	83
32	CARICARE DATI GRASS	87
33	CONOSCERE IL MODELLO DEL DATO VETTORIALE IN GRASS	92
34	CREAZIONE DI UNA TABELLA ATTRIBUTI PER UN NUOVO LIVELLO VETTORIALE GRASS	92
35	DIGITALIZZARE POLIGONI IN GRASS	93
36	CREARE UN 'LAYER' GRASS AGGIUNTIVO CON QGIS	94
37	PERMESSI DI MODIFICA IN GRASS	97
38	MOSTRARE I RISULTATI IMMEDIATAMENTE	99
39	SALVARE UN LAYOUT DI STAMPA	105
40	IL CRASH DEI PLUGINS	109

41	PLUGINS SETTINGS SAVED TO PROJECT	114
42	SCEGLIERE IL TIPO DI TRASFORMAZIONE	126
43	AGGIUNGERE PIÙ PLUGIN ESTERNI	147
44	DUE CARTELLE QGIS PYTHON PLUGIN	168
45	DOCUMENTAZIONE PER PYQGIS	189

1 Premessa

Benvenuti nel meraviglioso mondo dei Sistemi Informativi Geografici (Geographical Information Systems, GIS)! Quantum GIS (QGIS) è un Sistema Informativo Geografico a codice aperto (Open Source). Il progetto è nato nel maggio 2002 ed è stato ospitato su SourceForge nel giugno dello stesso anno. Abbiamo lavorato duramente per rendere il software GIS (che è tradizionalmente un software proprietario costoso) una valida prospettiva per chiunque avesse disponibilità di un Personal Computer. QGIS gira attualmente su molte piattaforme Unix (incluso ovviamente Linux!), su Windows, e OS X. QGIS è sviluppato in Qt (<http://www.trolltech.com>) e C++. Ciò fa sì che QGIS appaia responsivo nell'uso e piacevole e semplice da usare nell'interfaccia grafica (graphical user interface, GUI).

QGIS si prefigge lo scopo di essere un GIS facile da usare, in grado di fornire funzioni e caratteristiche di uso comune. L'obiettivo iniziale era di fornire un visore di dati GIS, ma attualmente QGIS ha oltrepassato questo punto nel suo sviluppo, ed è usato da molti per il loro lavoro quotidiano nel campo GIS. QGIS supporta nativamente un considerevole numero di formati raster e vettoriali, il supporto a nuovi formati può essere facilmente inserito mediante plugin (si veda l'Appendice A per una lista completa dei formati attualmente supportati).

QGIS è rilasciato con licenza GNU General Public License (GPL). Lo sviluppo di QGIS con questa licenza implica che si possa ispezionare e modificare il codice sorgente in modo da garantirvi di avere sempre accesso ad un programma GIS esente da costi di licenza e modificabile liberamente secondo le vostre esigenze. Dovresti aver ricevuto una copia completa della licenza con la tua copia di QGIS, puoi comunque trovarla nell'Appendice N.

Tip 1 DOCUMENTAZIONE AGGIORNATA

La versione più recente di questo documento è sempre reperibile all'indirizzo

<http://download.osgeo.org/qgis/doc/manual/>, o nell'area documentazione del sito di QGIS all'indirizzo <http://qgis.osgeo.org/documentation/>

1.1 Caratteristiche

QGIS offre nativamente e mediante plugin molte funzioni GIS di uso comune. È possibile offrirne una panoramica iniziale raggruppandole sinteticamente in sei categorie.

Visualizzazione di dati

Possono essere visualizzati e sovrapposti dati vettoriali e raster in diversi formati e proiezioni senza necessità di conversioni verso un formato comune interno. Tra i formati supportati sono inclusi:

- tabelle PostgreSQL con estensione spaziale usando PostGIS, formati vettoriali ¹ supportati dalla libreria OGR installata, inclusi gli shapefile ESRI e i formati MapInfo, SDTS e GML.
- formati raster e immagine supportati dalla libreria GDAL (Geospatial Data Abstraction Library) installata, come GeoTiff, Erdas Img., ArcInfo Ascii Grid, JPEG, PNG,
- formati raster GRASS e dati vettoriali da database GRASS (location/mapset),
- dati spaziali forniti da servizi di mappa online conformi agli standard OGC quali Web Map Service (WMS) o Web Feature Service (WFS).

Esplorazione dati e creazione di mappa

Possono essere composte mappe e esplorati interattivamente dati spaziali con un'amichevole interfaccia grafica. I molti strumenti utili disponibili nell'interfaccia grafico includono:

- proiezione al volo
- compositore di mappa
- pannello vista panoramica
- segnalibri geospaziali
- identifica/seleziona elementi
- modifica/vedi/cerca attributi
- etichetta elementi
- cambio simbologia raster/vettoriale
- aggiunta reticolato su un nuovo layer
- decorazione della la mappa con una freccia del nord, una barra di scala e un'etichetta di copyright
- salva e ricarica progetti

Creazione, modifica, gestione ed esportazione di dati

Possono essere creati, modificati, gestiti ed esportati dati vettoriali in molteplici formati. I dati raster devono essere importati in GRASS per poter essere editati ed esportati in altri formati. QGIS offre le seguenti funzioni:

- strumenti per digitalizzare formati supportati da OGR e layer vettoriali GRASS
- creazione e modifica di shapefiles e layer vettoriali GRASS
- georeferenziazione di immagini con l'apposito plugin
- strumenti GPS per l'importazione ed esportazione del formato GPX, e conversione di altri formati GPS al formato GPX o down/upload dei dati direttamente ad unità GPS

¹i formati di database supportati da OGR come Oracle o mySQL non sono ancora supportati in QGIS.

- creazione di layers PostGIS da shapefiles con il plugin SPIT
- gestione di tabelle degli attributi di dati vettoriali con il plugin gestione tabelle

Analisi di dati

Possono essere eseguite analisi spaziali di dati PostgreSQL/PostGIS e di altri formati supportati da OGR per mezzo del plugin python ftools. QGIS offre attualmente strumenti per l'analisi, il campionamento, il geoprocessing, la gestione delle geometrie e del database di dati vettoriali. Possono inoltre essere usati gli strumenti GRASS integrati, che includono l'intera gamma delle funzioni di GRASS di oltre 300 moduli (Si veda la Sezione [9](#)).

Pubblicazione di mappe su internet

QGIS può essere usato per esportare dati in un mapfile che può essere pubblicato su internet mediante un webserver sul quale sia installato UMN MapServer. QGIS può anche essere impiegato come client WMS or WFS, e come server WMS.

Estensione delle funzioni di QGIS per mezzo di plugins

QGIS può essere adattato a particolari esigenze grazie all'architettura estensibile per mezzo di plugin. QGIS fornisce le librerie che possono essere usate per creare i plugins. Possono addirittura essere creati nuovi programmi in C++ or Python!

- **Plugin inclusi nel software**

- Aggiunta layer WFS
- Aggiunta layer a partire da testo delimitato
- Cattura coordinate
- Decorazioni (Etichetta Copyright, Freccia Nord e Barra di Scala)
- Georeferenziatore
- Convertitore Dxf2Shp
- Strumenti GPS
- Integrazione con GRASS
- Generatore di reticolo
- Plugin per interpolazione
- Conversione tra layer supportati da OGR
- Stampa rapida
- Interfaccia di importazione di shapefile verso PostgreSQL/PostGIS (SPIT)
- Esportazione verso Mapserver
- Console Python
- Installatore plugins Python

- **Plugins Python**







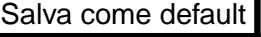

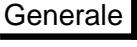










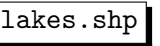
QGIS offre un numero crescente di plugin esterni in Python forniti dalla comunità. Questi plugin sono ospitati sul repository ufficiale PyQGIS, e possono essere facilmente installati usando il gestore di plugins python (Si veda la Sezione [11](#)).

1.2 Convenzioni

Questa sezione descrive le convenzioni di rappresentazione grafica usate nel manuale. Esse sono le seguenti:

Convenzioni per la GUI

Gli stili convenzionali per la GUI hanno lo scopo di rassomigliare all'effettivo aspetto della GUI. In generale, si è evitato di usare immagini o indicazioni che compaiono solo al passaggio del mouse sopra l'indicazione, in modo che l'utente possa scorrere visivamente la GUI per trovare la migliore corrispondenza con l'istruzione rappresentata nel manuale.

- Opzioni da menù:  >  Aggiungi un layer raster
- o
-  >  >  Digitalizzazione
- Strumenti:  Aggiungi un layer raster
- Bottone:  Salva come default
- Titolo casella di dialogo:  Proprietà layer
- Tab:  Generale
- Oggetto della toolbox:  nviz - Open 3D-View in NVIZ
- Casella di controllo:  Render
- Radio Button:  Postgis SRID  EPSG ID
- Scelta numerica: Hue  60 
- Scelta di una stringa: Outline style  —Solid Line
- Cerca file: 
- Scelta colore: Colore contorno 
- Barra scorrimento: Trasparenza 0% 
- Inserimento testo: Nome da mostrare 

L'ombra caratterizza un componente della GUI cliccabile.

Le schermate visibili nella guida sono state create su diversi sistemi operativi, indicati da apposite icone alla fine della didascalia.

2 Introduzione al GIS

Un Sistema Informativo Geografico (Geographical Information System, GIS)⁽¹⁾² è un insieme di programmi che permettono di creare, visualizzare, interrogare e analizzare dati geospaziali. I dati geospaziali riportano informazioni inerenti la posizione geografica di un oggetto. Questo spesso implica l'uso di coordinate geografiche, quali valori di latitudine e longitudine. Dato spaziale è un altro termine comunemente usato, così come lo sono: dato geografico, dato GIS, mappa, location, coordinate e geometrie spaziali.

Le applicazioni che usano dati geospaziali eseguono varie funzioni. Quella più conosciuta e facilmente compresa è la produzione di mappe. I programmi per la realizzazione di mappe impiegano i dati geospaziali e li rappresentano in una forma che sia visibile, tipicamente su uno schermo o stampati su carta. Si possono presentare mappe statiche (una semplice immagine) o mappe dinamiche che sono personalizzate dall'utente che ne usufruisce attraverso un'applicazione desktop o una pagina web.

Molte persone ritengono erroneamente che le applicazioni geospaziali si riducano alla produzione di mappe, ma l'analisi geospaziale del dato è un'altra primaria funzione di tali programmi. Alcune tipiche analisi che possono essere condotte sono:

1. distanze tra posizioni geografiche
2. misurazione dell'area (ad. es. metri quadrati) di una certa regione geografica
3. quali elementi geografici si sovrappongono ad altri
4. determinazione dell'entità della sovrapposizione tra elementi
5. il numero di posizioni comprese entro una certa distanza da un'altra
6. e così via...

Ciò può apparire riduttivo, eppure può essere applicato in innumerevoli maniere in disparate discipline. Il risultato dell'analisi può essere mostrato su una mappa, ma è spesso tabulato in un report a supporto di decisioni gestionali.

La recente diffusione di servizi basati sulla posizione lascia intravedere l'introduzione di ogni sorta di ulteriori caratteristiche, ma molte di esse saranno basate sulla combinazione di mappatura e analisi. Per esempio i cellulari tracciano la loro posizione geografica. Con un programma adatto, il telefono può dire che tipo di ristoranti possono essere raggiunti a piedi dalla propria posizione entro una certa distanza. Per quanto questa sia un'applicazione recentissima della tecnologia geospaziale, è essenzialmente basata su analisi geospaziale del dato e presentazione dei risultati.

²Questo capitolo è stato scritto da Tyler Mitchell (<http://www.oreillynet.com/pub/wlg/7053>) e usato con Licenza Creative Commons License. Tyler is the author of *Web Mapping Illustrated*, pubblicato da O'Reilly, 2005.

2.1 Come mai così tante novità?

Bene, in realtà non è vero. Ci sono molti nuovi dispositivi hardware che consentono servizi geospaziali in modalità mobile. Sono disponibili anche molti programmi geospaziali open source, ma l'esistenza di hardware e software finalizzati all'uso del dato geospaziale non sono per niente una novità. I ricevitori GPS (Global positioning system, GPS) sono diventati molto comuni, ma sono usati in diverse industrie da più di un decennio. In maniera analoga, programmi per la mappatura e l'analisi sono stati un preminente mercato commerciale, focalizzato primariamente su industrie dell'ambito della gestione delle risorse naturali.

Ciò che è nuovo è come le recenti tecnologie hardware e software sono usate e da chi sono usate. Gli utenti tradizionali di applicazioni per la mappatura e l'analisi erano analisti GIS fortemente specializzati o tecnici della mappatura digitale istruiti ad usare strumenti tipo CAD. Attualmente le capacità di processamento degli home PC e il software open source (open source software, OSS) hanno consentito ad un esercito di hobbysti, professionisti, sviluppatori web ecc. ecc. di interagire con il dato geospaziale. La curva di apprendimento si è abbassata. I costi sono diminuiti. La quantità di tecnologia geospaziale è cresciuta.

Come sono archiviati i dati geospaziali? In estrema sintesi, ci sono due tipi di dati geospaziali di uso comune attualmente. Ciò in aggiunta al tradizionale dato in forma tabellare che è comunque diffusamente impiegata dalle applicazioni geospaziali.

2.1.1 Dati raster

Un tipo di dato geospaziale è detto dato raster o semplicemente un raster. Il più noto dato raster è l'immagine satellitare o la foto aerea. Le ombreggiature altimetriche o i modelli digitali di terreno (Digital Elevation Model, DEM) sono anch'esse rappresentate come dati raster. Qualunque elemento di mappa può essere rappresentato come dato raster, ma ci sono delle limitazioni.

Un raster è una griglia regolare fatta di celle o nel caso di semplici immagini, di pixels. Essi hanno un numero fisso di righe e colonne. Ogni cella ha un valore numerico e una certa dimensione geografica (ad es. 30x30 metri).

Più raster sovrapposti sono utilizzati per rappresentare immagini che utilizzano più di un colore (es: un raster per ogni set di rosso, verde e blu viene combinato per creare il colore dell'immagine). Anche le immagini satellitari sono un esempio di dati in bande multiple. Ogni banda è essenzialmente un livello sovrapposto al precedente dove vengono salvati i valori della lunghezza della luce. Come è facile immaginare, un raster di grosse dimensioni occupa maggiore spazio su disco. Un raster con celle piccole può fornire maggior dettaglio ma richiede anche più spazio. L'astuzia sta nel trovare il giusto equilibrio tra la dimensione della cella ai fini dell'archiviazione e la dimensione della cella ai fini analitici o di mappatura.

2.1.2 Dati vettoriali

Anche i dati vettoriali vengono usati nelle applicazioni geospaziali. Nel suo senso più semplice, i vettori sono un metodo di descrizione di una posizione utilizzando un insieme di coordinate. Ogni coordinata si riferisce ad una posizione geografica utilizzando un sistema di valori y e x.

Si può pensare a tutto ciò in riferimento ad un piano cartesiano, i diagrammi di scuola che mostravano un asse x e un asse y. Potreste averli usati per diagrammare decrescenti risparmi per la pensione o crescenti interessi per il mutuo, ma gli stessi concetti sono alla base dell'analisi e della mappatura del dato geospaziale.

Ci sono vari modi di rappresentare queste coordinate geografiche secondo lo scopo che ci si è prefissi. Questo riguarda l'area di studio dei sistemi di proiezione geografica.

I dati vettoriali si presentano in tre forme, di complessità crescente e basate su quella precedente.




1. Punti - Una coppia di coordinate (x y) rappresenta una precisa posizione geografica
2. Linee - Coordinate multiple (x1 y1, x2 y2, x3 y4, ... xn yn) collegate tra loro in un certo ordine. Equivale a disegnare una linea dal punto (x1 y1) al punto (x2 y2) e così via. Queste parti fra ogni punto sono considerate segmenti. Hanno una lunghezza ed ad essi si può attribuire una direzione basata sull'ordine dei punti. Tecnicamente, una linea è data da una singola coppia di coordinate collegate insieme; una polilinea è costituita da linee multiple collegate insieme.
3. Poligoni - Quando le linee sono collegate tra loro da più di due punti, con l'ultimo punto coincidente con il primo, denominiamo questa un poligono. Triangoli, cerchi, rettangoli ecc. ecc. sono tutti poligoni. La caratteristica principale di un poligono è che essi racchiudono un'area.

3 Come iniziare

Questo capitolo fornisce una veloce panoramica dell'installazione di QGIS, alcuni dati campione dall'homepage di QGIS e su come avviare una prima semplice sessione in cui vengano visualizzati layer raster e vettoriali.


3.1 Installazione

L'installazione di QGIS è molto semplice. Pacchetti standard per l'installazione sono disponibili per MS Windows e Mac OS X. Per le distribuzioni GNU/Linux sono disponibili pacchetti binari (rpm and deb) o repositories software da aggiungere al gestore di installazione. Informazioni aggiornate possono essere reperite sul sito web di QGIS <http://qgis.osgeo.org/download/>.



La compilazione di QGIS da sorgenti è documentata nell'Appendice D per MS Windows , Appendice E per Mac OS X  e Appendice F per GNU/Linux . Le istruzioni per l'installazione sono distribuite con il codice sorgente di QGIS e sono anche disponibili su <http://qgis.osgeo.org>.

3.2 Dati campione

La guida utente contiene esempi basati sul set di dati campione di QGIS.

 L'installer per Windows comprende un'opzione per scaricare il set di dati campione di QGIS. Se selezionato, i dati verranno scaricati nella vostra cartella My Documents e posizionata in una cartella denominata GIS Database. Può essere usato Windows Explorer per spostare in questa cartella in qualunque altra posizione. Qualora non fosse stata selezionata l'opzione di installare il set di dati campione durante l'installazione iniziale di QGIS è possibile:

- usare dati GIS già posseduti;
- scaricare il set di dati dal sito di QGIS <http://qgis.osgeo.org/download>; oppure
- disinstallare QGIS e reinstallarlo selezionando l'opzione per lo scaricamento dei dati.

  Per GNU/Linux e Mac OS X non sono ancora disponibili pacchetti di installazione del set di dati campione in formato rpm, deb or dmg. Per usare il set di dati campione scaricate il file `qgis_sample_data` come archivio ZIP o TAR da <http://download.osgeo.org/qgis/data/> e decomprimetelo sul vostro sistema. Il dataset Alaska include tutti i dati GIS che sono usati come esempi e schermate nella guida utente, e include anche un piccolo database GRASS. La proiezione usata per il set di dati di QGIS è Alaska Albers Equal Area con unità in piedi. Il codice EPSG di questa proiezione è 2964.

```
PROJCS["Albers Equal Area",
```




```
GEOGCS["NAD27",  
  DATUM["North_American_Datum_1927",  
    SPHEROID["Clarke 1866",6378206.4,294.978698213898,  
      AUTHORITY["EPSG","7008"]],  
    TOWGS84[-3,142,183,0,0,0,0],  
    AUTHORITY["EPSG","6267"]],  
  PRIMEM["Greenwich",0,  
    AUTHORITY["EPSG","8901"]],  
  UNIT["degree",0.0174532925199433,  
    AUTHORITY["EPSG","9108"]],  
  AUTHORITY["EPSG","4267"]],  
PROJECTION["Albers_Conic_Equal_Area"],  
PARAMETER["standard_parallel_1",55],  
PARAMETER["standard_parallel_2",65],  
PARAMETER["latitude_of_center",50],  
PARAMETER["longitude_of_center",-154],  
PARAMETER["false_easting",0],  
PARAMETER["false_northing",0],  
UNIT["us_survey_feet",0.3048006096012192]]
```

Se si intende usare QGIS come frontend a GRASS, delle locations campione (ad es. Spearfish or South Dakota) sono disponibili sul sito ufficiale di GRASS GIS <http://grass.osgeo.org/download/data.php>.



3.3 Sessione di esempio

Ora che si è installato QGIS e si ha a disposizione un set di dati campione, dimostreremo una breve e semplice sessione di QGIS. Visualizzeremo un layer raster ed uno vettoriale. Useremo il layer raster dell'uso del suolo `qgis_sample_data/raster/landcover.img` e il layer vettoriale dei laghi `qgis_sample_data/gml/lakes.gml`.


Avvio di QGIS

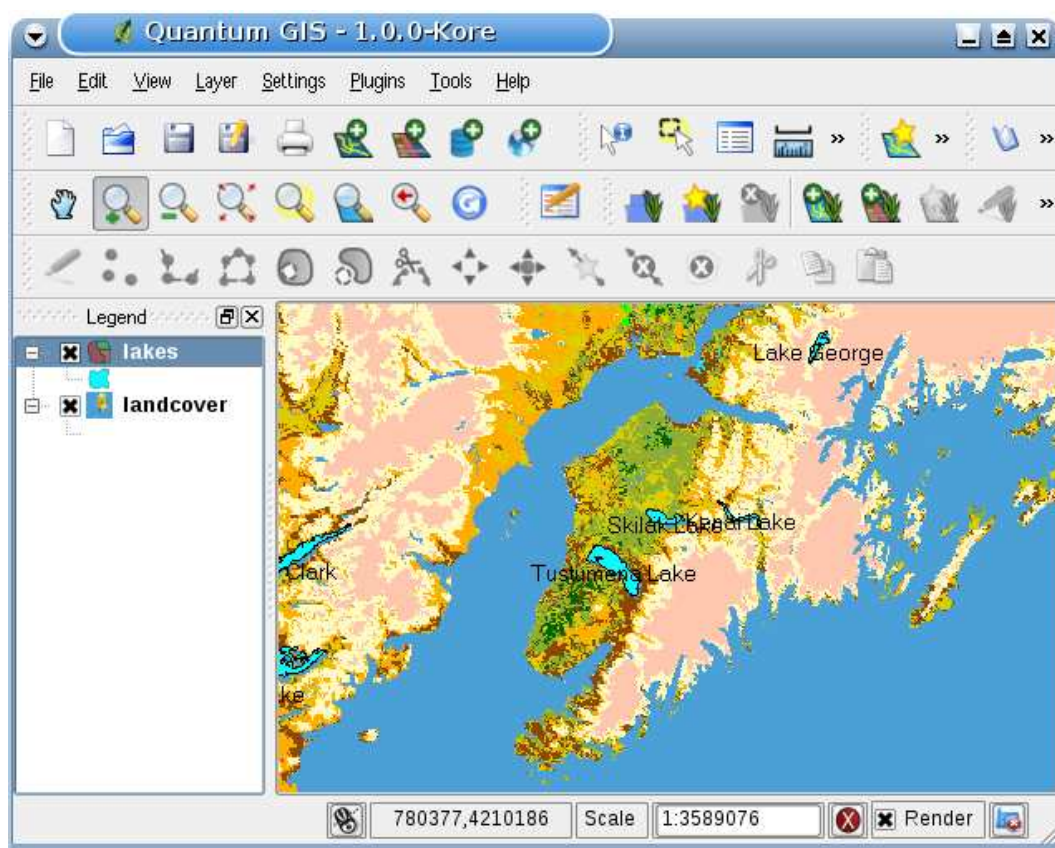
-  Avviate QGIS scrivendo: `qgis` al prompt dei comandi.
-  Avviate QGIS usando il menu Start o l'icona sul desktop, oppure facendo doppio click su un file di progetto QGIS.
-  Doppio click sull'icona nella cartella Applicazioni.

Caricare dati raster e vettoriali dal set di dati campione

1. Cliccate sull'icona  **Carica layer raster**.
2. Individuate la cartella `qgis_sample_data/raster/`, selezionate il file `ERDAS Img landcover.img` e cliccate **Apri**.
3. Ora cliccate sull'icona  **Carica layer vettoriale**.
4. Individuate la cartella `qgis_sample_data/gml/`, selezionate il file `GML lakes.gml` e cliccate **Apri**.
5. Ingrandite un pò la visuale su un'area a vostra scelta con alcuni laghi.
6. Fate doppio click sul layer `lakes` nella legenda per aprire la finestra **Proprietà del layer**.
7. Cliccate sulla linguetta **Simbologia** e selezionate blu come colore di riempimento.
8. Cliccate sulla linguetta **Etichette** e spuntate l'opzione ☒ **Mostra etichette** per abilitare l'etichettatura.
9. Cliccate **Applica**.

Visto come è facile visualizzare layer raster e vettoriali in QGIS? Proseguiamo alla sezione seguente per imparare ulteriori funzioni disponibili, caratteristiche e settaggi e come usarle.

Figura 1: Una sessione di esempio in QGIS 





4 Panoramica sulle caratteristiche

Dopo un primo sguardo e una semplice sessione d'esempio nella Sezione 3 forniamo in questo capitolo una panoramica maggiormente dettagliata delle caratteristiche di QGIS. Molte delle caratteristiche presentate nei successivi capitoli saranno spiegate e descritte successivamente nelle apposite sezioni del manuale.


4.1 Avvio e chiusura di QGIS

Nella sezione 3.3 abbiamo già imparato come avviare QGIS. Ripeteremo questa operazione qui per mostrare come QGIS fornisca ulteriori opzioni all'avvio da riga di comando.

-  assumendo che QGIS sia installato nel tuo PATH, si può avviare QGIS digitando: `qgis` al prompt dei comandi o facendo doppio click sul collegamento all'applicazione (o allo shortcut) sul desktop.
-  avvia GIS usando il menu Avvio (Start) o il collegamento sul desktop, o facendo doppio click su un progetto QGIS precedentemente salvato.
- **X** fare doppio click sull'icona QGIS nella cartella Applicazioni (Applications).

Per uscire da QGIS, cliccare sul menu   File **X** QGIS} > Esci, o usate la scorciatoia da tastiera Ctrl+Q.

4.1.1 Opzioni da linea di comando

 QGIS supporta un certo numero di opzioni se avviato da riga di comando. Per avere una lista delle opzioni possibili, digitare `qgis --help` al prompt dei comandi. La sintassi d'uso di QGIS è:

```
qgis --help
Quantum GIS - 1.0.0 'Kore'
Quantum GIS (QGIS) is a viewer for spatial data sets, including
raster and vector data.
Usage: qgis [options] [FILES]
  options:
    [--snapshot filename]  emit snapshot of loaded datasets to given file
    [--lang language]      use language for interface text
    [--project projectfile] load the given QGIS project
    [--extent xmin,ymin,xmax,ymax] set initial map extent
    [--help]                this text

FILES:
```


4 PANORAMICA SULLE CARATTERISTICHE

Files specified on the command line can include rasters, vectors, and QGIS project files (.qgs):

1. Rasters - Supported formats include GeoTiff, DEM and others supported by GDAL
2. Vectors - Supported formats include ESRI Shapefiles and others supported by OGR and PostgreSQL layers using the PostGIS extension

Tip 2 ESEMPIO DI UTILIZZO DELLE OPZIONI DA RIGA DI COMANDO

QGIS può essere avviato specificando uno o più files da riga di comando. Per esempio, assunto che ci si trovi nella directory `qgis_sample_data`, si può avviare QGIS con un layer vettoriale ed un file raster inserendo il seguente comando: `qgis ./raster/landcover.img ./gml/lakes.gml`

Opzione da linea di comando `--snapshot`

L'opzione consente di creare uno snapshot in formato PNG della vista corrente. Questo può essere utile quando si hanno molti progetti e si vogliono generare schermate dai propri dati.

Il file PNG generato ha una risoluzione di 800x600 pixels. Dopo l'opzione `--snapshot` può essere specificato il nome del file con cui si vuole salvare l'immagine.

Opzione da linea di comando `--lang`

L'interfaccia di QGIS si presenta nella lingua definita dal setting del locale di sistema. Se si desidera l'interfaccia in un'altra lingua, può essere specificato all'avvio. Ad esempio: `--lang=en` fa sì che QGIS si avvii localizzato in inglese. Un elenco delle lingue correntemente supportate è fornito all'indirizzo <http://wiki.qgis.org/qgiswiki/TranslatorsCorner>

Opzione da linea di comando `--project`

È possibile avviare QGIS anche con un file di progetto. Basta semplicemente aggiungere l'opzione da riga di comando `-project` seguita dal percorso e dal nome del progetto e QGIS si aprirà caricando tutti i layer indicati nel file specificato.

Opzione da linea di comando `--extent`

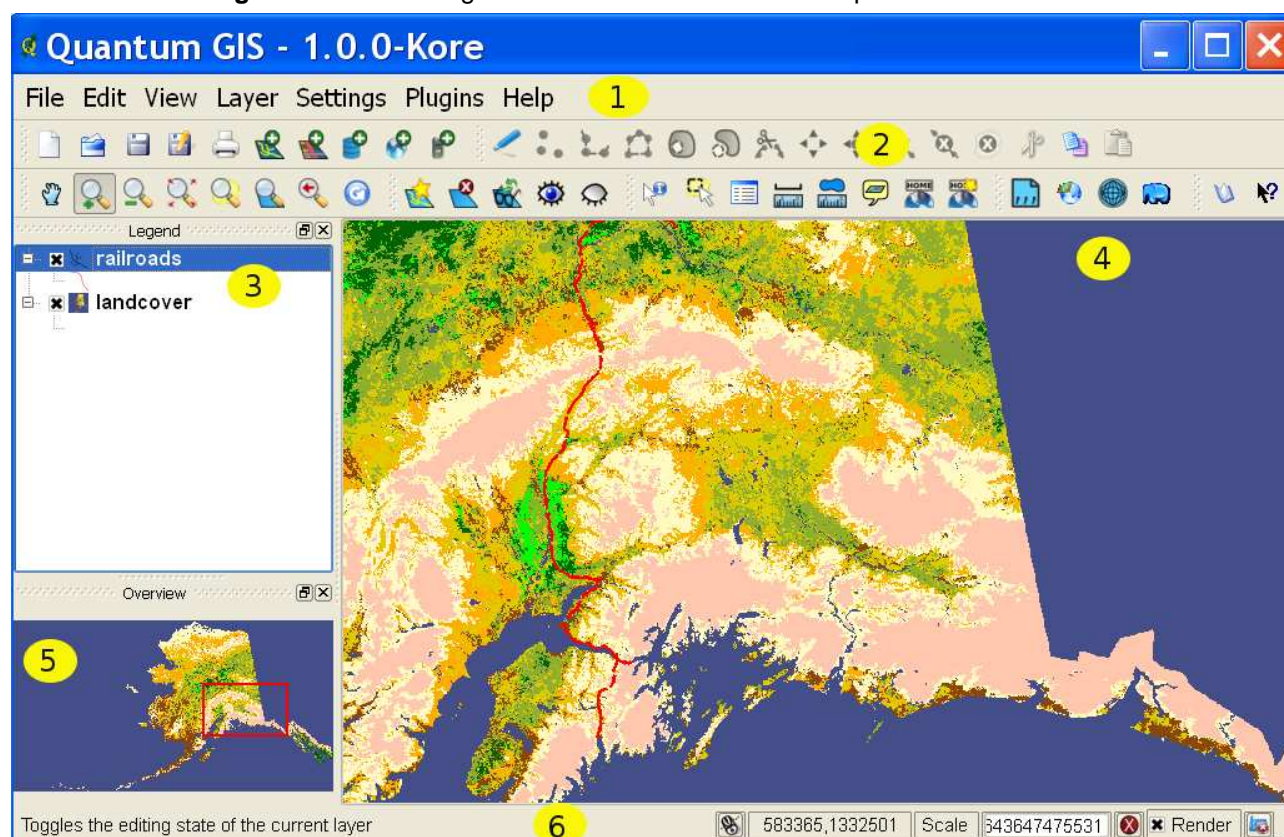
Per fare sì che QGIS si avvii visualizzando una specifica porzione di mappa, è necessario specificare i limiti dell'estensione che si intende visualizzare secondo il seguente ordine, con ogni valore separato da virgole:

```
--extent xmin,ymin,xmax,ymax
```

4.2 QGIS GUI

Quando QGIS viene avviato, viene mostrata l'interfaccia grafica come di seguito mostrato (i numeri da 1 a 6 negli ovali gialli fanno riferimento alle sei principali aree della interfaccia come qui specificato):

Figura 2: Interfaccia grafica di QGIS GUI con dati campione dell'Alaska 



Nota: L'aspetto delle finestre (barra del titolo, ecc.) può apparire diverso secondo il sistema operativo e il gestore di finestre adottato.
















L'interfaccia di QGIS è divisa in sei aree:

- | | |
|--------------------|-------------------|
| 1. Barra Menu | 4. Vista Mappa |
| 2. Barra Strumenti | 5. Panoramica |
| 3. Legenda | 6. Barra di Stato |

Le sei componenti dell'interfaccia di QGIS saranno descritte con maggior dettaglio nelle sezioni seguenti.

4.2.1 Barra Menù

La barra menù fornisce accesso alle varie caratteristiche di QGIS utilizzando un menù gerarchico standard. Le voci gerarchicamente più elevate e una sintesi di alcune opzioni di menù sono elencate di seguito, assieme alle icone dello strumento corrispondente così come appaiono nella barra strumenti e alle scorciatoie da tastiera per richiamarle. Sebbene molte voci di menù abbiano uno strumento corrispondente e viceversa, i menù non sono organizzati esattamente come le barre strumenti. La barra strumenti che contiene lo strumento descritto è indicata dopo ogni voce di menù con l'aspetto di una casella di controllo (checkbox). Per ulteriori informazioni su strumenti e barre degli strumenti, si veda la Sezione 4.2.2.

Voce di Menù	Scorciatoia	Riferimento	Barra strumenti
• File			
 Nuovo progetto	Ctrl+N	vedi Sezione 4.5	<input checked="" type="checkbox"/> File
 Apri progetto	Ctrl+O	vedi Sezione 4.5	<input checked="" type="checkbox"/> File
Apri progetti recenti		vedi Sezione 4.5	
 Salva progetto	Ctrl+S	vedi Sezione 4.5	<input checked="" type="checkbox"/> File
 Salva progetto con nome	Ctrl+Shift+S	vedi Sezione 4.5	<input checked="" type="checkbox"/> File
 Salva come immagine		vedi Sezione 4.6	
 Compositore stampe	Ctrl+P	vedi Sezione 10	<input checked="" type="checkbox"/> File
 Esci	Ctrl+Q		
• Modifica			
 Taglia geometrie	Ctrl+X	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
 Copia geometrie	Ctrl+C	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
 Incolla geometrie	Ctrl+V	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
 Inserisci punto	.	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
 Inserisci linea	/	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
 Inserisci poligono	Ctrl+/,	vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
E altre voci del menù Modifica		vedi Sezione 5.4.3	<input checked="" type="checkbox"/> Digitalizza
• Visualizza			
 Sposta mappa			<input checked="" type="checkbox"/> Navigazione mappa
 Ingrandisci	Ctrl++		<input checked="" type="checkbox"/> Navigazione mappa

Rimpicciolisci	Ctrl+-
Seleziona geometrie	
Informazioni geometrie	I
Misura linea	M
Calcola l'area	J
Vista massima	F
Zoom sul layer	
Zoom sulla selezione	Ctrl+J
Ultimo zoom	
Zoom dimensione corrente	
Suggerimenti mappa	
Nuovo segnalibro	Ctrl+B
Mostra segnalibri	B
Aggiorna	Ctrl+R

☒ Navigazione mappa
☒ Attributi
☒ Attributi
☒ Attributi
☒ Attributi
☒ Navigazione mappa
☒ Navigazione mappa
☒ Navigazione mappa
☒ Navigazione mappa
☒ Attributi
☒ Attributi
☒ Attributi
☒ Navigazione mappa

vedi Sezione 4.8

vedi Sezione 4.8

• Layer

Nuovo layer vettoriale	N
Aggiungi layer vettoriale	V
Aggiungi layer raster	R
Aggiungi layer PostGIS	D
Aggiungi layer WMS	W
Apri tabella attributi	
Attiva/disattiva modifica	
Salva come shapefile	
Salva selezione come shapefile	
Elimina Layer	Ctrl+D
Proprietà	
Aggiungi alla panoramica	A
Aggiungi tutto alla panoramica	A
Rimuovi tutto dalla panoramica	A
Nascondi tutti i layer	H
Mostra tutti i layer	S

vedi Sezione 5.4.4

vedi Sezione 5

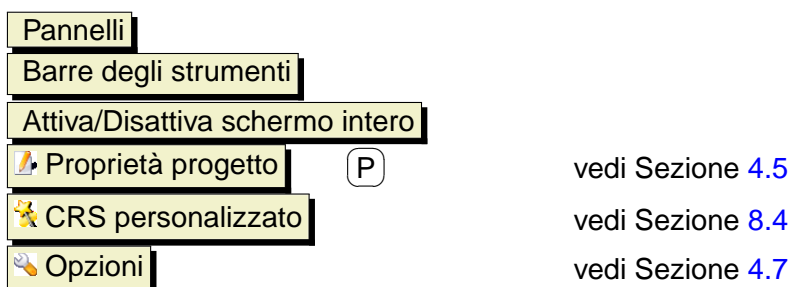
vedi Sezione 6

vedi Sezione 5.2

vedi Sezione 7.2

☒ Gestisci layer
☒ File
☒ File
☒ File
☒ File
☒ Attributi
☒ Digitalizza
☒ Gestisci layer
☒ Gestisci layer
☒ Gestisci layer
☒ Gestisci layer

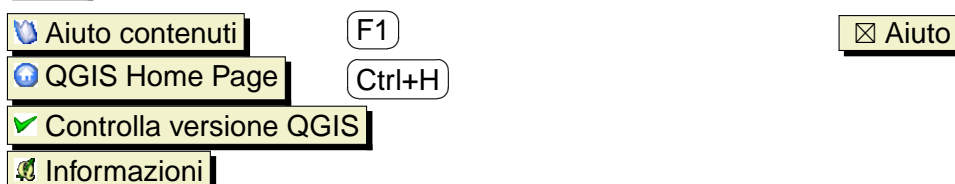
• Impostazioni



- **Plugins** - (Ulteriori voci di menu vengono aggiunte quando si installano e caricano nuovi plugins.)



- **Aiuto**



4.2.2 Barre degli strumenti

Le barre degli strumenti forniscono accesso alla maggior parte delle medesime funzioni presenti nei menù, oltre a funzioni aggiuntive per interagire con la mappa. Ogni oggetto della barra strumenti ha un aiuto a comparsa disponibile. Lasciando il cursore del mouse sopra l'icona verrà visualizzata una breve descrizione della funzione fornita da quello strumento.

Ogni barra può essere spostata secondo le preferenze personali. Inoltre ognuna di esse può essere disattivata dal menù contestuale ottenibile cliccando con il tasto destro sulla toolbar.

Tip 3 RIPRISTINARE BARRE DEGLI STRUMENTI

Se avete accidentalmente disattivato tutte le barre strumenti, possono essere ripristinate dalla voce di menù

Impostazioni > **Barre degli strumenti**.

4.2.3 Legenda

L'area di legenda è usata per impostare la visibilità e l'ordine di sovrapposizione dei layers: i layers in cima alla legenda coprono nella mappa quelli sottostanti. La casella di controllo in ogni legenda può essere usata per attivare/disattivare la visibilità di un layer.

È possibile raggruppare layer nella legenda aggiungendo un gruppo e trascinando i layer nel gruppo. Per fare ciò cliccare con il tasto destro nell'area di legenda, scegliere **Aggiungi gruppo**. Apparirà una nuova cartella nella quale possono essere trascinati i livelli che si intendono raggruppare. È quindi possibile attivare/disattivare la visibilità di tutti i layer nel gruppo con un solo click. Per estrarre un layer dal gruppo, muovere il puntatore del mouse sul simbolo del layer, cliccare con il tasto destro e scegliere **Crea all'oggetto in cima**. Per rinominare il gruppo, fare click con il tasto destro sulla cartella e scegliere **Rinomina** dal menù contestuale.

Il contenuto del menù contestuale varia a seconda che si faccia click col tasto destro su un layer raster o vettoriale. Per i layer vettoriali di GRASS la voce di menù **Attiva/disattiva modifica** non è abilitata. Si veda la sezione 9.7 per informazioni su come editare layers vettoriali GRASS.

- **Menù relativo ai raster attivabile con il tasto destro del mouse**

- Zoom all'estensione del layer
- Zoom alla scala migliore (100%)
- Aggiungi alla panoramica
- Rimuovi
- Proprietà
- Rinomina
- Aggiungi gruppo
- Espandi tutto
- Comprimi tutto
- Mostra gruppi

- **Menù relativo ai vettoriali attivabile con il tasto destro del mouse**

- Zoom all'estensione del layer
- Aggiungi alla panoramica
- Rimuovi
- Apri tabella attributi
- Attiva/disattiva modifica (non disponibile per layers GRASS)
- Salva come shapefile
- Salva selezione come shapefile
- Proprietà
- Crea all'oggetto in cima
- Rinomina
- Aggiungi gruppo

- Espandi tutto
- Comprimi tutto
- Mostra gruppi

- **Menù relativo ai gruppi di strati attivabile con il tasto destro del mouse**

- Rimuovi
- Rinomina
- Aggiungi gruppo
- Espandi tutto
- Comprimi tutto
- Mostra gruppi

Se diversi vettori hanno lo stesso tipo di geometria e gli stessi attributi le loro proprietà di visualizzazione possono essere raggruppate. Questo significa che se la proprietà di visualizzazione di uno strato è cambiata automaticamente gli altri prenderanno lo stesso tipo di visualizzazione. Per raggruppare la simbologia scegli la voce **Mostra gruppi** dal menù contestuale dei Gruppi di strati e sarà possibile trascinare i file da un file di gruppo ad un altro. Se questo verrà fatto la simbologia sarà raggruppata. Nota che QGIS permette di spostare i file di gruppo solo se i due strati sono compatibili (stessa geometria e stessi attributi).

4.2.4 Il visualizzatore di mappa

Questa è l'area in cui le mappe vengono visualizzate. La mappa visualizzata in questa finestra sarà il risultato dei livelli vettoriali e raster che avete scelto di caricare (vedere le sezioni che seguono per ulteriori informazioni su come caricare i livelli). La zona di visualizzazione della mappa può essere modificata (spostando la messa a fuoco dell'esposizione della mappa ad un'altra regione) ed è possibile effettuare operazioni di zoom in ed out (+ e -). Varie altre operazioni sono descritte nella sezione relativa alla barra dei menù. La vista nell'area di mappa e la legenda sono strettamente legate l'una all'altra - le mappe che vengono visualizzate riflettono i cambiamenti che fate nella zona della legenda.


Tip 4 ZOOMARE RAPIDAMENTE CON LA ROTELLA DEL MOUSE

Può essere usata la rotella del mouse per le operazioni di zoom. Posizionando il puntatore nell'area di visualizzazione delle mappe si aumenta lo zoom girando la rotella verso lo schermo, lo si riduce girandola verso di sé. La posizione del puntatore costituisce il centro per l'ingrandimento. Il comportamento della funzione di zoom con la rotella del mouse può essere regolata dalla linguetta **Strumenti di mappa** sotto il menù **Impostazioni** > **Opzioni**.

Tip 5 MUOVERE LA MAPPA CON I TASTI FRECCIA E LA BARRA SPAZIATRICE

È possibile usare i tasti freccia per muovere la mappa. Posizionando il puntatore del mouse nell'area di visualizzazione delle mappe, ci si muove verso Est con la freccia destra, verso Ovest con quella sinistra, verso Nord con la freccia Sù e verso Sud con la freccia Giù. Si può spostare la mappa anche tenendo premuta la barra spaziatrice e muovendo il mouse.

4.2.5 Panoramica della mappa

La panoramica della mappa fornisce una vista completa dei livelli aggiunti ad essa. All'interno della panoramica c'è un rettangolo che mostra l'estensione corrente della mappa. Ciò permette di determinare rapidamente quale area della mappa state attualmente osservando. Si noti che le etichette non sono visualizzate nella panoramica della mappa anche se i livelli hanno la funzione di visualizzazione delle etichette attiva. Può essere aggiunto un singolo livello alla panoramica facendo click col tasto destro su di esso nella legenda e scegliendo poi  Aggiungi alla panoramica. Possono anche essere aggiunti o rimossi tutti i livelli nella panoramica usando lo strumento 'Panoramica' nella barra degli strumenti. Cliccando e trascinando nella panoramica il rettangolo rosso che mostra l'estensione corrente della vostra visuale, la mappa visualizzata si sposta di conseguenza.

4.2.6 Barra di stato

La barra di stato mostra la posizione del cursore attuale nella mappa nelle coordinate specificate (es. metri o gradi decimali). Nella barra di stato a sinistra del visore delle coordinate è presente un piccolo bottone che consente di passare dalla visione delle coordinate al movimento del puntatore a quella dell'estensione della vista quando si effettua lo zoom.

Una barra di avanzamento nella barra di stato mostra il progredire della visualizzazione di ogni strato nella vista mappa. In alcuni casi, come quando vengono raccolte informazioni statistiche sul layer raster, questa barra è utilizzata per mostrare lo stato di tali processi, in genere molto lunghi.


Se è disponibile un nuovo plugin o un aggiornamento ad un plugin installato, nella barra di stato apparirà un avviso. Sulla destra della barra di stato è presente una casella di controllo (checkbox) che può essere usata per disattivare temporaneamente la visualizzazione dei layer nella vista (vedi Sezione 4.3 più avanti). All'estrema destra della barra di stato è presente l'icona di un proiettore. Cliccando su di essa si apre la finestra con le proprietà della proiezione del progetto corrente.

4.3 Visualizzazione

Come impostazione di default, QGIS mostra nella vista tutti i layer visibili ogni qualvolta la vista debba essere rivisualizzata. Gli eventi che richiedono una rivisualizzazione della vista includono:

Tip 6 CALCOLARE LA SCALA CORRETTA DELLA MAPPA

Quando si avvia QGIS, l'unità di default sono i gradi, il che fa sì che QGIS interpreti ogni coordinata del layer in gradi. Per avere il corretto valore della scala, è possibile sia cambiare manualmente l'unità di misura a metri nella linguetta **Generale** sotto il menù **Impostazioni** > **Proprietà della proiezione** oppure può essere scelto un Sistema di Riferimento per le Coordinate (Coordinate Reference System, CRS), cliccando

sull'icona  **Stato CRS** nell'angolo in basso a destra della barra di stato. In quest'ultimo caso, le unità della mappa sono impostate a quelle che specifica la proiezione scelta, ad es. '+units=m'.

- Aggiunta di un layer
- Spostamento, ingrandimento o riduzione
- Ridimensionamento della finestra di QGIS
- Cambiamento della visibilità di uno o più layer

QGIS consente di controllare il processo di resa a video in diverse maniere.

4.3.1 Visualizzazione in funzione della scala

La visualizzazione in funzione della scala consente di specificare la scala minima e massima rispettivamente al di sotto e al di sopra della quale un layer è visibile. Per impostare la visualizzazione in funzione della scala, aprire la finestra di dialogo **Proprietà** facendo doppio click sul layer nella legenda. Nella linguetta **Generale**, spuntare la casella di controllo

☒ Utilizzare una scala in relazione al rendering e impostare i valori per la scala minima e massima.

I valori di scala possono essere determinati zommando dapprima sul livello per il quale si vuole impostare l'opzione e prendendo nota del valore di scala visualizzato nella barra di stato di QGIS.

4.3.2 Controllo della resa a video della mappa

La resa a video mappa può essere controllata nei seguenti modi:

a) Sospensione della resa a video

Per sospendere la resa a video, cliccare sulla casella di controllo ☒ **Disegna** in basso a destra della barra di stato. Quando la casella ☒ **Disegna** non è selezionata, QGIS non ridisegna la vista quando si verifica uno degli eventi precedentemente descritti alla Sezione 4.3. Casi in cui si potrebbe voler sospendere la visualizzazione a video includono:

- Aggiunta di molti layer e applicazione di uno stile visuale prima della resa

- Aggiunta di uno o più layer di grosse dimensioni e impostazione di una scala prima della resa
- Aggiunta di uno o più layer di grossa dimensione e zoom ad un'area specifica prima della resa
- Combinazioni delle precedenti

Selezionando la casella di controllo ☒ Disegna abilita la resa a video e causa l'immediata rivisualizzazione della vista mappa.

b) Regolazione dell'opzione per controllare la visibilità degli strati quando sono aggiunti

Può essere scelta l'opzione di caricare i nuovi layer senza che essi vengano immediatamente resi a video. Ciò significa che quando si aggiungerà un layer al progetto la casella di controllo per la visibilità nella legenda sarà disabilitata di default. Per impostare questa opzione, scegliere l'opzione di menù **Impostazioni** > **Opzioni** e cliccate sulla linguetta **Disegno**. Deselezionare la casella di controllo ☒ Per impostazione predefinita i nuovi layer aggiunti alla mappa vengono visualizzati subito. Ogni layer aggiunto alla mappa sarà quindi spento (invisibile) di default.

c) Aggiornamento della mappa durante la visualizzazione

Può essere impostata un'opzione per aggiornare la mappa man mano che gli elementi del layer vengono visualizzati. Di default, QGIS non mostra alcun elemento dei layer fino a che l'intero layer è stato renderizzato. Per aggiornare il display man mano che gli elementi sono letti dall'archivio, selezionare la voce di menù **Impostazioni** > **Opzioni** e cliccare sulla linguetta **Disegno** tab. Impostare il numero di elementi che si desidera vengano letti prima che lo schermo venga aggiornato. Un valore pari a 0 disabilita l'aggiornamento durante la tracciatura degli oggetti (impostazione di default). Un valore troppo basso diminuisce le prestazioni in quanto la mappa viene continuamente aggiornata man mano che gli elementi del layer vengono letti. Il valore di prova suggerito per iniziare è 500.

d) Modificare la qualità della resa a video


Vi sono 3 opzioni per modificare la qualità della resa a video. Dal menù **Impostazioni** > **Opzioni** cliccare sulla linguetta **Disegno** e selezionare o deselezionare le seguenti caselle di controllo.

- ☒ Rendi le linee meno dettagliate in favore di migliori prestazioni nel disegno
- ☒ Risolvi problemi con poligoni riempiti non correttamente
- ☒ Ridisegna la mappa mentre viene spostato il divisore legenda/mappa

4.4 Misurare

È possibile effettuare misure unicamente nei sistemi di coordinate piani (es. UTM). Se la mappa caricata è definita in un sistema di coordinate geografiche (es. latitudine/longitudine), il risultato della misura di linee o di aree saranno errati. Per misurare è quindi necessario impostare correttamente il sistema di coordinate della mappa (si veda la Sezione 8).

4.4.1 Misurare lunghezze ed aree

 QGIS è in grado di fornire la misura di distanza reale tra due punti in funzione di un definito ellissoide. Ciò può essere configurato dall'opzione di menù **Impostazioni** > **Opzioni**, cliccando sulla linguetta **Strumenti mappa** e scegliendo l'ellissoide appropriato. Lo strumento consentirà allora di cliccare punti sulla mappa. La misura di ogni segmento verrà mostrata nella finestra dello strumento insieme alla misura totale. Per terminare la funzione misura cliccare con il tasto destro del mouse.



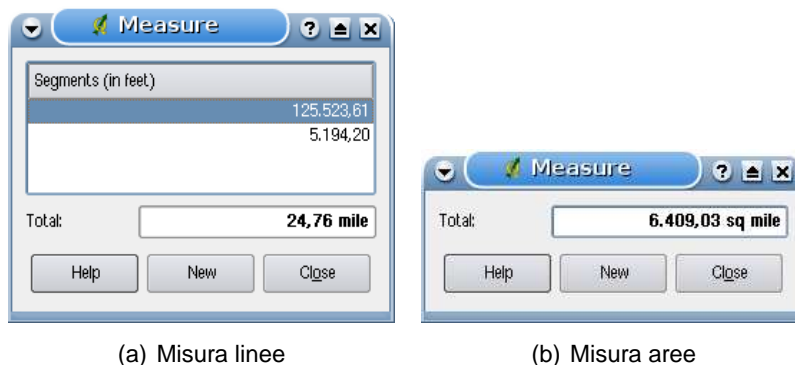
 Questo strumento consente la misura di aree, la finestra mostrerà unicamente l'area totale misurata.

Figura 3: Strumento di misura in azione 



4.5 Progetti

Lo stato della sessione QGIS è considerata un progetto. È possibile lavorare su un progetto alla volta. Le impostazioni possono essere settate per ogni singolo progetto oppure come default per tutti i nuovi progetti (si veda la Sezione 4.7). Lo stato della sessione corrente può essere salvato in un progetto usando la voce di menù **File** > **Salva progetto** o **File** > **Salva progetto con nome**.

Per caricare progetti salvati usare **File** > **Apri progetto** o **File** > **Apri progetti recenti**. Se si

vuole eliminare la sessione corrente e ricominciare da zero, scegliere **File** > **Nuovo progetto**. Ognuna di queste voci di menù chiederà se si vuole salvare la sessione corrente se sono stati effettuati cambiamenti dall'ultima volta in cui essa è stata aperta o salvata.

Le informazioni salvate nel file di progetto includono:

- Layers aggiunti
- Proprietà dei layers, inclusa la loro rappresentazione grafica
- Proiezione usata per la vista
- Ultima estensione della vista (scala e inquadramento)

Il file di progetto è salvato in formato XML, così che sia possibile editarlo esternamente a QGIS con qualunque editor se se conosce la sintassi. Il formato del file di progetto è stato modificato parecchie volte rispetto a quello delle precedenti versioni di QGIS, di conseguenza file di progetto salvati con precedenti versioni di QGIS potrebbero non funzionare più correttamente. Si può essere avvertiti preventivamente di ciò selezionando dalla linguetta **Generale** nel menù **Impostazioni** > **Opzioni** la casella di controllo

☒ Avvisa quando un file di progetto viene salvato con una vecchia versione di QGIS.

4.6 Output

Ci sono diversi modi di generare file di output dalla sessione QGIS. Il primo è stato descritto alla Sezione 4.5 e consiste nel salvataggio su file di progetto. Altri modi di produrre file di output sono ad esempio:

- L'opzione di menù **Salva come immagine** apre una finestra del file browser nella quale indicare nome, percorso ed estensione del formato immagine (PNG or JPG).
- L'opzione di menù **Compositore stampe** apre una finestra nella quale è possibile comporre un layout per stampare la vista mappa corrente (vedi Sezione 10).

4.7 Opzioni dell'interfaccia grafica (GUI)



Alcune opzioni di base per QGIS possono essere impostate dalla finestra **Opzioni** dialog. Selezionare la voce di menù **Impostazioni** > **Opzioni**. Le linguette nelle quali possono essere regolate le opzioni sono:

Generale

- ☒ Richiedi di salvare i cambiamenti di progetto se necessario

- ☒ Avvisa quando un file di progetto viene salvato con una vecchia versione di QGIS
- ☒ Scelta del colore per evidenziare la selezione e lo sfondo della vista mappa
- Cambio del tema delle icone (scelta tra default, classica, gis e nkids)
- ☒ Rendi maiuscoli i nomi dei layer nella legenda
- ☒ Visualizza i nomi degli attributi della classificazione della legenda
- ☒ Nascondi lo splash screen all'avvio
- ☒ Apri la tabella attributi in una finestra separata
- Comportamento della tabella attributi (scelta tra mostra tutte le geometrie, mostra le geometrie selezionate, mostra le geometrie della vista attuale)

Disegno

- ☒ Per impostazione predefinita i nuovi layer aggiunti alla mappa vengono visualizzati subito
- Definizione del numero di geometrie da disegnare prima di aggiornare lo schermo.
- ☒ Rendi le linee meno dettagliate in favore di migliori prestazioni di disegno
- ☒ Risolvi problemi con i poligoni riempiti non correttamente
- ☒ Ridisegna la mappa mentre viene spostato il divisore legenda/mappa

Strumenti mappa

- Definizione del raggio di ricerca per identificare gli oggetti e visualizzare le relative informazioni sulla mappa in percentuale della larghezza della mappa
- Definizione dell'ellissoide da impiegare per il calcolo delle distanze
- Definizione del colore della traccia (colore elastico) nello strumento misura
- Definizione del comportamento della rotellina del mouse (Zoom, Zoom e centramento, Zoom al cursore del mouse, Niente)
- Definizione del fattore di zoom quando si aziona la rotellina del mouse

Digitalizzazione

- Definizione del colore e della larghezza della traccia quando si digitalizza
- Definizione della modalità di aggancio predefinita (al vertice, al segmento o entrambe)
- Definizione della tolleranza (distanza massima) per attivare lo snapping in unità del layer

- Definizione del raggio di ricerca per la cattura e modifica di vertici in unità del layer
- Definizione del simbolo per i vertici (Croce o cerchio semitrasparente)

CRS

- ☒ Richiesta per CRS
- ☒ CRS predefinito utilizzato dal progetto
- ☒ Verrà utilizzato il seguente CRS globale predefinito visualizzato qui sotto
- Selezioni globali predefinite




Lingua

- ☒ Sovrascrivi lingua in uso
- Informazioni sulla lingua correntemente impostata nel sistema

Proxy

- ☒ Utilizza un proxy per l'accesso web, definizione di host, porta, utente e password.

Queste opzioni possono essere modificate in funzione delle specifiche esigenze. Alcuni cambiamenti potrebbero rendere necessario riavviare QGIS prima che divengano attivi.

-  Le impostazioni sono salvate in un file di testo: `$HOME/.config/QuantumGIS/qgis.conf`
-  Le impostazioni vengono collocate in: `$HOME/Library/Preferences/org.qgis.qgis.plist`
-  Le impostazioni vengono inserite nel registro di sistema alla voce:
`\\HKEY\\CURRENT\\USER\\Software\\QuantumGIS\\qgis`

4.8 Segnalibri geospaziali

I segnalibri geospaziali consentono di "memorizzare" una posizione geografica alla quale ritornare in un secondo momento.

4.8.1 Creazione di un segnalibro

Per creare un segnalibro:

1. Zommare o muovere la mappa all'estensione di interesse.

2. Selezionare la voce di menù **Visualizza** > **Nuovo segnalibro** o premere **Ctrl-B**.
3. Inserire un nome descrittivo per il segnalibro (fino a 255 caratteri).
4. Cliccare su **OK** per aggiungere il segnalibro o **Cancel** per uscire senza aggiungere il segnalibro.

Si noti che è possibile avere più di un segnalibro con lo stesso nome.

4.8.2 Uso e gestione dei segnalibri

Per usare o gestire i segnalibri, selezionare la voce di menù **Visualizza** > **Mostra segnalibri**. La finestra **Segnalibri geospaziali** consente di zoommare a un segnalibro o di eliminarne uno. Non è possibile editare il nome o le coordinate di un segnalibro.

4.8.3 Zoommare a un segnalibro

Dalla finestra **Segnalibri geospaziali** dialog, selezionare il segnalibro desidera cliccando su di esso, quindi cliccare su **Zoom a**. Si può zoommare su un segnalibro anche facendo doppio click su di esso.

4.8.4 Cancellare un seganlibro

Per cancellare un segnalibro dalla finestra **Segnalibri geospaziali**, cliccare su di esso e poi sul bottone **Elimina**. Confermare la scelta cliccando su **OK** o annullare l'eliminazione cliccando su **Cancel**.

5 Lavorare con i dati vettoriali

QGIS supporta un gran numero di formati vettoriali, inclusi quelli supportati dalla libreria OGR inclusa, come gli ESRI shapefiles, , il formato di interscambio MapInfo MIF e il formato nativo MapInfo TAB (native format). Una lista dei formati vettoriali supportati da OGR si trova nell'Appendice [A.1](#).

QGIS supporta anche strati PostGIS immagazzinati in un database PostgreSQL usando il plugin di accesso a PostgreSQL. Il supporto per altri formati (ad es. testo delimitato) è fornito da ulteriori plugins specifici.

Questa sezione descrive come lavorare con due formati di uso comune: ESRI shapefiles e strati PostGIS. Molti degli strumenti disponibili in QGIS funzionano allo stesso modo con le differenti sorgenti di dati vettoriali (ad es. l'identificazione, la selezione, la visualizzazione delle etichette ed altre funzioni).

La Sezione [9](#) illustra come lavorare con i dati di GRASS.

5.1 ESRI Shapefiles

Il formato di file usato come default in QGIS è ESRI Shapefile. Il supporto a tale formato è fornito dalla libreria OGR Simple Feature Library (<http://www.gdal.org/ogr/>) . Uno shapefile consiste di un minimo di tre files:

- .shp file contenente le geometrie.
- .dbf file contenente gli attributi in formato dBase.
- .shx file di indice.

Idealmente dovrebbe essere presente un altro file con estensione .prj, che contiene le informazioni sulla proiezione dello shapefile. Ci possono essere ulteriori files che compongono il dataset in formato shape. Per uno sguardo più ravvicinato al formato shapefile si raccomanda di prendere visione delle specifiche tecniche del formato disponibili sul sito <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf> .

5.1.1 Caricare uno shapefile



Per caricare uno shapefile avviare QGIS e cliccare sul pulsante



Aggiungi layer vettoriale

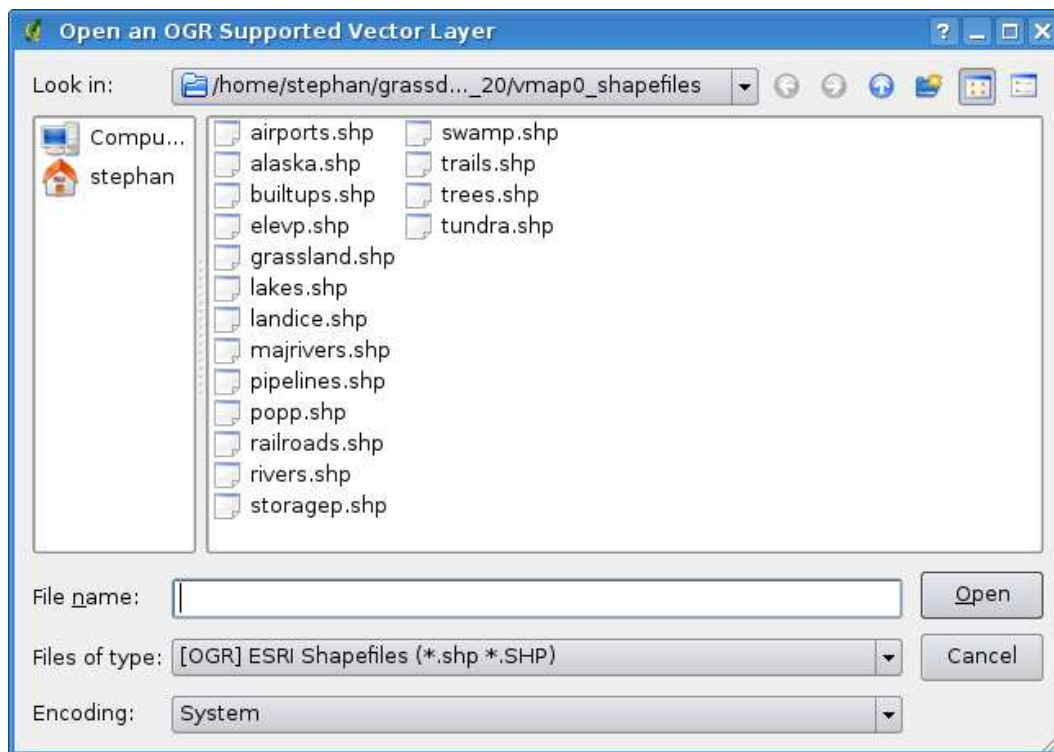
o semplicemente digitare **V**. Lo stesso strumento può essere usato per caricare qualunque dei formati supportati dalla libreria OGR.

Cliccando sullo strumento si apre una finestra di dialogo standard (si veda la Figura [4](#)) che consente

di cercare nel filesystem lo shapefile o qualunque altro dato vettoriale si intenda caricare. La casella di selezione **Files of type** [...] consente di preselezionare alcuni formati supportati da OGR.

Se lo si desidera, può essere inoltre selezionata la codifica (encoding) da utilizzare per le porzioni testuali della tabella dello shapefile.

Figura 4: Finestra di dialogo Apri un layer vettoriale supportato da OGR 🐧

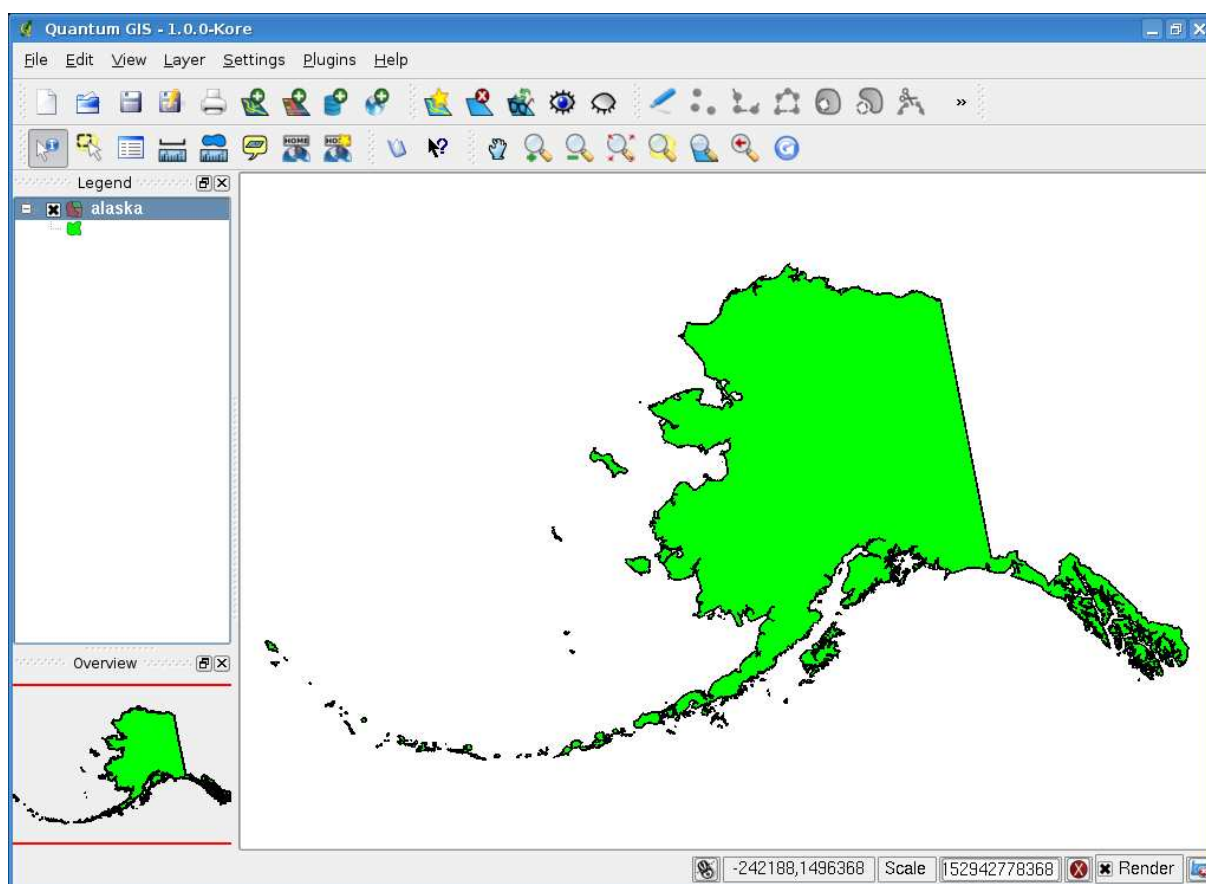


Selezionando uno shapefile dalla lista e cliccando su **Open** esso viene caricato in QGIS. La figura 5 mostra come appare l'interfaccia di QGIS dopo aver caricato il file `alaska.shp`.

Tip 7 COLORI DEL LAYER

Quando uno strato viene aggiunto alla mappa, gli viene assegnato un colore a caso. Aggiungendo più layer in una sola volta, ad ognuno di essi viene assegnato un colore differente.

Una volta caricato, si può zoommare nella mappa usando gli strumenti di navigazione. Per cambiare la rappresentazione di un layer, aprire la finestra **Proprietà** facendo doppio click sul nome del layer e quindi sulla linguetta Simbologia o cliccando con il tasto destro sul nome del layer nella legenda e scegliendo **Proprietà** dal menù contestuale. Si veda la Sezione 5.3.2 per ulteriori informazioni su come settare la simbologia dei layer vettoriali.

Figura 5: QGIS con lo shapefile Alaska caricato 



5.1.2 Ottimizzare le prestazioni

Per migliorare le prestazioni di disegno di uno shapefile, può essere creato un indice spaziale. Un indice spaziale migliorerà la velocità di disegno quando si usano le funzioni di zoom e di spostamento. Gli indici spaziali usati da QGIS hanno estensione `.qix`.


Per creare un indice, seguire queste indicazioni:

- Caricare uno shapefile.
- Aprire la finestra di dialogo **Proprietà** facendo doppio click sul nome dello shapefile nella legenda o cliccando su di esso con il tasto destro e scegliendo la voce **Proprietà** dal menù contestuale.
- Nella linguetta **Generale** cliccare sul pulsante **Crea indice spaziale**.

5.1.3 Caricare uno strato MapInfo

Per caricare uno strato MapInfo, cliccare nella barra strumenti sul pulsante  **Aggiungi layer vettoriale** o digitare **V**, cambiare il filtro sul tipo di file nel menù a tendina a **Files of Type** **[OGR] MapInfo (*.mif *.tab *.MIF *.TAB)**  e selezionare lo strato che si intende caricare.

5.1.4 Caricare una coverage ArcInfo

Per caricare una coverage ArcInfo si usa lo stesso metodo precedentemente visto per shapefiles e layers MapInfo. Cliccare nella barra strumenti sul pulsante  **Aggiungi layer vettoriale** o digitare **V**, scorrere il filesystem per individuare la cartella contenente la coverage che si vuole caricare e selezionare uno dei seguenti file (se presenti nella propria coverage):

- **.lab** - per caricare un livello etichette (etichette di poligoni o di punti).
- **.cnt** - per caricare un livello contenente i centroidi dei poligoni
- **.arc** - per caricare un livello arc (linee).
- **.pal** - per caricare un livello di poligoni.

5.2 Gli strati PostGIS

I layer layers sono immagazzianti in un database PostgreSQL. I vantaggi nell'uso di PostGIS stanno nella capacità di creazione dell'indice spaziale, di filtraggio e di interrogazione fornite. Usando PostGIS, le funzioni vettoriali come la selezione e l'identificazione in QGIS lavorano con maggiore precisione che con i livelli OGR.

Per usare livelli PostGIS layers bisogna:

- Creare una connessione in QGIS con il database PostgreSQL (se non è già definita).
- Connettersi al database.
- Selezionare i layer da aggiungere alla mappa.
- Fornire eventualmente una query SQL di tipo *where* per definire quali elementi del layer caricare.
- Caricare il layer.

5.2.1 Creare una connessione




 La prima volta in cui viene usata una fonte dati PostGIS, bisogna creare una connessione al database PostgreSQL che contiene i dati. Cliccare nella barra strumenti sul pulsante  **Aggiungi layer PostGIS**, oppure selezionare l'opzione  **Aggiungi layer PostGIS...** dal menù **Layer** o digitare **D**. Si aprirà la finestra di dialogo **Aggiungi tabella(e) PostGIS**. Per accedere al gestore della connessione, cliccare sul tasto **Nuovo** per far comparire la finestra di dialogo **Creare una nuova connessione PostGIS**. I parametri richiesti per la connessione sono mostrati nella tabella 1.

Tabella 1: Parametri di connessione al geodatabase PostGIS



Name	Nome della connessione. Può essere uguale a quello del <i>Database</i> .
Host	Nome del server che ospita il database. Deve essere un host con indirizzo raggiungibile, lo stesso che potrebbe essere usato per aprire una connessione telnet o per pingare l'host. Se il database è sullo stesso computer sul quale è installato QGIS, inserire semplicemente 'localhost'.
Database	Nome del database.
Port	Numero della porta sulla quale il database PostgreSQL è in ascolto. La porta di default è 5432.
Username	Nome dell'utente che accede al database.
Password	Password usata dall' <i>Username</i> per collegarsi al database.

Come opzione, possono essere attivate le seguenti caselle di controllo:

- ☒ Salva password
- ☒ Cercare solamente nella tabella geometry_columns
- ☒ Cerca solamente nello schema 'public'

Quando tutti i parametri sono impostati, la connessione può essere testata cliccando sul pulsante **Prova connessione**.

Tip 8 SETTAGGIO E SICUREZZA DELLE IMPOSTAZIONI UTENTE IN QGIS


Le impostazioni personalizzate di QGIS sono salvate in modo diverso in base al sistema operativo. , le impostazioni sono salvate nella cartella home dell'utente nel file `.qt/qgisrc`. , le impostazioni sono salvate nel registro di sistema. Secondo il sistema operativo, il salvataggio delle password in QGIS può essere più o meno a rischio.

5.2.2 Caricare un layer PostGIS



Una volta definite una o più connessioni, possono essere caricati livelli dal database PostgreSQL. Ovviamente questo richiede avere dati in PostgreSQL. Si veda la Sezione 5.2.4 per informazioni sul come importare dati nel database.

Per caricare livelli da PostGIS, seguire i seguenti passaggi:

- Se la finestra di dialogo **Aggiungi tabella(e) PostGIS** non è già aperta, cliccare nella barra strumenti sul pulsante  **Aggiungi layer PostGIS**.
- Scegliere la connessione dal menù a tendina e cliccare su **Connetti**.
- Individuare il livello che si vuole aggiungere nella lista fra quelli disponibili.
- Selezionarlo cliccando sul nome. È possibile selezionare più livelli tenendo premuto il tasto **shift** mentre si seleziona. Si veda la Sezione 5.5 per informazioni su come usare il costruttore di query PostgreSQL per definire una selezione al momento del caricamento.
- Cliccare sul tasto **Aggiungere** per aggiungere il livello alla mappa.

Tip 9 LIVELLI POSTGIS

Di solito un livello PostGIS è definito da un record nella tabella `geometry_columns`. Dalla versione 0.11.0 in avanti, QGIS può caricare livelli che non hanno tale record nella tabella `geometry_columns`. Ciò vale sia per le tabelle che per le viste. La definizione di una vista spaziale fornisce un mezzo molto potente per visualizzare i dati. Fare riferimento al manuale PostgreSQL per informazioni su come creare le viste.

5.2.3 Alcuni dettagli sui livelli PostgreSQL

Questa sezione contiene alcuni dettagli su come QGIS accede agli strati PostgreSQL. La maggior parte delle volte QGIS dovrebbe semplicemente fornire una lista di tabelle del database che possono essere caricate, e caricarle su richiesta. Tuttavia, se avete difficoltà a caricare una tabella di PostgreSQL in QGIS, le informazioni qui sotto possono aiutare a capire tutti i messaggi di QGIS ed a dare un'indicazione su come cambiare la definizione di tabella o di vista di PostgreSQL per permettere a QGIS di caricarla.

QGIS richiede che gli strati di PostgreSQL contengano una colonna che possa essere usata come chiave unica per lo strato. Per le tabelle questo significa che esse devono contenere una chiave primaria o presentino una colonna con un vincolo unico su essa. Se una tabella manca di questi elementi, verrà usata la colonna `oid`. QGIS richiede inoltre che questa colonna sia di tipo `int4` (un numero intero del formato 4 byte). Le prestazioni saranno migliorate se la colonna è indicizzata (notare che le chiavi primarie sono automaticamente indicizzate in PostgreSQL).

Se lo strato di PostgreSQL è una vista, esistono gli stessi requisiti, ma le viste non hanno chiavi primarie o colonne con i vincoli unici su di loro. In questo caso QGIS proverà a trovare una colonna nella vista che provenga da una colonna della tabella appropriata. Se non ne viene trovata alcuna, QGIS non caricherà il livello. Se questo accade, la soluzione è di alterare la vista in modo che includa una colonna adatta (un tipo di int4 e una chiave primaria o un vincolo unico, spostato e preferibilmente indicizzato).

5.2.4 Importazione di dati in PostgreSQL

shp2pgsql

I dati possono essere importati in PostgreSQL in diverse maniere. PostGIS include un programma di utilità chiamato `shp2pgsql` che può essere usato per importare shapefiles in un database PostGIS. Per esempio, per importare lo shapefile chiamato `lakes.shp` nel database PostgreSQL chiamato `gis_data`, usare il comando seguente:

```
shp2pgsql -s 2964 lakes.shp lakes_new | psql gis_data
```

Questo comando crea un nuovo layer chiamato `lakes_new` nel database `gis_data`. Il nuovo layer avrà uno spatial reference identifier (SRID) di 2964. Si veda la Sezione 8 per ulteriori informazioni sui sistemi di spatial reference systems e le proiezioni.


Tip 10 ESPORTARE DATI DA POSTGIS

Come è presente lo strumento per l'importazione `shp2pgsql` c'è anche lo strumento per l'esportazioni di dati PostGIS come shapefiles: `pgsql2shp`. Esso è incluso nella versione di PostGIS installata.

SPIT Plugin



QGIS include un plugin denominato SPIT (Shapefile to PostGIS Import Tool). SPIT può essere usato per caricare più shapefiles in una volta sola e include il supporto per gli schemi. Per usare SPIT, aprire il Gestore plugin dal menù **Plugins**, selezionare la casella di controllo vicina a **SPIT** e cliccate su **OK**. L'icona di SPIT verrà aggiunta alla barra degli strumenti plugin.

Per importare uno shapefile, cliccare sull'icona  nella barra degli strumenti per aprire la finestra di dialogo **SPIT**. Selezionare il database PostGIS al quale si desidera connettersi e cliccare su **Connetti**. Ora è possibile aggiungere uno o più files alla coda cliccando su **Aggiungi**. Per processare i files selezionati, cliccare su **OK**. L'avanzamento dell'importazione ed eventuali errori/avvertimenti saranno mostrati mentre ciascuno shapefile viene elaborato.

Tip 11 IMPORTARE SHAPEFILES CONTENENTI PAROLE RISERVATE A POSTGRESQL

Se uno shapefile contenente campi che sono parole riservate per il database PostgreSQL viene aggiunto alla coda, comparirà una finestra di dialogo che darà informazioni sullo stato di ogni campo. È necessario editare i nomi dei campi contenenti tali parole (ed è possibile eventualmente editare anche il nome degli altri campi) prima dell'importazione, altrimenti il processo di importazione non andrà a buon fine.

ogr2ogr



Oltre a `shp2pgsql` e `SPIT` c'è un altro strumento per inserire geodati in PostGIS: `ogr2ogr`. Esso fa parte della versione di GDAL installata. Per importare uno shapefile in PostGIS con `ogr2ogr`, digitare questo comando:

```
ogr2ogr -f "PostgreSQL" PG:"dbname=postgis host=myhost.de user=postgres \
password=topsecret" alaska.shp
```

L'espressione importerà lo shapefile `alaska.shp` nel database PostGIS `postgis` usando l'utente `postgres` e la password `topsecret` sull'host *myhost.de*.

Notare che OGR deve essere compilato con il supporto a PostgreSQL per poter effettuare tale operazione. La presenza del supporto a PostgreSQL-PostGIS può essere verificata digitando da riga di comando:

```
ogrinfo --formats | grep -i post
```

Se si volesse usare il comando interno di PostgreSQL `COPY` al posto del metodo predefinito `INSERT INTO` bisogna settare le variabili d'ambiente come segue (su piattaforme  e ):

```
export PG_USE_COPY=YES
```

`ogr2ogr` non crea indici spaziali come `shp2pgsql`. Bisogna crearli manualmente usando il comando SQL `CREATE INDEX` dopo l'importazione come passaggio aggiuntivo (come descritto nella prossima sezione [5.2.5](#)).

5.2.5 Migliorare le prestazioni

Richiamare dati geografici da un database PostgreSQL può richiedere molto tempo, specialmente se il server dei dati si trova in rete. È possibile migliorare le prestazioni di resa a video di strati PostgreSQL assicurandosi di creare un indice spaziale su ogni layer nel database. PostGIS supporta la creazione di un indice GiST (indice dell'albero generalizzato di ricerca, Generalized Search Tree) per velocizzare le ricerche spaziali di dati.

La sintassi per la creazione di un indice GiST è: ³

```
CREATE INDEX [indexname] ON [tablename]
  USING GIST ( [geometryfield] GIST_GEOMETRY_OPS );
```

Si noti che per tabelle molto grandi, la creazione dell'indice può richiedere parecchio tempo. Non appena l'indice è stato creato, bisognerebbe effettuare una `VACUUM ANALYZE`. Si veda la documentazione di PostGIS (4) per ulteriori informazioni.

Ciò che segue è un esempio di come creare un indice GiST:

```
gsherman@madison:~/current$ psql gis_data
Welcome to psql 8.3.0, the PostgreSQL interactive terminal.
```


```
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit
```

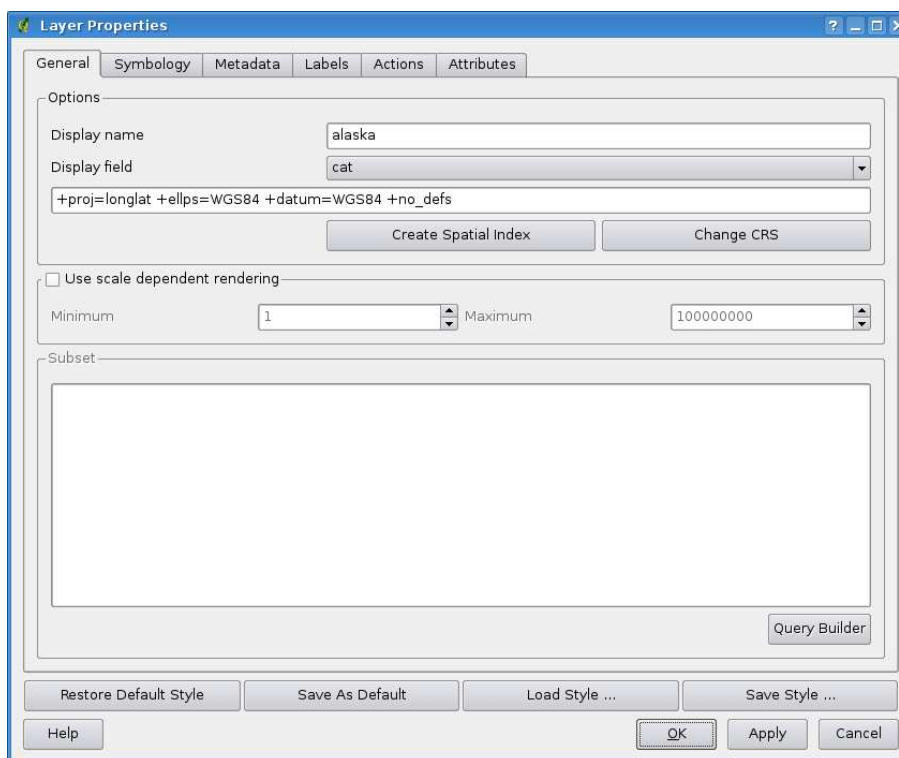
```
gis_data=# CREATE INDEX sidx_alaska_lakes ON alaska_lakes
gis_data=# USING GIST (the_geom GIST_GEOMETRY_OPS);
CREATE INDEX
gis_data=# VACUUM ANALYZE alaska_lakes;
VACUUM
gis_data=# \q
gsherman@madison:~/current$
```

5.3 La finestra delle proprietà dei vettori

La finestra di dialogo **Proprietà del vettoriale** fornisce informazioni sul layer, sulla sua rappresentazione grafica (simbologia) e opzioni per la visualizzazione di etichette sugli elementi che lo compongono. Se il layer vettoriale è stato caricato da un archivio dati PostgreSQL/PostGIS, è possibile modificare anche l'espressione SQL che lo ha generato - sia a mano editando l'espressione SQL nella linguetta **Generale** o richiamando la finestra di dialogo **Costruttore di query** dalla linguetta **Generale**. Per accedere alla finestra di dialogo **Proprietà del vettoriale**, fare doppio click sul layer nella legenda o click con il tasto destro sul layer e selezionare **Proprietà** dal menù contestuale.

³le informazioni sull'indice GiST sono tratte dalla documentazione PostGIS disponibile su <http://postgis.refractory.net>

Figura 6: Finestra delle proprietà di un vettoriale 



5.3.1 Linguetta Generale

La linguetta **Generale** è sostanzialmente simile a quella dei raster. Essa consente di cambiare il nome del file mostrato, impostare la visualizzazione in base alla scala, creare un indice spaziale (solo per i formati supportati da OGR e per livelli PostGIS) e vedere o cambiare la proiezione dello specifico layer vettoriale.

Il pulsante **Creatore di query** consente di selezionare un sottoinsieme di elementi nel layer - ma questo pulsante è attualmente disponibile unicamente quando viene aperta la tabella attributi cliccando sul pulsante **Avanzate ...**.

5.3.2 Linguetta simbologia

QGIS supporta diverse modalità di rappresentazione per controllare come gli elementi del vettoriale sono mostrati. Attualmente sono disponibili le seguenti modalità:


Simbolo singolo - lo stesso stile è applicato a tutti gli elementi del vettore

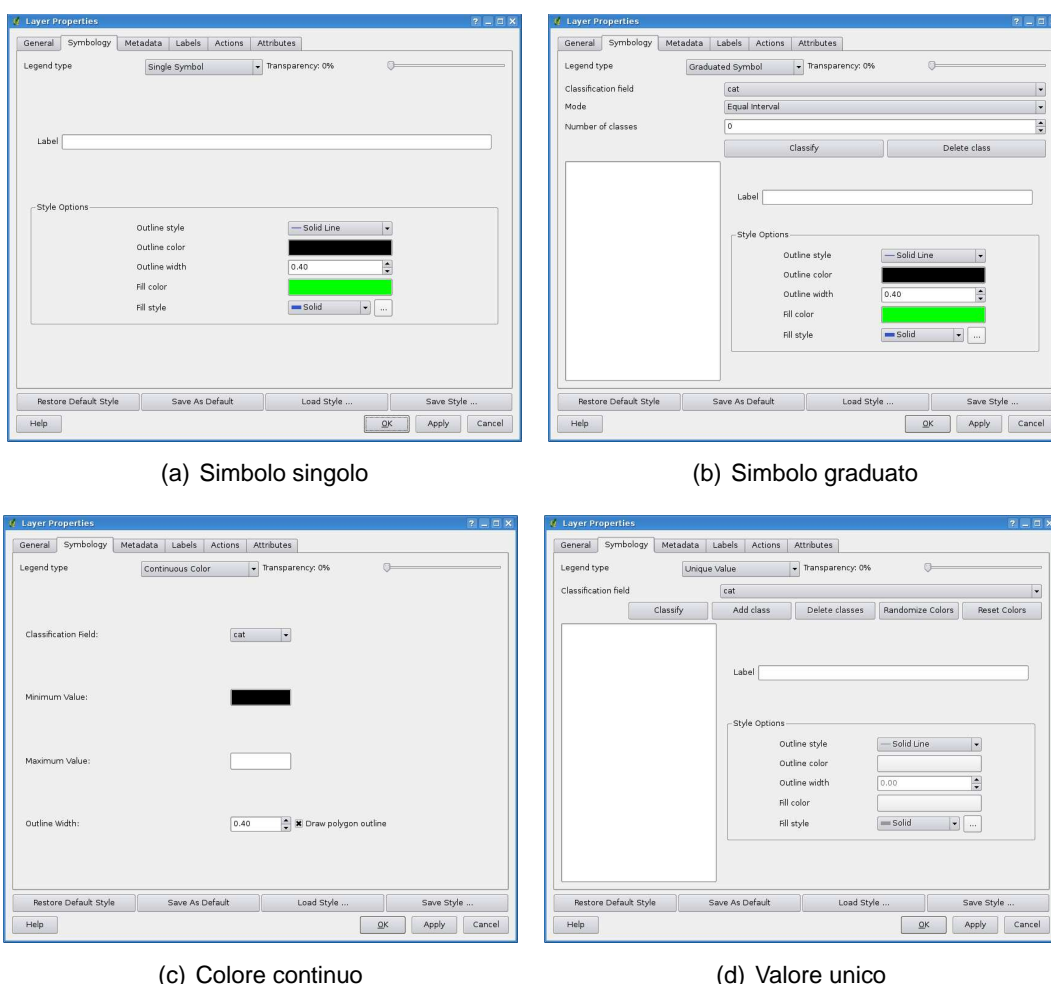
Simbolo graduato - lo stile applicato ai diversi elementi dipende dal valore di un campo particolare nella tabella associata.

Colore continuo - gli elementi del layer sono mostrati con una gradazione di colori compresa entro due estremi specificati in base ai valori numerici di uno specifico campo.

Valore univoco - gli oggetti sono classificati in base ai valori unici di uno specifico campo, ad ogni valore viene assegnata una simbologia differente.

Per modificare la simbologia di un layer, fare semplicemente doppio click sulla relativa voce di legenda per fare apparire la finestra di dialogo **Proprietà del vettoriale**.

Figura 7: Opzioni per la simbologia dei layer 



Opzioni per lo stile

In questa finestra di dialogo è possibile scegliere lo stile di rappresentazione del layer vettoriale. Secondo l'opzione di visualizzazione scelta tra quelle descritte precedentemente si ha la possibilità

di classificare anche gli elementi della mappa.



Le seguenti opzioni dovrebbero essere disponibili per pressoché tutti le simbologie:

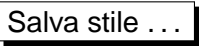

Stile esterno - tipo di tratteggio del contorno degli elementi. Si può anche impostare l'opzione 'Nessuna penna' per escludere la rappresentazione del contorno.

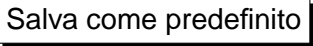

Colore esterno - colore del contorno degli elementi

Dimensione stile esterno - larghezza della linea di contorno

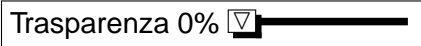
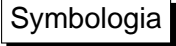
Colore di riempimento - colore di riempimento degli elementi.

Stile di riempimento - stile per il riempimento. Oltre ai retini forniti è possibile scegliere  e cliccare su  per selezionare un retino personalizzato. Attualmente sono supportati i formati *.jpeg, *.xpm, and *.png.

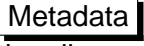
Una volta impostato lo stile del layer esso può essere salvato in un file separato (con estensione *.qml). Per fare ciò, cliccare sul pulsante . Ovviamente il pulsante  carica un file di stile precedentemente salvato.

Se si desidera usare sempre un particolare stile quando il layer viene caricato, cliccare su  per rendere predefinito lo stile impostato. Inoltre, se si effettuano allo stile modifiche delle quali non si è soddisfatti, cliccare su  per ritornare allo stile predefinito precedentemente impostato.


Applicare la trasparenza ad un vettore

QGIS 1.0.0 fornisce la possibilità di settare la trasparenza per ogni layer vettoriale. Ciò può essere fatto settando l'apposita barra  nella linguetta  (si veda la fig. 6). Ciò può essere molto utile per la visualizzazione di più layer vettoriali sovrapposti.

5.3.3 Linguetta Metadata

La linguetta  contiene informazioni relative allo strato quali il formato, la posizione, il numero ed il tipo di geometrie e le possibilità di editing. Nella sezione Sistema di riferimento spaziale del layer è visualizzata la proiezione associata al dato, mentre quella Informazioni campo attributi fornisce il nome e il formato dei campi della tabella associata. Questo è un metodo veloce per ottenere informazioni sul layer.

5.3.4 Linguetta Etichette

La linguetta  consente di abilitare la visualizzazione delle etichette associate agli elementi dello strato e controlla una serie di opzioni legate al posizionamento, allo stile ed ad altre

caratteristiche delle etichette..

Come esempio apporremo le etichette allo shapefile lakes del dataset_esempio_di_qgis:

1. Caricare lo shapefile `alaska.shp` e il file GML `lakes.gml` in QGIS.
2. Zoommare su un'area a scelta contenente alcuni laghi.
3. Rendere attivo il layer `lakes` cliccando su di esso nella legenda.
4. Aprire la finestra di dialogo **Proprietà del vettoriale**.
5. Cliccare sulla linguetta **Etichette**.
6. Selezionare la casella di controllo ☒ **Mostra etichette** per abilitarne la visualizzazione.
7. Scegliere il campo della tabella contenente le etichette da apporre. In questo esempio si userà il **Campo contenente etichetta** **NAMES**.
8. Inserire una etichetta di default per gli elementi del layer lakes che non hanno nome. Questa etichetta verrà quindi usata ogni volta che QGIS dovrà etichettare un lago al quale non corrisponda un valore nel campo NAMES.
9. Cliccare su **Apply**.

Adesso sono visualizzate le etichette. Il loro aspetto non è probabilmente gradevole, potrebbero essere troppo grandi e posizionate male in relazione al simbolo dei laghi.

Cliccare allora su **Carattere** e **Colore** per impostare il tipo di carattere e il colore. È possibile anche cambiare l'angolo e la posizione delle etichette testuali.

Per modificare la posizione relativa del testo rispetto agli elementi:

1. Cliccare sulla linguetta **Etichette**.
2. Cambiare la posizione selezionando una delle opzioni disponibili nel gruppo **Posizionamento**. Nel caso preso in esame, scegliere l'opzione **Destra**.
3. la voce **Unità della dimensione del carattere** consente di selezionare tra **Punti** o **Unità mappa**.
4. Cliccare su **Apply** per visualizzare i cambiamenti senza chiudere la finestra di dialogo.

Ora l'aspetto sarà migliore, ma le etichette appaiono ancora troppo vicine all'indicatore della loro posizione. Per sistemare il problema è possibile utilizzare l'opzione della voce **Posizione**. Può qui essere impostato uno scostamento nelle direzioni X e Y. Aggiungendo uno spostamento in X pari a 5 le etichette verranno scostate dall'indicatore della loro posizione e renderle più leggibili. Ovviamente più è grande l'indicatore o il carattere, maggiore sarà lo scostamento da applicare.

Aggiungiamo infine un buffer sulle etichette cliccando sulla voce **buffer**. In questo modo verrà aggiunto uno sfondo attorno alle lettere per farle risaltare maggiormente. Per mettere un buffer alle etichette dei laghi procedere come di seguito:

1. Cliccare sulla voce **Buffer**.
2. Abilitare la casella di controllo ☒ **Buffer sulle etichette?**.
3. Scegliere una dimensione (spessore) del buffer.
4. Scegliere un colore per il buffer cliccando sul pulsante **Colore**. È inoltre possibile assegnare una trasparenza in percentuale al buffer.
5. Cliccare su **Apply** per vedere giudicare i cambiamenti.

Modificare eventualmente i cambiamenti fino a quando non si è soddisfatti del risultato, cliccando su **Apply** dopo ogni modifica.

In genere un buffer di 1 punto fornisce risultati esteticamente gradevoli. Si noti che è anche possibile specificare la dimensione del buffer in unità della mappa se ciò rende più agevole l'impostazione.

Le rimanenti voci della linguetta **Etichette** consentono di controllare l'aspetto delle etichette usando, se adeguatamente preparati, gli attributi del layer. Le voci che iniziano con **Definizione** consentono di settare tutti i parametri delle etichette facendo riferimento a campi della tabella del livello.

Si noti che la linguetta **Etichette** fornisce una **Anteprima** nella quale viene mostrata l'etichetta predefinita.

5.3.5 Linguetta Azioni

QGIS offre la possibilità di effettuare azioni sulla base degli attributi associati ai singoli elementi dello strato vettoriale. Questo permette di effettuare un elevato numero di azioni, per esempio, lanciare un programma con argomenti costruiti tramite gli attributi delle geometrie o passando i parametri ad uno strumento di web reporting.


Definire delle azioni è utile quando si voglia lanciare un'applicazione esterna o la visualizzazione di una pagina web sulla base di uno o più valori associati allo strato vettoriale. Ad esempio si può lanciare una ricerca web basata sul valore di un attributo. Questo concetto è spiegato nel seguente paragrafo.

Definire le azioni

Le azioni legate agli attributi sono definite dalla finestra di dialogo **Proprietà del vettoriale**. Per impostare un'azione, aprire la finestra di dialogo **Proprietà del vettoriale** e cliccare sulla linguetta **Azioni**. Fornire una descrizione per l'azione nel campo Nome. L'azione in sé deve contenere il nome o il percorso di una applicazione che verrà eseguita quando l'azione viene richiamata. L'azione può venire fatta dipendere da uno o più campi della tabella attributi. Quando essa è richiamata ogni stringa testuale che inizia con % seguita dal nome di un campo della tabella attributi verrà rimpiazzata dal valore di quel campo. I caratteri speciali %% saranno rimpiazzati dal valore del campo

nell'elemento selezionato con lo strumento **Identifica geometrie** disponibile nella barra strumenti o dalla tabella attributi (si veda il seguente Usare le azioni). Le virgolette () possono essere usate per raggruppare il testo in un singolo argomento da passare al programma, allo script o al comando che si intende eseguire. Le virgolette saranno ignorate se precedute dalla barra inversa.

Se sono presenti nomi di campo che possono essere interpretati come sottostringhe di altri nomi di campo (ad es. `col1` e `col10`) è necessario indicare, racchiudendo il nome di campo (e il carattere %) tra parentesi quadre (ad es. `[%col10]`). Ciò impedirà che il nome di campo `%col10` possa essere confuso con `%col1` con uno 0 alla fine. Le virgolette saranno rimosse da QGIS man mano che vengono inseriti i valori del campo al posto dell'espressione. Se si vuole che i campi sostituiti vengano racchiusi entro parentesi quadre, aggiungere una seconda coppia di parentesi quadre in questo modo: `[[%col10]]`.

La finestra di dialogo **Risultati individuati** che compare quando si usa lo strumento **Identifica geometrie** ha una voce (*Derivato*) che contiene informazioni che sono dipendenti dal tipo di layer interrogato. Si può accedere ai valori di questa voce similmente a come possibile per i valori di campo della tabella attributi anteposendo al nome di campo disponibile alla voce (*Derivato*) l'espressione (*Derivato*).. Per esempio un layer puntuale ha due sottovoci X e Y e il valore di essi può essere usato nell'azione con l'espressione `%(Derivato).X` e `%(Derivato).Y`. Gli attributi derivati sono disponibili solo nella finestra **Risultati individuati** aperta dallo strumento  **Identifica geometrie** e non nella finestra **Tabella attributo**.

Due esempi di azioni sono di seguito indicati:

- `konqueror http://www.google.com/search?q=%nam`
- `konqueror http://www.google.com/search?q=%%`

Nel primo esempio, il browser konqueror viene richiamato con un URL da aprire. L'URL crea una ricerca Google sul valore del campo `nam` nel layer vettoriale. Si noti che il programma o lo script richiamato dall'azione deve essere nel path impostato come variabile d'ambiente oppure bisogna fornire il percorso completo all'eseguibile. Per sicurezza, è possibile riscrivere il primo esempio come: `/opt/kde3/bin/konqueror http://www.google.com/search?q=%nam`. In questo modo si sarà sicuri che l'applicazione konqueror sarà eseguita quando si richiama l'azione.



Nel secondo esempio viene usata la notazione `%%` che non richiede l'indicazione di un particolare campo. Quando si richiama l'azione, il `%%` sarà rimpiazzato dal valore selezionato con lo strumento




Identifica geometrie o nella tabella attributi.

Uso delle azioni

Le azioni possono essere richiamate sia dalla finestra **Risultati individuati** che da quella della **Tabella attributo**. (Si ricorda che queste finestre possono essere aperte rispettivamente cliccando

sullo strumento  **Identifica geometrie** o  **Apri tabella attributi**.) Per eseguire l'azione, fare click con il tasto destro sul record e scegliere azione dal menù a comparsa. Le azioni sono indicate nel menù a comparsa dal nome assegnatogli in fase di definizione dell'azione. Cliccare sull'azione che si vuole eseguire.

Se si esegue un'azione che usa la notazione `%%`, cliccare con il tasto destro sul valore di campo che si desidera passare all'azione nella finestra di dialogo **Risultati individuati** o in quella **Tabella attributo**.

In questo altro esempio viene illustrato come estrarre dati da un layer vettoriale per inserirli in un file usando la shell di sistema `bash` e il comando `echo` (dunque funzionerà solo su  e forse su **X**). Il layer in questione ha i seguenti campi nella tabella attributi: nome della specie `taxon_name`, latitudine `lat` e longitudine `long`. Si vorrebbe eseguire una selezione spaziale delle specie (`taxon`) presenti a determinate posizioni esportando i risultati in un file di testo per le posizioni selezionate (evidenziate di default in giallo nella vista mappa di QGIS). L'azione in grado di assolvere lo scopo è la seguente:

```
bash -c "echo \"%%taxon_name %lat %long\" >> /tmp/species_localities.txt"
```

Selezionando alcune posizioni, l'esecuzione dell'azione precedente su ognuna di esse genera un file in uscita che avrà l'aspetto seguente:

```
Acacia mearnsii -34.0800000000 150.0800000000
Acacia mearnsii -34.9000000000 150.1200000000
Acacia mearnsii -35.2200000000 149.9300000000
Acacia mearnsii -32.2700000000 150.4100000000
```

Come esercizio si può creare un'azione che generi una ricerca su Google sul layer `lakes`. Innanzitutto è necessario determinare la sintassi da impiegare nell'URL per eseguire una ricerca basata su una parola chiave. L'espressione si ricava facilmente eseguendo una ricerca dalla pagina di Google, la pagina dei risultati avrà un indirizzo, visibile nella barra indirizzi del browser, del tipo: <http://google.com/search?q=qgis>, in cui `qgis` è la parola ricercata. Forniti di questa informazione, si può procedere nel seguente modo:

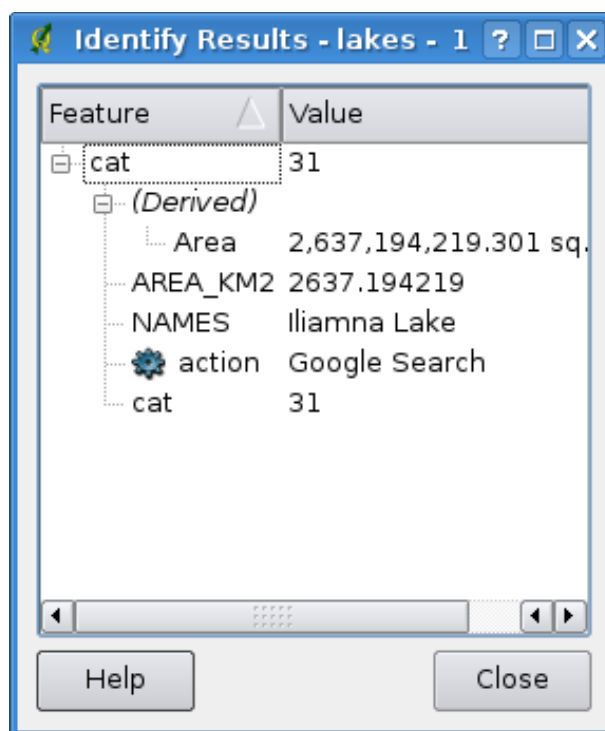
1. Assicurarsi che il livello `lakes` sia caricato.
2. Aprire la finestra di dialogo **Proprietà del vettoriale** facendo doppio click sul layer o cliccando su di esso nella legenda con il tasto destro e scegliendo **Proprietà** dal menù contestuale.
3. Cliccare sulla linguetta **Azioni**.
4. Inserire un nome descrittivo per l'azione, ad esempio `Ricerca su Google`.

5. Fornire il nome di un programma esterno da eseguire nell'azione. In questo caso useremo il browser Firefox. Se il programma non si trova in uno dei percorsi di sistema definiti dalla variabile d'ambiente PATH, bisogna specificare il percorso completo all'eseguibile.
6. Far seguire il nome del programma esterno dall'URL usato per la ricerca su Google senza includere la parola ricercata, ovvero: `http://google.com/search?q=`
7. A questo punto il testo nel campo Azioni dovrebbe apparire così:
firefox `http://google.com/search?q=`
8. Cliccare sul menù a tendina contenente i nomi dei campi della tabella associata al layer `lakes`, posizionato immediatamente a sinistra del pulsante **Inserisci campo**.
9. Dall'elenco apparso scegliere **NAMES ▼** e cliccare su **Inserisci campo**.
10. Il testo dell'azione dovrebbe ora apparire come segue:
firefox `http://google.com/search?q=%NAMES`
11. Per completare l'azione cliccare sul pulsante **Inserisci azione**.

Questo completa la definizione dell'azione che è così pronta per essere usata. La formulazione finale dell'azione dovrebbe apparire così:

```
firefox http://google.com/search?q=%NAMES
```

A questo punto l'azione è pronta per essere usata. Chiudere la finestra **Proprietà del vettoriale** e zoommare su un'area a scelta. Assicurarsi che il livello `lakes` sia attivo ed identifichiamo con l'apposito strumento un lago. Nella finestra risultante dovrebbe essere visibile l'azione:

Figura 8: Selezione di un elemento e scelta dell'azione 

Quando si clicca sull'azione, viene lasciato Firefox all'URL <http://www.google.com/search?q=Tustumena>. È anche possibile aggiungere ulteriori campi all'azione, inserendo un "+" alla fine della stringa che definisce l'azione, selezionando quindi un altro campo e cliccando sul pulsante **Inserisci campo**. Nell'esempio seguito finora semplicemente non c'è alcun altro campo sul quale avrebbe senso fare una ricerca.

È possibile definire più di un'azione per ogni layer, ognuna di esse verrà mostrata nella finestra **Risultati individuati**. Si possono anche eseguire azioni dalla tabella attributi cliccando con il tasto destro su una riga selezionata e scegliendo dal menù contestuale l'azione desiderata.

Si possono immaginare molti tipi di azione. Ad esempio se un layer di punti rappresenta le posizioni alle quali sono state scattate foto o alle quali corrispondono immagini e il nome dei file di tali foto o immagini, è possibile creare un'azione per lanciare un visualizzatore che mostri l'immagine. Le azioni possono essere usate anche per lanciare report sul web per uno o più campi della tabella attributo, definendole allo stesso modo definito nell'esempio per la ricerca con Google.

5.3.6 La linguetta Attributi

Nella linguetta **Attributi** è possibile modificare lo schema degli attributi del livello selezionato. I pulsanti **Nuova colonna** ed **Elimina colonna** vengono attivati quando il layer è in modalità editing, ovvero quando viene premuto il pulsante **Alterna la modalità di editing**. Al momento possono essere editate solo colonne appartenenti a layers PostGIS, in quanto questa funzione non è ancora supportata dalla libreria OGR.

modifica widget

Nella linguetta **Attributi** sono presenti anche le colonne `modifica widget` e `valori`. Esse possono essere usate per definire un valore o un campo di valori che possono essere aggiunti alla specifica colonna della tabella attributi. Sono usati per produrre diversi widgets di modifica nella finestra degli attributi. Questi widgets sono:

- **modifica linea**: un campo di modifica che consente di inserire un linea di testo semplice (o esclusivamente numeri se l'attributo è di tipo numerico).
- **valori univoci**: una lista di singoli valori da assegnare all'attributo proveniente da elementi preesistenti viene prodotta e presentata in una combo box per la selezione.
- **valori univoci (modificabile)**: una combinazione di 'modifica linea' e 'valori univoci'. Il campo modifica consente di modificare i valori preesistenti o di inserirne di nuovi.
- **valore mappa**: una combobox nella quale è possibile scegliere da un insieme di valori specificati nella colonna `valori` della linguetta **Attributi**. I valori possibili sono delimitati da una semicolon (ad es. `high;medium;low`). È anche possibile preporre un'etichetta ad ogni valore, la quale sarà delimitata dal segno dell'uguale (ad es. `high=1;medium=2;low=3`). L'etichetta è mostrata nella combobox al posto del valore.
- **classificazione**: if a unique value renderer is selected for the layer, the values used for the classes are presented for selection in a combobox.
- **intervallo (modificabile)**: un campo di modifica che consente di restringere i valori numerici entro un certo intervallo. L'intervallo viene specificato inserendo i valori massimi e minimi delimitati da semicolon (ad es. `0;360`) nella colonna `valori` della linguetta **Attributi**.
- **intervallo (cursore)**: un cursore scorrevole che consente la selezione di un valore entro un predefinito intervallo e con un certo passo. L'intervallo è definito da un valore minimo, massimo e dall'ampiezza del passo (ad es. `0;360;10`) nella colonna `valori` della linguetta **Attributi**.
- **nome file**: la linea di editazione del widget presenta anche un pulsante. Premendo il pulsante si può caricare un file usando la finestra di dialogo per l'esplorazione delle risorse.

5.4 Editing

QGIS fornisce capacità basilari di editazione delle geometrie vettoriali. Prima di procedere si evidenzia che a questo stadio il supporto per l'editing è ancora preliminare. Prima di eseguire una qualunque operazione di editing, quindi, fare sempre una copia di backup dei dati che si intendono modificare.

Nota - la procedura per la modifica di livelli GRASS è differente, si veda la Sezione 9.7 per dettagli.

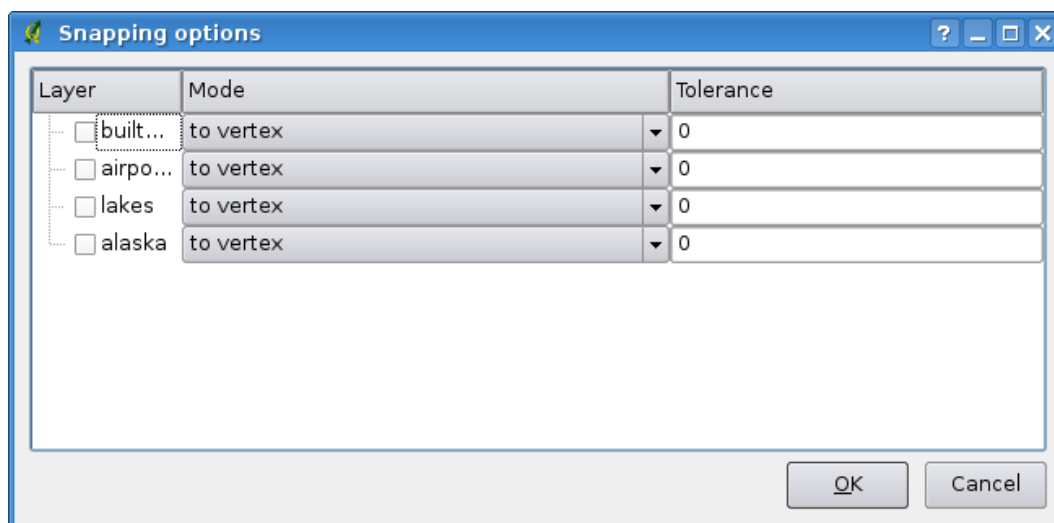

5.4.1 Settare la tolleranza dello snapping e il raggio di ricerca degli elementi

Prima di editare vertici, è molto importante sia impostare il livello di snapping che il valore del raggio di ricerca che ci consentano una modifica ottimale delle geometrie del layer vettoriale.

Tolleranza sullo snapping

La tolleranza sullo snapping è la distanza entro la quale QGIS cerca il più vicino vertice e/o segmento al quale si cerca di agganciarsi quando si crea un nuovo vertice o si sposta un vertice esistente. Se non si è entro la tolleranza di snapping, QGIS lascerà il vertice creato o spostato nella posizione in cui si rilascia il pulsante del mouse invece di agganciarlo ad un vertice e/o segmento esistente.

1. Una prima impostazione della tolleranza sullo snapping può essere impostata a livello dell'intero progetto scegliendo la voce di menù **Impostazioni** > **Opzioni**. Alla linguetta **Digitalizzazione** è possibile impostare la modalità di snap predefinita tra snap al vertice, al segmento o entrambe. Si può anche definire una tolleranza sullo snap e un raggio di ricerca per la modifica di un vertice, in unità del layer. Nel set di dati Alaska, le unità sono in piedi (feet). L'impostazione ottimale può variare, ma in genere 300 piedi può essere una impostazione ragionevole per lavorare alla scala 1:10.000.
2. È anche possibile impostare una tolleranza sullo snapping basata sul singolo layer scegliendo la voce di menù **Impostazioni** > **Proprietà progetto...**. Nella linguetta **Generale**, alla sezione **Digitalizzazione** si può cliccare sul pulsante **Opzioni di snap...** per abilitare e impostare la tolleranza per ogni singolo layer del progetto (si veda la Figura 9).

Figura 9: Modifica delle opzioni di snapping per singoli layers 

Raggio di ricerca


Il raggio di ricerca è la distanza che QGIS usa per cercare il più vicino vertice che si sta cercando di spostare quando si clicca nella mappa. Se non si è entro il raggio di ricerca, QGIS mostrerà un avvertimento in merito in una finestra pop-up. La tolleranza sullo snap e il raggio di ricerca sono impostati in unità di mappa e potrebbe essere necessario fare diversi tentativi per trovare l'impostazione migliore. Se si specifica una tolleranza troppo alta, QGIS potrebbe agganciare il vertice sbagliato, specialmente se si ha a che fare con molti vertici vicini all'area in cui si sta effettuando la modifica. Impostando invece un raggio di ricerca troppo piccolo impedirà invece a QGIS di trovare alcuna geometria da spostare.

Il raggio di ricerca per la modifica di vertici può essere definita in unità del layer dalla linguetta **Digitalizzazione** sotto il menù **Impostazioni** -> **Opzioni**, sotto lo stesso percorso dal quale è possibile impostare la tolleranza sullo snapping a livello di progetto.


5.4.2 Editing topologico

Oltre alle opzioni di snap a livello di singolo layer la linguetta **Generale** sotto la voce di menù **Impostazioni** -> **Proprietà progetto...** fornisce anche alcune funzionalità per la topologia. Alla voce Digitalizzazione è possibile selezionare l'opzione ☒ **Abilita la modifica topologica** e/o attivare l'opzione ☒ **Vieta le intersezioni per i nuovi poligoni**.

Abilitare la modifica topologica

L'opzione  **Abilita la modifica topologica** serve a mantenere limiti comuni tra poligoni adiacenti durante l'editazione. QGIS individua un limite condiviso in un insieme di poligoni e tutto ciò che si deve fare è spostare il vertice una volta sola, QGIS si occuperà di aggiornare anche il limite del poligono adiacente.

Impedire le intersezioni per i nuovi poligoni

La seconda opzione  **Vieta intersezioni per i nuovi poligoni** impedisce la sovrapposizione di poligoni adiacenti, rendendone più spedita la digitalizzazione. Se si ha già un poligono, è possibile con questa opzione abilitata digitalizzare il secondo in modo che entrambi si intersechino e qgis taglierà automaticamente il secondo sul limite comune, con il vantaggio che l'utente non deve digitalizzare tutti i vertici del limite comune.

5.4.3 Modifica di un layer esistente

Di default i layer sono caricati in QGIS in modalità sola lettura al fine di evitare modifiche involontarie. È comunque in ogni caso modificare qualunque layer se è consentito dalla libreria (ovvero se ad es. OGR supporta lo specifico formato in lettura/scrittura) e se il dato medesimo è anche scrivibile (ovvero i files non sono in modalità sola lettura).

Le funzioni di modifica di layer sono più versatili quando sono applicate a dati immagazzinati in database PostgreSQL/PostGIS.


Tip 12 INTEGRITÀ DEL DATO

È buona norma fare un back-up del dato originale prima di procedere alla modifica. Per quando siano stati fatti molti sforzi da parte dei programmatori di QGIS per preservare l'integrità del dato, non vi è alcuna garanzia che ciò avvenga.

Tip 13 MODIFICA DI ATTRIBUTI

Attualmente solo ai layers PostGIS possono essere aggiunte o eliminate colonne attributi da questa finestra. Nelle versioni di QGIS future saranno supportate altre fonti di dati, in quanto questa caratteristica è stata recentemente introdotta nelle librerie GDAL/OGR > 1.6.0

Tip 14 SALVATAGGIO AD INTERVALLI REGOLARI



Si ricordi di cambiare lo stato dello strumento  **Abilita/disabilita modifica** ad intervalli regolari, in modo da consentire il salvataggio delle modifiche recenti e per verificare che le stesse sono accettate dalla sorgente di dati.

Tip 15 MODIFICHE CONCORRENTI

Questa versione di QGIS non effettua alcuna verifica sulla possibilità che più utenti stiano effettuando contemporaneamente modifiche sullo stesso strato, è quindi l'ultimo utente che effettua il salvataggio ad apportare le modifiche definitive.

Tip 16 INGRANDIRE PRIMA DELLA MODIFICA

Prima di modificare uno strato, bisognerebbe effettuare un ingrandimento all'area di interesse, in modo da evitare che in modalità editing tutti i marker dei vertici visibili vengano resi a video per l'intero layer.

Ogni sessione di modifica può essere inizializzata dall'opzione  **Attiva/disattiva modifica**, che può essere attivata e disattivata cliccando con il tasto destro sul nome del layer nella legenda e scegliendo dal menù contestuale oppure dalla barra strumenti cliccando sul pulsante  **Attiva/disattiva modifica**. Quando il layer è in modalità modifica, i vertici verranno contrassegnati dai markers impostati (croci o cerchi semitrasparenti) e ulteriori pulsanti verranno resi disponibili nella barra degli strumenti modifica.

Zoommare con la rotella del mouse





Mentre si digitalizza è possibile usare la rotella del mouse per ingrandire e ridurre la mappa, posizionando il cursore nell'area di mappa e girando la rotellina verso di sé per ridurre e verso lo schermo per ingrandire. La posizione del puntatore del mouse determinerà centro dell'area da ingrandire. È possibile personalizzare il comportamento della rotella del mouse nella linguetta **Strumenti mappa** alla voce di menù **Impostazioni** > **Opzioni**.

Spostare la vista mappa con i tasti freccia


È possibile spostare la vista mappa mentre si digitalizza aiutandosi con i tasti freccia, posizionando il puntatore del mouse nella vista e cliccando sulla freccia destra per spostarsi verso est, sinistra per spostarsi verso ovest, su per spostarsi verso nord e giù per andare verso sud.











È anche possibile tenere premuta la barra spaziatrice mentre si sposta il mouse per spostare la vista mappa e usare i tasti PgUp e PgDown per aumentare o ridurre l'ingrandimento senza interrompere la sessione di digitalizzazione.

Sono disponibili le seguenti funzioni di modifica:




- Aggiunta elementi:  **Inserisci punti**,  **Inserisci linee** and  **Inserisci poligoni**
-  **Inserisci anello**




Tip 17 MARKER DEI VERTICI

La versione di QGIS attuale supporta due tipi di marker per i vertici (in modalità modifica): un cerchio semitrasparente o una croce. Per cambiare lo stile del marker, scegliere la voce  **Opzioni** dal menù **Impostazioni**, cliccare sulla linguetta **Digitalizzazione** e selezionare la voce appropriata.

-  **Inserisci isola**
-  **Dividi geometria**
-  **Sposta geometria**
-  **Sposta vertice**
-  **Aggiungi vertice**
-  **Elimina vertice**
-  **Elimina selezione**
-  **Taglia geometrie**
-  **Copia geometrie**
-  **Incolla geometrie**

Aggiungere elementi

Prima di iniziare ad aggiungere elementi ad un layer, usare gli strumenti  **Sposta mappa**  **Ingrandisci**  **Rimpicciolisci** per visualizzare l'area di mappa nella quale si desidera apporre modifiche.

A questo punto si possono usare gli strumenti  **Inserisci punti**,  **Inserisci linee** o  **Inserisci poligoni** per porre il puntatore di QGIS in modalità digitalizzazione.

Per ogni elemento, bisogna dapprima digitalizzarne la geometria e successivamente inserirne gli attributi.

Per digitalizzare la geometria, cliccare con il tasto sinistro sull'area di mappa per creare il primo punto del nuovo elemento.

Per linee e poligoni, continuare a cliccare con il tasto sinistro per ogni ulteriore vertice che si desidera inserire. Quando è terminato l'inserimento dei vertici o dei punti, cliccare con il tasto destro in qualunque punto della mappa per confermare di aver terminato l'inserimento della geometria dell'elemento.

Apparirà quindi la finestra degli attributi che consentirà di inserire le informazioni per l'elemento appena creato. La Figura 10 mostra l'inserimento degli attributi per un fiume fittizio creato in Alaska.

Figura 10: Finestra di inserimento degli attributi per un elemento di nuova digitalizzazione 🐧

Tip 18 TIPOLOGIE DI ATTRIBUTO

Almeno per gli shapefiles la modifica del tipo di attributo è validata durante l'inserimento. A causa di ciò, non è ovviamente possibile inserire un numero in una colonna testuale quando compare la finestra

Inserire i valori dell'attributo o viceversa. Se ciò fosse necessario, si dovrebbero modificare gli attributi in un secondo momento per mezzo della finestra **Tabella attributo**.

Spostare elementi

Si possono spostare elementi scegliendo dalla barra lo strumento



Sposta geometria.

Dividere elementi

Si possono dividere elementi scegliendo dalla barra lo strumento



Dividi geometria.

Modificare i vertici di un elemento

Sia per i layer PostgreSQL/PostGIS che per gli shapefiles, i vertici di un elemento possono essere modificati.

I vertici possono essere modificati direttamente senza che si debba selezionare l'elemento del quale si voglia modificare la geometria. In alcuni casi, più elementi possono condividere dei vertici, in questi casi valgono le seguenti regole quando si clicca con il mouse vicino ad un elemento del layer:

- **Linee** - La linea più vicina al cursore è scelta come elemento da modificare. Di conseguenza (per spostare e cancellare vertici) viene selezionato per la modifica il vertice più vicino su quella linea.
- **Polygoni** - Se il mouse è all'interno di un poligono, esso è l'elemento da modificare; altrimenti viene usato il più vicino poligono. Di conseguenza (per spostare e cancellare vertici) viene selezionato per la modifica il vertice più vicino su quel poligono.

Ricordarsi di impostare l'opzione sotto **Impostazioni** > **Opzioni** > **Digitalizzazione** alla casella Raggio di ricerca per la modifica di un vertice in unità layer ad un valore superiore a zero, altrimenti QGIS non sarà in grado di determinare quale elemento deve essere modificato.

Aggiungere vertici ad un elemento

Si possono aggiungere ulteriori vertici ad un elemento del layer scegliendo dalla barra lo strumento



Aggiungi vertice.

Si noti che è privo di senso aggiungere ulteriori vertici ad un elemento puntuale!

In questa versione di QGIS i vertici possono essere aggiunti solo ad un segmento di elementi lineari *esistenti*. Se si desidera estendere una linea oltre le sue estremità, è necessario spostare dapprima il vertice finale e quindi aggiungere un nuovo vertice nel punto in cui questo si trovava.


Spostare i vertici di un elemento

Si possono spostare vertici scegliendo dalla barra lo strumento



Sposta vertice.

Cancellare vertici di un elemento


Si possono cancellare vertici scegliendo dalla barra lo strumento  **Elimina vertice**.

Si noti che è privo di senso cancellare un vertice da un elemento puntuale, poiché questo equivarrebbe a cancellare l'elemento!

Similmente una linea costituita da un solo vertice o un poligono costituito da due vertici non hanno molto senso e potrebbero avere comportamenti imprevisti nell'uso di altre funzioni di QGIS (ad es. con gli strumenti di analisi) e di conseguenza bisogna evitare modifiche che portino a creare tali geometrie.


Attenzione: Un vertice è contrassegnato per l'eliminazione non appena si clicca con il mouse vicino ad un elemento selezionabile. Per annullare, sarà necessario disabilitare la modalità di modifica senza salvare i cambiamenti. (Ovviamente questo comporterà anche la perdita delle altre modifiche non salvate.)

Inserisci anello (buco)

So possono digitalizzare nuovi poligoni all'interno di poligoni esistenti al fine di creare un buco all'interno di questi ultimi scegliendo dalla barra lo strumento  **Inserisci anello**. In questo modo solo l'area compresa tra i margini del poligono interno e di quello esterno verrà evidenziata come un poligono ad anello.


Inserisci isola, ovvero creazione/modifica di un multipoligono

Si possono creare e modificare multipoligoni (insieme di elementi poligonali geometricamente indipendenti ovvero non condividenti alcun vertice o limite visti come entità unica quando si seleziona anche uno solo dei singoli elementi) aggiungendo ulteriori isole ad un singolo poligono o ad un mul-

tipoligono esistenti scegliendo dalla barra lo strumento  **Inserisci isola**. La nuova isola del multipoligono deve essere esterna al poligono che si intende rendere multipoligono o agli elementi già facenti parte di un multipoligono esistente che si intende modificare.

Tagliare, copiare ed incollare elementi

Gli elementi selezionati possono essere tagliati, copiati ed incollati tra strati dello stesso progetto di QGIS a patto che anche nello strato di destinazione sia stata abilitata la modalità di editing tramite




l'opzione  **Attiva/disattiva modifica**.

Gli elementi possono essere anche incollati in applicazioni esterne come testi: gli elementi verranno rappresentati nel formato CSV con le informazioni della geometria espresse nel formato testo OGC Well-Known Text (WKT).

Tuttavia in questa versione di QGIS elementi di testo formattato creati con applicazioni esterne non possono essere incollati in un livello dentro QGIS.

Le funzioni di copia/incolla possono essere utili quando si vogliono modificare più layer copiando le modifiche effettuate in uno di questi negli altri. Supponendo ad esempio di voler lavorare su un layer contenente un paio di laghi, diventa molto più agevole creare un nuovo layer vuoto nel quale incollare gli elementi dei quali necessitiamo invece di lavorare sul livello `big_lakes` contenente 5000 elementi.

Dovremo quindi effettuare le seguenti operazioni:


1. Caricare il layer dal quale vogliamo copiare gli elementi (layer sorgente)
2. Caricare o creare il layer nel quale vogliamo incollare gli elementi copiati (layer di destinazione)
3. Impostare entrambi gli strati in modalità modifica
4. Rendere attivo il layer sorgente cliccando sul relativo nome nella legenda
5. Attivare lo strumento  **Seleziona geometrie** per selezionare gli elementi dal layer sorgente
6. Cliccare sullo strumento  **Copia geometrie**
7. Rendere attivo il layer di destinazione cliccando sul relativo nome nella legenda
8. Attivare lo strumento  **Incolla geometrie**
9. Terminare le modifiche e salvare


Se il layer sorgente e quello di destinazione hanno un diverso schema (nomi e tipi dei campi) della tabella attributi QGIS popola, (se presenti) i campi comuni e ignora il resto. Se non è importante che vengano copiati anche gli attributi nel layer di destinazione, si può non prestare attenzione a come viene definito lo schema della tabella attributi, altrimenti è necessario definirlo in modo che quello del layer sorgente e di destinazione combacino.



Tip 19 CONGRUENZA DEGLI ELEMENTI INCOLLATI

Se il layer sorgente e quello di destinazione usano lo stesso sistema di proiezione, gli elementi incollati saranno assolutamente identici a quelli dello strato di origine. Nel caso in cui invece la proiezione del layer di destinazione sia differente QGIS non garantisce che la geometria sia identica a causa del pur ridotto errore di arrotondamento introdotto nel passaggio da un sistema di proiezione all'altro.

Cancellare elementi selezionati

Se si vuole eliminare un intero poligono, è possibile farlo selezionando dapprima l'elemento che intendiamo cancellare con lo strumento  **Seleziona geometrie**. È possibile anche selezionare contemporaneamente più poligoni da cancellare in una volta sola. Una volta definita la selezione, usare

lo strumento  **Elimina selezione** per cancellare la selezione. Non è possibile annullare l'eliminazione, tuttavia è sempre possibile annullare l'eliminazione (ma anche altre eventuali modifiche effettuate dall'ultimo salvataggio!) terminando la modalità editing senza salvare le modifiche.


Anche lo strumento  **Taglia geometrie** può essere usato per eliminare elementi, che vengono in questo caso spostati in un blocco appunti spaziale. In questo modo è però possibile annullare l'operazione incollando nuovamente gli elementi tagliati con lo strumento  **Incolla geometrie** fornendo in ultima analisi almeno un livello di annullamento.

Tutti gli strumenti taglia, copia e incolla lavorano sugli elementi attualmente selezionati, consentendo quindi di lavorare su più di un elemento alla volta.

Tip 20 SUPPORTO ALLA CANCELLAZIONE DI ELEMENTI

Quando si modificano shapefiles ESRI, la cancellazione di elementi funziona solo se QGIS è compilato contro una versione di GDAL pari a 1.3.2 o superiore, come accade per le versioni compilate per OS X and Windows disponibili sul sito.

Modalità di snap

QGIS consente di agganciare i vertici digitalizzati ad altri vertici dello stesso layer. Per impostare la tolleranza di aggancio, andare alla voce di menù **Impostazioni** >  **Opzioni** > **Digitalizzazione**. Si noti che la tolleranza di aggancio è in unità di mappa.

Salvare i livelli modificati

Quando un layer è in modalità modifica, tutti i cambiamenti rimangono nella memoria di QGIS e quindi non sono immediatamente applicati e salvati nei dati su disco. Quando viene disabilitata la modalità di modifica (o si termina la sessione di QGIS mentre questa non è stata finalizzata), viene chiesto se si desidera salvare o scartare le modifiche.

Se le modifiche non possono essere salvate (ad es. perché il disco di destinazione è pieno o gli attributi contengono valori esterni agli estremi ammissibili), lo stato della memoria di QGIS è preservato, consentendo dunque di correggere gli errori e riprovare il salvataggio.

5.4.4 Creare un nuovo layer


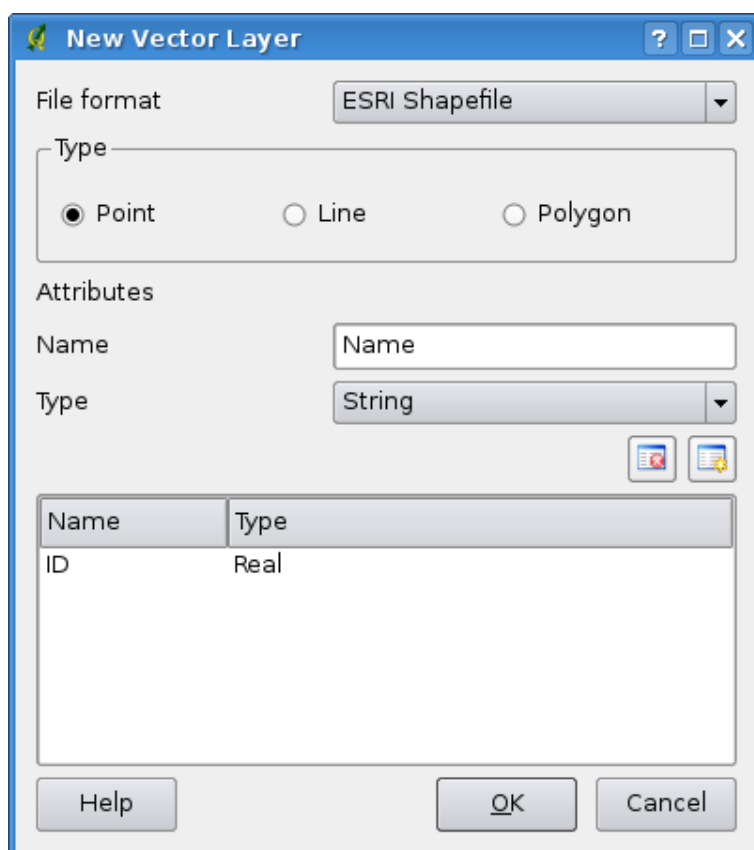
Per creare un nuovo layer da modificare, scegliere l'opzione  **Nuovo layer vettoriale** dal menù **Layer**. Verrà quindi aperta la finestra **Nuovo layer vettoriale** come mostrato in Figure 11. Scegliere qui il tipo di geometria che si intende inserire nel layer (punto, linea o poligono).

Figura 11: Finestra di creazione di un nuovo layer vettoriale 

Si noti che QGIS non supporta la creazione di elementi 2.5D (ad es. elementi con coordinate X,Y,Z) o il conteggio degli elementi. Ad oggi inoltre possono essere creati solo shapefiles. Il supporto per la creazione di layer OGR o PostgreSQL sarà inserito in future versioni di QGIS.


La creazione di layer di GRASS è supportata dal plugin GRASS. Si faccia riferimento alla Sezione [9.6](#) per ulteriori informazioni sulla creazione di layer vettoriali GRASS.

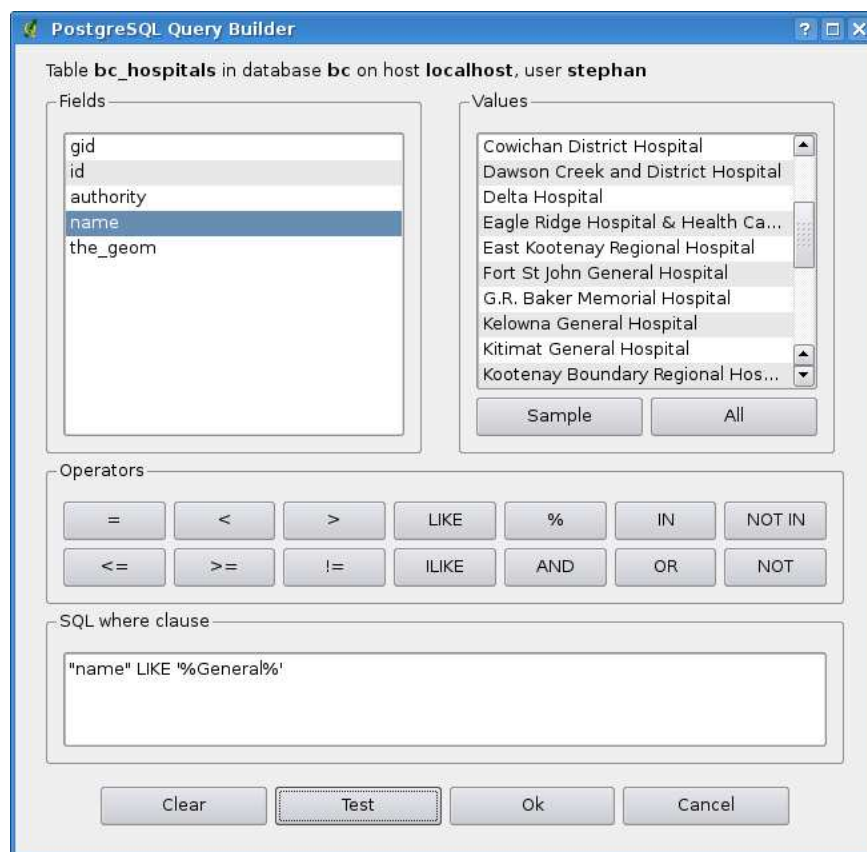
Per completare la creazione del nuovo layer, si definisca lo schema degli attributi specificando il nome e tipo della colonna da inserire nella tabella e cliccando sul pulsante **Aggiungi attributo**.

Sono supportati solo attributi di tipo **Tipo real ▼**, **Tipo Intero ▼**, and **Tipo Stringa ▼**. Una volta definito lo schema, cliccare su **OK** e assegnare un nome allo shapefile. QGIS aggiungerà automaticamente l'estensione .shp al nome indicato. Una volta creato il layer, esso sarà aggiunto alla mappa e potrà essere modificato nello stesso modo descritto alla precedente Sezione [5.4.3](#).

5.5 Costruttore di interrogazioni (query builder)

Il costruttore di query consente di definire (tramite il linguaggio Structured Query Language, SQL) un sottoinsieme di una tabella e mostrarlo come strato in QGIS. Può essere usato per tutti i formati OGR, i files di GRASS e gli strati di PostGIS. Per esempio, se si ha il file `towns` che contiene un campo `population` nella tabella attributi indicante il numero di residenti è possibile selezionare solo le città più grandi inserendo l'espressione `population > 100000` nella casella SQL del costruttore di query. La Figura 12 mostra un esempio per il costruttore di query popolato con dati provenienti da livelli PostGIS immagazzinati nel database PostgreSQL.

Figura 12: La finestra del costruttore di query 



La finestra del costruttore di query elenca i campi del database nella casella sulla sinistra. Un campione dei dati selezionabili con l'espressione impostata è visibile cliccando sul pulsante **Campione**. In questo modo vengono richiamati i primi 25 valori distinti dai campi del database. Per avere una lista completa di tutti i valori di un campo, cliccare sul pulsante **All**. Per aggiungere un campo o un valore selezionato alla query, fare doppio-click su di esso. È possibile usare i vari pulsanti per costruire le interrogazioni oppure digitare direttamente il testo dell'interrogazione nella casella SQL.


Per testare un'interrogazione, cliccare sul pulsante **Prova**. Se l'espressione è corretta, verrà restituito un conteggio dei record che verranno inclusi nella selezione. Quando l'espressione dell'interrogazione è soddisfacente, cliccare sul pulsante **OK**. Nella casella SQL verrà mostrato il testo della richiesta.

Tip 21 CAMBIARE LA DEFINIZIONE DI UNO STRATO

Si può cambiare la definizione SQL di uno strato caricato da PostGIS anche dopo che questo è stato caricato, aprendo la finestra **Proprietà del vettoriale** facendo doppio click sul nome del layer nella legenda e cliccando sul pulsante **Query Builder** nella linguetta **Generale**. Si veda la Sezione 5.3 per ulteriori informazioni.

5.6 Selezione mediante interrogazione

Con QGIS è possibile selezionare elementi per mezzo di un'interfaccia simile a quella del costruttore di interrogazioni usata in 5.5. Nella sezione precedente il costruttore di query è stato usato unicamente per mostrare gli elementi di un layer che soddisfacevano i filtri imposti in un layer virtuale sottoinsieme di quello originale. Lo scopo della selezione mediante interrogazione è invece quello di evidenziare gli elementi di un layer caricato che soddisfano particolare criteri. La selezione con query può essere usata con tutte le sorgenti di dati vettoriali supportate.

Per effettuare una selezione mediante interrogazione su un layer caricato, cliccare sul pulsante  **Apri tabella attributi** e cliccare nella finestra sul pulsante **Avanzate...** in basso a destra. In questo modo viene avviato il costruttore di interrogazioni che consente di definire un sottoinsieme degli elementi e mostrarli come descritto alla Sezione 5.5.

6 Lavorare con i dati raster

Questa Sezione descrive come visualizzare ed impostare le proprietà dei dati raster. I formati raster attualmente supportati in QGIS includono:

- Arc/Info Binary Grid
- Arc/Info ASCII Grid
- GRASS Raster
- GeoTIFF
- JPEG
- Spatial Data Transfer Standard Grids (con alcune limitazioni)
- USGS ASCII DEM
- Erdas Imagine

Considerato che il supporto ai raster in QGIS è fornito dalla libreria GDAL, oltre a quelli elencati è possibile che sia possibile caricare anche altri formati supportati da GDAL - nel dubbio è comunque possibile provare ad aprire il file in QGIS per verificare l'esito. Ulteriori dettagli sono forniti all'Appendice A.2 o all'indirizzo http://www.gdal.org/formats_list.html. Per caricare dati raster di GRASS, fare riferimento alla Sezione 9.2.

6.1 Cosa sono i dati raster?


I dati raster nei GIS sono matrici di celle discrete che rappresentano le caratteristiche della superficie terrestre o dell'ambiente al di sopra o al di sotto di essa. Ogni cella nella matrice raster presenta lo stesso formato e le celle sono solitamente rettangolari (in QGIS saranno sempre rettangolari). Esempi tipici di dati raster sono quelli provenienti dal telerilevamento come le fotografie aeree o immagini provenienti dal satellite e dati modellistici quali matrici dell'altitudine.

Diversamente dai dati vettoriali, i dati raster di solito non hanno associato un database contenente i dati descrittivi di ogni cella.

I dati raster, infine, sono geocodificati in base alla risoluzione del pixel e alle coordinate x/y di un angolo del raster che ne consente un corretto posizionamento nella vista mappa.

QGIS utilizza le informazioni di georeferenziazione incorporate nel layer raster (ad es. GeoTiff) o in un apposito world file per visualizzare correttamente i dati.

6.2 Caricati dati raster in QGIS

I layers raster possono essere caricati sia selezionando nella barra lo strumento  **Aggiungi layer raster** o scegliendo la voce di menù **Layer** > **Aggiungi layer raster**. È possibile caricare più di un livello alla volta tenendo premuto il tasto **Control** o **Shift** mentre si effettua la selezione multipla con il mouse sui livelli nella finestra di dialogo **Apri un raster supportato da GDAL**.

Quando il layer è caricato è possibile cliccare nella legenda sul relativo nome con il tasto destro per selezionare ed attivare opzioni specifiche o per aprire la finestra per l'impostazione delle proprietà del layer raster.

Menù contestuale per layers raster

- **Zoom all'estensione del layer**
- **Zoom alla scala migliore (100%)**
- **Aggiungi alla panoramica**
- **Rimuovi**
- **Proprietà**
- **Rinomina**
- **Aggiungi gruppo**
- **Espandi tutto**
- **Comprimi tutto**
- **Mostra gruppi**

6.3 Finestra delle proprietà raster

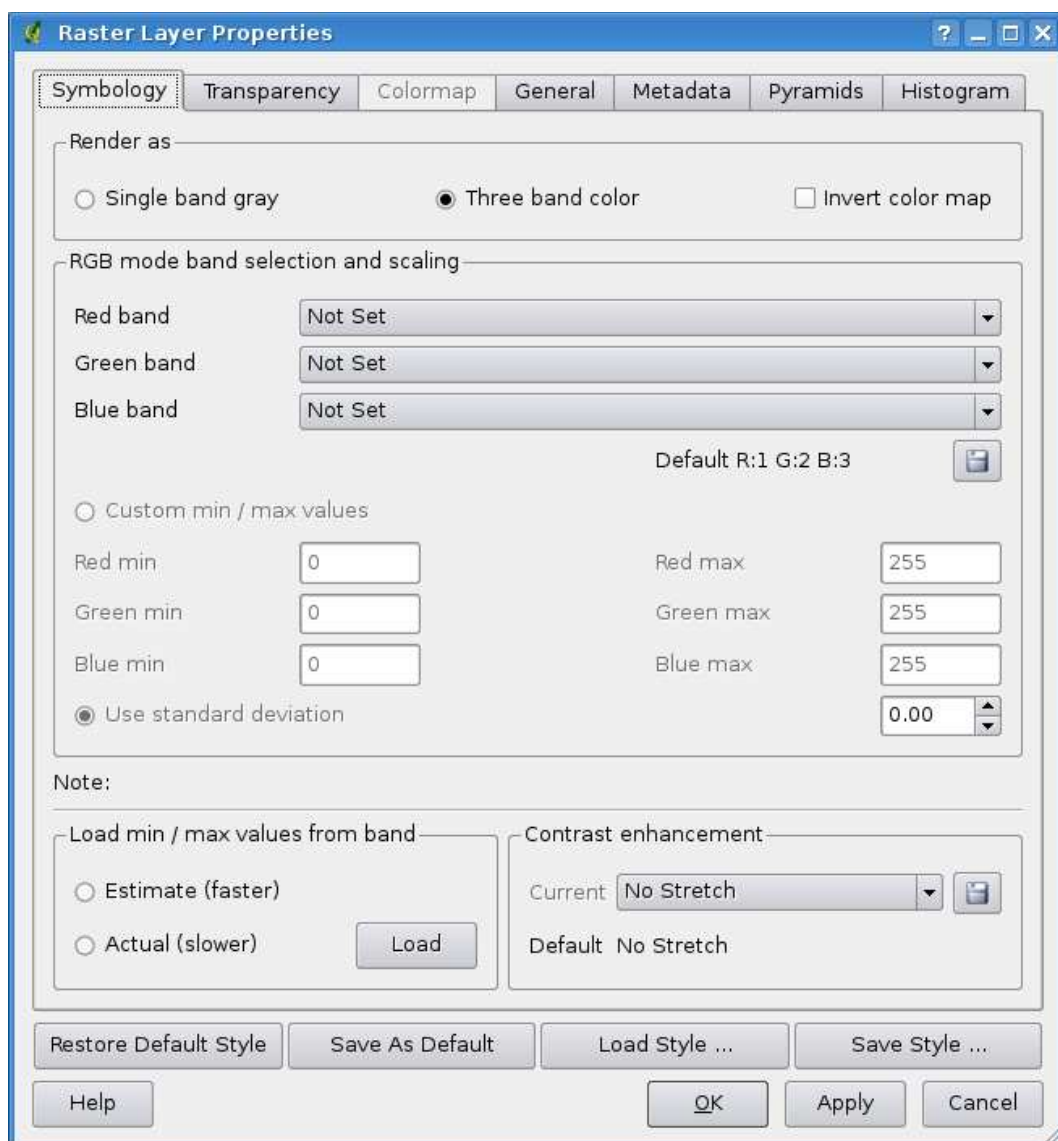
Per visualizzare ed impostare le proprietà di un layer raster, fare doppio click sul nome del raster nella legenda o cliccare su di esso con il tasto destro e scegliere **Proprietà** dal menù contestuale:

La figura 13 mostra la finestra **Proprietà raster**. Ci sono diverse linguette nella finestra:

- **Simbologia**
- **Trasparenza**
- **Mappa colore**

- **Generale**
- **Metadata**
- **Piramidi**
- **Istogramma**

Figura 13: Finestra delle proprietà dei livelli raster 🐧



6.3.1 Linguetta Simbologia

QGIS può rendere a video i raster in due modi:

- Banda singola grigia - viene campionata una sola banda dell'immagine e resa in gradazioni di grigio o in pseudocolore.
- Tre bande di colore - vengono rese a video tre bande dell'immagine ognuna rappresentante le componenti rosso, verde o blu che saranno composte per creare un'immagine a colori.

È possibile invertire i colori in modo che i colori chiari divengano scuri e viceversa selezionando la casella di controllo ☒ Inverti mappa colore.

Resa in banda singola

Selezionando questa opzione vengono offerte due possibilità tra le quali scegliere. Innanzitutto se il layer è multibanda è possibile scegliere quale banda si desidera venga impiegata per la resa a video.

La seconda opzione offre una selezione di tabelle colori preimpostate per la resa a video.

Le scelte possibili disponibili nel menù a tendina sono Mappa colore Scala di grigi ▼, impostazione di default. Altre scelte possibili sono

- Pseudo colore
- Freak Out
- Mappa colore

Quando viene selezionata l'opzione Mappa colore Mappa colore ▼, viene abilitata la linguetta Mappa colore. Si vedano ulteriori informazioni al capitolo [6.3.3](#).

Per immagini in pseudocolore in QGIS è possibile limitare i dati visualizzati per mostrare soltanto le celle i cui valori ricadono all'interno di una deviazione standard definita. Ciò può essere utile quando nella griglia raster si hanno una o due celle con valori anormalmente alti che hanno un impatto negativo sulla resa a video del raster.

Resa in tre bande

Questa opzione offre un considerevole numero di possibilità di modifica dell'aspetto del raster. Ad esempio è possibile cambiare il normale ordine RGB delle bande e/o applicare una scala di colori in base ai valori min/max del raster.

Tip 22 VEDERE UNA SINGOLA BANDA DI UN RASTER MULTIBANDA


Il modo corretto di visualizzare ad es. la sola banda rossa di un'immagine multibanda non è quello di impostare le bande verde e blu a Non impostato ma di impostare il tipo di immagine come scala di grigi e quindi scegliere il rosso come banda da usare per il grigio.

6.3.2 Linguetta Trasparenza

Questa linguetta consente di impostare in QGIS un diverso grado di trasparenza per ogni livello raster. Per settare una Trasparenza globale, impostare il cursore a scorrimento al livello di trasparenza desiderato che consenta di vedere gli eventuali layers sottostanti presenti. L'uso di questa impostazione può essere molto utile nei casi in cui si desideri sovrapporre ad es. ad una mappa delle ombreggiature del rilievo (shaded relief-map) una mappa raster contenente delle classificazioni, così da dare un aspetto tridimensionale a quest'ultima.



La sezione Nessun valore consente invece di definire la trasparenza per un certo valore del raster, che sarà quindi letto come un campo privo di dati del tipo *NODATA*. Di conseguenza in corrispondenza di tutte le celle del raster contenenti quel valore non verrà reso a video niente, determinando quindi una trasparenza per il valore specificato.

È possibile definire la trasparenza in maniera ancora più dettagliata e personalizzata nelle Sezione Opzioni di trasparenza personalizzate, nella quale è possibile impostare il grado di trasparenza per ogni valore del raster. Ad esempio se si volesse utilizzare quest'ultima sezione per visualizzare l'acqua del raster di esempio *landcover.tif* con una trasparenza del 20%, bisognerà seguire la seguente procedura:

1. Caricare il raster *landcover*
2. Aprire la finestra di dialogo **Proprietà** facendo doppio click sul nome del file nella legenda o cliccando su di esso con il tasto destro e scegliendo **Proprietà** dal menù contestuale
3. Selezionare la linguetta **Trasparenza**
4. Cliccare sul pulsante  **Aggiungi un valore manualmente**. Nella lista pixel apparirà una nuova riga
5. Inserire il valore del raster per il quale si desidera modificare la trasparenza (si supponga 0 per questo esempio) e si imposti la trasparenza al 20%
6. Cliccare sul pulsante **Applica** e guardare la mappa

È possibile ripetere i passaggi 4 e 5 per impostare ulteriori valori ai quali si desideri applicare una trasparenza personalizzata.

Appare quindi semplice impostare la trasparenza personalizzata, ma può risultare una procedura laboriosa specie se si hanno molti valori da impostare. È quindi possibile salvare la lista delle im-


postazioni di trasparenza cliccando sul pulsante  **Esporta in file** per non ripetere la procedura. Il pulsante  **Importa da file** infatti carica il file salvato e applica le impostazioni al raster selezionato.

6.3.3 Mappa colore

La linguetta **Mappa colore** viene abilitata solo quando si imposta la resa a video del raster come Banda singola grigia nella linguetta **Simbologia** (si veda il capitolo [6.3.1](#)).

Sono disponibili tre modalità di interpolazione del colore:

- Discrete
- Lineare
- Esatto

Il pulsante **Aggiungi oggetto** aggiunge un colore all'elenco. Facendo doppio click sul valore comparso nella colonna Value (di default 0.0) è possibile specificare il valore specifico del raster per il quale si vuole modificare il colore. Facendo doppio click sulla casella colorata nella colonna Color appare infatti la finestra **Select color** nella quale è possibile definire il colore da applicare alle celle raster corrispondenti al valore che si sta editando. In alternativa è possibile cliccare sul pulsante  **Carica mappa colore dalla banda**, con il quale viene caricata, se viene trovata o se disponibile, la tabella dalla banda.

Il blocco Genera nuova mappa colore consente di creare nuove mappe colore per categoria secondo i parametri impostati. È sufficiente impostare il numero di classi (valori) per i quali si intende impostare il colore nella sezione **Numero di oggetti** e cliccare poi su **Classifica**. Attualmente è possibile effettuare una classificazione solo in modalità **Modo di classificazione** .

6.3.4 Linguetta Generale

La linguetta **Generale** visualizza le informazioni di base sui raster selezionati, incluso il percorso dal quale il layer è caricato e il nome (modificabile a piacere) con il quale esso sarà visualizzato in legenda. Viene inoltre mostrata una miniatura dello strato, la legenda dei simboli e la gamma di colori (se disponibili).

È qui possibile attivare la funzione che setta la visibilità dello strato in base alla scala della mappa,

attivando l'apposita casella di controllo ed impostando l'intervallo di scale entro il quale si vuole che il layer venga reso a video.

Viene inoltre fornita l'indicazione del sistema di riferimento spaziale impostato per il livello raster in formato stringa PROJ.4. Esso è modificabile cliccando sul pulsante **Cambia**.

6.3.5 Linguetta Metadata

La linguetta **Metadata** mostra una serie di informazioni sul livello raster, come ad es. le statistiche riguardanti ogni banda. Le statistiche sono disponibili in questa linguetta solo dopo averle raccolte cliccando sulla linguetta **Istogramma** e aver cliccato sul pulsante **Aggiorna** in basso a destra (si veda il capitolo [6.3.7](#)).

Si tratta quindi di una linguetta meramente informativa nella quale non è possibile modificare alcun valore.

6.3.6 Linguetta Piramidi

I livelli raster ad alta risoluzione possono rallentare notevolmente l'esplorazione della mappa in QGIS. Creando copie a minor risoluzione dei dati (Piramidi), le prestazioni possono essere incrementate, poiché QGIS seleziona la risoluzione più appropriata in relazione al livello di zoom.

Per creare i piramidali è innanzitutto necessario avere i permessi in scrittura sulla cartella contenente il dato originale nel quale verranno salvate le copie a varia risoluzione.

Sono disponibili i seguenti metodi di ricampionamento:

- Media
- Il più vicino possibile (metodo Nearest Neighbour)

Quando viene attivata la casella di controllo ☒ **Crea piramidi interne se possibile** QGIS cerca di far sì che, se il formato lo supporta, i vari livelli di risoluzione siano immagazzinati nella stesso file raster piuttosto che in multiple copie separate.

Si evidenzia che la costruzione dei piramidali può alterare il dato originale in maniera irreversibile, quindi si raccomanda di fare una copia di backup del raster di partenza prima di eseguire l'operazione.

6.3.7 Linguetta Istogramma

La linguetta **Istogramma** consente di visualizzare la distribuzione delle bande di colore nel livello raster. Innanzitutto è necessario generare le statistiche del raster cliccando sul pulsante **Aggiorna**.

È possibile selezionare quale banda mostrare nel diagramma selezionandola nella lista sulla sinistra della linguetta (è possibile effettuare una selezione multipla cliccando su più voci). Possono essere visualizzati due tipi di diagrammi:

- Diagramma a colonna
- Diagramma a linea

Può essere definito in Conteggio colonna il numero di colonne da usare nel grafico (rappresentante il numero di intervalli in cui vengono raggruppati i dati) e decidere se si desidera abilitare ☒ Permetti approssimazione e/o mostrare valori fuori scala abilitando la casella ☒ Fuori intervallo OK?. Una volta visualizzato il grafico, verranno popolati i campi sulle statistiche per banda nella linguetta **Metadata**.

Tip 23 OTTENERE LE STATISTICHE DEL RASTER

Per ottenere le statistiche di un livello, impostarne la rappresentazione in pseudocolore e cliccare su **Applica**. L'operazione può richiedere molto tempo in funzione della quantità di dati contenuti nel raster, delle impostazioni di analisi settate e delle capacità di calcolo della macchina usata, attendere quindi pazientemente che QGIS termini l'analisi dei dati!

7 Lavorare con i dati OGC

QGIS supporta sorgenti di dati WMS e WFS. Il supporto WMS è nativo, quello per WFS è fornito tramite plugin.

7.1 Cosa sono i dati OGC

L'Open Geospatial Consortium (OGC), è un'organizzazione internazionale che raggruppa più di 300 organizzazioni commerciali, governative, nonprofit e di ricerca. I suoi membri sviluppano e implementano standards per contenuti e servizi geospaziali, analisi GIS e scambio dati.

Dal Consorzio è stato quindi elaborato un numero crescente di specifiche per i modelli di dati per garantire bisogni specifici riguardanti l'interoperabilità nell'ambito della tecnologia geospaziale, inclusi i GIS. Ulteriori informazioni all'indirizzo <http://www.opengeospatial.org/>.

Importanti specifiche OGC sono:

- **WMS** - Web Map Service
- **WFS** - Web Feature Service
- **WCS** - Web Coverage Service
- **CAT** - Web Catalog Service
- **SFS** - Simple Features for SQL
- **GML** - Geography Markup Language

Ad oggi i servizi OGC-sono sempre più di uso comune per scambiare dati geografici fra differenti implementazioni GIS. QGIS ora può gestire tre delle specifiche esposte sopra tra cui SFS (tramite il supporto a PostgreSQL/PostGIS, vedi Sezione 5.2); e WFS e WMS come client.

7.2 Client WMS

7.2.1 Panoramica sul servizio WMS

QGIS può agire come client WMS, nel rispetto delle specifiche 1.1, 1.1.1 e 1.3. E' stato particolarmente testato nei confronti di server accessibili pubblicamente quali DEMIS e JPL OnEarth.

I server WMS rispondono alle richieste da parte dei clients (ad es. QGIS) di una mappa raster di una determinata estensione, con un determinato insieme di strati, simboli e trasparenza. Il server WMS quindi consulta le sue risorse (locali o remote), genera il raster e lo invia al client in formato raster, per QGIS tipicamente come immagini JPEG o PNG.

WMS è un servizio REST (Representational State Transfer) piuttosto che un servizio web completo. Come tale, si può prendere la URL (indirizzo del server con specifiche) generata da QGIS e usarla in un browser web per ottenere la stessa immagine che QGIS usa internamente. Questo può essere utile per identificare le cause di eventuali problemi, dato che esistono vari tipi di server WMS e ciascuno ha la sua propria interpretazione degli standards WMS.

Gli strati WMS possono essere aggiunti molto semplicemente, una volta disponibile l'indirizzo (URL) per accedere al server WMS, una connessione adatta e posto che il server usi l'HTTP come meccanismo di trasferimento dati.

7.2.2 Scegliere un server WMS

La prima volta in cui si vuole utilizzare un servizio WMS, non sono presenti server predefiniti.




Si può avviare lo strumento cliccando sul pulsante  **Aggiungi layer WNS** nella barra strumenti, oppure dalla voce di menù **Layer** >  **Aggiungi layer WMS...**. Si aprirà la finestra di dialogo **Aggiungi layer dal server**. È possibile aggiungere alcuni servers cliccando sul pulsante **Aggiungi server predefiniti**. Verranno quindi aggiunti almeno tre servers WMS, incluso il server della NASA (JPL). Per definire un nuovo server WMS nella sezione **Connessioni server**, cliccare su **Nuovo** ed inserire i parametri di connessione al server WMS desiderato, seguendo le indicazioni della tabella 2:

Tabella 2: Parametri del collegamento WMS

Nome	nome per la connessione che consenta di individuarlo nella lista dei server WNS nel menù a tendina.
URL	indirizzo URL del server che fornisce i dati. Deve essere un indirizzo raggiungibile, nello stesso formato che verrebbe usato per aprire una connessione telnet o pingare un host.

È possibile, se necessario, impostare nelle opzioni i parametri del proxy per ricevere i servizi WMS da internet. Selezionare la voce di menù **Impostazioni** >  **Opzioni** e cliccare sulla linguetta **Proxy**, nella quale è possibile inserire le impostazioni abilitando la casella di controllo ☒ **Utilizza un proxy per l'accesso web**.

Una volta creata la connessione al server WMS, essa sarà memorizzata e disponibile per le successive sessioni di QGIS.

La tabella 3 mostra alcuni esempi di indirizzi di server WMS con i quali iniziare. Questi links sono stati controllati l'ultima volta nel Dicembre 2006, ma potrebbero essere nel frattempo essere stati modificati:

Tip 24 A PROPOSITO DI INDIRIZZI DEI SERVER WMS

Quando si inserisce l'indirizzo del server URL, assicurarsi di inserire l'indirizzo di base. Ad esempio non bisogna inserire frammenti tipo `request=GetCapabilities` o `version=1.0.0` nell'indirizzo.

Tabella 3: Esempi di indirizzi di server WMS pubblici

Name	URL
Atlas of Canada	http://atlas.gc.ca/cgi-bin/atlaswms_en?
DEMIS	http://www2.demis.nl/wms/wms.asp?wms=WorldMap&
Geoscience Australia	http://www.ga.gov.au/bin/getmap.pl?dataset=national
NASA JPL OnEarth	http://wms.jpl.nasa.gov/wms.cgi?
QGIS Users	http://qgis.org/cgi-bin/mapserv?map=/var/www/maps/main.map&

Un elenco esaustivo di servers WMS è reperibile all'indirizzo <http://wms-sites.com>.

7.2.3 Caricare livelli WMS

Una volta compilati correttamente i campi, si può premere sul pulsante **Connetti** per ottenere le capacità del server. Tra queste sono inclusi i formati immagine, i layer disponibili e i sistemi di proiezione forniti dal server. Considerato che si tratta di operazioni in rete, la velocità nella risposta dipenderà dalla qualità della connessione verso il server WMS. Mentre si scaricano i dati dal server, l'avanzamento dell'operazione viene visualizzato nella porzione inferiore sinistra della finestra.

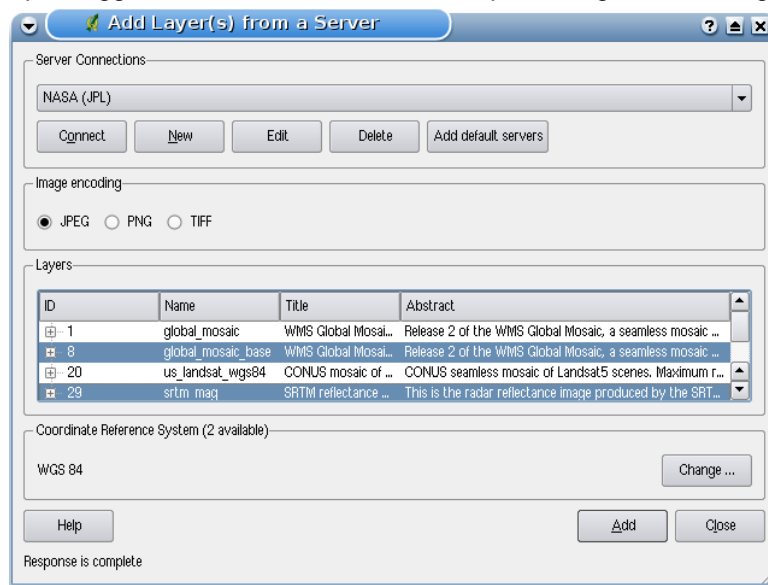
Lo schermo dovrebbe rassomigliare alla Figura 14, che mostra la risposta fornita dal server WMS NASA JPL OnEarth.

Codifica immagine

La sezione **Codifica immagine** elenca i formati supportati sia dal client che dal server. La scelta dipenderà dalla risoluzione che necessita allo scopo prefisso.

Tip 25 CODIFICA IMMAGINE

Solitamente un server WMS offrirà la scelta tra le codifiche JPEG e PNG. Mentre la compressione JPEG comporta una perdita di qualità, quella PNG riproduce fedelmente il dato di origine. Usare JPEG se non interessa avere una certa perdita di qualità dell'immagine, riducendo d'altro canto il tempo per il download dei dati di 5 volte rispetto al formato PNG. Usare invece PNG se si vuole una rappresentazione precisa del dato originale e non ci si preoccupa del tempo necessario per il trasferimento dei dati.

Figura 14: Finestra per l'aggiunta di un server WMS, nella quale vengono mostrati gli strati disponibili 


Layers

La sezione **Layer** elenca i livelli disponibili dal server WMS selezionato. Si vedrà che alcuni layers sono espandibili, il che comporta che essi possono essere mostrati in diversi stili di immagine.

Possono essere selezionati diversi layers alla volta, ma un solo stile di immagine per ognuno di essi. Quando si effettua una selezione multipla, i livelli vengono richiesti al server e trasmessi a QGIS in un solo blocco.

Tip 26 ORDINE DEGLI STRATI WMS

In questa versione di QGIS, i layers WMS caricati sono sovrapposti in base all'ordine in cui sono elencati nella sezione Layer, dall'alto verso il basso. Se si desidera sovrapporre gli strati nel senso opposto occorre

selezionare una seconda volta  **Aggiungi layer WMS**, scegliere nuovamente lo stesso server e selezionare il secondo gruppo di strati che si desidera sovrapporre al primo.

Trasparenza

In questa versione di QGIS la trasparenza è impostata per essere sempre attiva, se disponibile.

Tip 27 TRASPARENZA DEI LAYER WMS

La possibilità di rendere trasparenti gli strati WMS dipende dalla codifica tramite la quale sono stati caricati: PNG e GIF gestiscono la trasparenza mentre il JPEG no.

Sistema di proiezione delle coordinate (Coordinate Reference System)

Un sistema di proiezione delle coordinate (Coordinate Reference System, CRS) è il termine OGC per una proiezione in QGIS.


Ogni layer WMS può essere restituito in molteplici CRS, in funzione delle capacità del server. Si noti che il numero di sistemi di riferimento tra cui scegliere viene indicato nella dicitura *Coordinate Sistema di Riferimento (x disponibili)* quando si seleziona/deseleziona un livello nella sezione **Layer**.

Per scegliere uno dei CRS disponibili, cliccare su **Cambia...** per fare apparire una finestra simile a quella della Figura 17 alla Sezione 8.3. La differenza principale rispetto alla versione WMS mostrata nello schermo è che saranno mostrati solo i CRS supportati dal server al quale ci si sta effettivamente connettendo.

Tip 28 LE PROIEZIONI WMS

Per ottenere i migliori risultati, aggiungere per primo al progetto il layer WMS, in modo che il sistema di riferimento dell'intero progetto sia lo stesso attribuito al layer WMS. Si potrà quindi usare la proiezione al volo (si veda la Sezione 8.2) per adattare qualunque altro livello vettoriale successivamente aggiunto. In questa versione di qgis, aggiungere strati WMS successivamente ad altri con un CRS diverso rispetto a quello del progetto, può causare errori.

7.2.4 Uso dello strumento di identificazione

Una volta aggiunto un server WMS, e se uno dei layer disponibili è interrogabile, è possibile usare lo strumento  **Informazioni geometrie** per selezionare un pixel sulla mappa, determinando una interrogazione verso il server WMS per ogni selezione.

I risultati dell'interrogazione vengono restituiti come testo semplice la cui formattazione dipenderà dalle impostazioni del server WMS.

7.2.5 Visualizzazione delle proprietà del server

Una volta aggiunto un server WMS, è possibile visualizzarne le proprietà cliccando con il tasto destro sul suo nome nella legenda e selezionando **Proprietà**.

Linguetta Metadata

La linguetta **Metadata** mostra parecchie informazioni sul server WMS, generalmente raccolte dalla affermazione Capabilities ritornata dal server alla apposita richiesta.

Molte definizioni possono essere dedotte leggendo gli standards WMS (5), (6), alcune definizioni utili sono le seguenti:

- **Proprietà server**

- **Versione WMS** - La versione WMS supportata dal server.
- **Formati immagine** - Elenco dei tipi MIME che il server può fornire per disegnare la mappa. QGIS supporta qualunque formato sia supportato dalle librerie Qt contro le quali è compilato, che sono solitamente almeno `image/png` e `image/jpeg`.
- **Interroga formati** - L'elenco dei tipi MIME con i quali il server può fornire risposta quando si usa lo strumento Identifica geometrie. Attualmente QGIS supporta il tipo `text-plain`.


- **Proprietà layer**

- **Selezionato** - Indica se il layer era selezionato quando il server è stato aggiunto al progetto.
- **Visibilità** - Indica se il layer sia stato impostato come visibile in legenda. (funzione non ancora utilizzata in questa versione di QGIS.)
- **Può interrogare** - Indica se il layer fornisca o meno informazioni se si usa lo strumento Identifica geometrie.
- **Può essere trasparente** - Indica se il layer può essere o meno reso trasparente a video. Questa versione di QGIS farà sempre uso della trasparenza se questa voce visualizza *Sì* e se il formato immagine la supporta.
- **Può ingrandire** - Indica se questo strato possa o meno essere ingrandito dal server. Questa versione di QGIS suppone che tutti i livelli WMS abbiano questa opzione settata su *Sì*. Layers carenti in questa impostazione possono essere resi a video in modo anomalo.
- **Conteggio a cascata** - I server WMS possono fungere da proxy per altri server WMS dai quali ottengono i dati raster per un certo layer. La voce mostra quindi quante richieste per questo layer vengono inoltrate ai nodi per ottenere un risultato.
- **Larghezza fissa, Altezza fissa** - Indica se lo strato abbia o meno una dimensione del pixel fissata alla sorgente. Questa versione di QGIS assume che tutti i layer WMS abbiano vuota questa voce. Layers con impostazioni diverse possono essere resi a video in modo anomalo.
- **Perimetro WGS 84** - Estensione dello strato in coordinate WGS84. Alcuni server WMS non settano questo parametro correttamente (ad es. usano coordinate UTM invece di WGS84). In questo caso sembrerà che la vista iniziale dello strato sia ad uno zoom molto ridotto. Bisognerebbe informare di questi errori il webmaster del server WMS, il quale li dovrebbe identificare come elementi WMS XML `LatLonBoundingBox`, `EX_GeographicBoundingBox` o `CRS:84 BoundingBox`.
- **Disponibile in CRS** - Sistemi di proiezione nel quale il layer può essere rappresentato dal server WMS, elencati nel formato nativo WMS.
- **Disponibile in stile** - Stili visuali applicabili al layer dal server WMS.

7.2.6 Limitazioni del client WMS

Non tutte le possibili funzionalità WMS sono state incluse in questa versione di QGIS. Le eccezioni più rilevanti sono:

Modificare le impostazioni del layer WMS

Una volta completata la procedura mostrata dalla finestra  **Aggiungi layer WMS**, non è più possibile modificarne i parametri.

Una possibile soluzione è quella di eliminare il livello completamente e ricaricarlo reimpostando i parametri.

Servers WMS che richiedono un'autenticazione


Sono accessibili solo server pubblici. Non è infatti possibile impostare user name e password per l'autenticazione al server WMS.

Tip 29 ACCESSO A LIVELLI OGC CON PASSWORD

Se fosse necessario accedere a livelli protetti con password, è possibile usare InteProxy come proxy trasparente, che supporta molti metodi di autenticazione. Ulteriori informazioni sono fornite dal manuale di InteProxy al sito web <http://inteproxy.wald.intevation.org>.

7.3 Client WFS


In QGIS, un layer WFS si comporta come un qualsiasi altro livello vettoriale. È possibile identificare e selezionare elementi e visualizzare la tabella attributi. L'unica eccezione è che al momento non è supportata la modifica del livello. Per avviare il plugin the WFS bisogna andare alla voce di menù **Plugins** > **Gestione Plugins...**, attivare la casella di controllo ☒ **plugin WFS** e cliccare su **OK**.

Apparirà nella barra una nuova icona per lo strumento  **Aggiungi layer WFS** vicino all'icona WMS. Cliccando su di essa si aprirà la finestra di dialogo. La procedura per l'aggiunta di un layer WFS si svolge in maniera molto simile a quella vista per i WMS. La differenza sta nel fatto che non vi sono server predefiniti, di conseguenza è necessario aggiungere manualmente quelli noti.

7.3.1 Caricare un layer WFS

Come esempio è possibile caricare il server WFS DM Solutions e mostrare un layer. L'indirizzo da inserire è:

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap?VERSION=1.0.0&SERVICE=wfs&REQUEST=GetCapabilities
```

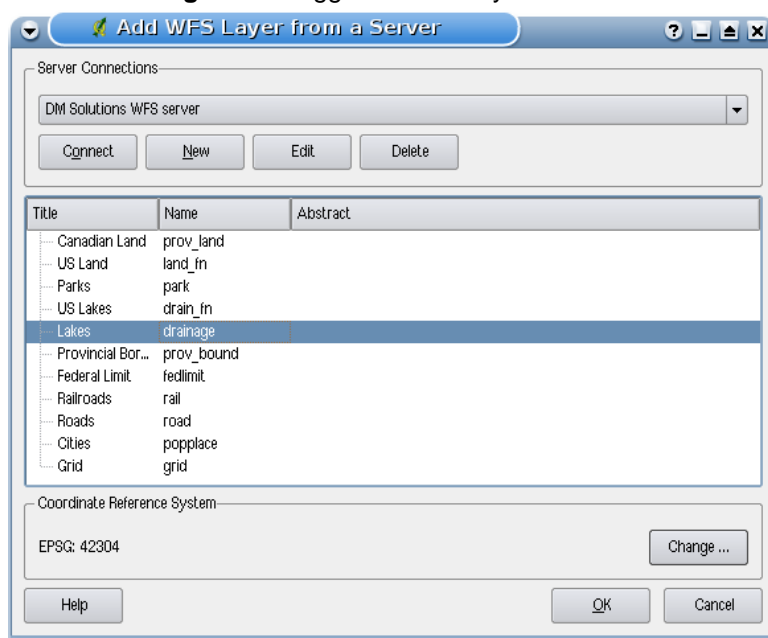

1. Assicurarsi che il plugin WFS sia caricato; in caso contrario aprire il Gestore plugins e caricarlo
2. Cliccare sullo strumento  **Aggiungi layer WFS** nella barra strumenti plugins
3. Cliccare su **Nuovo**
4. Inserire **Nome** **DM Solutions** come nome
5. Inserire l'indirizzo precedentemente indicato
6. Cliccare su **OK**
7. Selezionare **Connessioni server** **DM Solutions ▼** dal menù a tendina
8. Cliccare su **Connetti**
9. Attendere la ricezione dell'elenco layers
10. Cliccare sul livello **Canadian Land**
11. Cliccare su **OK** per aggiungere il livello alla mappa
12. Attendere pazientemente che gli elementi del livello appaiano nella vista mappa dopo essere scaricati dal server

Si noti che l'avanzamento della ricezione dei dati viene visualizzato nella parte inferiore sinistra della finestra principale di QGIS. Quando il layer è caricato, è possibile identificare e selezionare alcuni elementi e visualizzare la tabella attributi.

Tenere a mente che il plugin funziona al meglio con servers WFS basati su UMN MapServer. Sono ancora possibili comportamenti anomale e crash dello stesso, ma ci saranno miglioramenti in future versioni.

Tip 30 TROVARE SERVERS WMS E WFS

È possibile trovare ulteriori server WMS e WFS usando Google o altro motore di ricerca preferito. Ci sono anche diversi elenchi di URL pubblici, alcuni dei quali aggiornati e altri non più mantenuti.

Figura 15: Aggiunta di un layer WFS 

8 Lavorare con le proiezioni

QGIS consente all'utente di definire un CRS (Coordinate Reference System) globale o per ogni singolo progetto per i layers privi di un CRS predefinito. Consente inoltre di definire sistemi di coordinate personalizzati e supporta la riproiezione al volo (on-the-fly, OTF) dei layers vettoriali. Tutte queste caratteristiche permettono all'utente di visualizzare contemporaneamente layers con diversi CRS correttamente sovrapposti.

8.1 Panoramica del supporto alle proiezioni

QGIS supporta all'incirca 2,700 CRS noti. Le definizioni di ognuno di questi CRS sono immagazzinate in un database SQLite che viene installato con QGIS. Normalmente non è necessario manipolare il database direttamente, in effetti tale operazione può causare il malfunzionamento del supporto alla proiezione. I CRS personalizzati sono salvati in un database utente. Si veda la Sezione 8.4 per informazioni sulla gestione dei CRS personalizzati.

I CRS disponibili in QGIS sono basati su quelli definiti da EPSG e sono largamente ispirati alle tabelle `spatial_references` di PostGIS versione 1.x. Gli identificatori EPSG sono presenti nel database e possono essere usati per richiamare e definire i CRS in QGIS.

Per usare la proiezione al volo (OTF), i dati devono contenere informazioni sul proprio sistema di coordinate oppure bisogna definire un CRS per il layer, per il progetto oppure globale. Per livelli PostGIS QGIS usa l'identificatore dello spatial reference specificato quando il layer è stato creato. Per i dati supportati da OGR, QGIS fa affidamento sulla presenza di un file con formato caratteristico che definisce il CRS. Per gli shapefiles, questo significa avere l'indicazione del CRS in formato Well Known Text (WKT). Il file della proiezione ha lo stesso nome dello shapefile ma ha estensione `prj`. Per esempio uno shapefile chiamato `alaska.shp` avrà un corrispondente file di proiezione chiamato `alaska.prj`.

8.2 Specificare una proiezione

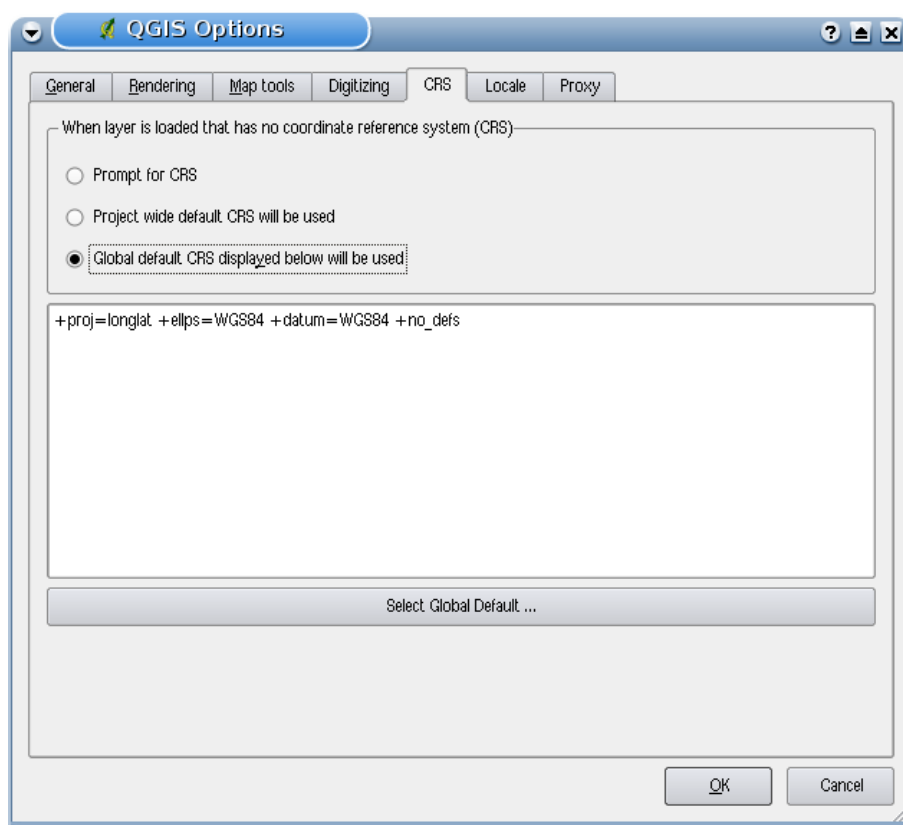
QGIS non imposta più il CRS della mappa al sistema di coordinate per primo layer caricato. Quando si avvia una sessione QGIS con livelli privi di CRS, è necessario controllare e definire il CRS di questi layers. Questo può essere fatto globalmente o per ogni progetto nella linguetta **CRS** sotto la voce di menù **Impostazioni** > **Opzioni** (Si veda la Figura 16).

- ☒ Richiesta per CRS
- ☒ CRS predefinito utilizzato dal progetto

- X Verrà utilizzato il seguente CRS globale predefinito visualizzato qui sotto

In QGIS il CRS globale predefinito è `proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs` ma può essere cambiato, la nuova impostazione verrà salvata per le successive sessioni.

Figura 16: Linguetta CRS nella finestra opzioni di QGIS 




Per definire il CRS di un layer privo di tale informazione, si può anche agire nella linguetta **Generale** della finestra delle proprietà raster (6.3.4) e delle proprietà vettore (5.3.1). Se il CRS è già stato definito, verrà mostrato come nella figura 6.

8.3 Definire la proiezione al volo (OTF)


In QGIS la proiezione al volo (OTF) non è abilitata di default, tale funzione è attualmente supportata solo per layers vettoriali. Per usare la proiezione OTF, bisogna aprire la finestra di dialogo **Proprietà progetto**, selezionare un CRS e attivare la casella di controllo X **Abilita la 'modifica al volo' della trasformazione CRS**. Ci sono due modi per aprire questa finestra:

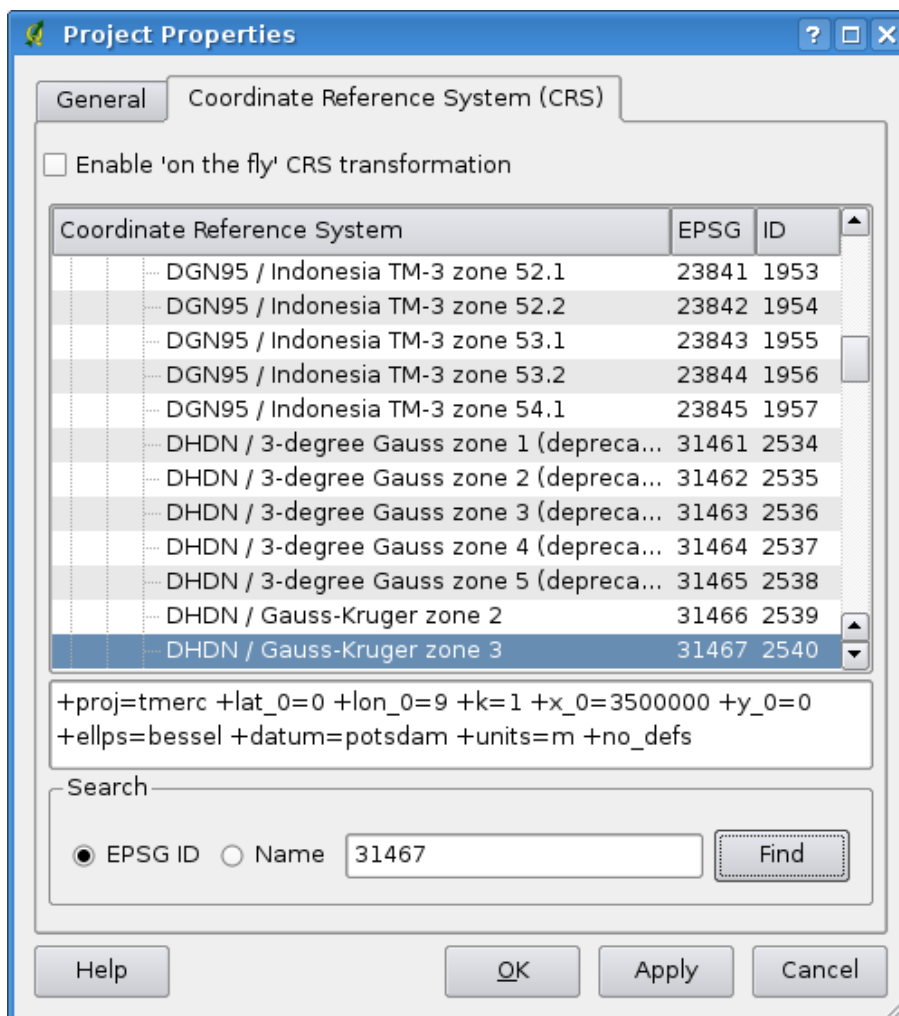
1. Selezionare **Proprietà progetto** dal menù **Impostazioni**.

2. Cliccare sull'icona  **Stato CRS** nell'angolo destro della barra di stato.

Se è già stato caricato un livello, e si vuole abilitare la proiezione OTF, è buona norma aprire la linguetta **Sistema di coordinate di riferimento spaziale (CRS)** della finestra di dialogo **Proprietà progetto** e individuare nell'elenco il CRS attualmente impostato, quindi attivare la casella di controllo ☒ **Abilita la 'modifica al volo' della trasformazione CRS**. Ogni layer caricato successivamente sarà quindi riproiettato al volo nel CRS definito.

La linguetta **Sistema di coordinate di riferimento spaziale (CRS)** della finestra **Proprietà progetto** contiene quattro importanti componenti come individuate nella Figura 17 e di seguito descritte.


Figura 17: Finestra di dialogo per l'impostazione della proiezione 



1. **Abilita la 'modifica al volo' della trasformazione CRS** - questa casella di controllo è usata per abilitare o disabilitare la proiezione OTF. Quando è deselezionata, ogni layer è tracciato usando le coordinate lette dal dato sorgente. Se abilitata, le coordinate di ogni layer sono riproiettate verso il sistema di coordinate definito per la mappa.
2. **Coordinate del Sistema di Riferimento** - è la lista di tutti i CRS supportati da QGIS, inclusi i sistemi di coordinate geografiche, piane e quelli personalizzati. Per usare un CRS, selezionarlo dalla lista espandendo la voce appropriata. Il CRS attivo è preselezionato.
3. **Stringa Proj4** - è la stringa CRS usata dal motore per le proiezioni the Proj4. È un testo in sola lettura a solo scopo informativo.
4. **Trova** - se si conosce l'identificatore EPSG o il nome del CRS che si vuole impostare, può essere usata questa area di ricerca per trovarlo nell'elenco inserendo l'identificatore e cliccando su **Trova**.

Tip 31 FINESTRA DELLE PROPRIETÀ DELLA PROIEZIONE

Se si apre la finestra **Proprietà progetto** dal menù **Impostazioni**, bisogna cliccare sulla linguetta **Sistema di coordinate di riferimento spaziale (CRS)** per vedere le impostazioni del CRS. Aprendo la

finestra dall'icona  **Stato CRS** verrà portata invece immediatamente in primo piano la linguetta **Sistema di coordinate di riferimento spaziale (CRS)**.

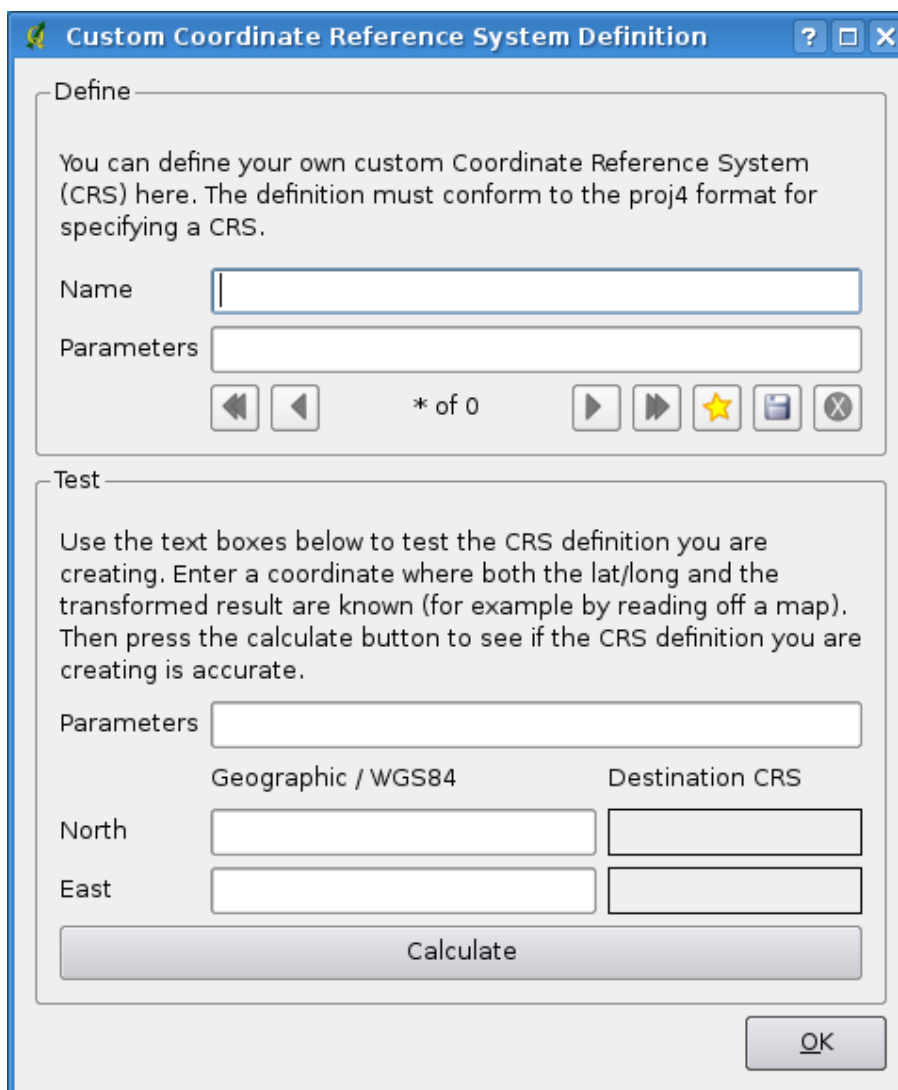
8.4 Proiezione definita dall'utente

Se in QGIS non si trova la proiezione di cui si necessita, è possibile definire un CRS personalizzato. Per fare ciò, selezionare **★ CRS personalizzato** dal menù **Impostazioni**. I CRS personalizzati sono salvati nel database utente di QGIS. Oltre ai CRS personalizzati, questo database contiene anche i segnalibri geospaziali e altri dati personalizzati.

La definizione di un CRS personalizzato in QGIS richiede una buona comprensione delle librerie Proj.4. Per iniziare, fare riferimento al documento Cartographic Projection Procedures for the UNIX Environment - A User's Manual by Gerald I. Evenden, U.S. Geological Survey Open-File Report 90-284, 1990 (disponibile all'indirizzo <ftp://ftp.remotesensing.org/proj/OF90-284.pdf>). Questo manuale descrive l'uso di proj.4 e delle relative utilità da riga di comando. I parametri cartografici usati da proj.4 sono descritti nel manuale utente e sono identici a quelli usati da .

La finestra **Definizione Sistema Riferimento Spaziale Personalizzato** richiede solo due parametri per definire un CRS personalizzato:

1. un nome descrittivo e
2. i parametri cartografici in formato PROJ.4.

Figura 18: Finestra del CRS personalizzato 


Custom Coordinate Reference System Definition

Define

You can define your own custom Coordinate Reference System (CRS) here. The definition must conform to the proj4 format for specifying a CRS.

Name

Parameters

Navigation buttons: back, forward, home, save, close

Test



Use the text boxes below to test the CRS definition you are creating. Enter a coordinate where both the lat/long and the transformed result are known (for example by reading off a map). Then press the calculate button to see if the CRS definition you are creating is accurate.

Parameters

	Geographic / WGS84	Destination CRS
North	<input type="text"/>	<input type="text"/>
East	<input type="text"/>	<input type="text"/>

Calculate

OK

Per creare un nuovo CRS, cliccare il pulsante  **Nuovo** e inserire il nome descrittivo e i parametri CRS. Successivamente salvare il CRS cliccando sul pulsante  **Salva**.

Si noti che la voce Parametri deve iniziare con un blocco `+proj=`, per rappresentare il nuovo CRS.

È possibile testare i parametri CRS per vedere se danno risultati validi cliccando sul pulsante **Calcola** nella sezione Prova dopo aver incollato i parametri CRS personalizzati nel campo Parameters. Inserire valori di latitudine e longitudine WGS 84 nei campi Nord e Est rispettivamente. Cliccare su **Calcola** e confrontare i risultati con i valori noti nel sistema CRS personalizzato.

9 Integrazione con GRASS GIS

Il plugin GRASS consente l'accesso ai dati e alle funzioni di GRASS GIS (3), inclusa la visualizzazione di strati raster e vettoriali, la digitalizzazione di strati vettoriali, la modifica degli attributi e la creazione di nuovi vettori, e l'analisi di dati GRASS 2D e 3D tramite più di 300 moduli GRASS.

In questa Sezione verranno forniti un'introduzione sulle funzionalità del plugin e qualche esempio sulla gestione e l'utilizzo di dati GRASS. Quando viene abilitato il plugin GRASS come descritto alla Sezione 9.1, nella barra sono disponibili i seguenti strumenti:

-  Apri mapset
-  Nuovo mapset
-  Chiudi mapset
-  Aggiungi vettoriale GRASS
-  Aggiungi raster GRASS
-  Crea nuovo vettoriale GRASS
-  Modifica vettoriale GRASS
-  Apri strumenti GRASS
-  Visualizza GRASS regione attuale
-  Modifica region GRASS attuale

9.1 Avviare il plugin GRASS




Per usare le funzioni GRASS e/o visualizzare livelli raster e vettoriali in formato GRASS in QGIS, bisogna selezionare e caricare il plugin GRASS con il Gestore plugin. Cliccare quindi sul menù **Plugins** > **Gestione plugins**, selezionare **GRASS** e cliccare **OK**.

Ora è quindi possibile caricare livelli raster e vettoriali da una LOCATION GRASS esistente (si veda la Sezione 9.2). È anche possibile creare una nuova LOCATION GRASS in QGIS (si veda la Sezione

9.3.1) e importare in essa dati raster e vettoriali (si veda la Sezione 9.4) per ulteriori analisi con gli strumenti GRASS (si veda la Sezione 9.9).

9.2 Caricare livelli raster e vettoriali GRASS

Con il plugin GRASS, possono essere caricati layer raster o vettoriali usando il pulsante appropriato nella barra strumenti. Come esempio si consideri il set di dati Alaska (si veda la Sezione 3.2). Esso include un piccola LOCATION GRASS campione contenente 3 layers vettoriali e una mappa di altitudine raster.

1. Creare una nuova cartella grassdata, nella quale scaricare il set di dati Alaska denominato qgis_sample_data.zip dall'indirizzo web <http://download.osgeo.org/qgis/data/> e decomprimere il file nella cartella grassdata.
2. Avviare QGIS.
3. Se non è già stato fatto in precedenti sessioni di QGIS, caricare il plugin GRASS cliccando su **Plugins** > **Gestione plugins** e selezionare **GRASS**. La barra GRASS apparirà nella barra strumenti.
4. Nella barra strumenti GRASS, cliccare sull'icona  **Apri mapset** per aprire la finestra Seleziona Mapset.
5. Alla voce Gisdbase inserire l'indirizzo completo o navigare fino alla cartella grassdata appena creata.
6. Dovrebbe ora essere possibile selezionare la LOCATION alaska e il MAPSET demo.
7. Cliccare su **OK**. Si noti che ora alcuni degli strumenti precedentemente disabilitati sono divenuti attivi.
8. Cliccare su  **Aggiungi raster GRASS**, scegliere la mappa denominata gtopo30 e cliccare su **OK**. Verrà visualizzato il layer delle elevazioni del terreno.
9. Cliccare su  **Aggiungi vettoriale GRASS**, selezionare la mappa denominata alaska e cliccare su **OK**. Il confine dell'Alaska verrà sovrapposto alla mappa gtopo30 map. Ora è possibile adattare le proprietà del layer come descritto al capitolo 5.3, ovvero cambiare la trasparenza, il colore di riempimento e del contorno dell'elemento.
10. Caricare anche gli altri due layer vettoriali denominati rivers e airports e modificarne le proprietà.

Come dimostrato è quindi molto semplice caricare dati raster e vettoriali in formato GRASS in QGIS. Si veda la sezione seguente per sapere come editare dati GRASS e creare nuove

LOCATION. Ulteriori LOCATIONs campione di GRASS sono disponibili sui siti di GRASS all'indirizzo <http://grass.osgeo.org/download/data.php>.

Tip 32 CARICARE DATI GRASS



Se si presentano problemi nel caricare dati o QGIS termina inaspettatamente, assicurarsi di aver caricato il plugin GRASS correttamente come descritto alla Sezione 9.1.

9.3 LOCATION e MAPSET in GRASS

GRASS organizza i propri in cartelle alle quali si fa riferimento con la denominazione GISDBASE. Queste cartelle, spesso chiamate *grassdata*, devono essere create prima di iniziare a lavorare con il plugin GRASS in QGIS. In queste directori, i dati GRASS sono organizzati per progetti inseriti in sottocartelle chiamate LOCATION. Ogni LOCATION è definita da un sistema di coordinate, da un sistema di proiezione e dall'estensione geografica. La LOCATION può avere a sua volta molte sottocartelle MAPSETs usate per suddividere il progetto in diversi argomenti, sottoregioni o spazi di lavoro per i diversi membri del team che vi sta lavorando (Neteler & Mitasova 2008 (2)). Per analizzare layers raster e vettoriali con i moduli GRASS, bisogna importarli in una LOCATION GRASS.⁴

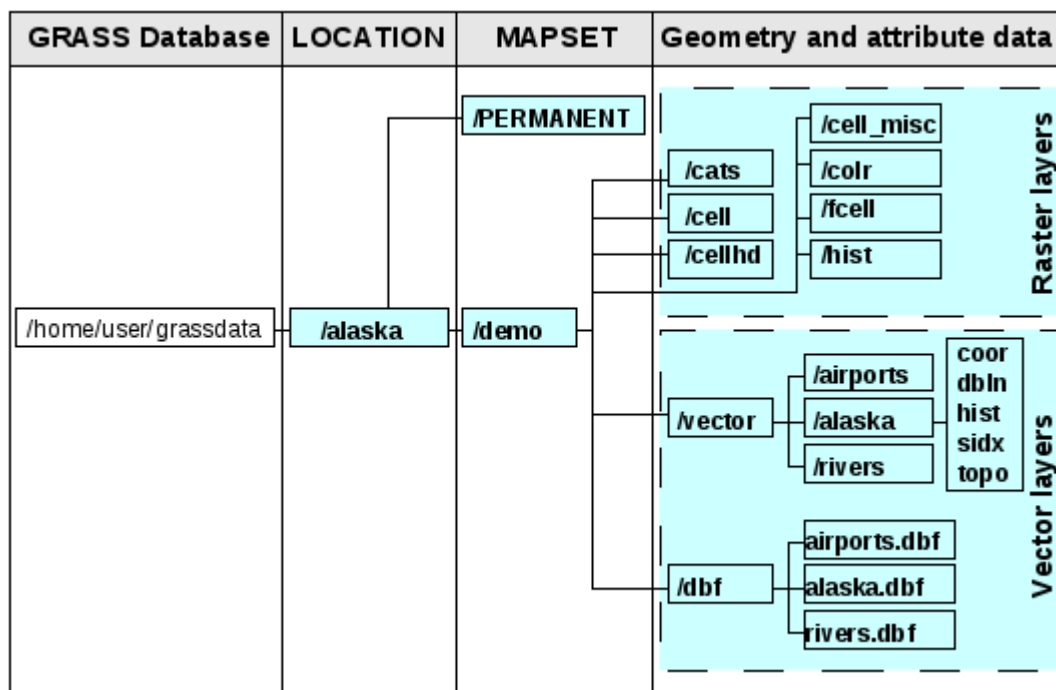
9.3.1 Creare una nuova LOCATION GRASS



Per questo e per tutti i successivi esempi riguardanti GRASS GIS verrà usata la LOCATION *alaska* campione, che è proiettata nel sistema Albers Equal Area con unità di misura in metri, creata dal set di dati campione di QGIS. Sarà utile scaricare ed installare il set di dati sul proprio computer (3.2).

1. Avviare QGIS e assicurarsi che il plugin GRASS sia caricato
2. Visualizzare lo shapefile *alaska.shp* (si veda la Sezione 5.1.1) dal set di dati *alaska* (3.2).
3. Nella barra strumenti GRASS, cliccare sull'icona  **Nuovo mapset** per avviare la procedura guidata.
4. Selezionare la cartella contenente il database GRASS (GISDBASE) denominata *grassdata* o crearne una nuova in cui ospitare la nuova LOCATION usando il gestore di file installato sul proprio computer. Cliccare su **Next**.
5. Per creare un nuovo MAPSET in una LOCATION esistente (si veda la Sezione 9.3.2) o per creare una nuova LOCATION, selezionare l'opzione  Crea nuova location (si veda la Figura 20).
6. Inserire il nome della LOCATION - nell'esempio abbiamo usato *alaska*, e cliccare su **Next**

⁴Questo non è proprio vero - con i moduli GRASS *r.external* e *v.external* è possibile creare collegamenti in sola lettura ai dati GDAL/OGR supportati senza che sia necessario effettuare l'importazione. Tuttavia siccome questa non è la modalità predefinita per i principianti questa funzione non verrà descritta.

Figura 19: Organizzazione dei dati GRASS nella LOCATION campione alaska (adattato da from Neteler & Mitasova 2008 (2))



7. Definire la proiezione cliccando sull'opzione  Proiezione per abilitare l'elenco delle proiezioni
8. Scegliere la proiezione Albers Equal Area Alaska (feet). Siccome ne conosciamo l'identificativo EPSG che è 2964, inserirlo nella casella di ricerca. (Nota: se si vuole ripetere il processo per un'altra LOCATION e non si è memorizzato l'identificativo EPSG della proiezione, cliccare sull'icona  **Stato CRS** nella parte destra inferiore della barra di stato (si veda la sezione 8.3)).
9. Cliccare su **Trova** per selezionare la proiezione
10. Cliccare su **Next**
11. Per definire l'estensione della regione di default, bisogna inserire i limiti della LOCATION verso nord, sud, est e ovest. Nel nostro esempio cliccare semplicemente sul pulsante **Imposta estensione attuale di QGIS**, per applicare l'estensione del layer caricato alaska.shp come estensione di default della region GRASS.
12. Cliccare su **Next**
13. Abbiamo anche bisogno di definire un MAPSET interno alla location LOCATION. Il nome è a scelta, nell'esempio abbiamo usato demo.⁵

⁵Quando si crea una nuova LOCATION, GRASS crea automaticamente un MAPSET speciale chiamato PERMANENT desi-

Figura 20: Creazione di una nuova LOCATION GRASS o di un nuovo MAPSET in QGIS 

14. Controllare il riassunto per assicurarsi che le impostazioni siano corrette e cliccare su **Finish**
15. La nuova LOCATION alaska e i MAPSETs demo e PERMANENT vengono creati. Il set di lavoro impostato per il lavoro corrente è il MAPSET demo.
16. Si noti che alcuni strumenti della barra di GRASS precedentemente disabilitati sono ora attivi.



Per quanto possa essere apparsa lunga, questa procedura costituisce un modo veloce per creare una LOCATION. La LOCATION alaska è ora pronta per l'importazione dei dati (si veda la Sezione 9.4). È anche possibile usare dati raster e vettoriali esistenti nella LOCATION alaska campione in formato GRASS inclusi nel dataset alaska di QGIS 3.2 e procedere alla Sezione 9.5.

9.3.2 Aggiungere un nuovo MAPSET

Ogni utente ha accesso in scrittura solo ai MAPSET GRASS che ha creato. Ciò implica che oltre ad accedere ai propri MAPSET, l'utente può anche leggere i dati contenuti nei MAPSETs creati da altri, ma può modificare o rimuovere solo i dati nei suoi MAPSET. Tutti i MAPSETs includono un file denominato



gnato a contenere i dati di base del progetto, l'estensione spaziale di default e la definizione del sistema di coordinate (Neteler & Mitasova 2008 (2)).

WIND nel quale sono immagazzinati i valori di coordinate dei limiti della regione impostata e la risoluzione spaziale impostata per i raster (Neteler & Mitsova 2008 (2), si veda la Sezione 9.8). Per creare un nuovo MAPSET:

1. Avviare QGIS e assicurarsi che il plugin GRASS sia abilitato
2. Nella barra strumenti GRASS, cliccare sull'icona  **Apri mapset** per aprire la procedura guidata di creazione del MAPSET.
3. Selezionare la cartella GRASS (GISDBASE) `grassdata` con il nome `LOCATION alaska`, nel quale si vuole aggiungere un ulteriore MAPSET, che chiameremo test.
4. Cliccare su **Next**.
5. Con questa procedura possiamo creare un nuovo MAPSET all'interno di una `LOCATION` esistente i creare anche una nuova `LOCATION` contemporaneamente. Cliccare sull'opzione  **Selezionare location** (si veda la Figura 20) e cliccare su **Next**.
6. Inserire il nome `test` per il nuovo MAPSET. Più sotto nella finestra è visibile una lista di MAPSETs esistenti e i relativi proprietari.
7. Cliccare su **Next**, controllare che il riassunto per assicurarsi che le impostazioni siano corrette e cliccare **Finish**.

9.4 Importare dati nelle LOCATION GRASS

Questa Sezione fornisce un esempio su come importare dati raster e vettoriali nella `LOCATION GRASS alaska` fornita dal set di dati QGIS alaska. Verrà usata la mappa raster dell'uso del suolo `landcover.img` e il vettoriale poligonale denominato `lakes.gml` dal set di dati medesimo 3.2.

1. Avviare QGIS e assicurarsi che il plugin GRASS sia caricato.
2. Nella barra strumenti GRASS, cliccare sull'icona  **Open MAPSET** per avviare la finestra di selezione MAPSET.
3. Selezionare come GRASS database la cartella `grassdata` nel set di dati QGIS alaska, come `LOCATION alaska`, come MAPSET `demo` e cliccare su **OK**.
4. Cliccare ora sullo strumento  **Apri strumenti GRASS**. Apparirà la finestra degli strumenti di GRASS (si veda la Sezione 9.9).
5. Per importare la mappa raster `landcover.img`, cliccare sul modulo `r.in.gdal` nella linguetta **Albero moduli**. Questo modulo GRASS consente l'importazione di file supportati da GDAL nella `LOCATION GRASS`. Apparirà la finestra di dialogo del modulo `r.in.gdal`.
6. navigare alla cartella raster nel set di dati QGIS alaska e selezionare il file `landcover.img`.

7. Come nome del raster in uscita inserire `landcover_grass` e cliccare su **Esegui**. Nella linguetta **Output** è possibile verificare l'avanzamento del comando `GRASS r.in.gdal -o input=/path/to/landcover.img output=landcover_grass`.
8. Quando compare la dicitura **Operazione conclusa con successo** cliccare sul pulsante **Visualizza output**. Il layer raster `landcover_grass` è ora importato in GRASS e verrà visualizzato nella vista mappa di QGIS.
9. Per importare il vettoriale in formato shape `lakes.gml`, cliccare sul module `v.in.ogr` nella linguetta **Albero moduli**. Questo modulo GRASS consente di importare i formati vettoriali supportati da OGR in una LOCATION GRASS. Apparirà la finestra di dialogo `v.in.ogr`.
10. Navigare alla cartella `gml` nel set di dati QGIS `alaska` e selezionare il file `lakes.gml` come file OGR.
11. Come nome della mappa vettoriale in uscita inserire `lakes_grass` e cliccare su **esegui**. In questo esempio è possibile trascurare le altre opzioni. Nella linguetta **Output** è possibile verificare l'avanzamento del comando `GRASS v.in.ogr -o dsn=/path/to/lakes.shp output=lakes_grass`.
12. Quando compare la dicitura **Operazione conclusa con successo** cliccare su **Visualizza output**. Il layer vettoriale `lakes_grass` è ora importato in grass e verrà visualizzato nella vista mappa di QGIS.

9.5 Il modello di dato vettoriale in GRASS

Prima di digitalizzare dati in GRASS è importante capirne la struttura del dato vettoriale. In generale, GRASS usa un modello di dato vettoriale topologico. Questo significa che le aree non sono rappresentate con poligoni chiusi singoli, ma da uno o più contorni (boundaries). Un contorno (boundary) tra due aree adiacenti è digitalizzato una sola volta, e condiviso da entrambe le aree. Perché un'area sia topologicamente corretta, i contorni devono essere collegati senza soluzione di continuità. Un'area è identificata (etichettata, labeled) dal centroide dell'area stessa.

Oltre a contorni e centroidi, una mappa vettoriale può contenere anche punti e linee. Tutti questi elementi possono essere mischiati in un singolo layer vettoriale e saranno rappresentati in cosiddetti 'layers' differenti in una singola mappa vettoriale di GRASS. Quindi in GRASS con layer non si intende una mappa raster o vettoriale ma un livello all'interno di un dato vettoriale. Questa è una distinzione molto importante da tenere presente.⁶

È possibile salvare più 'layers' in un set di dati vettoriale. Per esempio campi, foreste e laghi possono essere salvati in un vettoriale. Foreste e laghi adiacenti possono condividere lo stesso contorno, ma

⁶Sebbene sia possibile mescolare elementi geometrici di diverso tipo (punti, linee, contorni e centroidi), ciò è abbastanza insolito e perfino in GRASS viene usato solo in casi speciali come ad es. quando si esegue l'analisi di una rete vettoriale. Di solito è preferibile che elementi geometrici diversi vengano digitalizzati su file distinti.

avranno tabelle degli attributi distinte. È anche possibile assegnare attributi ai contorni. Ad esempio se il controrno tra un lago ed una foresta è una strada, può avere una diversa tabella degli attributi.

Il 'layer' di un elemento è definito dal 'layer' in GRASS. Al 'layer' viene assegnato un indice numerico che definisce se c'è più un gruppo geometrico nel set di dati vettoriale, ad es. se la geometria è foresta o lago. Attualmente tale indice può essere solo un numero, in versioni di GRASS successive saranno supportati anche i nomi dei campi nell'interfaccia utente.

Gli attributi degli elementi geometrici possono essere salvati nella LOCATION GRASS in formato DBase o SQLITE3 o in database esterni come PostgreSQL, MySQL, Oracle, ecc.

Gli attributi contenuti nelle tabelle del database sono collegate alla geometria per mezzo di un valore 'category'. 'Category' (key, ID) è un valore intero collegato alle primitive geometriche ed è usato come collegamento ad una colonna chiave nella tabella del database.

Tip 33 CONOSCERE IL MODELLO DEL DATO VETTORIALE IN GRASS

Il miglior modo per capire il modello di dato vettoriale di GRASS e le sue capacità è quello di scaricare uno dei molti tutorial di GRASS nei quali tale modello è descritto più approfonditamente. Si veda <http://grass.osgeo.org/gdp/manuals.php> per informazioni, libri e tutorials in diverse lingue.

9.6 Creazione di un nuovo layer vettoriale GRASS

Per creare un nuovo layer vettoriale GRASS tramite il plugin GRASS cliccare sullo strumento



Crea nuovo vettoriale GRASS. Inserire un nome nella casella di testo e iniziare la digitalizzazione di punti, linee o poligoni, seguendo la procedura descritta alla Sezione 9.7.

Come detto in GRASS è possibile inserire ogni tipo di geometria (punti, linee ed aree) in un singolo livello, in quanto viene impiegato un modello di dato vettoriale topologico, di conseguenza non è necessario selezionare a priori il tipo di geometria che si intende digitalizzare quando si crea un vettoriale in formato GRASS. In questo si differenzia ad es. dalla creazione di shapefiles con QGIS, in quanto questi ultimi usano un modello vettoriale denominato Simple Feature (si veda la Sezione 5.4.4).

Tip 34 CREAZIONE DI UNA TABELLA ATTRIBUTI PER UN NUOVO LIVELLO VETTORIALE GRASS

Se si desidera assegnare attributi alla geometria digitalizzata, accertarsi di definire lo schema della tabella prima di iniziare a digitalizzare (si veda la Figura 25).















Per creare poligoni in GRASS, bisogna iniziare con il digitalizzarne i contorni, impostando preliminarmente il modo *Nessuna categoria*. Una volta chiuso il poligono, aggiungere un centroide (marcatore dell'etichetta) all'interno del contorno chiuso impostando preliminarmente la modalità *Prossimo non in uso*. Questa procedura è necessaria in quanto il modello di dato vettoriale topologico collega le informazioni sull'attributo del poligono sempre al centroide e non al contorno.

Nella figura 21 sono mostrate le icone della barra strumenti per la digitalizzazione in GRASS fornita dal plugin GRASS. La Tabella 4 mostra le funzioni disponibili.

La linguetta **Categoria** consente di definire il modo in cui i valori della categoria verranno assegnati al nuovo elemento geometrico.

- **Modalità:** modalità con la quale viene assegnata la categoria (colonna cat nella tabella) alle geometrie digitalizzate.
 - Prossimo non in uso - applica il primo valore non utilizzato in ordine numerico crescente all'elemento.
 - Inserimento manuale - definizione manuale della categoria da assegnare all'elemento.
 - Nessuna categoria - non assegnare alcun valore all'elemento. Questa modalità è in genere usata ad es. per i contorni dei poligoni ai quali la categoria viene collegata tramite il centroide.
- **Categoria** - Il numero (ID) inserito o visualizzato viene associato ad ogni elemento digitalizzato. Viene usato per collegare ogni elemento geometrico con i relativi attributi.

Tabella 4: Strumenti per la digitalizzazione in GRASS

Icone	Strumento	Scopo
	Nuovo punto	Digitalizza un nuovo punto
	Nuova linea	Digitalizza una nuova linea (annullare selezionando un altro strumento)
	Nuovo contorno	Digitalizza nuovo contorno (annullare selezionando un altro strumento)
	Nuovo centroide	Digitalizza un nuovo centroide (imposta l'etichetta per area esistente)
	Sposta vertice	Sposta un vertice di un'esistente linea o contorno in una nuova posizione
	Aggiungi vertice	Aggiunge un vertice ad una linea o contorno esistente
	Elimina vertice	Cancella vertici da linee e contorni esistenti (confermare l'eliminazione del vertice selezionato cliccando una seconda volta)
	Sposta elemento	Sposta il confine, la linea, il punto o il centroide selezionato in una nuova posizione
	Dividi linea	Divide una linea o un contorno in due parti nel punto selezionato (confermare cliccando una seconda volta)
	Elimina elemento	Elimina un contorno, una linea, un punto o un centroide esistente (confermare cliccando una seconda volta)
	Modifica attributi	Edita gli attributi dell'elemento selezionato (si noti che ad un elemento possono essere associati più attributi, si veda sopra)
	Chiudi	Chiudi la sessione e salva lo stato attuale (ricostruisce conseguentemente la topologia)

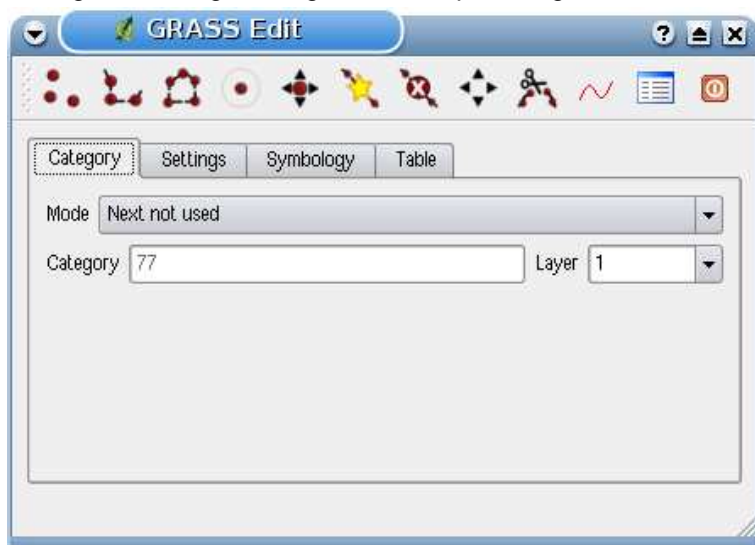
- **Layer** - Ogni elemento geometrico può essere collegato con molteplici tabelle attributo usando diversi 'layers' secondo il modello GRASS. Il layer di default è il numero 1.

Tip 36 CREARE UN 'LAYER' GRASS AGGIUNTIVO CON QGIS

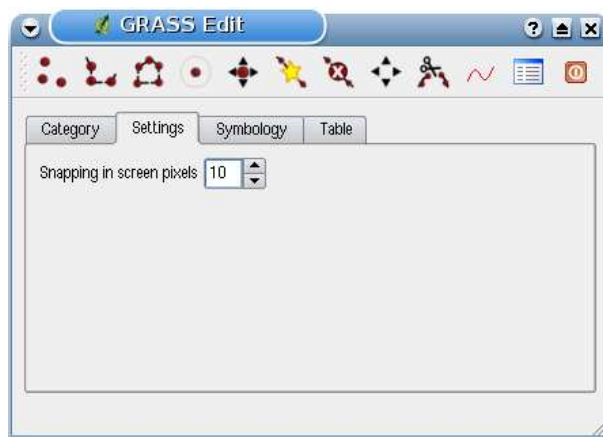

Se si vogliono aggiungere ulteriori layer al dato, inserire semplicemente un numero nel campo 'Layer' e dare invio. Nella linguetta Tabella sarà a questo punto possibile creare il nuovo schema degli attributi da associare a questo 'layer'.

Linguetta Preferenze

La linguetta **Preferenze** consente di impostare la tolleranza per l'aggancio automatico tra elementi (snapping) in pixels dello schermo. La soglia definisce a quale distanza massima nuovi punti o linee

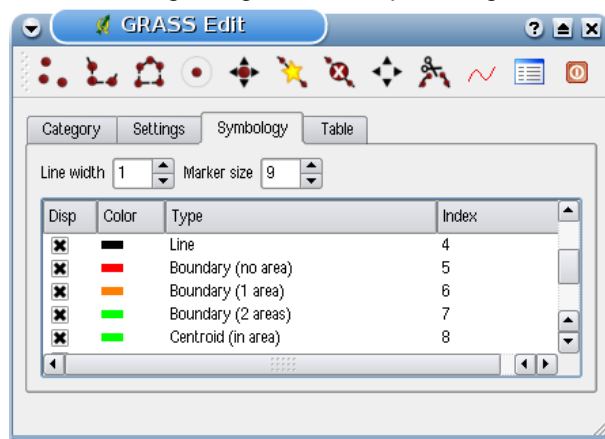
Figura 22: Linguetta Categoria negli strumenti per la digitalizzazione in GRASS 

sono agganciati ad altri nodi esistenti. Ciò aiuta ad evitare interruzioni o incroci tra contorni. Il valore preimpostato è 10 pixels.

Figura 23: Linguetta Preferenze negli strumenti per la digitalizzazione in GRASS 

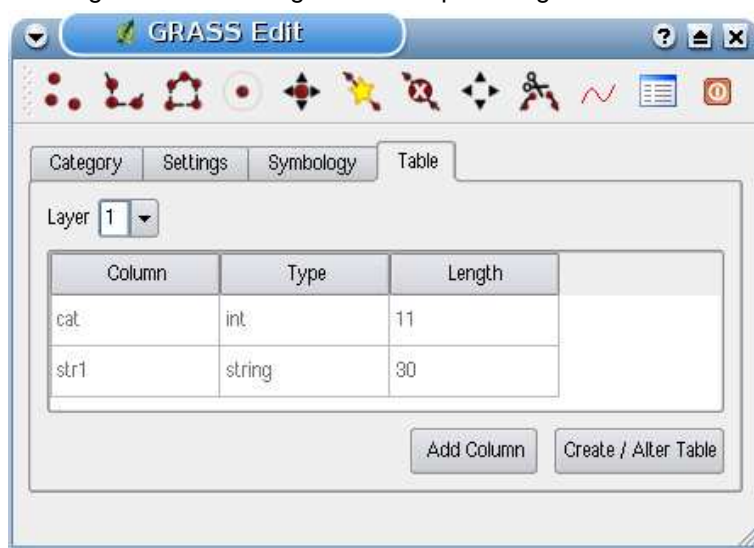
Linguetta Simbologia

La linguetta **Simbologia** consente di visualizzare e impostare la simbologia e i colori dei vari tipi geometrici nei vari stati topologici (ad es. contorni aperti/chiusi).

Figura 24: Linguetta Simbologia negli strumenti per la digitalizzazione in GRASS

Linguetta Tabella

La linguetta **Tabella** fornisce informazioni sulla struttura della tabella nel database per un determinato 'layer'. Qui si possono aggiungere nuove colonne ad una tabella attributi esistente o creare un nuovo schema tabella per un nuovo layer vettoriale GRASS o per un nuovo 'layer' (si veda la Sezione 9.6).

Figura 25: Linguetta Tabella negli strumenti per la digitalizzazione in GRASS


Tip 37 PERMESSI DI MODIFICA IN GRASS


È necessario essere il proprietario del MAPSET GRASS che si vuole editare. Non è possibile modificare dati in un MAPSET del quale non si è proprietari, anche se si possiedono su di esso permessi in scrittura.

9.8 Lo strumento Region di GRASS


L'impostazione di una regione (ovvero di una porzione di spazio geografico nella quale operare) è molto importante in GRASS quando si lavora con dati raster. L'analisi vettoriale infatti di default non è limitata dall'impostazione della regione ma viene fatta su tutta l'estensione del layer. Tutti i raster di nuova creazione avranno l'estensione e risoluzione spaziale della regione GRASS definita, indipendentemente dalla loro estensione e risoluzione originale. L'impostazione corrente della regione GRASS è salvata nel file `$LOCATION/$MAPSET/WIND` che ne definisce i limiti nord, sud est e ovest, il numero di righe e colonne e la risoluzione spaziale in senso orizzontale e verticale.

È possibile abilitare/disabilitare la visualizzazione della region di GRASS nella vista mappa in QGIS

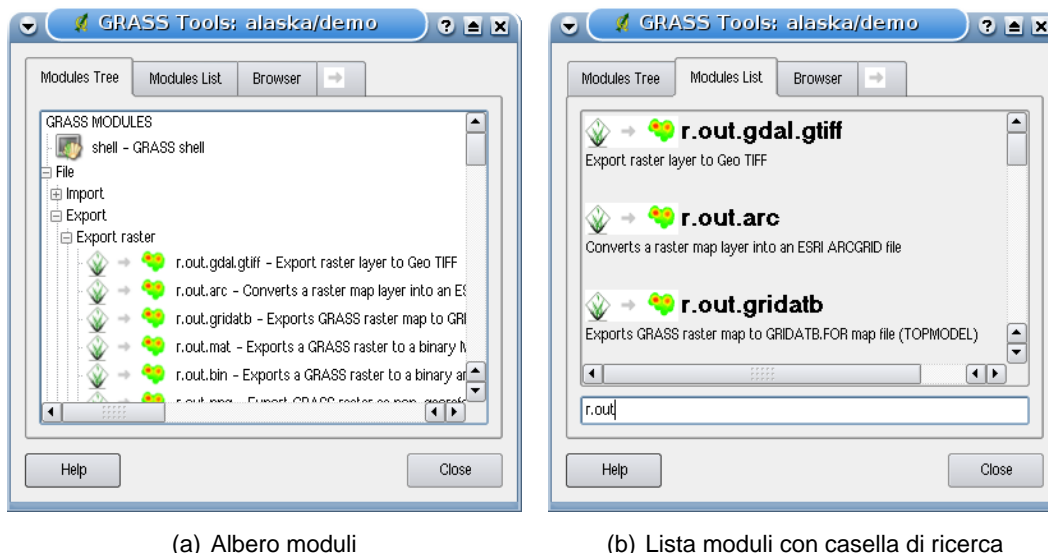
usando il pulsante  **Visualizza GRASS regione attuale**.

Con lo strumento  **Modifica region GRASS attuale** è possibile aprire una finestra di dialogo per cambiare le impostazioni attuali della regione e la simbologia con la quale il rettangolo che la rappresenta viene visualizzato nella vista mappa di QGIS. Inserire i nuovi limiti della regione e la risoluzione e cliccare su **OK**. Lo strumento consente anche di selezionare l'estensione della regione interattivamente con il mouse nella vista mappa di QGIS. Cliccando quindi con il tasto sinistro del mouse nella vista mappa si imposta il primo angolo del rettangolo che definirà la regione e cliccando in un altro punto lo si chiuderà. Cliccare quindi su **OK** per confermare. Il modulo GRASS `g.region` mette a disposizione molti più parametri per definire l'estensione della regione e la risoluzione con la quale si vuole condurre l'analisi raster. Si possono usare questi parametri usando lo strumento appropriato nella barra GRASS, descritta nella Sezione [9.9](#).

9.9 La barra strumenti GRASS

Lo strumento  **Apri strumenti GRASS** fornisce accesso alle funzionalità dei moduli GRASS con i quali lavorare nella LOCATION e nel MAPSET impostati. Per usare gli strumenti di GRASS è necessario aprire una LOCATION e un MAPSET sui quali si abbiano permessi di scrittura (in genere concessi se si è l'utente che ha creato il MAPSET). Ciò è necessario in quanto i nuovi layer raster o vettoriali creati durante l'analisi devono poter essere scritti nella LOCATION e nel MAPSET selezionato.

9.9.1 Lavorare con i moduli GRASS

Figura 26: Strumenti di GRASS e Lista Moduli con possibilità di ricerca 

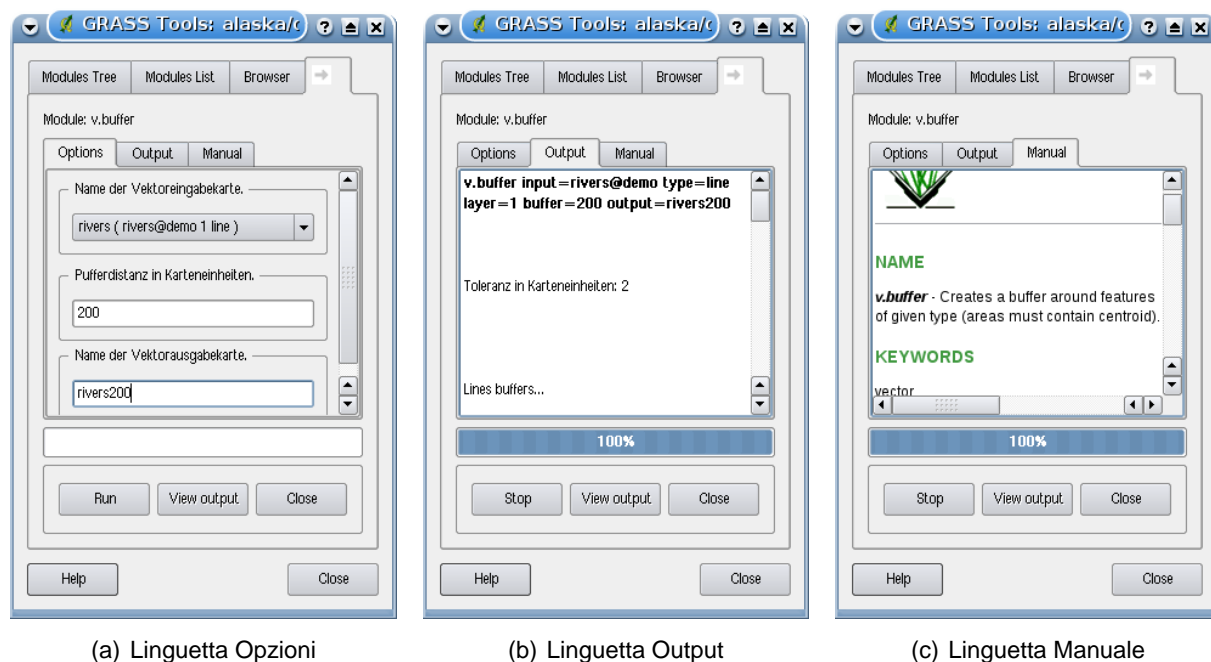
La shell di GRASS negli strumenti fornisce accesso a praticamente tutti gli oltre 300 moduli GRASS in modalità riga di comando. Per offrire un ambiente di lavoro maggiormente user-friendly, circa 200 di questi moduli e delle relative funzionalità sono presentate in finestre di dialogo. Questi moduli sono raggruppate in blocchi tematici ma possono anche essere cercati. Una lista completa dei moduli GRASS disponibili nella versione di QGIS 1.0.0 è fornita in Appendice B. È anche possibile personalizzare il contenuto degli strumenti GRASS come descritto alla Sezione 9.9.3.

Come mostrato in Figura 26, è possibile cercare il modulo GRASS desiderato cercandolo per aree tematiche nella linguetta **Albero moduli** o nella linguetta con casella di ricerca **Lista moduli**.

Cliccando sull'icona di un modulo grafico verrà aggiunta una nuova linguetta alla finestra degli strumenti. In questa linguetta si avranno tre ulteriori sottolinguette denominate **Opzioni**, **Output** e **Manuale**. Un esempio è mostrato in Figura 27 per il modulo `v.buffer`.

Opzioni

La linguetta **Opzioni** fornisce una finestra semplificata nel quale di solito è possibile selezionare un layer raster o vettoriale ed inserire ulteriori opzioni specifiche per l'esecuzione del modulo. Per mantenere la leggibilità della finestra non sempre sono presenti tutte le opzioni, se si volessero usare ulteriori parametri per il modulo è necessario avviare la shell di GRASS e eseguire il modulo dalla riga di comando.

Figura 27: Finestre degli strumenti GRASS 

Output

La linguetta **Output** fornisce informazioni sull'avanzamento delle operazioni eseguite dal modulo. Quando si clicca sul pulsante **Esegui**, viene portata in primo piano la linguetta **Output** nella quale vengono visualizzate le informazioni sul processo in corso. Se l'operazione va a buon fine, si vedrà il messaggio *Operazione conclusa con successo*.

Manuale

La linguetta **Manual** mostra la pagina di aiuto in formato HTML del modulo GRASS scelto. Qui può essere controllata la disponibilità di ulteriori parametri o ottenere una conoscenza più approfondita sulle operazioni che il modulo può eseguire. Alla fine di ogni pagina di manuale vi sono ulteriori collegamenti al *Main Help index*, al *Thematic index* o al *Full index*. Questi links forniscono le stesse informazioni che si avrebbero usando il modulo `g.manual`.

Tip 38 MOSTRARE I RISULTATI IMMEDIATAMENTE

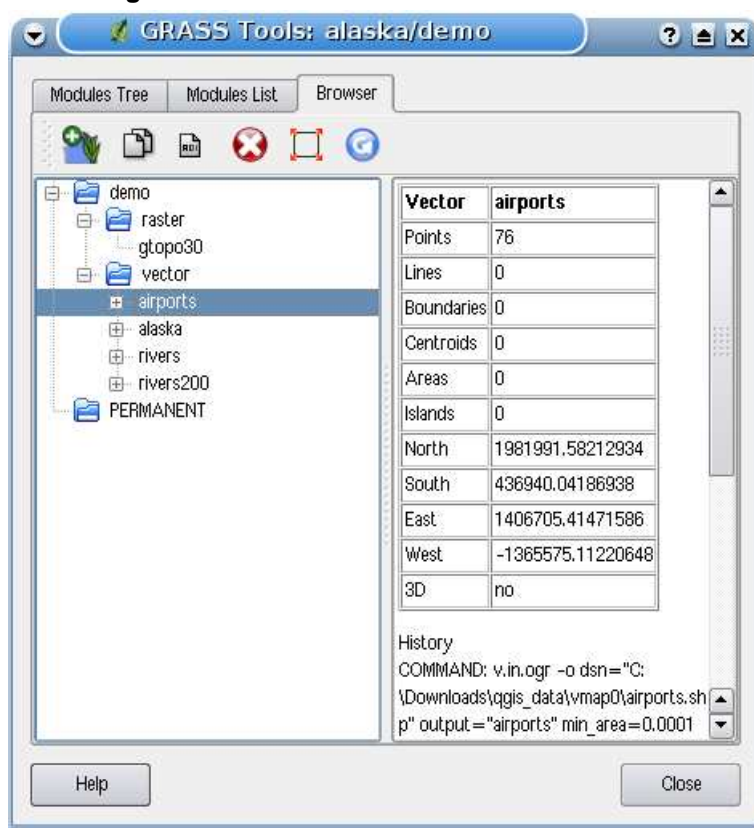
Se si desidera visualizzare il risultato dell'analisi immediatamente nella vista mappa, è possibile cliccare sul pulsante 'Visualizza Output' nella porzione inferiore della linguetta.

9.9.2 Lavorare con il browser delle LOCATION GRASS




Un'altra utile funzione tra quelle presenti negli strumenti GRASS è il browser delle LOCATION. In Figura 28 è possibile vedere un esempio che mostra la LOCATION impostata e i relativi MAPSETs.




Nella parte sinistra della finestra del browser si può navigare attraverso tutti i MAPSETs contenuti nella LOCATION impostata. La porzione di destra mostra invece alcuni metadati del raster o del vettoriale selezionato, come la risoluzione, l'estensione spaziale, la fonte del dato, il percorso alla tabella attributi associata per i dati vettoriali e lo storico comandi che ha generato quel dato.



Figura 28: Browser delle LOCATION GRASS 



La barra strumenti nella linguetta **Browser** tab offre i seguenti strumenti per la gestione della LOCATION selezionata:

-  Aggiungi la mappa selezionata all'area di lavoro
-  Copia la mappa selezionata
-  Rinomina la mappa selezionata

-  Elimina la mappa selezionata
-  Imposta la regione corrente con la mappa selezionata
-  Aggiorna

Gli strumenti  Rinomina la mappa selezionata e  Elimina la mappa selezionata funzionano solo su mappe contenute nel MAPSET attivo. Tutti gli altri strumenti funzionano anche con livelli raster e vettoriali di altri MAPSET.

9.9.3 Personalizzazione degli strumenti GRASS

Praticamente tutti i moduli GRASS possono essere aggiunti agli strumenti GRASS. Per incorporare i file XML di configurazione dei moduli è fornita un'interfaccia XML nella quale definire l'aspetto del modulo e i parametri da visualizzare nello strumento.

Un esempio di file XML che genera il module `v.buffer` (`v.buffer.qgm`) ha il seguente aspetto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE qgisgrassmodule SYSTEM "http://mrcc.com/qgisgrassmodule.dtd">

<qgisgrassmodule label="Vector buffer" module="v.buffer">
  <option key="input" typeoption="type" layeroption="layer" />
  <option key="buffer"/>
  <option key="output" />
</qgisgrassmodule>
```

Il parser legge questa definizione e crea una nuova linguetta negli strumenti quando si seleziona il modulo. Informazioni più dettagliate su come aggiungere moduli, cambiare i gruppi di moduli ecc. sono reperibile sul Wiki di QGIS all'indirizzo


http://wiki.qgis.org/qgiswiki/Adding_New_Tools_to_the_GRASS_Toolbox.

10 Compositore di stampe

Il compositore di stampe fornisce funzionalità per la creazione di layout di stampa che saranno continuamente migliorate. Consente di aggiungere elementi al layout come la vista mappa, la legenda, la barra di scala, immagini esterne e campi testuali. È possibile cambiare le dimensioni, raggruppare e spostare ogni elemento e regolarne le proprietà per creare il layout desiderato. Il risultato può essere stampato (anche come Postscript e PDF), esportato come immagine o come disegno vettoriale in formato SVG.⁷ Si veda l'elenco degli strumenti nella tabella 5:

Tabella 5: Strumenti del compositore di stampe

Icona	Scopo	Icona	Scopo
	Esporta come immagine		Esporta come SVG
	Stampa o esporta come PDF o Postscript		Zoomma all'estensione massima
	Ingrandisci		Rimpicciolisci
	Aggiorna la vista		Aggiungi una nuova vista mappa da QGIS
	Aggiungi immagine al layout di stampa		Aggiungi caselle di testo al layout di stampa
	Aggiungi una nuova legenda al layout di stampa		Aggiungi una barra di scala al layout di stampa
	Seleziona/sposta oggetti del layout di stampa		Sposta centro della vista mappa
	Raggruppa oggetti del layout		Separa oggetti del layout
	Alza di livello l'oggetto selezionato nel layout		Abbassa di livello l'oggetto selezionato nel layout
	Porta l'oggetto selezionato in primo piano		Porta l'oggetto selezionato sullo sfondo

Per accedere al compositore di stampe, cliccare sul pulsante  **Print** nella barra strumenti o scegliere la voce di menù **File** > **Compositore stampe**.

⁷L'esportazione in SVG è supportata, ma non funziona correttamente con alcune recenti versioni di QT4. È necessario fare delle prove e dei controlli individuali sul proprio sistema

10.1 Uso del compositore di stampe


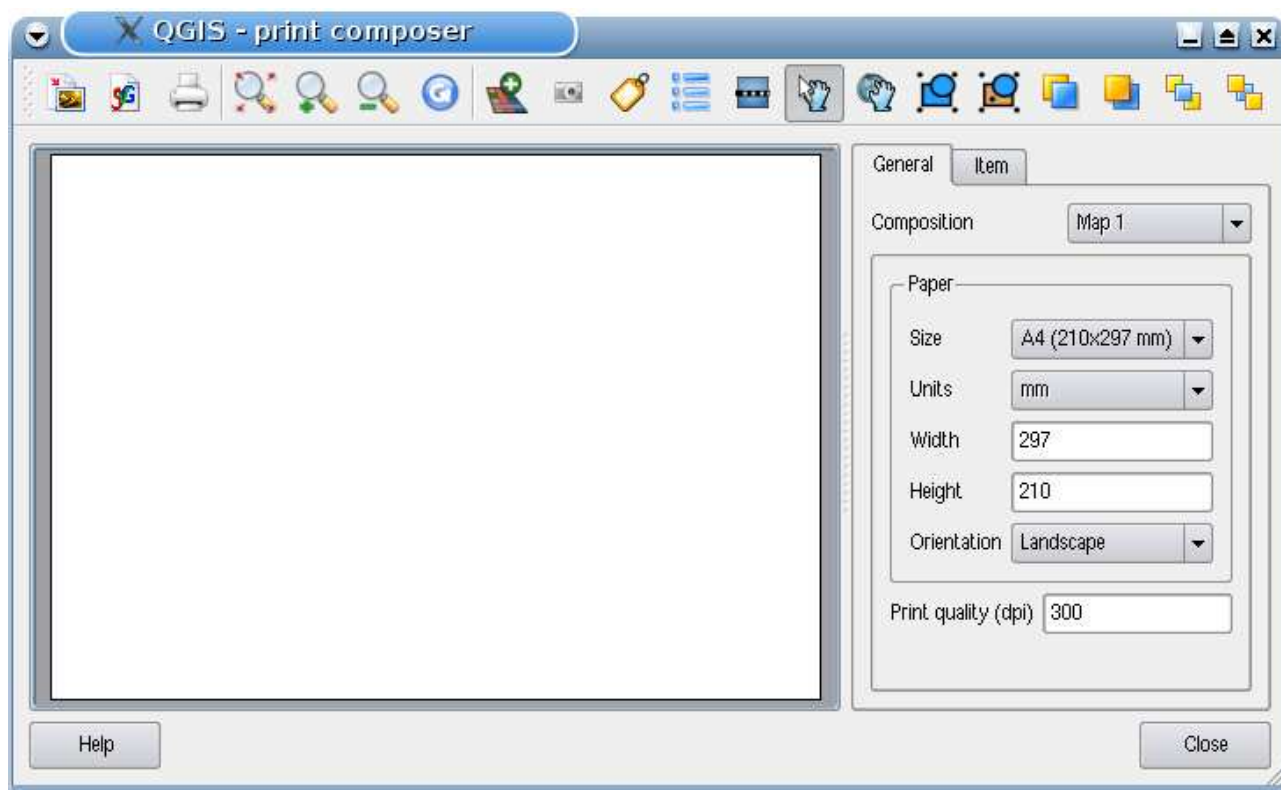

Prima di iniziare a lavorare con il compositore di stampe, è necessario caricare alcuni livelli raster e vettoriali nella vista mappa di QGIS e regolarne le proprietà secondo le proprie esigenze. Una volta effettuate tutte le impostazioni e applicata la simbologia cliccare sul pulsante  **Compositore stampe**.

Figura 29: Compositore di stampe 



Aprendo il compositore di stampe viene visualizzato un foglio bianco al quale aggiungere vista mappa, legenda, barra di scala, immagini e testo. La Figura 29 mostra la vista iniziale del compositore di stampe prima dell'aggiunta di un qualunque elemento. Nel compositore di stampe ci sono due linguette:

- La linguetta **Generale** consente di impostare la dimensione del foglio, l'orientamento e la qualità di stampa del file in uscita in dpi.
- La linguetta **Oggetto** mostra le proprietà dell'elemento selezionato nel layout di stampa. Cliccare sull'icona  **Seleziona/Sposta oggetto** per selezionare un elemento (ad es. legenda,

barra di scala o etichetta testuale) nel layout. Cliccare dunque sulla linguetta **Oggetto** e personalizzare le impostazioni dell'elemento selezionato.

Possono essere aggiunti diversi elementi al compositore ed è anche possibile avere più di una vista mappa o legenda o barra di scala nel layout di stampa. Ogni elemento ha le sue proprietà e, nel caso delle viste mappa, la propria estensione.

10.1.1 Aggiungere una vista mappa al layout nel compositore di stampe



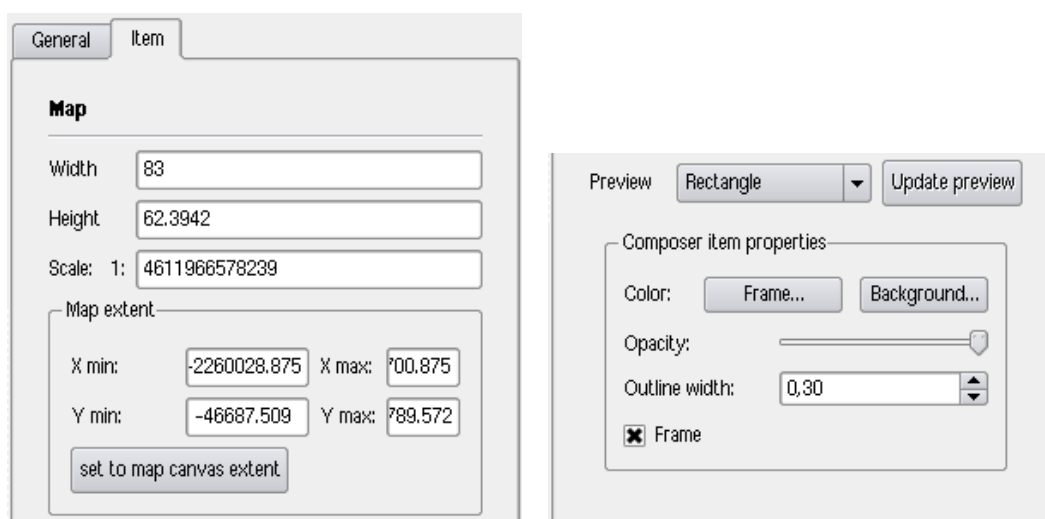

Per aggiungere una vista mappa, cliccare sul pulsante  **Aggiungi nuova mappa** nella barra strumenti del compositore di stampe e trascinare il mouse per definire un rettangolo nell'area occupata dal foglio tenendo premuto il tasto sinistro per aggiungere la vista mappa. Verrà visualizzato un riquadro vuoto con la scritta *La mappa verrà stampata qui*. Per visualizzare effettivamente la mappa, selezionare **Anteprima** **Cache**  nella linguetta **Oggetto**.


Figura 30: Contenuti della linguetta Oggetto nel compositore di stampe 




(a) Sezione per definizione larghezza, altezza ed estensione

(b) Sezione proprietà


È possibile ridimensionare la mappa in un momento successivo cliccando sul pulsante  **Seleziona/Sposta oggetto**, selezionando un elemento e trascinando una delle maniglie blu agli angoli della mappa. Con la mappa selezionata, è possibile regolare ulteriori proprietà della vista mappa nella linguetta **Oggetto**. Ridimensionare la mappa specificando la larghezza e l'altezza o la scala. Definire l'estensione della mappa usando i valori massimi/minimi per gli assi X e Y o cliccando sul pulsante **Imposta all'estensione della mappa**. Aggiornare l'anteprima della mappa e selezionare

se si desidera avere un'anteprima della mappa dalla cache o un rettangolo vuoto con il messaggio *La mappa verrà stampata qui*. Definire i colori e lo spessore del contorno della cornice dell'elemento, impostare un colore di sfondo e l'opacità del foglio mappa. È anche possibile scegliere se visualizzare o meno una cornice attorno agli elementi con la casella di controllo  **Cornice** (si veda la Figura 30). Se si modifica la vista mappa nel programma zoommando o spostando la vista o cambiando le proprietà dei vettoriali o dei raster, è possibile aggiornare la vista nel compositore di stampe cliccando sul pulsante **Aggiorna la vista** nella linguetta **Oggetto** (si veda la Figura 30).

Per spostare l'area visualizzata nella vista mappa aggiunta nel compositore, cliccare sul pulsante  **Sposta contenuto oggetto** e spostare la vista nella cornice della vista mappa trascinando con il tasto sinistro del mouse premuto.

Tip 39 SALVARE UN LAYOUT DI STAMPA


Se si desidera salvare lo stato attuale di una sessione del compositore di stampe, cliccare su **File** >


 **Salva Progetto con nome** per salvare lo stato dello spazio di lavoro incluso quello della sessione attuale del compositore di stampe. In futuro sarà possibile salvare separatamente lo stato del compositore di stampa.

10.1.2 Aggiunta di altri elementi al compositore di stampa


Oltre ad aggiungere una vista mappa al layout di stampa, è anche possibile aggiungere, spostare e personalizzare legenda, barra di scala, immagini e caselle testuali.

Etichette ed immagini

Per aggiungere una etichetta o un'immagine, cliccare sugli strumenti 

Aggiungi nuova etichetta o  **Aggiungi immagine** e posizionare gli elementi con il tasto sinistro del mouse al layout di stampa.

Legenda e barra di scala

Per aggiungere una legenda o una barra di scala, cliccare sul pulsante 


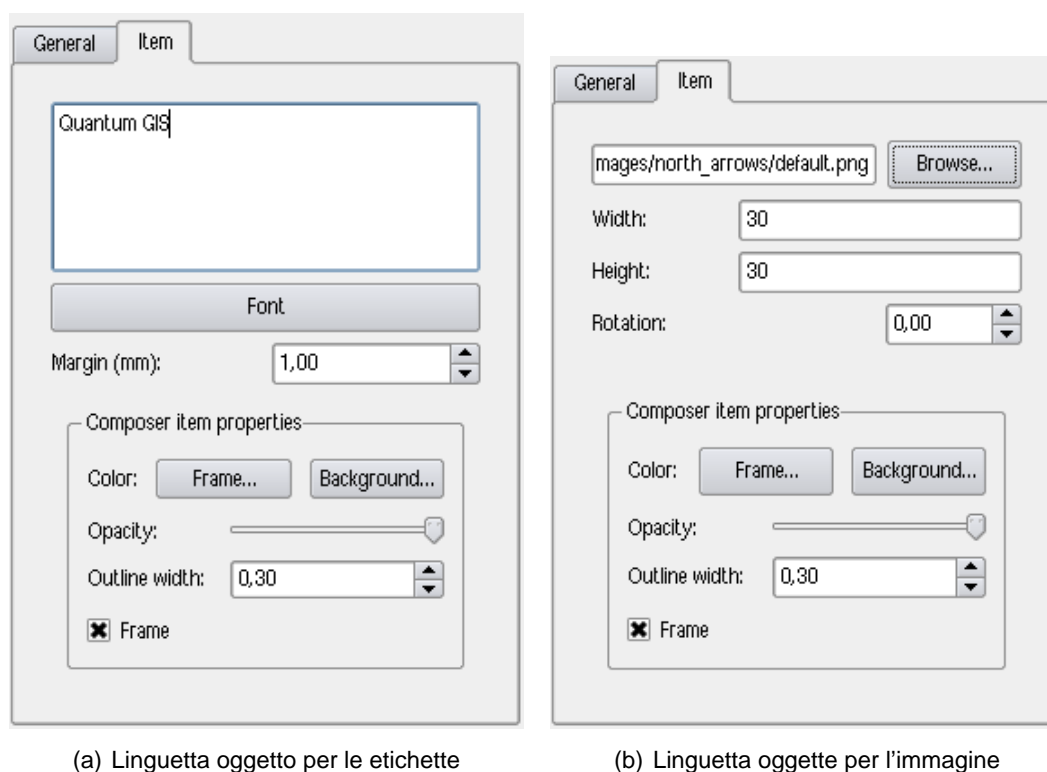
Aggiungi nuova legenda vettoriale o  **Aggiungi nuova barra di scala** e posizionare l'elemento con il tasto sinistro del mouse nel layout di stampa.

Figura 31: Personalizzazione di immagini ed etichette nel layout di stampa 🐧

10.1.3 Strumenti per l'esplorazione del layout di stampa

Per l'esplorazione del layout nel compositore di stampe sono forniti 4 strumenti:





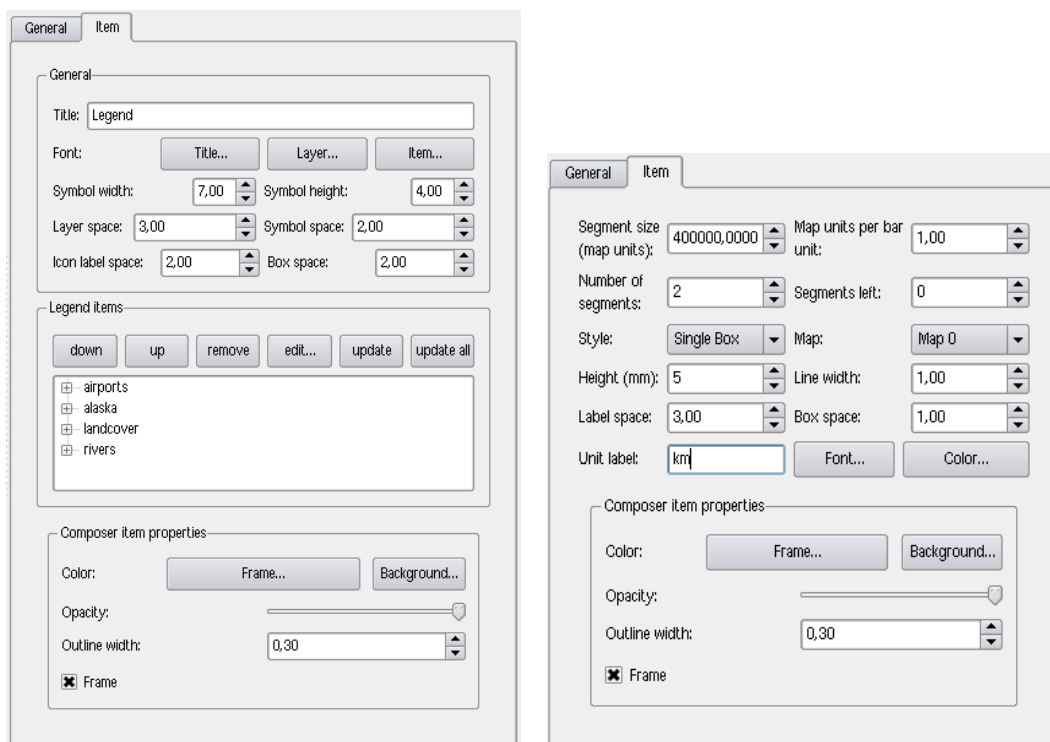
-  **Ingrandisci** ,
-  **Rimpicciolisci** ,
-  **Vista ad estensione massima** e
-  **Aggiorna la vista** , che serve nel caso in cui la vista nel layout non rispecchi quanto presente nella vista mappa in QGIS.

Figura 32: Personalizzazione della legenda e della barra di scala nel compositore di stampe 🐧

(a) Linguetta oggetto per la legenda

(b) Linguetta oggetto per la barra di scala

10.1.4 Creazione di file in uscita

La Figura 33 mostra il compositore di stampe con un layout di stampa completo di ognuna degli elementi precedentemente descritti.

Il compositore di stampe consente di creare diversi formati in uscita ed è possibile definirne la risoluzione (qualità di stampa) e il formato pagina:




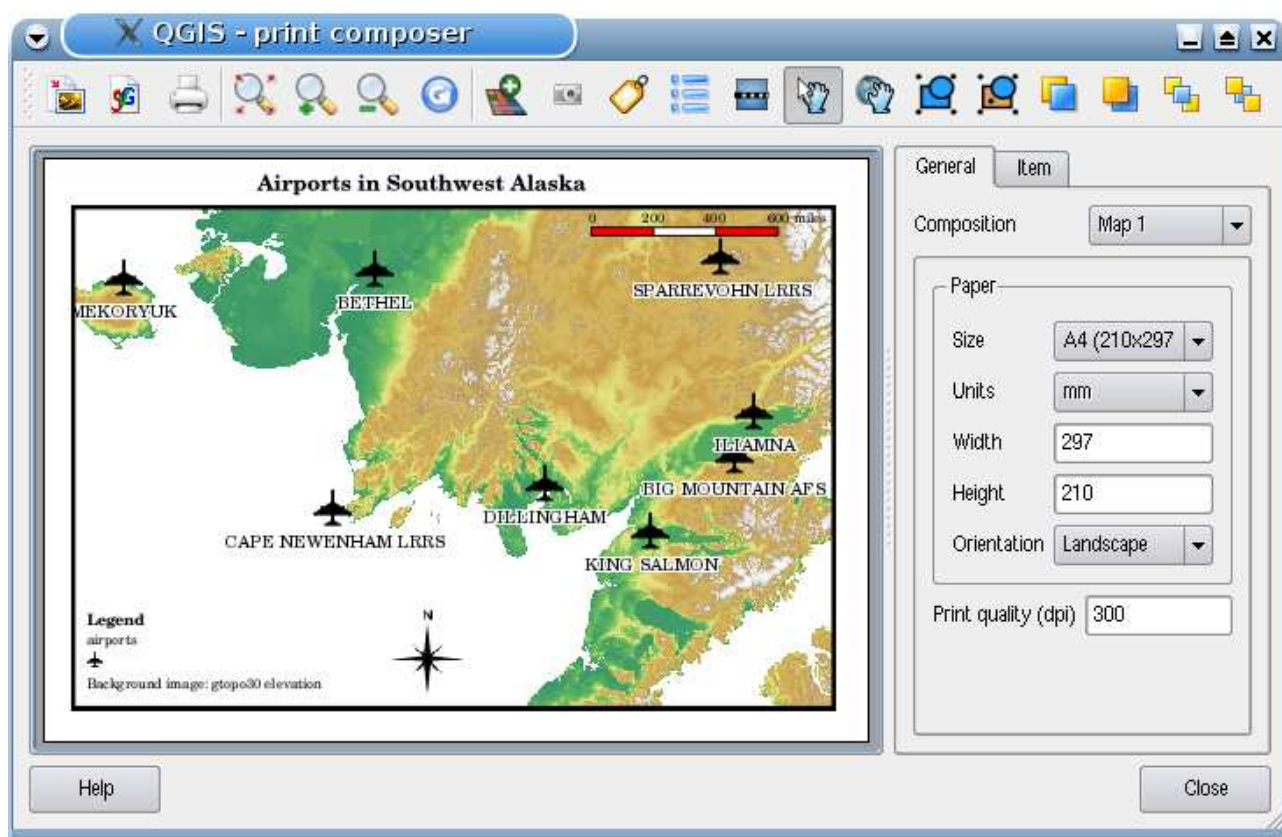
- L'icona  **Stampa** consente di stampare il layout su una stampante collegata o su un file PDF o Postscript in funzione del driver stampante installato.
- L'icona  **Esporta come immagine** esporta il layout in diversi formati immagine come PNG, BPM, TIF, JPG, ...
- L'icona  **Esporta come SVG** salva il layout di stampa in formato SVG (Scalable Vector Graphic). **Nota:** Attualmente l'uscita in SVG è ad un livello molto iniziale. Il problema non sta in QGIS, ma nella sottostante libreria Qt. Ci si augura che questo problema venga risolto nelle prossime versioni della libreria.

Figura 33: Compositore di stampe con layout completo di vista mappa, legenda, barra di scala e testo 



11 Plugins di QGIS

Quantum GIS è stato progettato con un'architettura a plugins. Ciò permette di aggiungere nuove caratteristiche e funzioni all'applicazione. Molte delle caratteristiche in QGIS sono in effetti implementate come plugins di base **Core** o contribuiti dagli utenti **Esterni**.

- **Plugins Core** sono mantenuti dal team di sviluppo di QGIS e fanno parte automaticamente di ogni distribuzione di QGIS. Sono scritti in uno dei due linguaggi: C++ or Python. Ulteriori informazioni riguardanti i plugins core sono disponibili nella sezione [12](#).
- **Plugins esterni** sono attualmente tutti scritti in Python. Sono immagazzinati in archivi dei pacchetti esterni e mantenuti dai singoli autori. Possono essere aggiunti a QGIS usando il plugin core chiamato *Plugin Installer*. Ulteriori informazioni riguardanti i plugins esterni sono disponibili nella sezione [13](#).

11.1 Gestire i Plugins

La gestione dei plugins consiste nella loro abilitazione o disabilitazione usando il plugin *Plugin Manager*. I plugin esterni devono prima essere installati usando il plugin *Plugin Installer*.

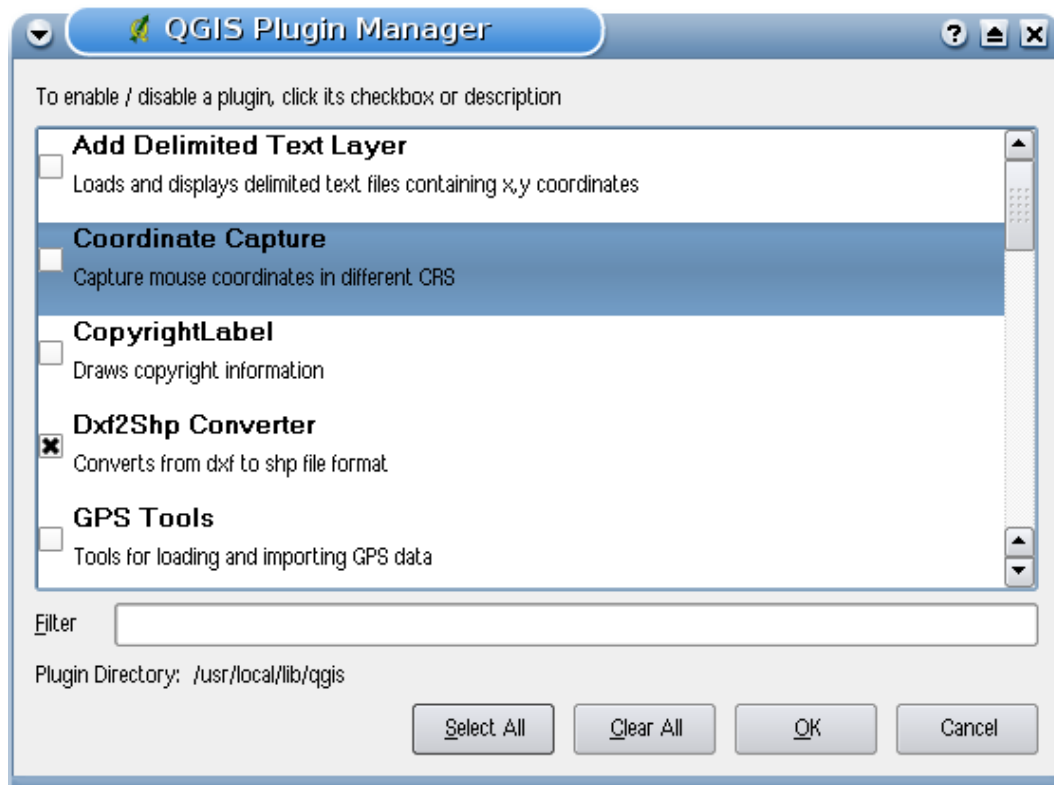
11.1.1 Abilitare un Plugin Core

L'abilitazione di un Plugin Core si ottiene dal menù principale **Plugins** > **Manage Plugins...**.

Il Gestore dei Plugins (*Plugin Manager*) elenca tutti i plugins disponibili e il loro stato (abilitato o disabilitato). Tutti i plugins disponibili comprende sia tutti i Core sia tutti gli Esterni che sono stati aggiunti usando il plugin *Plugin Installer* (vedere la sezione [13](#)). la figura [34](#) mostra la finestra di dialogo del Gestore dei Plugin. I plugin abilitati vengono 'ricordati' quando si esce dall'applicazione e riabilitati la prossima volta che QGIS viene lanciato.

Tip 40 IL CRASH DEI PLUGINS

Se vi accorgete che QGIS va in crash all'avvio, la colpa potrebbe essere di un plugin. Potete bloccare il caricamento di tutti i plugins attraverso l'editing del suo file di settaggio (vedere [4.7](#) per il settaggio). Individuate il settaggio dei plugins e cambiate i valori di tutti i plugins su 'false' in modo da impedire il loro caricamento. 🐧 Per es *Delimited text*, la modifica da effettuare sul file `$HOME/.config/QuantumGIS/qgis.conf` di Linux dovrebbe apparire così: `Add Delimited Text Layer=false`. Farlo per ogni plugin nella sezione *Plugins*. Quindi avviare QGIS ed aggiungere i plugins uno alla volta dal Gestore dei Plugins per determinare quale stia causando il problema.

Figura 34: Plugin Manager 

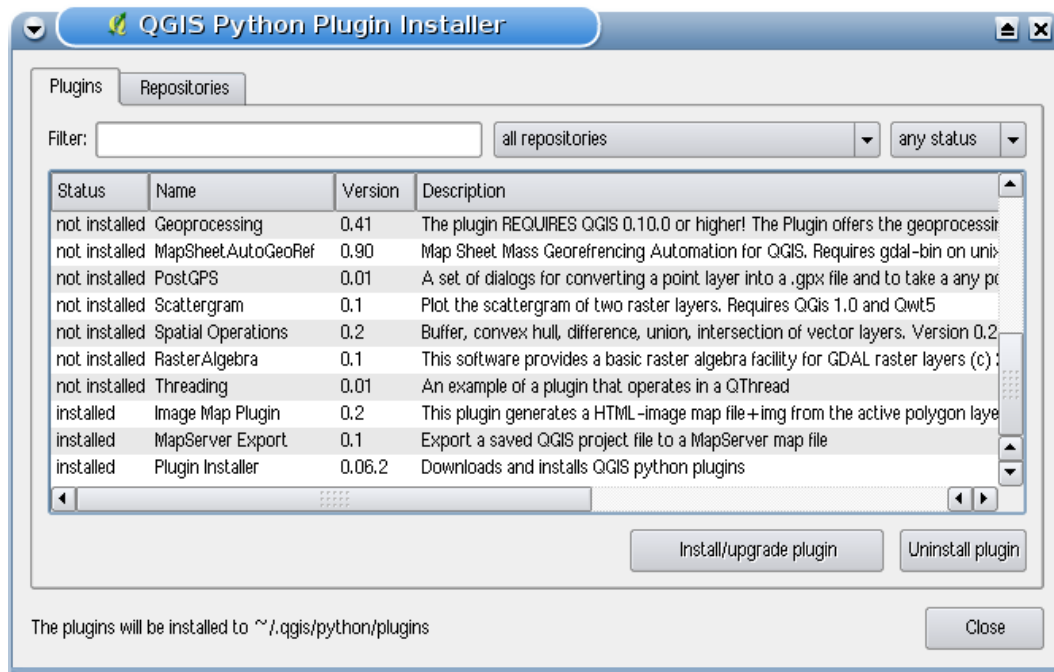

11.1.2 Caricamento di un plugin esterno di QGIS

Per integrare i Plugins Esterni in QGIS si deve prima caricare il plugin Plugin Installer come descritto nella sezione 11.1.1. Successivamente si possono caricare i Plugins Esterni Python in due passi:

1. Scaricare il Plugin Esterno dall'archivio dei pacchetti usando Plugin Installer (Sezione 11.1.3). Il nuovo Plugin Esterno verrà aggiunto all'alista dei Plugins disponibili nel Plugin Manager.
2. Caricare il Plugin usando il Plugin Manager.

11.1.3 Uso dell'Installatore di Plugins Python


Per scaricare e installare un Plugin Python Esterno, selezionare il menu **Plugins** > **Fetch Python Plugins...**. Apparirà la finestra Plugin Installer (figure 35) con il tab **Plugins**,

Figura 35: Installazione di Plugin Esterni Python 

contenente la lista sia di tutti i Plugins Python disponibili negli archivi remoti sia di quelli installati. Ogni plugin può essere:

- **not installed** - significa che il plugin è disponibile nell'archivio remoto, ma non ancora installato. Per installarlo, selezionarlo dalla lista e premere il pulsante **Install plugin**.
- **new** - idem come sopra, ma il plugin è visto per la prima volta.
- **installed** - il plugin è installato. Se è anche disponibile in qualsiasi archivio remoto il pulsante **Reinstall plugin** è abilitato. Ma la versione disponibile in remoto è più vecchia di quella installata, appare invece il pulsante **Downgrade plugin**.
- **upgradeable** - il plugin è installato, ma è disponibile una versione più recente. Il pulsante **Upgrade plugin** è abilitato.
- **invalid** - il plugin è installato, ma è inutilizzabile. La ragione è spiegata nella descrizione del plugin.

Plugins tab

Per installare un plugin, selezionarlo dalla lista e premere il pulsante **Install plugin**. Il plugin è installato nella sua propria directory, per esempio per  sotto `$HOME/.qgis/python/plugins` ed è visibile solo per l'utente che lo ha installato. Vedere una lista di altre subdirectories usate per i plugins spe-

cifiche per ogni sistema operativo nella Sezione 15.3. Se l'installazione va a buon fine, compare un messaggio di conferma. A questo punto andare su **Plugins** > **Manage Plugins...** e caricare il plugin istallato.

Se l'installazione fallisce, ne viene indicata la ragione. I problemi più frequenti sono correlati a errori di connessione e moduli Python mancanti. Nel primo caso bisognerà attendere alcuni minuti o anche ore, nel secondo è necessario installare ne sistema operativo i moduli mancanti prima di usare il plugin. 🐧 In Linux, i moduli più richiesti dovrebbero essere disponibili nel gestore dei pacchetti. 🌐. Per istruzioni sull'installazione in Windows, visitare la homepage del modulo. Se si usa un proxy, può essere necessario configurarlo nel menu **Settings** > **Options** nella tab **Proxy**.

Il pulsante **Uninstall plugin** è abilitato solo se il plugin selezionato è istallato e non è un plugin Core. Da notare che se si è istallato un aggiornamento (update) di un Core plugin, si può sempre disinstallare tale aggiornamento con il pulsante **Uninstall plugin** e ritornare alla versione contenuta nel pacchetto di installazione di Quantum GIS. Questa non può essere disinstallata.

Repositories tab

Il secondo tab **Repositories** contiene una lista di archivi di plugins disponibili per il Plugin Installer. Per impostazioni di default, viene usato solamente l'archivio ufficiale di QGIS (QGIS Official Repository). Si possono aggiungere archivi contribuiti dagli utenti, incluso l'archivio centrale 'QGIS Contributed Repository' ed alcuni altri archivi usando il pulsante **Add 3rd party repositories**. Questi archivi contengono un gran numero di più o meno utili plugins ma bisogna tener presente che non sono mantenuti dal Team di Sviluppo di QGIS, che quindi non se ne assume la responsabilità. Si può anche gestire la lista dei plugins manualmente, cioè aggiungere, rimuovere o editare le singole voci. E' possibile disabilitare temporaneamente un particolare archivio usando il pulsante **Edit...**.

La casella di scelta ☒ **Check for updates on startup** fa sì che QGIS cerchi gli aggiornamenti e le news dei plugins. Se la funzione è selezionata tutti gli archivi elencati e abilitati nella tab **Repositories** sono verificati ogni volta che il programma viene aperto. Se è disponibile un nuovo plugin o l'aggiornamento di uno di quelli già istallati, compare una messaggio di notifica selezionabile nella Barra di Stato. Se la casella di scelta non è selezionata, la ricerca di nuovi plugins e di aggiornamenti viene effettuata solamente quando si lancia il Plugin Installer dal menu.

Nel caso di problemi con la connessione internet, può esser visibile durante l'intera sessione di QGIS un indicatore *Looking for new plugins...* nella Barra di Stato e può causare il crash del programma alla chiusura. In questo caso deselectare la casella di scelta.

11.2 Data Providers


















I Data Providers sono plugins speciali che danno accesso ad un archivio di dati. Per default, QGIS supporta i livelli (layers) PostGIS e gli archivi di dati su disco supportati dalla libreria GDAL/OGR (Appendice [A.1](#)). Un plugin Data Provider estende l'abilità di QGIS di utilizzare altre sorgenti di dati.

I plugins Data Provider sono registrati automaticamente all'avvio di QGIS. Essi non sono gestiti dal Plugin Manager, ma usati dietro le quinte quando un tipo di dati viene aggiunto come livelli in QGIS.

12 Using QGIS Core Plugins

QGIS al momento contiene 17 plugins di base che possono essere caricati tramite il Plugin Manager. La tabella 6 mostra una lista dei plugins di base, insieme ad una descrizione del loro scopo e l'icona che viene mostrata nella barra dei menù.⁸

Tabella 6: QGIS Core Plugins

Icon	Plugin	Description
	Aggiungi layer testo delimitato	Carica e mostra files di testo delimitato contenenti coordinate x,y
	Cattura Coordinate	Cattura le coordinate del mouse usando SRS diverso
	Copyright Label	Draws a copyright label with information
	DXF2Shape Converter	Converts from DXF to SHP file format
	GPS Tools	Tools for loading and importing GPS data
	GRASS	Activates the mighty GRASS Toolbox
	Georeferencer	Adding projection info to Rasterfiles
	Gaticule Creator	Create a latitude/longitude grid and save as a shapefile
	Interpolation plugin	Interpolation on base of vertices of a vector layer
	MapServer Export Plugin	Export a saved QGIS project file to a MapServer map file
	North Arrow	Displays a north arrow overlayed onto the map
	OGR Layer Converter	Translate vector layers between OGR supported formats
	Plugin Installer	Downloads and installs QGIS python plugins
	SPIT	Shapefile to PostgreSQL/PostGIS Import Tool
	Qucik Print	Quickly print a map with minimal effort
	Scalebar	Draws a scale bar
	WFS	Load and display WFS layer

Tip 41 PLUGINS SETTINGS SAVED TO PROJECT

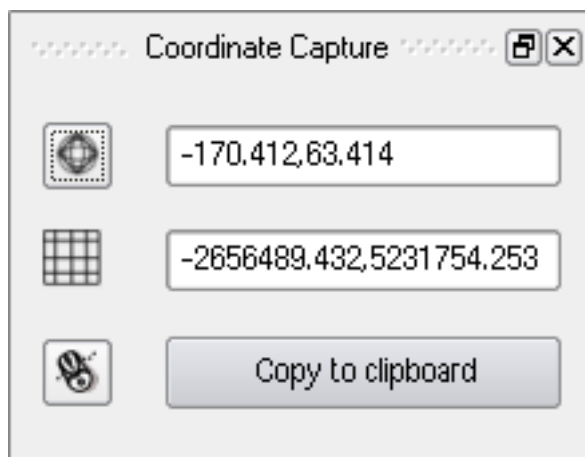
When you save a .qgs project, any changes you have made to NorthArrow, ScaleBar and Copyright plugins will be saved in the project and restored next time you load the project.






⁸Il Plugin MapServer Export Plugin ed il Plugin Installer Plugin sono Plugins Python esterni, ma sono parte delle fonti QGIS e automaticamente caricati e selezionabili nel QGIS Plugin Manager.

12.1 Plugin di Cattura delle Coordinate

Il plugin di Cattura delle Coordinate è facile da usare e offre la capacità di mostrare sulla mappa le coordinate per due Sistemi di Riferimento tramite Coordinate (Coordinate Reference Systems - CRS). Basta cliccare un certo punto e copiare le coordinate negli appunti o usare la funzionalità di tracciatura del mouse.

Figura 36: Plugin di cattura delle Coordinate 



1. Lanciare QGIS, selezionare  **Proprietà Progetto** dal menu **Impostazioni** e scegliere la tab **Sistema Riferimento Spaziale (SRS)**. In alternativa, cliccare l'icona  **Stato SRS** nell'angolo in basso a destra della barra di stato.
2. Selezionare ☒ **Abilita la modifica immediata SRS Information** **NAD27/Alaska Albers** con EPSG 2964 (vedi anche la Sezione 8).
3. Caricare il layer vettoriale `alaska.shp` dal set di dati campione QGIS.
4. Caricare il plugin di Cattura di Coordinate nel Gestore dei Plugins (vedere la Sezione 11.1.1) e selezionare l'icona  **Cattura delle Coordinate**. Compare una finestra di dialogo della Cattura di Coordinate come mostrato in Figura 36.
5. Selezionare l'icona  **Clicca per selezionare il SRS da usare durante la visualizzazione delle coordinate** e espandere la lista Sistemi di Coordinate Geografiche e scegliere WGS84 (EPSG 4326).
6. Ora si può cliccare ovunque sulla mappa e il plugin mostrerà le coordinate NAD27/Alaska Albers e WGS84 per il punto selezionato come mostrato nella Figura 36.
7. Per abilitare la tracciatura mouse delle coordinate selezionare l'icona  **tracciatura mouse**.

8. Si possono anche copiare le coordinate selezionate negli appunti.

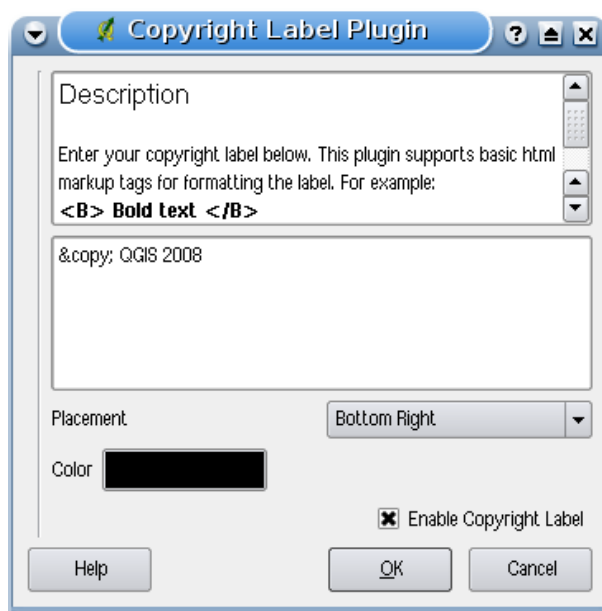
12.2 Plugins Decorativi



I plugins decorativi includono il plugin Etichetta di Copyright, il plugin Freccia Nord ed il plugin Barra di Scala. Sono usati per 'decorare' la mappa aggiungendo elementi cartografici.

12.2.1 Plugin etichetta di Copyright

Il nome del plugin è un po' fuorviante - si può aggiungere qualsiasi testo alla mappa.

Figura 37: Plugin Etichetta di Copyright 



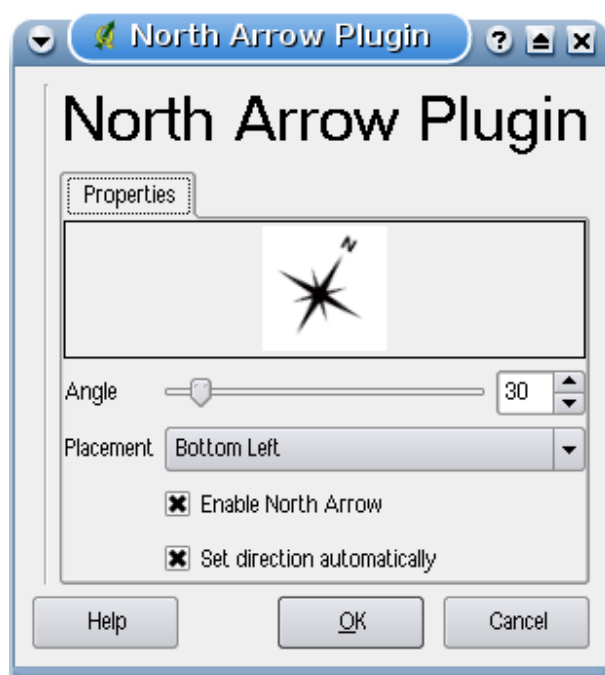
1. Assicurarsi che il plugin sia caricato
2. Selezionare su **Plugins** > **Decorazioni** > **Etichetta di Copyright** o selezionare il pulsante  **Etichetta di Copyright** dalla barra degli Strumenti.
3. Digitare il testo che si vuole aggiungere alla mappa. Si può usare HTML come mostrato nell'esempio
4. Scegliere il posizionamento dell'etichetta dal menu a tendina **Posizione** **In basso a destra** 
5. Assicurarsi che la casella ☒ **Abilita etichetta di Copyright** sia selezionata
6. Premere **OK**

Nell'esempio sopra, la prima linea è in grassetto, la seconda (creata usando
) contiene un simbolo di copyright, seguito dal nome della nostra compagnia in corsivo.

12.2.2 Plugin Freccia Nord

Il plugin Freccia Nord aggiunge alla mappa una semplice freccia indicante il nord. Al momento c'è un solo stile disponibile. Si può aggiustare manualmente l'angolo della freccia o lasciare che QGIS indichi automaticamente la direzione. Se si sceglie di lasciar fare a QGIS, il programma fa la sua migliore congettura circa come la freccia dovrebbe essere orientata. Per il posizionamento della freccia si hanno quattro possibilità, corrispondenti ai quattro angoli della mappa.

Figura 38: Plugin Freccia Nord 🐧



12.2.3 Plugin Barra di Scala

Il plugin Barra di scala aggiunge una semplice barra di scala alla mappa. Si può controllare lo stile ed il posizionamento, come anche l'etichettatura della scala.

QGIS supporta solamente di mostrare la scala nella stessa unità di misura della mappa. Cioè, se l'unità di misura dei layer è il metro, non si può creare una mappa in piedi. Allo stesso modo, se si usano gradi decimali, non si può creare una barra di scala che mostri le distanze in metri.

Per aggiungere una barra di scala:



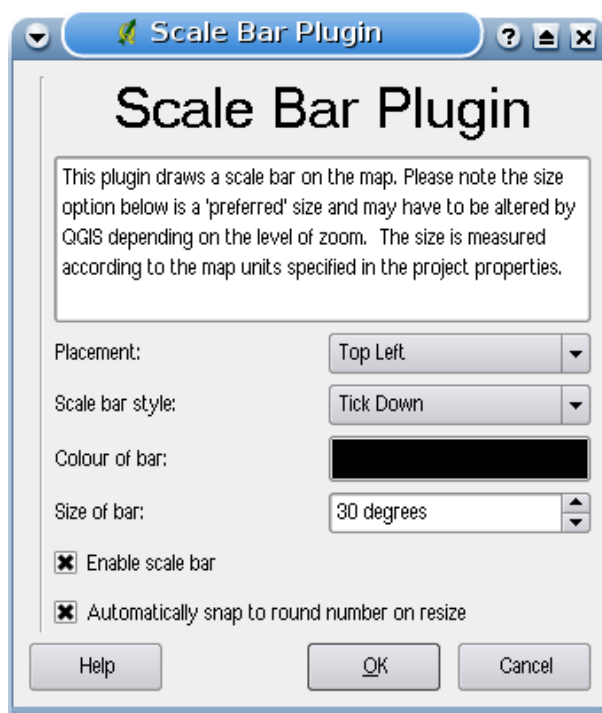
1. Selezionare **Plugins** > **Decorazioni** > **Barra di Scala** o premere il pulsante  **Barra di Scala** dalla barra degli Strumenti.
2. Scegliere il posizionamento dal menu a tendina **Posizionamento** **In basso a sinistra** ▼
3. Scegliere lo stile dalla lista **Stile della Barra di Scala** **Porta in basso** ▼
4. Scegliere il colore della barra di scala **Colore della barra**  o usare il colore nero di default
5. Impostare la dimensione della barra e la sua etichetta **Dimensione della barra** **30 gradi** ▲▼
6. Assicurarsi che la casella ☒ **Abilita barra di scala** sia selezionata
7. Se si vuole, scegliere di arrotondare automaticamente il numero quando la mappa viene ridimensionata ☒ **Arrotonda automaticamente il numero durante il ridimensionamento**
8. Premere **OK**

Figura 39: Plugin Barra di Scala 



12.3 Plugin Testo Delimitato

Il plugin Testo Delimitato permette di caricare un file di testo delimitato come layer in QGIS.

Requisiti

Per vedere un file di testo delimitato come layer, il testo deve contenere:

1. Una riga capopagina delimitata di nomi di campo. Questa deve essere la prima riga del file di testo.
2. La riga capopagina deve contenere un campo X ed uno Y. Questi campi possono avere qualsiasi nome.
3. Le coordinate x e y devono essere specificate da numeri. Il sistema di coordinate non è importante.

Come esempio di un file di testo valido importiamo il file di dati dell'elevazione di un punto `elevp.csv` presente nel dataset di esempi di QGIS (Vedere la Sezione [3.2](#)):


```
X;Y;ELEV
-300120;7689960;13
-654360;7562040;52
1640;7512840;3
[...]
```



Alcune note circa il file di testo:

1. Il file di testo usato come esempio usa ; come delimitatore. Qualsiasi carattere può essere usato per delimitare i campi.
2. La prima riga è la riga capopagina. Essa contiene i campi X, Y e ELEV.
3. Nessun tipo di virgoletta (") possono essere usati per delimitare i campi di testo.
4. Le coordinate x sono contenuto nel campo X.
5. Le coordinate y sono contenuto nel campo Y.

Uso del Plugin

Per usare il plugin si deve aver QGIS in funzione ed usare il Gestore dei Plugins per caricarlo:

Avviare QGIS, quindi aprire il Gestore dei Plugins scegliendo **Plugins** >  **Gestione Plugins**. Il Gestore dei Plugins mostra una lista dei plugins disponibili. Quelli già caricati hanno una crocetta nella casella alla sinistra del loro nome. Cliccare sulla casella alla sinistra del plugin

 Aggiungi layer di testo delimitato e premere il pulsante  per caricarlo come descritto nella Sezione 11.1.



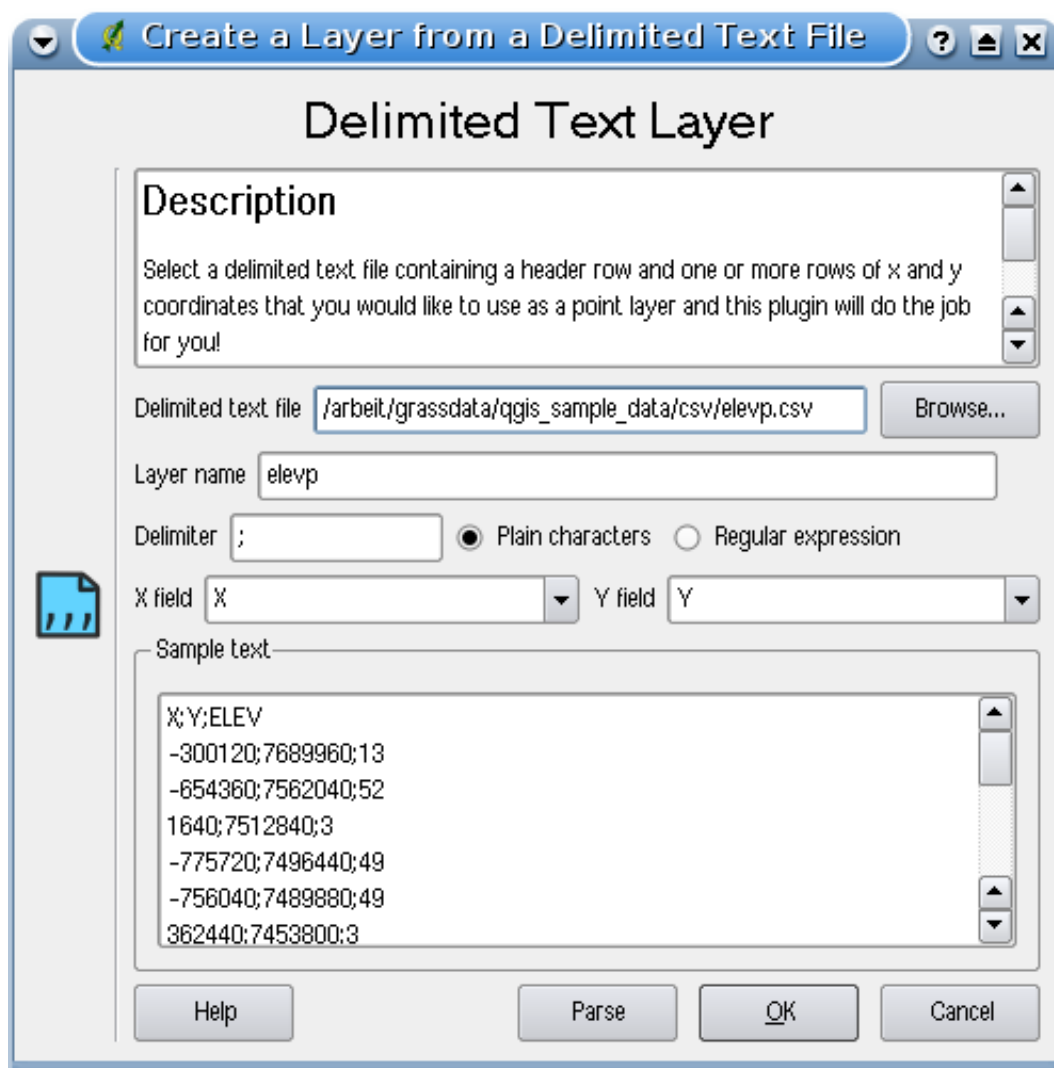


Cliccare la nuova barra degli strumenti   Aggiungi layer di testo delimitato per aprire la finestra di dialogo Aggiungi layer di testo delimitato come mostrato in Figura 40.

Figura 40: finestra di dialogo Aggiungi layer di testo delimitato 



Prima selezionare il file `qgis_sample_data/csv/elevp.csv` per importare premendo il pulsante  **Sfoglia**. Una volta selezionato il file, il plugin cerca di processare il file usando l'ultimo delimitatore utilizzato, in questo caso ;. Per processare per bene il file, è importante selezionare il giusto delimitatore. Per cambiare delimitatore a tab usare `\t` (questa è un'espressione regolare per il carattere tab). Dopo aver cambiato il delimitatore premere  **Processa**.

scegliere i campi X e Y dai menu a tendina e introdurre il nome di un layer `elevp` come mostrato in Figura 40. Per aggiungere il layer alla mappa, premere il pulsante **Aggiungi Layer**. il file di testo delimitato si comporta ora come qualsiasi altro layer di QGIS.




12.4 Plugin Convertitore Dxf2Shp

The plugin Convertitore dxf2shape permette di convertire i dati vettoriali dal formato DXF al formato Shape. E' molto semplice da maneggiare e offre la seguente funzionalità come mostrato in Figura 41.


- **File DXF in Input:** Scrivere il percorso del file DXF da convertire
- **Output file:** Scrivere il nome che si desidera attribuire al file Shape che verrà creato
- **Tipo di file in output:** Specifica il tipo di file in output. Al momento sono supportati i tipi: polilinea, poligono e punto.
- **Esporta le etichette di testo:** Se si seleziona questa casella, viene creato un layer di punti in forma di file Shape aggiuntivo, e la tabella dbf associata conterrà informazioni circa i campi 'TESTO' trovati nel file dxf e le stringhe di testo propriamente dette.

Figura 41: Plugin Convertitore Dxf2Shape



1. Avviare QGIS, caricare il plugin Dxf2Shape nel Gestore dei Plugins (Vedere Sezione 11.1.1) e premere l'icona  Convertitore Dxf2Shape che compare nel menu della barra degli strumenti di QGIS. La finestra di dialogo del plugin Dxf2Shape appare come mostrato in Figura 41.
2. introdurre il file DXF in Input, un nome per il file Shape in Output ed il tipo di file Shape.
3. Abilitare la casella  Esporta le etichette di testo, se si vuole creare un layer aggiuntivo di punti con le etichette.
4. Premere .

12.5 Plugin Georeferenziatore

Il  **Georeferenziatore** permette di generare file di georeferenziazione per raster. Occorre selezionare punti sul raster, aggiungere le loro coordinate ed il plugin calcola i parametri per il file di georeferenziazione. più sono le coordinate che si forniscono, migliore sarà il risultato.

Come esempio genereremo un file di georeferenziazione per una carta topografica del Sud Dakota da SDGS. Essa potrà essere visualizzata insieme agli altri dati della Location Spearfish60 di GRASS. E' possibile scaricare la carta topografica qui: http://grass.osgeo.org/sampleddata/spearfish_toposheet.tar.gz

Come primo passo scaricare il file e decomprimerlo.

```
wget http://grass.osgeo.org/sampleddata/spearfish_toposheet.tar.gz
tar xvzf spearfish_toposheet.tar.gz
cd spearfish_toposheet
```

Il passo successivo è avviare QGIS, caricare il plugin di georeferenziazione e selezionare il file `spearfish_topo24.tif`.


Figura 42: Selezionare un'immagine da georeferenziare 

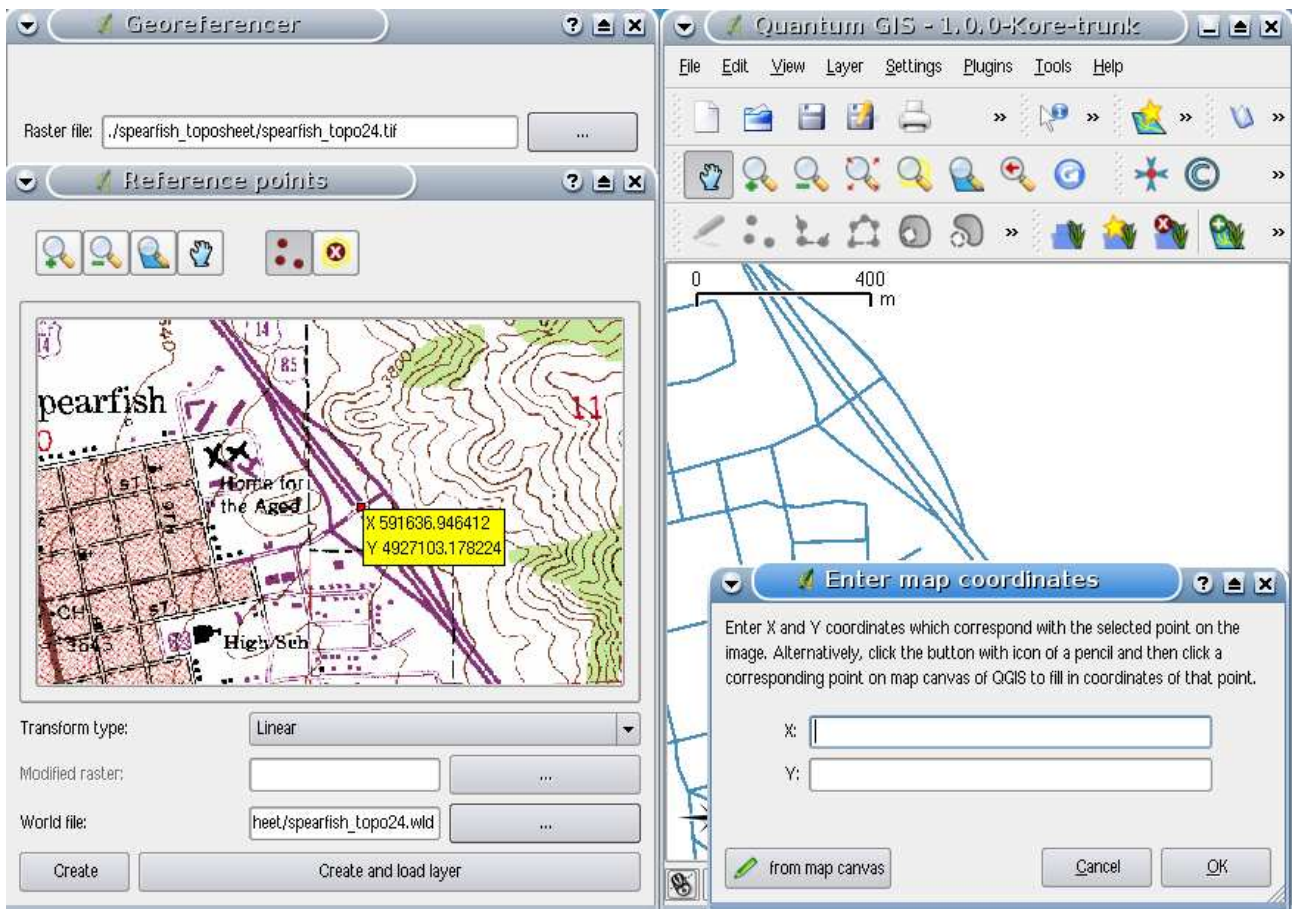


Ora cliccare il pulsante **Organizza le finestre di plugin** per aprire l'immagine nel georeferenziatore e organizzarla nella mappa di riferimento nella mappa qgis sulla propria scrivania (Vedere Figura 43).

Con il pulsante **Aggiungi punto** è possibile cominciare ad aggiungere punti sull'immagine raster ed inserire le loro coordinate, ed il plugin calcolerà i parametri del file di georeferenziazione (Vedere Figura 44). Più sono le coordinate che si forniscono, migliore sarà il risultato. Per procedere ci sono due opzioni:

1. Si può cliccare sul raster inserendo manualmente le coordinate X e Y del punto inserito

Figura 43: Organizzare le finestre di plugin con la mappa qgis 



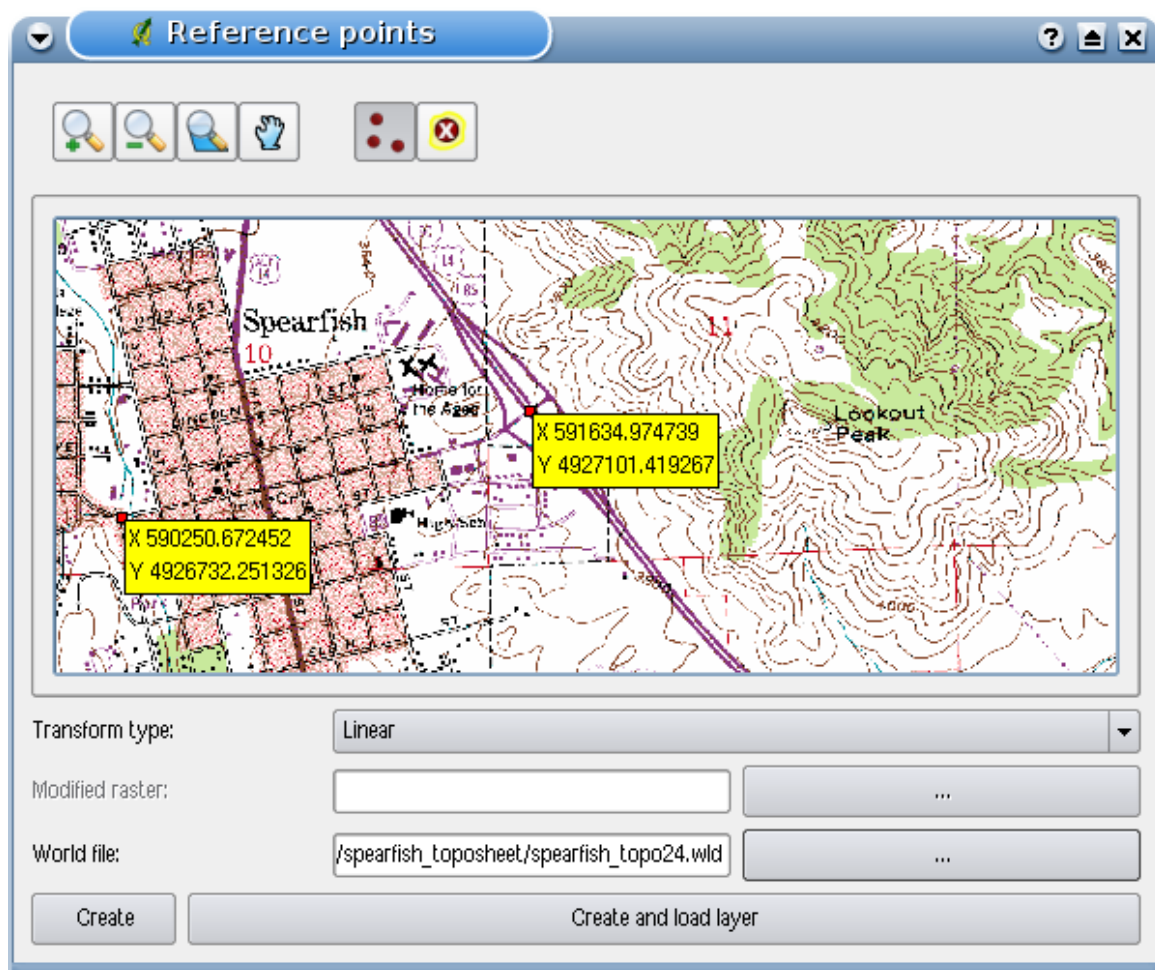
2. Si può cliccare sul raster e premere il pulsante **da mappa** per aggiungere le coordinate X e Y con l'aiuto di una mappa già georeferenziata già caricata in QGIS.

Per questo esempio useremo la seconda opzione e inseriremo le coordinate per i punti selezionati con l'aiuto della mappa roads contenuta nella location spearfish60 da: http://grass.osgeo.org/sampled/spearfish_grass60data-0.3.tar.gz

Se non si sa come integrare la location spearfish60 con il plugin GRASS, vedere le informazioni nella Sezione 9. Come si può vedere in Figura 44, il georeferenziatore fornisce tasti di zoom, pan, aggiunta e rimozione di punti dall'immagine.

Dopo aver aggiunti abbastanza punti all'immagine, occorre selezionare il tipo di trasformazione per il processo di georeferenziazione e salvare il risultante file di georeferenziazione insieme con il file Tiff. Nel nostro esempio scegliamo una **Transform type trasformazione lineare**

Transform type trasformazione di Helmert sarebbe sufficiente.

Figura 44: Aggiungere punti all'immagine raster **Tip 42 SCEGLIERE IL TIPO DI TRASFORMAZIONE**

La trasformazione lineare (affine) è una trasformazione di primo ordine e si usa per scalare, traslare e ruotare immagini geometricamente corrette. Con la trasformazione di Helmert semplicemente si aggiunge l'informazione delle coordinate all'immagine come in un geocoding. Se l'immagine è contorta, è necessario usare software che fornisca trasformazioni polinomiali di secondo o terzo ordine, ad esempio GRASS GIS.

I punti aggiunti alla mappa saranno salvati nel file `spearfish_topo24.tif.points` Insieme all'immagine raster. Questo permette di riaprire il plugin Georeferenziatore e aggiungere o rimuovere punti per ottimizzare il risultato. Il file `spearfish_topo24.tif.points` di quest'esempio mostra i punti:

mapX	mapY	pixelX	pixelY
591630.196867999969982	4927104.309682800434530	591647	4.9271e+06
608453.589164100005291	4924878.995150799863040	608458	4.92487e+06

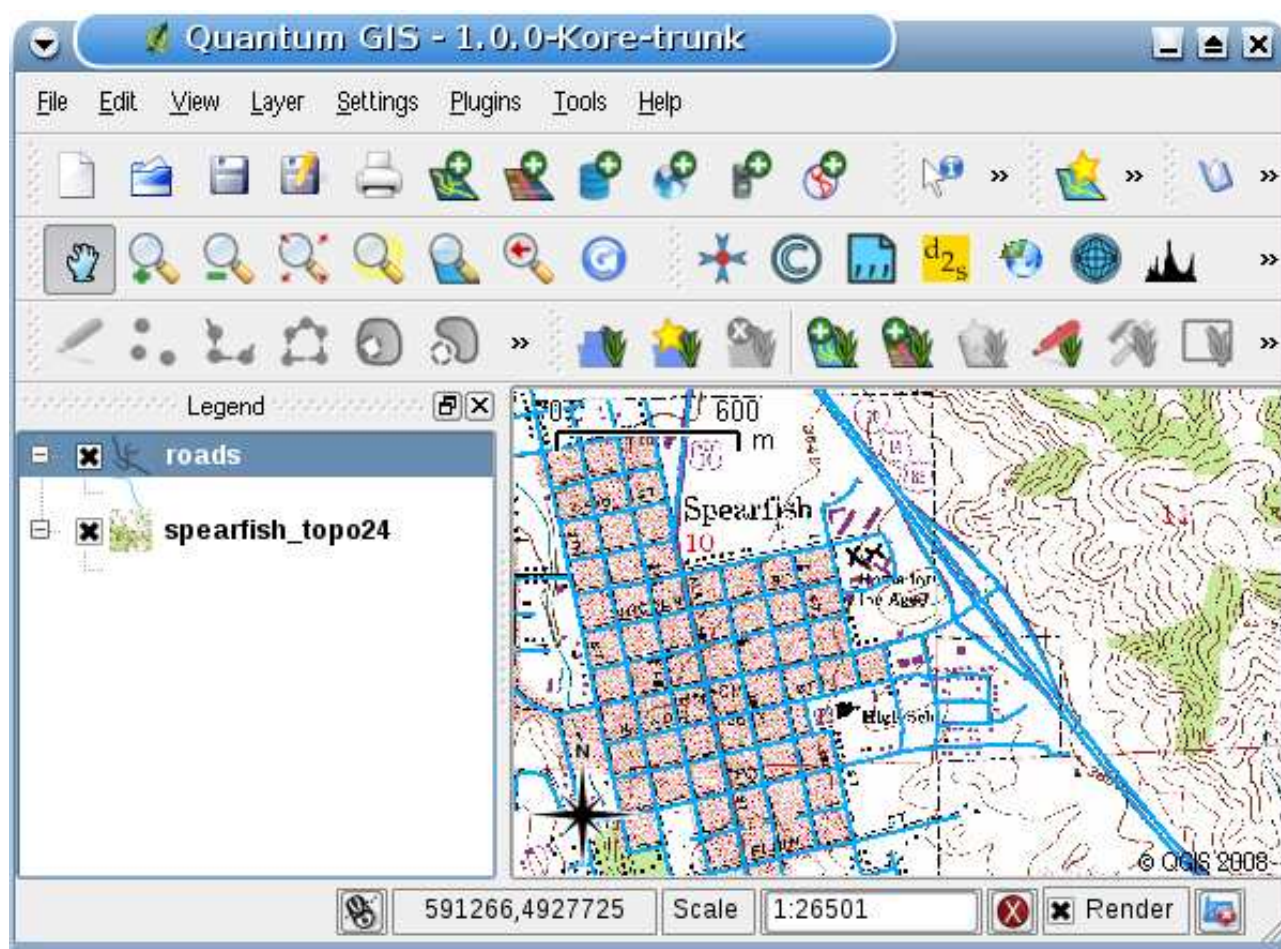

```

602554.903929700027220 4915579.220743400044739 602549 4.91556e+06
591511.138448899961077 4915952.302661700174212 591563 4.91593e+06
602649.526155399973504 4919088.353569299913943 602618 4.91907e+06

```

Usiamo le 5 coordinate dei punti per georeferenziare l'immagine raster. Per ottenere un risultato corretto, è importante posizionare i punti con regolarità nell'immagine. Infine, si può controllare il risultato, caricare la nuova mappa georeferenziata `spearfish_topo24.tif` e sovrapporla alla mappa roads della location `spearfish60`.

Figura 45: Mappa georeferenziata con la mappa roads della location `spearfish60` sovrapposta 



12.6 Quick Print Plugin



Il plugin  **Quick Print** permette di stampare la mappa corrente con minimo sforzo in formato PDF. Tutto quello che l'utente deve fare è aggiungere un Titolo Mappa, un Nome Mappa e scegliere la dimensione della pagina (Vedere Figura 46).

Figura 46: Finestra di dialogo Quick Print 

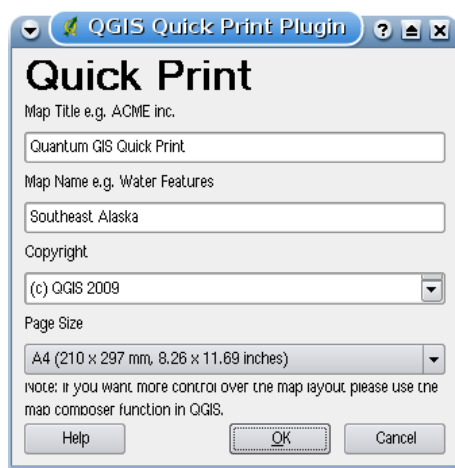
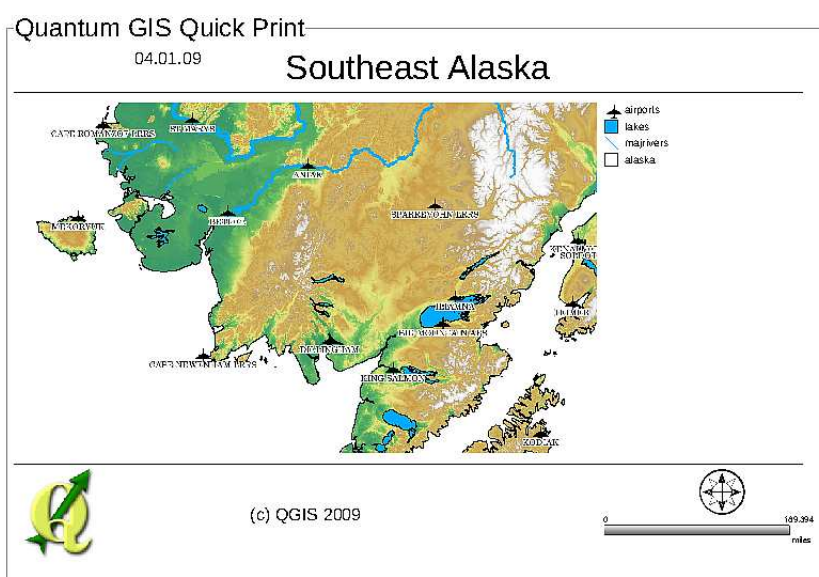


Figura 47 qui sotto mostra il risultato di una stampa veloce DIN A4 dal dataset campione dell'Alaska. Se si vuole maggior controllo sul layout della mappa, è meglio usare il plugin Compositore di Stampe, descritto nella Sezione 10.

Figura 47: Risultato ottenuto con Quick Print come DIN A4 PDF 



12.7 Plugin GPS


12.7.1 Cos'è un GPS?

GPS sta per Global Positioning System ed indica un sistema satellitare che permette a chiunque possieda un ricevitore GPS di trovare la propria posizione ovunque nel mondo. Viene usato come ausilio nella navigazione, ad esempio in aereo, in barca o anche facendo escursioni. Il ricevitore GPS usa i segnali provenienti dai satelliti per calcolare la propria latitudine, longitudine and (talvolta) elevazione. La maggior parte dei ricevitori ha anche la capacità di archiviare posizioni (note come *waypoints*), sequenze di posizioni che costituiscono una rotta o *route* pianificata e una traccia or *track* dei propri movimenti nel tempo. Waypoints, routes e tracks sono i tre elementi base dei dati GPS. QGIS mostra i waypoints in layer di punti, mentre routes e tracks vengono mostrati in layer di linee.

12.7.2 Caricare i dati GPS da un file

Ci sono dozzine di diversi formati di file usati per archiviare dati GPS. Il formato usato da QGIS si chiama GPX (GPS eXchange format, formato di scambio GPS), che è un formato standard di interscambio che può contenere qualunque numero di waypoints, routes e tracks nello stesso file.

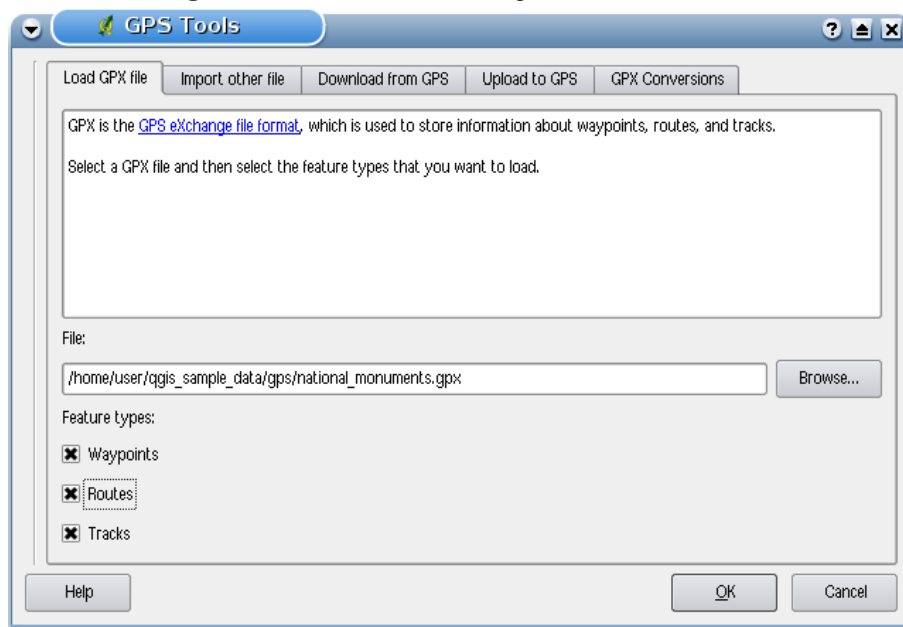

Per caricare un file GPX bisogna prima caricare il plugin. **Plugins** > **Gestione Plugin...** > **x Strumenti GPS**. Quando questo plugin è caricato, un tasto indicante un piccolo dispositivo manuale GPS compare nella barra degli strumenti. Un file GPX di esempio è disponibile nel dataset: `/qgis_sample_data/gps/national_monuments.gpx`. Vedere la Sezione 3.2 per avere maggiori informazioni sui dati di esempio.

1. Cliccare l'icona  **strumenti GPS** e selezionare la tab **Carica file GPX**.
2. **Sfoglia** la cartella `qgis_sample_data/gps/`, selezionare il file GPX `national_monuments.gpx` e premere **Apri**.

Usare il pulsante Sfoglia **...** per selezionare il file GPX, quindi selezionare con i checkbox i tipi di dati che si desidera caricare dal file GPX. Ogni tipo di dato verrà caricato in uno layer separato quando si preme il pulsante **OK**. Il file `national_monuments.gpx` include soltanto waypoints.

12.7.3 GPSTBabel

Dato che QGIS usa file in formato GPX, è necessario un sistema per convertire gli altri formati di file GPS in GPX. Questo può essere fatto per molti formati usando il programma gratuito GPSTBabel, che è disponibile a <http://www.gpsbabel.org>. Questo programma può anche trasferire i dati GPS tra il

Figura 48: La finestra di dialogo *Strumenti GPS* 

computer e il dispositivo GPS. QGIS usa GPSTools per fare queste cose, quindi si raccomanda di installare questo programma. Tuttavia, se si vuole solamente caricare dati GPS da file GPX, l'installazione non è necessaria. E' noto che la versione 1.2.3 di GPSTools funziona con QGIS, ma non ci dovrebbero essere problemi anche con versioni successive.

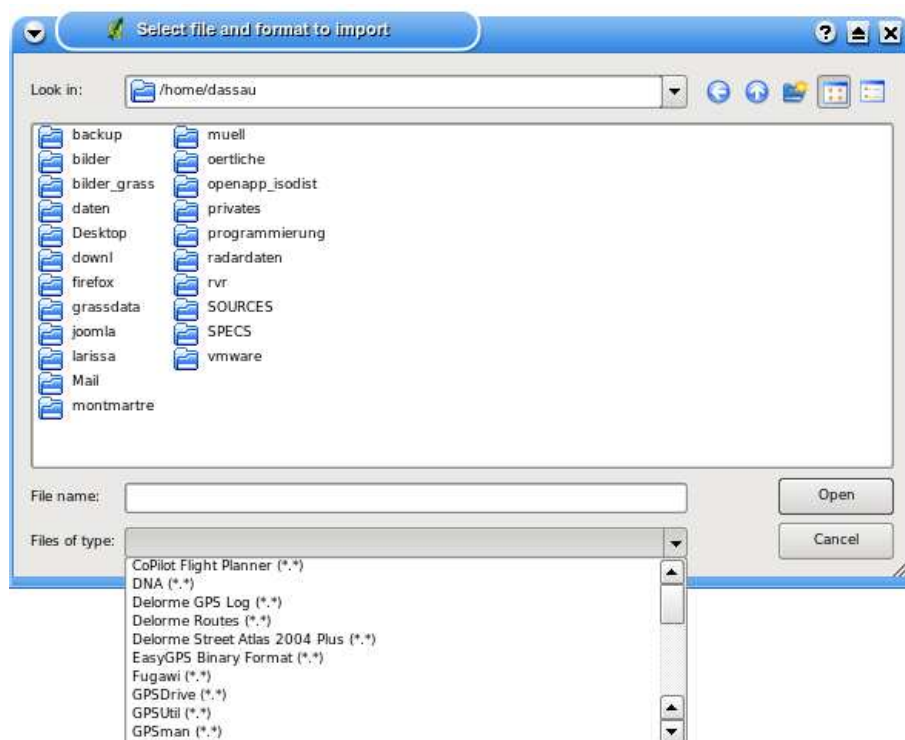
12.7.4 Importazione di dati GPS

Per importare dati GPS da un file che non è in formato GPX, si può usare lo strumento **Importa un altro file** nella finestra di dialogo Strumenti GPS. Qui si seleziona il file che si vuole importare, quali tipi di dati si vogliono importare da esso, dove deve essere archiviato il file convertito GPX e quale deve essere il nome del nuovo layer.

Quando si seleziona il file da importare, si deve anche specificare il formato di quel file usando il menu nella finestra di dialogo di selezione file (vedere figura 49). non tutti i formati supportano tutti e tre i tipi di dati, quindi per molti formati si potranno scegliere solamente uno o due tipi.

12.7.5 Scaricare dati GPS da un dispositivo

QGIS può usare GPSTools per scaricare i dati da un dispositivo GPS direttamente in layer vettoriali. Per questo si usa lo strumento **Download dal GPS** (vedere la Figura 50), dove si seleziona il tipo

Figura 49: Finestra di dialogo dello strumento di selezione del file da importare 🐧

di dispositivo GPS, La porta di comunicazione alla quale lo si è connesso, i tipi di dati che si vogliono scaricare, il nome del file GPX dove i dati devono essere archiviati e il nome del nuovo layer.

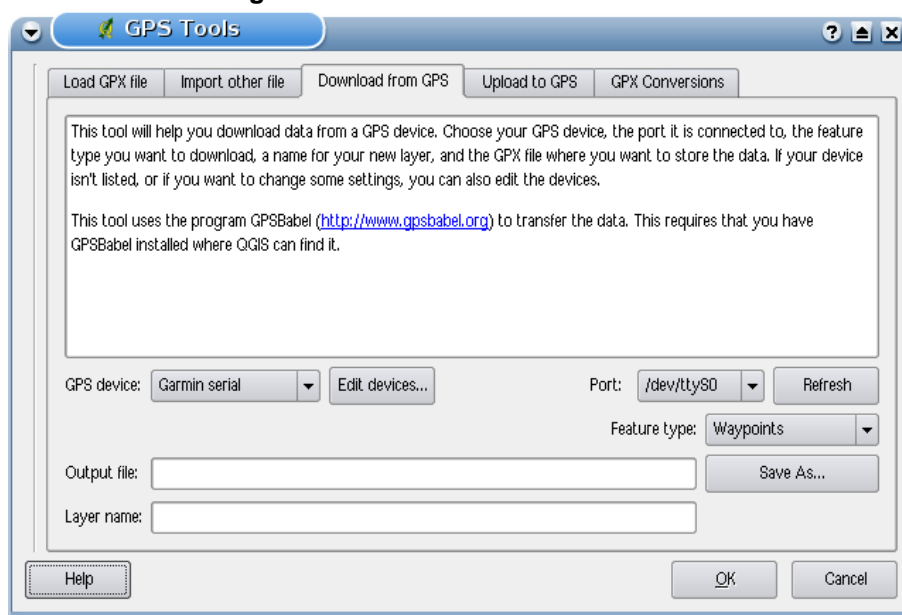
Il tipo di dispositivo che si seleziona nel menu dispositivi GPS determina il modo con cui GPSTabel cerca di comunicare con il dispositivo in questione. Se nessuno di questi tipi funziona con il dispositivo che si ha, si può creare un tipo nuovo (vedere sezione [12.7.7](#)).

la porta è un nome di file o un qualche altro nome che il sistema operativo usa nper riferirsi alla porta fisica del computer cui è connesso il dispositivo GPS. 🐧 In Linux è qualcosa come /dev/ttyS0 or /dev/ttyS1 e in 🌐 Windows è COM1 o COM2.

Quando si preme il pulsante **OK** i dati verranno scaricati dal dispositivo ed appariranno come layer in QGIS.

12.7.6 Caricare dati GPS In un dispositivo

Si può anche caricare dati direttamente da un layer vettoriale di QGIS in un dispositivo GPS, usando lo strumento **Upload sul GPS**. Il layer deve essere un layer GPX. Per far questo semplicemente si seleziona il layer che si vuole caricare, il tipo di dispositivo GPS e la porta cui è connesso. Come per

Figura 50: Lo strumento di download 

lo strumento di download, bisogna specificare un nuovo tipo di dispositivo se quello che si sta usando non è nella lista.

Questo strumento è molto utile insieme con le capacità di pubblicazione vettoriale di QGIS. Si può caricare una mappa, creare alcuni waypoints e routes e poi caricarli ed utilizzarli nel proprio dispositivo GPS.


12.7.7 Definizione di un nuovo dispositivo

Ci sono moltissimi tipi diversi di dispositivi GPS. Gli sviluppatori di QGIS non possono testarli tutti, quindi se ne ha uno che non funziona con nessuno dei tipi elencati negli strumenti **Download from GPS** and **Upload to GPS** si può definire il proprio tipo di dispositivo per esso. Questo può essere fatto usando l'editor di periferiche GPS, che si lancia premendo il pulsante **Modifica periferiche** nelle finestre di dialogo di download o di upload.

Per definire un nuovo dispositivo, semplicemente si preme il pulsante **Nuova periferica**, si sceglie un nome, un comando di download ed uno di upload per il dispositivo, e infine si preme il pulsante **Aggiorna periferiche**. Il nome verrà elencato nel menu dei dispositivi nelle finestre di dialogo di upload e download, e può essere qualsiasi stringa. Il comando di download è il comando usato per scaricare i dati dal dispositivo GPS al file GPX. Questo probabilmente sarà un comando di GPSTabel, ma si può usare qualunque altra linea di comando che possa creare un file GPX. QGIS sostituirà le

parole chiave %type, %in, and %out quando fa funzionare il comando.

%type sarà sostituito da “-w” se si stanno scaricando waypoints, “-r” se si stanno scaricando routes e “-t” se si stanno scaricando tracks. Queste sono opzioni della linea di comando che dicono a GPSTBabel quali tipi di dati scaricare.

%in sarà sostituito dal nome della porta scelta nella finestra di dialogo di download e %out sarà sostituito dal nome scelto per il file GPX in cui saranno archiviati i dati scaricati. Così, se si crea un tipo di periferica con il comando di download “gpsbabel %type -i garmin -o gpx %in %out” (questo è in realtà il comando di download per il tipo di dispositivo predefinito ) e lo si usa poi per scaricare waypoints dalla porta “/dev/ttyS0” al file “output.gpx”, QGIS sostituirà le parole chiave ed eseguirà il comando “gpsbabel -w -i garmin -o gpx /dev/ttyS0 output.gpx”.


Il comando di upload è il comando che si usa per caricare dati sul dispositivo GPS. Si usano le stesse parole chiave, ma %in è sostituita con il nome del file GPX per il layer che si sta caricando e %out è sostituito dal nome della porta.

Per ulteriori informazioni su GPSTBabel e per le opzioni disponibili per la sua linea di comando: <http://www.gpsbabel.org>

Una volta creata il nuovo tipo di periferica, essa apparirà nella lista dei dispositivi per gli strumenti di download e upload.


12.8 Plugin Creatore di Griglia

Il creatore di griglia permette di creare una griglia di punti o poligoni per coprire un'area di interesse. Tutte le unità di misura devono essere inserite in gradi decimali. L'output è un file shape che può essere riproiettato al volo su altri dati.

Figura 51: Creare uno strato griglia 



Ecco un esempio di come creare una griglia:


1. Lanciare QGIS, caricare il Plugin Creatore di Griglia nel Gestore dei Plugin (vedere Sezione 11.1.1) e cliccare sull'icona  **Creatore di Griglia** che appare nella barra degli strumenti QGIS.
2. Scegliere il tipo di griglia che si vuole creare: punti o poligoni.
3. Inserire valori di latitudine e longitudine per gli angoli in basso a sinistra ed in alto a destra della griglia.
4. Inserire l'intervallo da usare nella costruzione della griglia. Si possono inserire valori diversi per le direzioni X e Y (longitudine, latitudine).
5. Scegliere il nome e la cartella in cui creare il file shape.

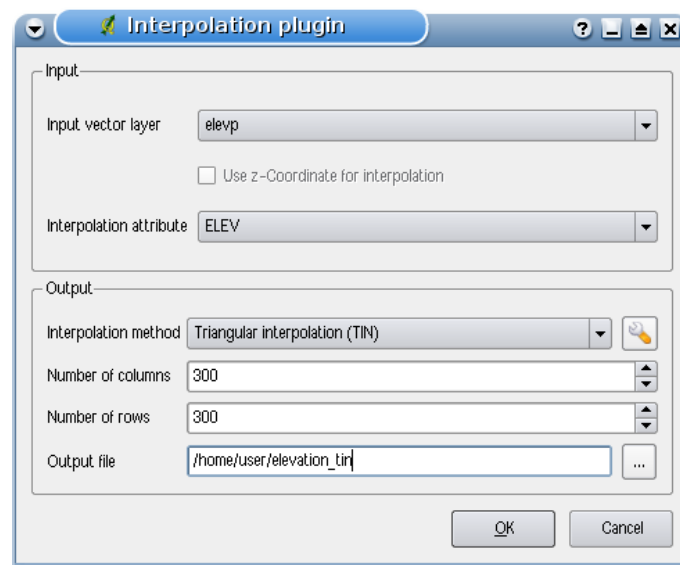
6. Cliccare **OK** per creare la griglia ed aggiungerla alla mappa.


12.9 Interpolation Plugin



Il plugin di interpolazione permette di interpolare un layer raster TIN o IDW a partire da un layer vettoriale caricato in QGIS. E' molto semplice da maneggiare e fornisce le funzionalità mostrate in Figura 52.

- **Layer vettoriale in input:** Seleziona il layer vettoriale caricato in QGIS.
- **Attributo di interpolazione:** Seleziona la colonna attributo usata per l'interpolazione o abilita la casella ☒ Usa Coordinata -Z.
- **Metodo di interpolazione:** Seleziona il metodo di interpolazione Interpolazione triangolare (TIN) o Distanza Inversa Ponderata (IDW) .
- **Numero di colonne/righe:** definisce il numero per il file raster in output
- **Output file:** Definisce un nome per il file raster in output


Figura 52: Plugin di interpolazione 

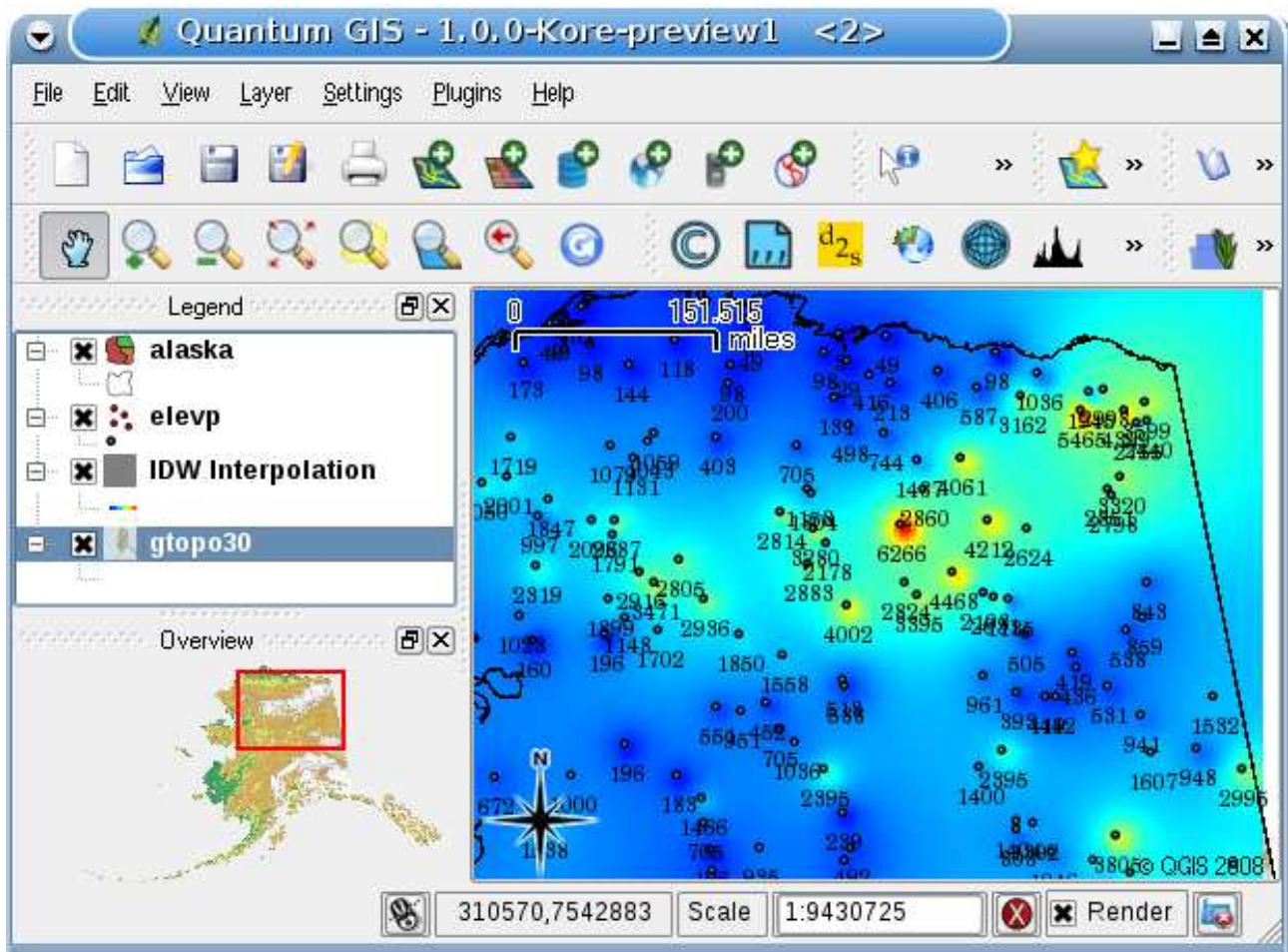


1. Avviare QGIS e caricare la `elevp.csv` tabella CSV con i punti elevazione in QGIS usando il plugin di testo delimitato come descritto nella Sezione 12.3.
2. Caricare il plugin di Interpolazione nel Gestore dei Plugin (see Section 11.1.1) e premere il pulsante  **Interpolazione** che appare nella barra menu strumenti in QGIS. la finestra di dialogo del plugin di Interpolazione appare come in Figura 52.
3. Selezionare `elevp` come vettore di input e la colonna `ELEV` per interpolazione.

4. Selezionare **Interpolazione triangolare**  come metodo di interpolazione, definire 3663 colonne e 1964 righe (questo è equivalente ad una risoluzione pixel di 1000 metri) come nome del file raster di output `elevation_tin`.
5. Premere **Ok**.
6. Premere due volte `elevation_tin` nella legenda della mappa per aprire la finestra di dialogo Proprietà del Layer Raster e selezionare **Pseudocolor**  come mappa di colore nella scheda **Simbologia**. Oppure si può definire una nuova tabella di colori come descritto nella Sezione [6.3](#).

Nella Figura [53](#) si vede il risultato di una interpolazione IDW con una risoluzione 366 colonne x 196 righe (10 km) per i dati `elevp.csv` visualizzati usando la tabella di colore Pseudocolor. Il processamento impiega circa un paio di minuti, sebbene i dati riguardano soltanto la parte nord dell'Alaska.

Figura 53: Interpolazione di dati elevp usando il metodo IDW 



12.10 Plugin MapServer Export

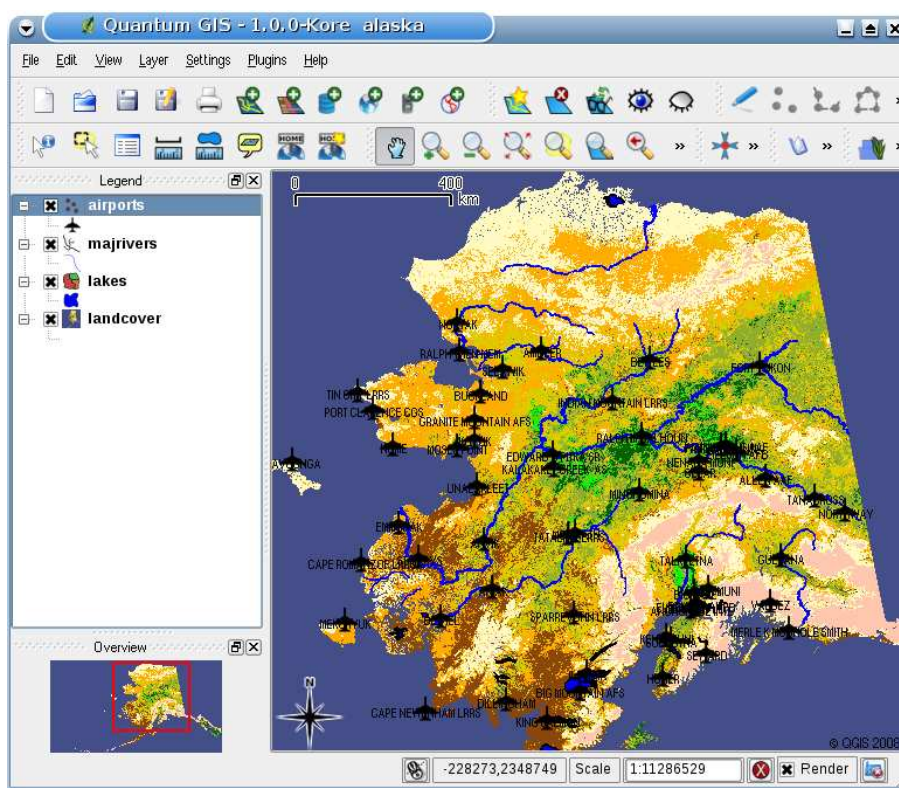
Si può usare QGIS per 'comporre' la propria mappa aggiungendo e arrangiando i layers, simbolizzandoli, personalizzando i colori, e infine creando un map file per Mapserver. Per usare il plugin MapServer Export bisogna aver installato Python ≥ 2.4 nel proprio sistema e aver compilato QGIS con il supporto per esso. Tutti i pacchetti binari includono il supporto Python.

Il plugin MapServer Export in QGIS 1.0.0 è un plugin Python, il che significa che viene caricato automaticamente nel Gestore dei Plugin come plugin core (see Section 12).



12.10.1 Creare il file Progetto

Il Plugin MapServer Export opera su un progetto QGIS salvato e **non** sui contenuti correnti della finestra aperta e della sua legenda. Questo ha generato confusione per molta gente. Come descritto più avanti, prima di cominciare ad utilizzare il Plugin MapServer Export, è necessario predisporre i layers vettoriali e raster che si vuole usare in MapServer e salvare queste impostazioni nel file progetto QGIS


Figura 54: predisporre i layers vettoriali e raster in un file progetto QGIS 



In questo esempio vengono mostrati i quattro passi per arrivare ad essere pronti a creare il map file in MapServer. Vengono usati file vettoriali e raster dal dataset campione di QGIS 3.2.

1. Aggiungere il layer raster `landcover.tif` premendo il tasto  **Aggiungere Layer Raster**.
2. Aggiungere i file Shape `lakes.shp`, `majrivers.shp` e `airports.shp` dal dataset campionei QGIS premendo il tasto  **Aggiungere Layer Vettoriale**.
3. Cambiare i colori e simbolizzare i dati a piacere (vedere la Figura 54)
4. Salvare un nuovo progetto chiamato `mapserverproject.qgs` usando **File** > **Salva Progetto**.

12.10.2 Creazione del Map File

Lo strumento `msexport` per esportare un file progetto QGIS in un file mappa MapServer è installato nella directory binaria QGIS e può essere usato indipendentemente da QGIS. da QGIS bisogna prima caricare il Plugin MapServer Export con il Gestore dei Plugin. Selezionare **Plugins** > **Gestione Plugins...** per aprire il Gestore dei Plugin, scegliere il Plugin MapServer Export e premere **OK**. Ora avviare la finestra di dialogo  **MapServer Export** (vedere Figura 55) premendo il tasto nel menu strumenti.

File mappa

Scegliere un nome per il file mappa da creare. si può usare il tasto a destra per scegliere per la directory dove si vuole che il file venga creato.

File progetto QGIS

Introdurre il percorso completo per il file progetto QGIS (.qgs) che si vuole esportare. Si può usare il tasto a destra per cercare il file progetto QGIS.

Nome mappa

Un nome per la mappa. Questo nome è prefissato a tutte le immagini generate dal mapserver.

Larghezza mappa

Larghezza dell'immagine in output in pixel.

Altezza della mappa

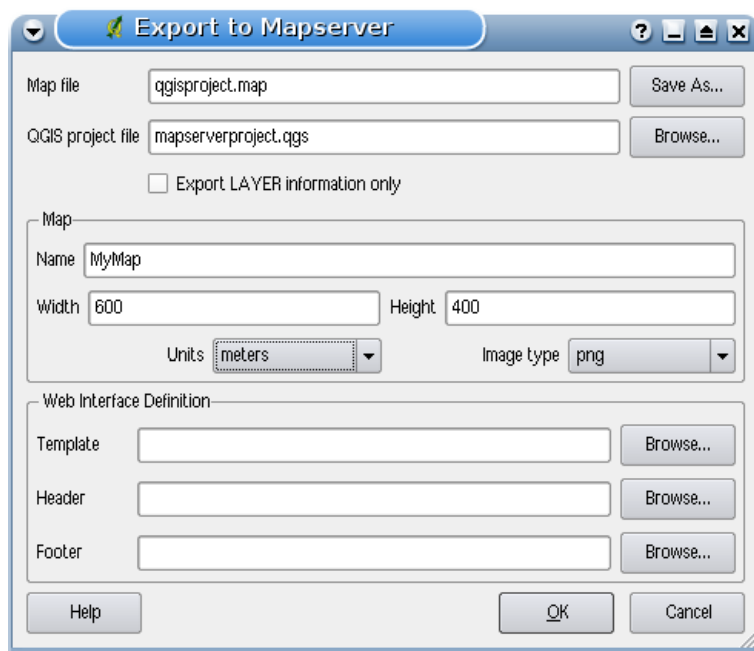
Altezza dell'immagine in output in pixel.

Unità della mappa

Unità di misura usate per l'output.

Tipo di immagine

Formato dell'immagine in output generata da MapServer.

Figura 55: Finestra di dialogo MapServer Export **Modello web**

Percorso completo al file modello di Mapserver che sarà usato con il file mappa.


Intestazione web

Percorso completo al file intestazione di Mapserver che sarà usato con il file mappa.

Piede pagina web

Percorso completo al file piede pagina di Mapserver che sarà usato con il file mappa.

Soltanto gli input File Mappa e file progetto QGIS sono richiesti per creare un file maoppla, tuttavia ci si può ritrovare con un file mappa non funzionante, a seconda di quale è l'uso che se ne vuole fare. Anche se QGIS è capace di creare un file mappa da un file progetto, può essere necessario qualche aggiustamento per raggiungere il risultato desiderato. Proviamo comunque a creare il file mappa usando il file progetto `mapserverproject.qgs` appena creato (vedere Figura 55):

1. Aprire il Plugin MapServer Export premendo il tasto  **MapServer Export**.
2. Introdurre il nome `qgisproject.map` per il nuovo file mappa.
3. Selezionare il file progetto QGIS `mapserverproject.qgs` appena salvato.
4. Introdurre un nome `MyMap` per la mappa.
5. Introdurre 600 per la larghezza e 400 per l'altezza.
6. I layer sono in metri, quindi si cambiano le unità a metri.

7. Scegliere 'png' come tipo di immagine.

8. Premere **OK** per generare il nuovo file mappa `qgisproject.map`. QGIS mostrerà il risultato.

Si può visualizzare il file mappa con qualsiasi editoriale visualizzatore di testo. Se si apre il file, si noterà che lo strumento di esportazione aggiunge i metadati necessari per abilitare la mappa per WMS.

12.10.3 Testare il File Mappa

Si può testare il risultato fin qui ottenuto usando lo strumento `shp2img` per creare un'immagine dal file mappa. L'utilità `shp2img` è parte di MapServer e FWTools. Per creare un'immagine dalla mappa:

- Aprire una finestra terminale
- Se la mappa non è stata salvata nella directory Home, cambiare alla cartella in cui è stata salvata
- Lanciare `shp2img -m qgisproject.map -o mapserver_test.png` e mostrare l'immagine

Questo crea un PNG con tutti i layer inclusi nel file progetto QGIS. In aggiunta, l'estensione del PNG sarà la stessa di quando si è salvato il progetto. Come si vede in Figura 56, tutte le informazioni eccetto i simboli degli aeroporti sono incluse.

Figura 56: Test PNG creato con `shp2img` con tutti i layer MapServer Export 🗺️



Se si prevede di usare il file mappa per richieste WMS, probabilmente non sarà necessario alcun adattamento. Se invece si prevede di usarlo con un modello di mappatura o un'interfaccia persona-

lizzata, potrebbe essere necessario un po' di lavoro. Per vedere come è facile passare da QGIS alle mappe di servizio sul web, vedere il flash video di 5 minuti di Christopher Schmidt. Egli usa QGIS versione 0.8, ma è comunque utile.⁹

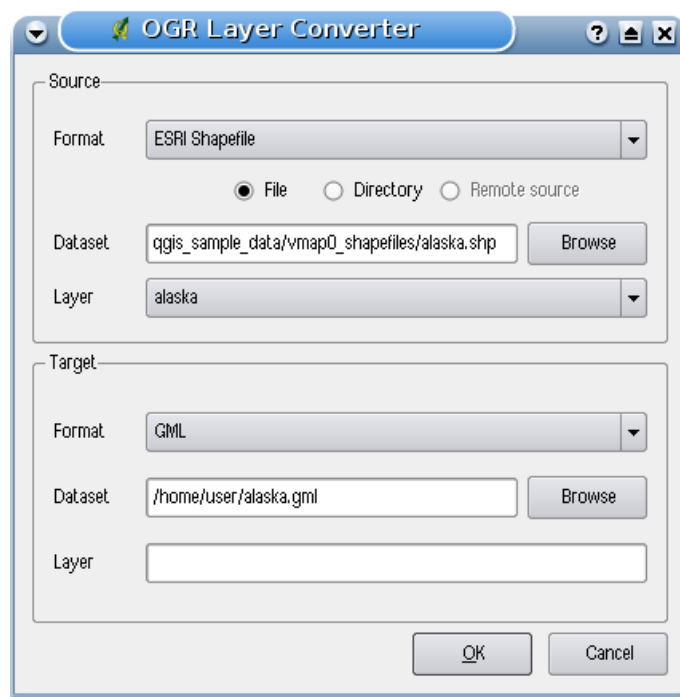
⁹<http://openlayers.org/presentations/mappingyourdata/>

12.11 Plugin Convertitore Layer OGR

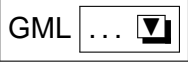

Il plugin convertitore layer OGR permette di convertire i dati vettoriali da un formato vettoriale supportato OGR ad un altro formato vettoriale supportato OGR. E' molto semplice da maneggiare e offre funzionalità come mostrato in Figura 57. I formati supportati possono variare a seconda del pacchetto GDAL/OGR installato.

- **Sorgente Formato/Set di dati/Layer:** Introduce il formato OGR e il percorso al file vettoriale da convertire
- **Destinazione Formato/Set di dati/Layer:** Introduce il formato OGR e il percorso al file vettoriale in output

Figura 57: Plugin Convertitore Layer OGR 



1. lanciare QGIS, caricare il Plugin Convertitore Layer OGR nel Gestore dei Plugin (vedere Sezione 11.1.1) e premere il tasto  **Convertitore Layer OGR** che appare nel menu della barra strumenti QGIS. Appare la finestra di dialogo del Plugin Convertitore Layer OGR dialog come mostrato in Figura 57.
2. Selezionare il formato OGR supportato **ESRI Shapefile**  e il percorso per il file vettoriale in input `alaska.shp` nell'area Sorgente.

3. Selezionare il formato OGr sopoprato  e definire un percorso e il nome per il file vettoriale in output `alaska.gml` nell'area destinazione.
4. Premere .


13 Uso dei Plugin esterni QGIS Python

I Plugin esterni QGIS sono scritti in python. Sono immagazzinati in un archivio ufficiale moderato e mantenuto dall'autore. La tabella 7 mostra una lista di plugin attualmente disponibile con una breve descrizione.^{10 11}

Quando questo manuale è stato distribuito, archivio ufficiale moderato e mantenuto non era ancora completamente costituito. Una documentazione dettagliata circa l'uso, l'autore ed altre importanti informazioni sono fornite con il plugin esterno stesso e non è parte di questo manuale.

Si può trovare una lista aggiornata dei plugin esterni moderati nell'Archivio ufficiale di QGIS a  Scarica Python Plugins...) e <http://qgis.osgeo.org/download/plugins.html>.







Tabella 7: Attuali Plugin QGIS esterni moderati

Icona	Plugin esterno	Descrizione
	Zoom al punto	Fa lo zoom alle coordinate specificate nella finestra di dialogo di input. Si può specificare il livello di zoom così come il controllo dell'estensione della vista.

Una descrizione dettagliata per i plugin python esterni si può trovare nella Sezione [11.1.2](#).

Archivio Plugin Python sviluppati dagli utenti e Archivi degli autori

In aggiunta ai plugin esterni moderati, esiste anche un altro archivio Plugin Python non ufficiale. Contiene plugin che non sono ancora abbastanza maturi da essere inclusi nell'archivio ufficiale, comunque alcuni possono essere abbastanza utili. Inoltre, alcuni dei nostri contributori mantengono i propri archivi.

Per aggiungere l'archivio non ufficiale e gli archivi degli autori aprire l'Installatore di Plugin Python ( Plugins >  Scarico Plugin Python), andare alla scheda  Repositories e premere  Aggiunge repository di terze parti. Se non si vuole uno o più degli archivi aggiunti, disabilitarli con il tasto  Edit... o rimuoverli completamente con il tasto  Delete.

Tip 43 AGGIUNGERE PIÙ PLUGIN ESTERNI

In aggiunta ai plugin esterni moderati dall'archivio ufficiale QGIS si possono aggiungere più archivi esterni. Quindi selezionare la scheda Repositories nell'Installatore di Plugin Python

¹⁰Anche gli aggiornamenti dei plugin core possono essere disponibili anche in questo archivio come overlay esterni.

¹¹L'Installatore di Plugin Python è anch'esso un plugin Python esterno, ma è parte della sorgente QGIS e caricato automaticamente e selezionabile nel Gestore dei Plugin di QGIS (vedere Sezione [11.1.2](#)).

14 Scrivere un plugin QGIS in C++

In questa sezione viene fornito un'esercitazione per principianti per scrivere un semplice plugin QGIS in C++. E' basata su un workshop tenuto dal Dr. Marco Hugentobler.

I plugin QGIS C++ sono librerie collegate dinamicamente (.so or .dll). Sono legate a QGIS al runtime quando richiesto nel Gestore dei Plugin ed estendono la funzionalità di QGIS. hanno accesso al GUI di QGIS e possono essere distinti in core ed ed esterni.

Tecnicamente il Gestore dei Plugin QGIS cerca nella directory lib/qgis per tutti i file .so e li carica quando è lanciato. Quando viene chiuso sono scaricati di nuovo, eccetto quelli con la casella vistata. Per i plugin caricati nuovi, il *classFactory* metodo crea un'istanza della classe del plugin e il *initGui* metodo del plugin è chiamato a mostrare gli elementi GUI nel menu plugin e nella barra degli strumenti. La funzione *unload()* del plugin è usata per rimuovere gli elementi GUI allocati e la classe stessa del plugin è rimossa usando il distruttore di classe. Per elencare i plugin, ognuno deve avere alcune funzioni esterne 'C' per descrizione e ovviamente il metodo *classFactory*.

14.1 Perché C++ e circa le licenze

QGIS stesso è scritto in C++, così è sensato scrivere anche i plugin in C++. E' un linguaggio di programmazione orientato all'oggetto (object-oriented programming, OOP) che è visto da molti sviluppatori come un linguaggio preferenziale per creare applicativi su larga scala.

I plugins QGIS C++ usano funzionalità di librerie libqgis*.so. Dato che hanno licenze GNU GPL, anche i plugins QGIS C++ devono avere licenze GPL. Questo significa che si possono usare i plugin per qualsiasi scopo e non si è costretti a pubblicarli. Se si vuole pubblicarli comunque, devono essere pubblicati sotto le condizioni della licenza GPL.

14.2 Programmare un plugin QGIS C++ in 4 passi

Il plugin di esempio è un convertitore di punti ed è intenzionalmente mantenuto semplice. Il plugin cerca lo layer vettoriale attivo in QGIS, converte tutti i vertici delle caratteristiche del layer a caratteristiche puntiformi mantenendo gli attributi e finalmente scrive le caratteristiche puntiformi in un file di testo delimitato. Il nuovo layer può quindi essere caricato in QGIS usando il plugin di testo delimitato (vedere la Sezione [12.3](#)).

Passo 1: Far riconoscere il plugin al gestore dei plugin

Come primo passo si creano i file `QgsPointConverter.h` e `QgsPointConverter.cpp`. Poi si aggiungono metodi virtuali ereditati da `QgsPlugin` (ma si lasciano vuoti per ora), si creano i necessari metodi

esterni 'C' e un file .pro, che è un meccanismo Qt per creare facilmente Makefiles. Quindi si compila i sorgenti, si sposta la libreria compilata nell cartella plugin e si carica con il gestore dei plugin QGIS.

a) Creare un nuovo file pointconverter.pro e aggiungere:

```
#base directory of the qgis installation
QGIS_DIR = /home/marco/src/qgis

TEMPLATE = lib
CONFIG = qt
QT += xml qt3support
unix:LIBS += -L/$$QGIS_DIR/lib -lqgis_core -lqgis_gui
INCLUDEPATH += $$QGIS_DIR/src/ui $$QGIS_DIR/src/plugins $$QGIS_DIR/src/gui \
    $$QGIS_DIR/src/raster $$QGIS_DIR/src/core $$QGIS_DIR
SOURCES = qgspointconverterplugin.cpp
HEADERS = qgspointconverterplugin.h
DEST = pointconverterplugin.so
DEFINES += GUI_EXPORT= CORE_EXPORT=
```

b) Creare un nuovo file qgspointconverterplugin.h e aggiungere:

```
#ifndef QGSPPOINTCONVERTERPLUGIN_H
#define QGSPPOINTCONVERTERPLUGIN_H

#include "qgisplugin.h"

/**A plugin that converts vector layers to delimited text point files.
 The vertices of polygon/line type layers are converted to point features*/
class QgsPointConverterPlugin: public QgisPlugin
{
public:
    QgsPointConverterPlugin(QgisInterface* iface);
    ~QgsPointConverterPlugin();
    void initGui();
    void unload();

private:
    QgisInterface* mIface;
};
#endif
```

c) Creare un nuovo file qgspointconverterplugin.cpp e aggiungere:

```
#include "qgspointconverterplugin.h"

#ifdef WIN32
#define QGISEXTERN extern "C" __declspec( dllexport )
#else
#define QGISEXTERN extern "C"
#endif

QgsPointConverterPlugin::QgsPointConverterPlugin(QgisInterface* iface): mIface(iface)
{
}

QgsPointConverterPlugin::~QgsPointConverterPlugin()
{
}

void QgsPointConverterPlugin::initGui()
{
}

void QgsPointConverterPlugin::unload()
{
}

QGISEXTERN QgisPlugin* classFactory(QgisInterface* iface)
{
    return new QgsPointConverterPlugin(iface);
}

QGISEXTERN QString name()
{
    return "point converter plugin";
}

QGISEXTERN QString description()
{
    return "A plugin that converts vector layers to delimited text point files";
}

QGISEXTERN QString version()
{
    return "0.00001";
}
```

```
}

// Return the type (either UI or MapLayer plugin)
QGISEXTERN int type()
{
    return QgisPlugin::UI;
}

// Delete ourself
QGISEXTERN void unload(QgisPlugin* theQgsPointConverterPluginPointer)
{
    delete theQgsPointConverterPluginPointer;
}
```

Passo 2: Creare un'icona, un pulsante e un menu per il plugin

Questo passo include l'aggiunta di un puntatore all'oggetto `QgisInterface` nella classe `plugins`. Quindi si crea un `QAction` e una funzione di richiamo (slot), si aggiunge al QGIS GUI usando `QgisInterface::addToolBarIcon()` and `QgisInterface::addPluginToMenu()` e alla fine si rimuove il `QAction` nel metodo `unload()`.

d) Aprire `qgspointconverterplugin.h` di nuovo e estendere il contenuto esistente a:

```
#ifndef QGSPPOINTCONVERTERPLUGIN_H
#define QGSPPOINTCONVERTERPLUGIN_H

#include "qgisplugin.h"
#include <QObject>

class QAction;

/**A plugin that converts vector layers to delimited text point files.
 The vertices of polygon/line type layers are converted to point features*/
class QgsPointConverterPlugin: public QObject, public QgisPlugin
{
    Q_OBJECT

public:
    QgsPointConverterPlugin(QgisInterface* iface);
    ~QgsPointConverterPlugin();
    void initGui();
    void unload();
}
```



```
private:
    QgisInterface* mIface;
    QAction* mAction;

    private slots:
        void convertToPoint();
};

#endif
```

e) Aprire qgspointconverterplugin.cpp di nuovo e estendere il contenuto esistente a:

```
#include "qgspointconverterplugin.h"
#include "qgisinterface.h"
#include <QAction>

#ifdef WIN32
#define QGISEXTERN extern "C" __declspec( dllexport )
#else
#define QGISEXTERN extern "C"
#endif

QgsPointConverterPlugin::QgsPointConverterPlugin(QgisInterface* iface): \
    mIface(iface), mAction(0)
{

}

QgsPointConverterPlugin::~QgsPointConverterPlugin()
{

}

void QgsPointConverterPlugin::initGui()
{
    mAction = new QAction(tr("&Convert to point"), this);
    connect(mAction, SIGNAL(activated()), this, SLOT(convertToPoint()));
    mIface->addToolBarIcon(mAction);
    mIface->addPluginToMenu(tr("&Convert to point"), mAction);
}
```

```
void QgsPointConverterPlugin::unload()
{
    mIface->removeToolBarIcon(mAction);
    mIface->removePluginMenu(tr("&Convert to point"), mAction);
    delete mAction;
}

void QgsPointConverterPlugin::convertToPoint()
{
    qWarning("in method convertToPoint");
}

QGISEXTERN QgisPlugin* classFactory(QgisInterface* iface)
{
    return new QgsPointConverterPlugin(iface);
}

QGISEXTERN QString name()
{
    return "point converter plugin";
}

QGISEXTERN QString description()
{
    return "A plugin that converts vector layers to delimited text point files";
}

QGISEXTERN QString version()
{
    return "0.00001";
}

// Return the type (either UI or MapLayer plugin)
QGISEXTERN int type()
{
    return QgisPlugin::UI;
}

// Delete ourself
QGISEXTERN void unload(QgisPlugin* theQgsPointConverterPluginPointer)
{

```

```
delete theQgsPointConverterPluginPointer;  
}
```

Passo 3: Leggere le caratteristiche del punto dal layer attivo e scriverle in file di testo

Per leggere le caratteristiche del punto dal layer attivo è necessario specificare il layer corrente e la posizione del nuovo file di testo. Quindi si ripete per tutte le caratteristiche dell layer corrente, si convertono le geometrie (vertici) a punti, si apre un nuovo file e si usa QTextStream per scrivere le coordinate X e Y.

f) Aprire qgspointconverterplugin.h di nuovo e estendere i contenuti esistenti a

```
class QgsGeometry;  
class QTextStream;  
  
private:  
  
void convertPoint(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;  
void convertMultiPoint(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;  
void convertLineString(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;  
void convertMultiLineString(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;  
void convertPolygon(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;  
void convertMultiPolygon(QgsGeometry* geom, const QString& attributeString, \  
    QTextStream& stream) const;
```

g) Aprire qgspointconverterplugin.cpp di nuovo e estendere i contenuti esistenti a:

```
#include "qgsgeometry.h"  
#include "qgsvectordataproducer.h"  
#include "qgsvectorlayer.h"  
#include <QFileDialog>  
#include <QMessageBox>  
#include <QTextStream>  
  
void QgsPointConverterPlugin::convertToPoint()  
{
```

```
qWarning("in method convertToPoint");
QgsMapLayer* theMapLayer = mIface->activeLayer();
if(!theMapLayer)
{
    QMessageBox::information(0, tr("no active layer"), \
        tr("this plugin needs an active point vector layer to make conversions \
            to points"), QMessageBox::Ok);
    return;
}
QgsVectorLayer* theVectorLayer = dynamic_cast<QgsVectorLayer*>(theMapLayer);
if(!theVectorLayer)
{
    QMessageBox::information(0, tr("no vector layer"), \
        tr("this plugin needs an active point vector layer to make conversions \
            to points"), QMessageBox::Ok);
    return;
}

QString fileName = QFileDialog::getSaveFileName();
if(!fileName.isNull())
{
    qWarning("The selected filename is: " + fileName);
    QFile f(fileName);
    if(!f.open(QIODevice::WriteOnly))
    {
        QMessageBox::information(0, "error", "Could not open file", QMessageBox::Ok);
        return;
    }
    QTextStream theTextStream(&f);
    theTextStream.setRealNumberNotation(QTextStream::FixedNotation);

    QgsFeature currentFeature;
    QgsGeometry* currentGeometry = 0;

    QgsVectorDataProvider* provider = theVectorLayer->dataProvider();
    if(!provider)
    {
        return;
    }

    theVectorLayer->select(provider->attributeIndexes(), \
        theVectorLayer->extent(), true, false);
```

```
//write header
theTextStream << "x,y";
theTextStream << endl;

while(theVectorLayer->nextFeature(currentFeature))
{
    QString featureAttributesString;

    currentGeometry = currentFeature.geometry();
    if(!currentGeometry)
    {
        continue;
    }

    switch(currentGeometry->wkbType())
    {
        case QGis::WKBPoint:
        case QGis::WKBPoint25D:
            convertPoint(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBMultiPoint:
        case QGis::WKBMultiPoint25D:
            convertMultiPoint(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBLineString:
        case QGis::WKBLineString25D:
            convertLineString(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBMultiLineString:
        case QGis::WKBMultiLineString25D:
            convertMultiLineString(currentGeometry, featureAttributesString \
theTextStream);
            break;

        case QGis::WKBPolygon:
```

```
        case QGis::WKBPolygon25D:
            convertPolygon(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBMultiPolygon:
        case QGis::WKBMultiPolygon25D:
            convertMultiPolygon(currentGeometry, featureAttributesString, \
theTextStream);
            break;
    }
}

//geometry converter functions
void QgsPointConverterPlugin::convertPoint(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsPoint p = geom->asPoint();
    stream << p.x() << "," << p.y();
    stream << endl;
}

void QgsPointConverterPlugin::convertMultiPoint(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsMultiPoint mp = geom->asMultiPoint();
    QgsMultiPoint::const_iterator it = mp.constBegin();
    for(; it != mp.constEnd(); ++it)
    {
        stream << (*it).x() << "," << (*it).y();
        stream << endl;
    }
}

void QgsPointConverterPlugin::convertLineString(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsPolyline line = geom->asPolyline();
    QgsPolyline::const_iterator it = line.constBegin();
    for(; it != line.constEnd(); ++it)
```

```
{
    stream << (*it).x() << "," << (*it).y();
    stream << endl;
}
}
```

```
void QgsPointConverterPlugin::convertMultiLineString(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
```

```
{
    QgsMultiPolyline ml = geom->asMultiPolyline();
    QgsMultiPolyline::const_iterator lineIt = ml.constBegin();
    for(; lineIt != ml.constEnd(); ++lineIt)
    {
        QgsPolyline currentPolyline = *lineIt;
        QgsPolyline::const_iterator vertexIt = currentPolyline.constBegin();
        for(; vertexIt != currentPolyline.constEnd(); ++vertexIt)
        {
            stream << (*vertexIt).x() << "," << (*vertexIt).y();
            stream << endl;
        }
    }
}
```

```
void QgsPointConverterPlugin::convertPolygon(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
```

```
{
    QgsPolygon polygon = geom->asPolygon();
    QgsPolygon::const_iterator it = polygon.constBegin();
    for(; it != polygon.constEnd(); ++it)
    {
        QgsPolyline currentRing = *it;
        QgsPolyline::const_iterator vertexIt = currentRing.constBegin();
        for(; vertexIt != currentRing.constEnd(); ++vertexIt)
        {
            stream << (*vertexIt).x() << "," << (*vertexIt).y();
            stream << endl;
        }
    }
}
```

```
void QgsPointConverterPlugin::convertMultiPolygon(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
```

```

{
    QgsMultiPolygon mp = geom->asMultiPolygon();
    QgsMultiPolygon::const_iterator polyIt = mp.constBegin();
    for(; polyIt != mp.constEnd(); ++polyIt)
    {
        QgsPolygon currentPolygon = *polyIt;
        QgsPolygon::const_iterator ringIt = currentPolygon.constBegin();
        for(; ringIt != currentPolygon.constEnd(); ++ringIt)
        {
            QgsPolyline currentPolyline = *ringIt;
            QgsPolyline::const_iterator vertexIt = currentPolyline.constBegin();
            for(; vertexIt != currentPolyline.constEnd(); ++vertexIt)
            {
                stream << (*vertexIt).x() << "," << (*vertexIt).y();
                stream << endl;
            }
        }
    }
}

```

Passo 4: Copiare gli attributi delle caratteristiche in un file di testo

Alla fine si estraggono gli attributi dal layer attivo usando `QgsVectorDataProvider::fieldNameMap()`. Per ogni caratteristica si estraggono i valori di campo usando `QgsFeature::attributeMap()` e si aggiungono i contenuti separati da virgole di seguito alle coordinate X e Y per ogni nuovo punto. Per questo step non c'è bisogno di alcun altro cambiamento in `qgspointconverterplugin.h`

h) Aprire `qgspointconverterplugin.cpp` Anuovo e estendere i contenuti esistenti a:

```

#include "qgspointconverterplugin.h"
#include "qgisinterface.h"
#include "qgsgeometry.h"
#include "qgsvectordataplayer.h"
#include "qgsvectorlayer.h"
#include <QAction>
#include <QFileDialog>
#include <QMessageBox>
#include <QTextStream>

#ifdef WIN32
#define QGISEXTERN extern "C" __declspec( dllexport )
#else

```



```
#define QGISEXTERN extern "C"
#endif

QgsPointConverterPlugin::QgsPointConverterPlugin(QgisInterface* iface): \
mIface(iface), mAction(0)
{

}

QgsPointConverterPlugin::~QgsPointConverterPlugin()
{

}

void QgsPointConverterPlugin::initGui()
{
    mAction = new QAction(tr("&Convert to point"), this);
    connect(mAction, SIGNAL(activated()), this, SLOT(convertToPoint()));
    mIface->addToolBarIcon(mAction);
    mIface->addPluginToMenu(tr("&Convert to point"), mAction);
}

void QgsPointConverterPlugin::unload()
{
    mIface->removeToolBarIcon(mAction);
    mIface->removePluginMenu(tr("&Convert to point"), mAction);
    delete mAction;
}

void QgsPointConverterPlugin::convertToPoint()
{
    qWarning("in method convertToPoint");
    QgsMapLayer* theMapLayer = mIface->activeLayer();
    if(!theMapLayer)
    {
        QMessageBox::information(0, tr("no active layer"), \
            tr("this plugin needs an active point vector layer to make conversions \
                to points"), QMessageBox::Ok);
        return;
    }
    QgsVectorLayer* theVectorLayer = dynamic_cast<QgsVectorLayer*>(theMapLayer);
    if(!theVectorLayer)
```

```
{
    QMessageBox::information(0, tr("no vector layer"), \
        tr("this plugin needs an active point vector layer to make conversions \
            to points"), QMessageBox::Ok);
    return;
}

QString fileName = QFileDialog::getSaveFileName();
if(!fileName.isNull())
{
    qWarning("The selected filename is: " + fileName);
    QFile f(fileName);
    if(!f.open(QIODevice::WriteOnly))
    {
        QMessageBox::information(0, "error", "Could not open file", QMessageBox::Ok);
        return;
    }
    QTextStream theTextStream(&f);
    theTextStream.setRealNumberNotation(QTextStream::FixedNotation);

    QgsFeature currentFeature;
    QgsGeometry* currentGeometry = 0;

    QgsVectorDataProvider* provider = theVectorLayer->dataProvider();
    if(!provider)
    {
        return;
    }

    theVectorLayer->select(provider->attributeIndexes(), \
        theVectorLayer->extent(), true, false);

    //write header
    theTextStream << "x,y";
    QMap<QString, int> fieldMap = provider->fieldNameMap();
    //We need the attributes sorted by index.
    //Therefore we insert them in a second map where key / values are exchanged
    QMap<int, QString> sortedFieldMap;
    QMap<QString, int>::const_iterator fieldIt = fieldMap.constBegin();
    for(; fieldIt != fieldMap.constEnd(); ++fieldIt)
    {
        sortedFieldMap.insert(fieldIt.value(), fieldIt.key());
    }
}
```

```
    }

    QMap<int, QString>::const_iterator sortedFieldIt = sortedFieldMap.constBegin();
    for(; sortedFieldIt != sortedFieldMap.constEnd(); ++sortedFieldIt)
    {
        theTextStream << "," << sortedFieldIt.value();
    }

    theTextStream << endl;

    while(theVectorLayer->nextFeature(currentFeature))
    {
        QString featureAttributesString;
        const QgsAttributeMap& map = currentFeature.attributeMap();
        QgsAttributeMap::const_iterator attributeIt = map.constBegin();
        for(; attributeIt != map.constEnd(); ++attributeIt)
        {
            featureAttributesString.append(",");
            featureAttributesString.append(attributeIt.value().toString());
        }

        currentGeometry = currentFeature.geometry();
        if(!currentGeometry)
        {
            continue;
        }

        switch(currentGeometry->wkbType())
        {
            case QGis::WKBPoint:
            case QGis::WKBPoint25D:
                convertPoint(currentGeometry, featureAttributesString, \
theTextStream);
                break;

            case QGis::WKBMultiPoint:
            case QGis::WKBMultiPoint25D:
                convertMultiPoint(currentGeometry, featureAttributesString, \
theTextStream);
                break;
```

```
        case QGis::WKBLineString:
        case QGis::WKBLineString25D:
            convertLineString(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBMultiLineString:
        case QGis::WKBMultiLineString25D:
            convertMultiLineString(currentGeometry, featureAttributesString \
theTextStream);
            break;

        case QGis::WKBPolygon:
        case QGis::WKBPolygon25D:
            convertPolygon(currentGeometry, featureAttributesString, \
theTextStream);
            break;

        case QGis::WKBMultiPolygon:
        case QGis::WKBMultiPolygon25D:
            convertMultiPolygon(currentGeometry, featureAttributesString, \
theTextStream);
            break;
    }
}
}

//geometry converter functions
void QgsPointConverterPlugin::convertPoint(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsPoint p = geom->asPoint();
    stream << p.x() << "," << p.y();
    stream << attributeString;
    stream << endl;
}

void QgsPointConverterPlugin::convertMultiPoint(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsMultiPoint mp = geom->asMultiPoint();
```

```
QgsMultiPoint::const_iterator it = mp.constBegin();
for(; it != mp.constEnd(); ++it)
{
    stream << (*it).x() << "," << (*it).y();
    stream << attributeString;
    stream << endl;
}
}
```

```
void QgsPointConverterPlugin::convertLineString(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsPolyline line = geom->asPolyline();
    QgsPolyline::const_iterator it = line.constBegin();
    for(; it != line.constEnd(); ++it)
    {
        stream << (*it).x() << "," << (*it).y();
        stream << attributeString;
        stream << endl;
    }
}
```

```
void QgsPointConverterPlugin::convertMultiLineString(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsMultiPolyline ml = geom->asMultiPolyline();
    QgsMultiPolyline::const_iterator lineIt = ml.constBegin();
    for(; lineIt != ml.constEnd(); ++lineIt)
    {
        QgsPolyline currentPolyline = *lineIt;
        QgsPolyline::const_iterator vertexIt = currentPolyline.constBegin();
        for(; vertexIt != currentPolyline.constEnd(); ++vertexIt)
        {
            stream << (*vertexIt).x() << "," << (*vertexIt).y();
            stream << attributeString;
            stream << endl;
        }
    }
}
```

```
void QgsPointConverterPlugin::convertPolygon(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
```

```

{
    QgsPolygon polygon = geom->asPolygon();
    QgsPolygon::const_iterator it = polygon.constBegin();
    for(; it != polygon.constEnd(); ++it)
    {
        QgsPolyline currentRing = *it;
        QgsPolyline::const_iterator vertexIt = currentRing.constBegin();
        for(; vertexIt != currentRing.constEnd(); ++vertexIt)
        {
            stream << (*vertexIt).x() << "," << (*vertexIt).y();
            stream << attributeString;
            stream << endl;
        }
    }
}

void QgsPointConverterPlugin::convertMultiPolygon(QgsGeometry* geom, const QString& \
attributeString, QTextStream& stream) const
{
    QgsMultiPolygon mp = geom->asMultiPolygon();
    QgsMultiPolygon::const_iterator polyIt = mp.constBegin();
    for(; polyIt != mp.constEnd(); ++polyIt)
    {
        QgsPolygon currentPolygon = *polyIt;
        QgsPolygon::const_iterator ringIt = currentPolygon.constBegin();
        for(; ringIt != currentPolygon.constEnd(); ++ringIt)
        {
            QgsPolyline currentPolyline = *ringIt;
            QgsPolyline::const_iterator vertexIt = currentPolyline.constBegin();
            for(; vertexIt != currentPolyline.constEnd(); ++vertexIt)
            {
                stream << (*vertexIt).x() << "," << (*vertexIt).y();
                stream << attributeString;
                stream << endl;
            }
        }
    }
}

QGISEXTERN QgisPlugin* classFactory(QgisInterface* iface)
{
    return new QgsPointConverterPlugin(iface);
}

```

```
}

QGISEXTERN QString name()
{
    return "point converter plugin";
}

QGISEXTERN QString description()
{
    return "A plugin that converts vector layers to delimited text point files";
}

QGISEXTERN QString version()
{
    return "0.00001";
}

// Return the type (either UI or MapLayer plugin)
QGISEXTERN int type()
{
    return QgisPlugin::UI;
}

// Delete ourself
QGISEXTERN void unload(QgisPlugin* theQgsPointConverterPluginPointer)
{
    delete theQgsPointConverterPluginPointer;
}
```

14.3 Ulteriori informazioni

Come si vede, c'è necessità di informazioni da diverse fonti per scrivere QGIS plugin in C++. Gli sviluppatori di plugin devono conoscere C++, l'interfaccia plugin di QGIS, nonché le classi e gli strumenti Qt4. All'inizio è meglio imparare dagli esempi e copiare i meccanismi dei plugin esistenti.

Online è disponibile una documentazione che può risultare utile per i programmatori QGIS C++:

- Debugging dei plugin QGIS: <http://wiki.qgis.org/qgiswiki/DebuggingPlugins>
- Documentazione QGIS API: http://svn.qgis.org/api_doc/html/
- DocumentationQt: <http://doc.trolltech.com/4.3/index.html>

15 Scrivere un Plugin QGIS in Python

In questa sezione si può trovare una guida per principianti per scrivere un plugin QGIS Python. È basata sul workshop *Extending the Functionality of QGIS with Python Plugins* che tenute al FOSS4G 2008 dal Dr. Marco Hugentobler, Dr. Horst Düster and Tim Sutton.

Oltre a scrivere un plugin QGIS Python, è anche possibile usare PyQGIS da una console linea di comando python che è prevalentemente interessante per il debugging o per scrivere applicativi in Python non collegati in rete con le loro proprie interfacce usando la funzionalità della libreria QGIS core.

15.1 Perché Python e circa la licenza

Python è un linguaggio di scrittura realizzato con lo scopo di essere facile da programmare. Ha un meccanismo che rilascia automaticamente la memoria non più in uso (collettore di spazzatura). Un ulteriore vantaggio è che in molti programmi scritti in C++ o Java offre la possibilità di scrivere estensioni in Python, ad es. OpenOffice o Gimp. Quindi è un buon investimento di tempo imparare il linguaggio Python.

I plugin PyQGIS usano funzionalità di `libqgis_core.so` e `libqgis_gui.so`. Dato che entrambi sono licenziati sotto GNU GPL, anche i plugin QGIS Python devono essere licenziati sotto GPL. Questo significa che si possono usare i propri plugin per qualsiasi scopo e non si è obbligati a pubblicarli. Comunque se li si pubblica devono essere pubblicati sotto le condizioni di licenza GPL.

15.2 Cosa è necessario installare per iniziare

Nei computer di laboratorio, tutto per il workshop è già installato. Se si programmano i plugin Python a casa, saranno necessarie le seguenti librerie e programmi:

- QGIS
- Python
- Qt
- PyQt
- strumenti di sviluppo PyQt

Se si usa Linux, ci sono pacchetti binary per tutte le maggiori distribuzioni. Per Windows, l'installatore PyQt contiene già Qt, PyQt ed gli strumenti di sviluppo PyQt. development tools.




15.3 Programmare un semplice plugin PyQGIS in quattro passaggi


Il plugin dell'esempio è volutamente tenuto semplice. Esso aggiunge un pulsante alla barra menu di QGIS. Se il pulsante viene premuto, appare una finestra di dialogo da cui l'utente può caricare un file shape.

Per ogni plugin python, deve essere creata una cartella dedicata che contiene i file del plugin. In modo predefinito, QGIS cerca i plugins in due posizioni: `$QGIS_DIR/share/qgis/python/plugins` e `$HOME/.qgis/python/plugins`. Notare che i plugin installati nella seconda posizione sono visibili per un solo utente.

Passo 1: Far sì che il gestore dei plugin riconosca il plugin

Ogni plugin Python è contenuto nella sua propria directory. Quando QGIS si avvia, esso scansiona ogni subdirectory specifica del sistema operativo e inizializza ogni plugin che trova.

-  Linux e altri unix:
./share/qgis/python/plugins
/home/\$USERNAME/.qgis/python/plugins
-  Mac OS X:
./Contents/MacOS/share/qgis/python/plugins
/Users/\$USERNAME/.qgis/python/plugins
-  Windows:
C:\Program Files\QGIS\python\plugins
C:\Documents and Settings\\$USERNAME\.qgis\python\plugins

Fatto questo, il plugin sarà visibile nel  **Gestore dei Plugin...**

Tip 44 DUE CARTELLE QGIS PYTHON PLUGIN

Ci sono due directory che contengono i plugin Python. `$QGIS_DIR/share/qgis/python/plugins` è ideata principalmente per i plugin core mentre `$HOME/.qgis/python/plugins` per la installazione semplice dei plugin estreni. I plugin nella posizione home sono visibili per un solo utente, ma mascherano anche i plugin core con lo stesso nome, che può essere usato per fornire l'aggiornamento dei plugin principali

Per fornire le necessarie informazioni per QGIS, il plugin deve implementare i metodi `name()`, `description()`, `version()`, `qgisMinimumVersion()` e `authorName()` che restituiscono delle stringhe descrittive. Il `qgisMinimumVersion()` dovrebbe restituire un modulo semplice, ad es. 1.0. Un plugin deve anche avere un metodo `classFactory(QgisInterface)` che è chiamato dal gestore dei plugin a creare un'istanza del plugin. L'argomento di tipo `QgisInterface` viene usato dal plugin per accedere alle funzioni della istanza di QGIS. Si lavorerà con quest'oggetto nel passaggio 2.

Notare che, in contrasto ad altri linguaggi di programmazione, l'indentatura è molto importante. L'interprete Python lancia un messaggio di errore se non è corretta.

Per il nostro plugin si crea la cartella plugin 'foss4g_plugin' in `$HOME/.qgis/python/plugins`. Quindi si aggiungono i due nuovi file di testo in questa cartella, `foss4gplugin.py` e `__init__.py`.

Il file `foss4gplugin.py` contiene la classe plugin:

```
# -*- coding: utf-8 -*-
# Import the PyQt and QGIS libraries
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from qgis.core import *
# Initialize Qt resources from file resources.py
import resources

class FOSS4GPlugin:

    def __init__(self, iface):
        # Save reference to the QGIS interface
        self.iface = iface


    def initGui(self):
        print 'Initialising GUI'

    def unload(self):
        print 'Unloading plugin'
```

Il file `__init__.py` contiene i metodi `name()`, `description()`, `version()`, `qgisMinimumVersion()` e `authorName()` and `classFactory`. Poichè si sta creando una nuova istanza della classe del plugin, è necessario importare il codice di questa classe:

```
# -*- coding: utf-8 -*-
from foss4gplugin import FOSS4GPlugin
def name():
    return "FOSS4G example"
def description():
    return "A simple example plugin to load shapefiles"
def version():
    return "0.1"
def qgisMinimumVersion():
    return "1.0"
def authorName():
    return "John Developer"
def classFactory(iface):
```

```
return FOSS4GPlugin(iface)
```

A questo punto il plugin ha già la necessaria infrastruttura per apparire nel QGIS  **Gestore dei Plugin...** per essere caricato o scaricato.

Passo 2: Creare un'icona per il plugin

Per rendere l'icona grafica disponibile per il programma, è necessario il cosiddetto file risorsa. In questo file, la grafica è codificata in notazione esadecimale. fortunatamente, non c'è da preoccuparsi della sua rappresentazione perchè si usa il compilatore `pyrcc`, uno strumento che legge il file `resources.qrc` e crea un file risorsa.

I file `foss4g.png` e `resources.qrc` usati in questo piccolo workshop possono essere scaricati da http://karlinapp.ethz.ch/python_foss4g. Salvare questi due file nella directory del plugin esempio `$HOME/.qgis/python/plugins/foss4g_plugin` e accedere: `pyrcc4 -o resources.py resources.qrc`.

Passo 3: Aggiungere un pulsante ed un menu

In questa sezione, si implementa il contenuto dei metodi `initGui()` e `unload()`. Serve una istanza della classe **QAction** che esegua il `run()` metodo del plugin. Con l'oggetto d'azione, si può quindi generare il menu e il pulsante:

```
import resources

def initGui(self):
    # Create action that will start plugin configuration
    self.action = QAction(QIcon(":/plugins/foss4g_plugin/foss4g.png"), "FOSS4G plugin",
self.iface.getMainWindow())
    # connect the action to the run method
    QObject.connect(self.action, SIGNAL("activated()"), self.run)

    # Add toolbar button and menu item
    self.iface.addToolBarIcon(self.action)
    self.iface.addPluginMenu("FOSS-GIS plugin...", self.action)

def unload(self):
    # Remove the plugin menu item and icon
    self.iface.removePluginMenu("FOSSGIS Plugin...", self.action)
    self.iface.removeToolBarIcon(self.action)
```

Passo 4: Caricare un layer da un file shape

In questo passaggio si implementa la reale funzionalità del plugin nel *run()* metodo. Il metodo Qt4 *QFileDialog::getOpenFileName* apre una finestra di dialogo e restituisce il percorso al file scelto. Se l'utente cancella la finestra di dialogo, il percorso è un oggetto nullo, per il quale si testa. Si chiama quindi il metodo *addVectorLayer* dell'oggetto interfaccia che carica il layer. Il metodo necessita di tre soli argomenti: il percorso per il file, il nome del layer che sarà mostrato nella legenda e il nome del fornitore di dati. Per i file shape, questo è 'ogr' perchè internamente QGIS utilizza la library OGR per accedere i file shape:

```
def run(self):
    fileName = QFileDialog.getOpenFileName(None,QString.fromLocal8Bit("Select a file:"),
    "", "*.shp *.gml")
    if fileName.isNull():
        QMessageBox.information(None, "Cancel", "File selection canceled")
    else:
        vlayer = self.iface.addVectorLayer(fileName, "myLayer", "ogr")
```

15.4 Fare il commit del plugin nell'archivio

Se si è scritto un plugin che si considera utile e lo si vuole condividere con gli altri utenti, lo si può caricare nell'archivio QGIS contribuito dagli utenti.

- Preparare una directory del plugin contenente solo i file necessari (assicurarsi che non ci siano file compilati .pyc files, Subversione .svn directory etc).
- Farne un archivio zip, inclusa la directory. Accertarsi che il nome del file zip sia esattamente lo stesso dell' directory contentua (eccetto l'estensione .zip extension naturalmente). In caso contrario l'installatore di plugin non riuscirà a correlare il plugin disponibile con la sua istanza installata localmente.
- Caricarlo nell'archivio: <http://pyqgis.org/admin/contributed> (sarà necessario registrarsi al primo uso). Prestare particolare attenzione al riempimento del modulo. Spesso è il campo Version Number è riempito in maniera errata il che confonde l'installatore di plugin e causa false notificazioni di aggiornamenti disponibili.

15.5 Ulteriori informazioni

Come si vede, sono necessarie informazioni da diverse fonti per scrivere plugin PyQGIS. gli sviluppatori di plugin devono conoscere Python e l'interfaccia plugin QGIS, come anche le classi e gli strumenti Qt4. All'inizio è benene seguire l'esempio e copiare i meccanismi dei plugin già esistenti. Usando

l'installatore di plugin QGIS, che è esso stesso uno dei plugin Python esistenti, è possibile scaricare un gran quantità di plugin Python esistenti e studiarne il comportamento.

Online è disponibile la documentazione che può essere utile per i programmatori PyQGIS:

- QGIS wiki: <http://wiki.qgis.org/qgiswiki/PythonBindings>
- documentazione QGIS API: <http://doc.qgis.org/index.html>
- documentazioneQt: <http://doc.trolltech.com/4.3/index.html>
- PyQt: <http://www.riverbankcomputing.co.uk/pyqt/>
- guida Python: <http://docs.python.org/>
- un libro su desktop GIS e QGIS. Contiene un capitolo sulla programmazione di plugin PyQGIS:
<http://www.pragprog.com/titles/gsdgis/desktop-gis>

Si possono anche scrivere plugin per QGIS in C++. Vedere la Sezione 14 per maggiori informazioni a riguardo.

16 Creare applicazioni C++

Non tutti vogliono un'applicazione GIS completa. Talvolta si vuole solamente avere un widget nella propria applicazione che mostri una mappa mentre lo scopo principale dell'applicazione è un altro. Forse un database frontend con una mappa? Questa Sezione fornisce due semplici esempi di scritti da Tim Sutton. Sono disponibili nell'archivio qgis subversion insieme con altre interessanti guide. L'intero archivio può essere esplorato: https://svn.osgeo.org/qgis/trunk/code_examples/

16.1 Creare un semplice widget mappa

Con questa guida si esplora come creare un semplice widget di mappazione. Esso non farà molto - solo caricare uno shape file e mostrarlo in colori casuali. Ma dovrebbe dare un'idea del potenziale per usare QGIS come un componente di mappazione incorporato. Prima di procedere, molti ringraziamenti a Francis Bolduc che ha scritto l'inizio di questo demo. Egli ha gentilmente acconsentito a rendere il suo lavoro pubblicamente disponibile.

Si comincia con la tipica aggiunta del necessario includes per la nostra applicazione:

```
//  
// QGIS Includes  
//  
#include <qgsapplication.h>  
#include <qgsproviderregistry.h>  
#include <qgssinglesymbolrenderer.h>  
#include <qgsmaplayerregistry.h>  
#include <qgsvectorlayer.h>  
#include <qgsmapcanvas.h>  
//  
// Qt Includes  
//  
#include <QString>  
#include <QApplication>  
#include <QWidget>
```

Si usa QgsApplication invece di Qt's QApplication, e si ottengono alcuni benefici aggiuntivi di vari metodi statici che possono essere usati per localizzare i percorsi della libreria e così via.

Il registro del provider è un singleton che tiene traccia dei plugin del provider di dati vettoriali. Fa tutto il lavoro di caricare i plugin e così via. Il generatore di singolo simbolo è la più basilare classe di simbologia. Genera punti linee o poligoni in un singolo colore che viene scelto predefinitamente

in modo casuale (anche se lo si può impostare in modo personalizzato). Ogni layer vettoriale deve avere una simbologia da esso associata.

Il registro del layer di mapp tiene traccia di tutti i layer in uso. La classe del layer vettoriale eredita da `maplayer` e lo estende ad includere funzionalità specialistiche per dati vettoriali.

Infine il `mapcanvas` è realmente il nocciolo della questione. E' un widget disegnabile su cui sarà disegnata la mappa.

Ora si può andare oltre alla inizializzazione della nostra applicazione....

```
int main(int argc, char ** argv)
{
    // Start the Application
    QgsApplication app(argc, argv, true);

    QString myPluginsDir      = "/home/timlinux/apps/lib/qgis";
    QString myLayerPath       = "/home/timlinux/gisdata/brazil/BR_Cidades/";
    QString myLayerBaseName   = "Brasil_Cap";
    QString myProviderName    = "ogr";
```

Così ora abbiamo un'applicazione qgs e abbiamo definito alcune variabili. Dato che ho testato su Ubuntu 8.10, specifico qui le posizioni dei plugin del provider vettoriale come nella mia directory di installazione sviluppo. Probabilmente avrebbe più senso in generale tenere le librerie QGIS libs in uno dei percorsi di ricerca delle library standard del vostro sistema (ad es. `/usr/lib`) ma per ora andrà bene anche così.

Le prossime due variabili definite qui indicano solo il file shape che si userà (e qui dovrete sostituire i vostri dati).

Il nome del provider è importante - dice a QGIS che provider di dati usare per caricare il file. Tipicamente si userà 'ogr' o 'postgres'.

Ora possiamo creare il layer oggetto.

```
// Instantiate Provider Registry
QgsProviderRegistry::instance(myPluginsDir);
```

Prima si inizializza il registro del provider. E' una classe singleton quindi si usa un'istanza di chiamata statica e si passa il percorso di ricerca della library del provider. Appena inizializzato scandirà questo percorso alla ricerca di librerie del provider.

Ora si crea un layer...

```

QgsVectorLayer * mypLayer =
    new QgsVectorLayer(myLayerPath, myLayerBaseName, myProviderName);
QgsSingleSymbolRenderer *mypRenderer = new
QgsSingleSymbolRenderer(mypLayer->geometryType());
QList <QgsMapCanvasLayer> myLayerSet;

mypLayer->setRenderer(mypRenderer);
if (mypLayer->isValid())
{
    qDebug("Layer is valid");
}
else
{
    qDebug("Layer is NOT valid");
}

// Add the Vector Layer to the Layer Registry
QgsMapLayerRegistry::instance()->addMapLayer(mypLayer, TRUE);
// Add the Layer to the Layer Set
myLayerSet.append(QgsMapCanvasLayer(mypLayer, TRUE));

```

Il codice è adeguatamente auto esplicativo qui. Si crea un layer usando le variabili definite prima. Poi si assegna al layer un generatore di immagini. Quando si crea un generatore di immagini, dobbiamo specificare il tipo di geometria, cosa che si fa chiedendo allo stato vettoriale il suo tipo di geometria. Poi si aggiunge il layer ad un set di layer (che è usato da QgsMapCanvas per tenere traccia di quali layer generare graficamente e in che ordine) e al registro di maplayer. Infine ci si assicura che il layer sarà visibile. Ora si crea un map canvas sul quale disegnare il layer.

```

// Create the Map Canvas
QgsMapCanvas * mypMapCanvas = new QgsMapCanvas(0, 0);
mypMapCanvas->setExtent(mypLayer->extent());
mypMapCanvas->enableAntiAliasing(true);
mypMapCanvas->setCanvasColor(QColor(255, 255, 255));
mypMapCanvas->freeze(false);
// Set the Map Canvas Layer Set
mypMapCanvas->setLayerSet(myLayerSet);
mypMapCanvas->setVisible(true);
mypMapCanvas->refresh();

```

Di nuovo, non c'è niente di particolarmente complesso in questo. Si crea il canvas e si imposta la

sua estensione a quella del layer. Poi si aggiusta un po' il canvas per disegnare vettori a distorsione minima. Infine si imposta il colore di sfondo, si sblocca il canvas, si rende visibile e quindi si aggiorna.

```
// Start the Application Event Loop
return app.exec();
}
```

Nell'ultimo passaggio semplicemente si avvia il loop di eventi Qt ed è fatta. Si può controllare, compilare e lanciare usando cmake come questo:

```
svn co
https://svn.osgeo.org/qgis/trunk/code_examples/1_hello_world_qgis_style
cd 1_hello_world_qgis_style
mkdir build
#optionally specify where your QGIS is installed (should work on all
platforms)
#if your QGIS is installed to /usr or /usr/local you can leave this next step
out
export LIB_DIR=/home/timlinux/apps
cmake ..
make
./timtut1
```

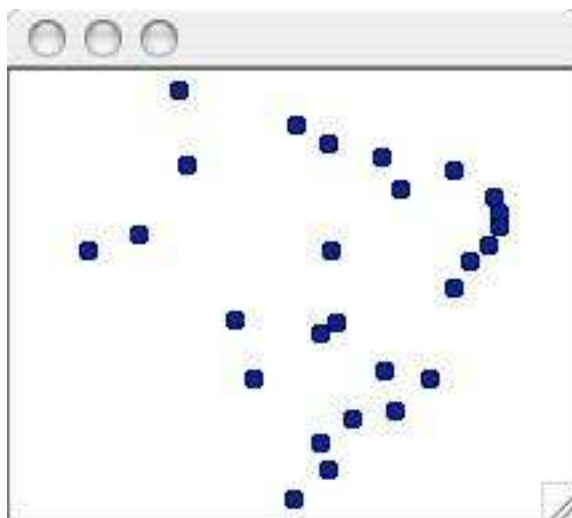
Quando lo si compila e lo si lancia ecco come appare l'applicazione funzionante:

16.2 Lavorare con QgsMapCanvas

nella Sezione 16.1 è stato illustrato l'uso del QgsMapCanvas api per creare una semplice applicazione che carica un file shape e mostra i punti in esso contenuti. Ma a cosa serve una mappa con la quale non si può interagire?

In questa seconda guida la guida precedente viene estesa rendendola un'applicazione QMainWindow con un menu, una barra degli strumenti e un'area canvas. Si illustra come usare QgsMapTool - la classe base per tutti gli strumenti che devono interagire con il canvas della mappa. Lo scopo è fornire un progetto dimostrativo, così non prometto di scrivere il più elegante o il più robusto dei codici C++. Il progetto fornirà 4 icon per barre degli strumenti per

- caricare un layer mappa (il nome del layer è incorporato nell'applicazione)
- ingrandire

Figura 58: Semplice applicazione C++ X

- ridurre
- panoramica

Nella directory di lavoro per il codice guida si trovano un certo numero di file incluse sorgenti c++, icone e un semplice file di dati sotto il nome dati. C'è anche il file .ui per la finestra principale.

Nota: Sarà necessario editare il file .pro nella directory svn soprastante per abbinarlo al proprio sistema.

dato che il codice è lo stesso della precedente guida, ci focalizzeremo sulle specifiche MapTool - il resto dei dettagli di implementazione può essere investigato esplorando il progetto da SVN. Un QgsMapTool è una classe che interagisce con il MapCanvas usando il puntatore del mouse. QGIS ha un gran numero di QgsMapTools implementati, e si possono fare delle sottoclassi in QgsMapTool per crearsi il proprio. In mainwindow.cpp si può vedere che ho incluso dei capipaginay per il QgsMapTools vicino all'inizionale file:

```
//
// QGIS Map tools
//
#include "qgsmaptoolpan.h"
#include "qgsmaptoolzoom.h"
//
// These are the other headers for available map tools
// (not used in this example)
//
// #include "qgsmaptoolcapture.h"
```

```
//#include "qgsmaptoolidentify.h"  
//#include "qgsmaptoolselect.h"  
//#include "qgsmaptoolvertexedit.h"  
//#include "qgsmeasure.h"
```

Come si vede, uso soltanto due tipi di sottoclassi MapTool per questa guida, ma ce ne sono altre disponibili nella libreria QGIS. Collegare MapTools al canvas è molto semplice usando il normale meccanismo Qt4 segnale/slot:

```
//create the action behaviours  
connect(mActionPan, SIGNAL(triggered()), this, SLOT(panMode()));  
connect(mActionZoomIn, SIGNAL(triggered()), this, SLOT(zoomInMode()));  
connect(mActionZoomOut, SIGNAL(triggered()), this, SLOT(zoomOutMode()));  
connect(mActionAddLayer, SIGNAL(triggered()), this, SLOT(addLayer()));
```

Quindi si crea una piccola barra degli strumenti per contenere i pulsanti degli strumenti. Da notare che le azioni mpAction* sono state create in designer.

```
//create a little toolbar  
mpMapToolBar = addToolBar(tr("File"));  
mpMapToolBar->addAction(mpActionAddLayer);  
mpMapToolBar->addAction(mpActionZoomIn);  
mpMapToolBar->addAction(mpActionZoomOut);  
mpMapToolBar->addAction(mpActionPan);
```

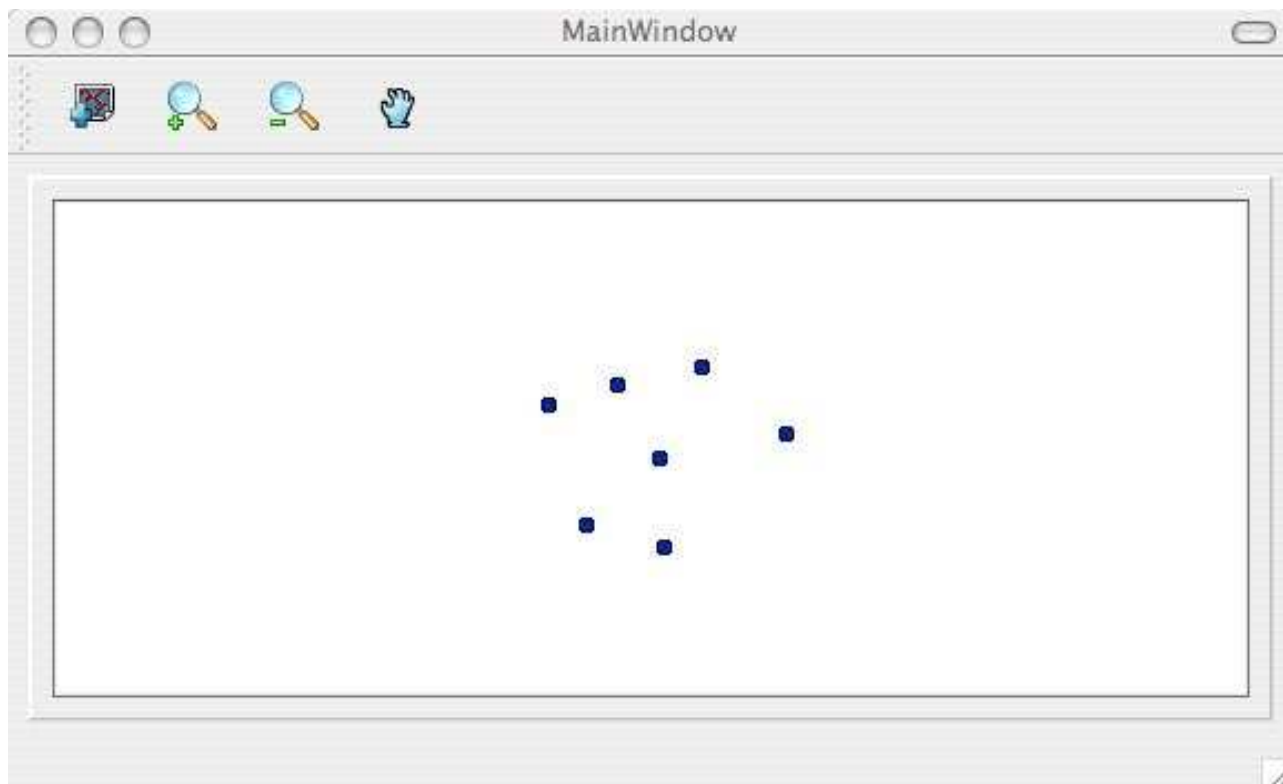
Anche la parte Qt è decisamente diretta. Ora si creano tre strumenti della mappa:

```
//create the maptools  
mpPanTool = new QgsMapToolPan(mpMapCanvas);  
mpPanTool->setAction(mpActionPan);  
mpZoomInTool = new QgsMapToolZoom(mpMapCanvas, FALSE); // false = in  
mpZoomInTool->setAction(mpActionZoomIn);  
mpZoomOutTool = new QgsMapToolZoom(mpMapCanvas, TRUE ); //true = out  
mpZoomOutTool->setAction(mpActionZoomOut);
```

Di nuovo, niente qui è molto complicato - si creano le istanze degli strumenti, ognuno dei quali è associato con lo stesso mapcanvas, ed una diversa QAction. Quando l'utente seleziona una delle icone della barra degli strumenti, il MapTool attivo per il canvas viene impostato. Per esempio, quando l'icona panoramica viene premuta, si fa questo:

```
void MainWindow::panMode()
{
    mpMapCanvas->setMapTool(mpPanTool);
}
```

Figura 59: Applicazione QMainWindow con un menu, una barra degli strumenti e un'area canvas **X**



Conclusioni

Come si vede, ampliare il precedente esempio a qualcosa di più funzionale usando MapTools è veramente molto facile e richiede solo poche righe di codice per ogni MapTool che si voglia fornire.

Si può controllare e costruire questa guida usando SVN e CMake usando i seguenti passaggi:

```
svn co https://svn.osgeo.org/qgis/trunk/code_examples/2_basic_main_window
cd 2_basic_main_window
mkdir build
#optionally specify where your QGIS is installed (should work on all platforms)
#if your QGIS is installed to /usr or /usr/local you can leave this next step out
export LIB_DIR=/home/timlinux/apps
```

```
cmake ..  
make  
./timtut2
```

17 Creare applicazioni PyQGIS

Uno degli obiettivi di QGIS è fornire non soltanto un'applicazione ma anche un set di librerie che possano essere usate per creare nuove applicazioni. Questo obiettivo è stato realizzato mediante la ristrutturazione delle librerie che ha avuto luogo dopo il rilascio della versione 0.8. Dal rilascio della versione 0.9, è possibile lo sviluppo di applicazioni indipendenti usando C++ o Python. Si raccomanda di usare QGIS 1.0.0 o superiore come base per le proprie applicazioni python poiché da questa versione si fornisce un API stabile e coerente.

In questo capitolo si darà un'occhiata al processo di creazione di un'applicazione Python indipendente. Il blog di QGIS ha vari esempi su come creare applicazioni PyQGIS¹². Useremo una di esse come punto di partenza per un'occhiata a come creare un'applicazione.

Le funzioni che vogliamo nell'applicazione sono:

- Caricare un layer vettoriale
- Panoramica
- Ingrandimento e riduzione
- Ingrandimento alla completa estensione del layer
- Impostazione di colori personalizzati al caricamento del layer

Questo è un set di funzioni piuttosto minimo. Cominciamo progettando il GUI usando Qt Designer.

17.1 Progettare il GUI

Dato che stiamo creando un'applicazione minimale, useremo lo stesso approccio per il GUI. Usando Qt Designer, si crea una semplice finestra principale (MainWindow) priva di menu e barre degli strumenti. Questo ci dà una lavagna bianca per lavorare. Per creare la finestra principale:

1. Creare una directory per sviluppare l'applicazione andare in essa
2. Lanciare Qt Designer
3. Dovrebbe apparire la finestra di dialogo New form. Se non appare, scegliere New form... dal menu File.
4. Scegliere Main Window dalla lista the templates/forms
5. Premere Create
6. Ridimensionare la nuova finestra ad una dimensione maneggevole
7. Trovare nella lista il widget Frame (sotto Containers) e trascinarlo nella finestra principale appena creata

¹²Un'applicazione creata usando Python ed i legami QGIS

8. Premere all'esterno della cornice per selezionare l'area della finestra principale
9. Premere lo strumento Lay Out in a Grid. Quando si fa questo, la cornice si allarga a riempire l'intera finestra principale
10. Salvare il modulo come `mainwindow.ui`
11. Exit Qt Designer

Ora compilare il modulo usando l'interfaccia compilatore PyQt:

```
pyuic4 -o mainwindow_ui.py mainwindow.ui
```

Questo crea la sorgente Python per la finestra principale GUI. Poi si deve creare il codice dell'applicazione per riempire la lavagna bianca con alcuni strumenti che si possono usare.

17.2 Creare la Finestra Principale

Adesso siamo pronti a scrivere la classe **MainWindow** che farà il vero lavoro. Dato che ci vogliono parecchie linee, ci daremo un'occhiata a pezzi, cominciando con la sezione importazione e il settaggio dell'ambiente:

```
1 # Loosely based on:
2 #   Original C++ Tutorial 2 by Tim Sutton
3 #   ported to Python by Martin Dobias
4 #   with enhancements by Gary Sherman for FOSS4G2007
5 # Licensed under the terms of GNU GPL 2
6
7 from PyQt4.QtCore import *
8 from PyQt4.QtGui import *
9 from qgis.core import *
10 from qgis.gui import *
11 import sys
12 import os
13 # Import our GUI
14 from mainwindow_ui import Ui_MainWindow
15
16 # Environment variable QGISHOME must be set to the 1.0 install directory
17 # before running this application
18 qgis_prefix = os.getenv("QGISHOME")
```

Un po' di questo dovrebbe apparire familiare dal nostro plugin, specialmente le importazioni Py-Qt4 e QGIS. Alcune cose specifiche da notare sono le importazioni del nostro GUI alla linea 14 e l'importazione nella libreria CORE alla linea 9.

La nostra applicazione necessita di sapere dove trovare l'installazione di QGIS. Per questo, impostiamo l'ambiente variabile QGISHOME in modo che indichi la directory di installazione di QGIS 1.x. Alla linea 20 memorizziamo questo valore dall'ambiente per un uso successivo.

Poi è necessario creare la nostra classe **MainWindow** che conterrà tutta la logica della nostra applicazione.

```

21 class MainWindow(QMainWindow, Ui_MainWindow):
22
23     def __init__(self):
24         QMainWindow.__init__(self)
25
26         # Required by Qt4 to initialize the UI
27         self.setupUi(self)
28
29         # Set the title for the app
30         self.setWindowTitle("QGIS Demo App")
31
32         # Create the map canvas
33         self.canvas = QgsMapCanvas()
34         # Set the background color to light blue something
35         self.canvas.setCanvasColor(QColor(200,200,255))
36         self.canvas.enableAntiAliasing(True)
37         self.canvas.useQImageToRender(False)
38         self.canvas.show()
39
40         # Lay our widgets out in the main window using a
41         # vertical box layout
42         self.layout = QVBoxLayout(self.frame)
43         self.layout.addWidget(self.canvas)
44
45         # Create the actions for our tools and connect each to the appropriate
46         # method
47         self.actionAddLayer = QAction(QIcon("(qgis_prefix + "/share/qgis/themes/classic/mActionZ
48         \
49             "Add Layer", self.frame)
50         self.connect(self.actionAddLayer, SIGNAL("activated()"), self.addLayer)
51         self.actionZoomIn = QAction(QIcon("(qgis_prefix + "/share/qgis/themes/classic/mActionZ
52             "Zoom In", self.frame)

```



```
53     self.connect(self.actionZoomIn, SIGNAL("activated()"), self.zoomIn)
54     self.actionZoomOut = QAction(QIcon("(qgis_prefix + "/share/qgis/themes/classic/mActionZo
55         "Zoom Out", self.frame)
56     self.connect(self.actionZoomOut, SIGNAL("activated()"), self.zoomOut)
57     self.actionPan = QAction(QIcon("(qgis_prefix + "/share/qgis/themes/classic/mActionPan.pn
58         "Pan", self.frame)
59     self.connect(self.actionPan, SIGNAL("activated()"), self.pan)
60     self.actionZoomFull = QAction(QIcon("(qgis_prefix + "/share/qgis/themes/classic/mActionZ
61         "Zoom Full Extent", self.frame)
62     self.connect(self.actionZoomFull, SIGNAL("activated()"),
63         self.zoomFull)
64
65     # Create a toolbar
66     self.toolbar = self.addToolBar("Map")
67     # Add the actions to the toolbar
68     self.toolbar.addAction(self.actionAddLayer)
69     self.toolbar.addAction(self.actionZoomIn)
70     self.toolbar.addAction(self.actionZoomOut);
71     self.toolbar.addAction(self.actionPan);
72     self.toolbar.addAction(self.actionZoomFull);
73
74     # Create the map tools
75     self.toolPan = QgsMapToolPan(self.canvas)
76     self.toolZoomIn = QgsMapToolZoom(self.canvas, False) # false = in
77     self.toolZoomOut = QgsMapToolZoom(self.canvas, True) # true = out
```

Le linee da 21 a 27 costituiscono la dichiarazione di base e l'inizializzazione della **MainWindow** e le impostazioni dell'interfaccia utente usando il metodo *setUpUi*. Questo è richiesto per tutte le applicazioni.

Quindi si imposta il titolo per l'applicazione così dice qualcosa di più interessante che **MainWindow** (linea 30). Una volta che questo è fatto, siamo pronti a completare l'interfaccia utente. Quando la si è creata in Designer, l'abbiamo lasciata abbozzata—solo una finestra principale e una cornice. Si può aver aggiunto un menu ed una barra degli strumenti usando Designer, comunque lo faremo con Python.

Nelle linee 33 fino a 38 si impostano il canvas della mappa, il colore di sfondo a celeste e si abilita la distorsione minima. Si specifica anche di non usare una **QImage** per la rappresentazione grafica (fidatevi a questo riguardo) e poi si imposta il canvas a visibile chiamando il metodo *show*.

Quindi si imposta il layer affinché usi uno schema a riquadro verticale all'interno della cornice e si aggiunge ad esso il canvas della mappa alla linea 43.

Le linee da 48 a 63 impostano le azioni e le connessioni per gli strumenti nella nostra barra degli strumenti. Per ogni strumento, si crea una **QAction** usando l'icona che abbiamo definito nel tema classico di QGIS. Poi colleghiamo il segnale `activated` dallo strumento al metodo nella nostra classe che gestirà l'azione. Questo è simile a come impostare le cose nell'esempio plugin.

Una volta che si hanno l'azione e la connessione, si deve aggiungerle alla barra degli strumenti. Nelle linee da 66 a 72 si crea la barra degli strumenti e si aggiunge ad esse ogni strumento.

Infine si creano tre strumenti mappa per l'applicazione (linee da 75 a 77). Si useranno gli strumenti mappa in un momento dopo aver definito i metodi per rendere l'applicazione funzionale. Vediamo i metodi per gli strumenti mappa.

```

78  # Set the map tool to zoom in
79  def zoomIn(self):
80      self.canvas.setMapTool(self.toolZoomIn)
81
82  # Set the map tool to zoom out
83  def zoomOut(self):
84      self.canvas.setMapTool(self.toolZoomOut)
85
86  # Set the map tool to
87  def pan(self):
88      self.canvas.setMapTool(self.toolPan)
89
90  # Zoom to full extent of layer
91  def zoomFull(self):
92      self.canvas.zoomFullExtent()
```

Per ogni strumento della mappa, è necessario un metodo che corrisponde alla connessione fatta per ogni azione. Nelle linee da 79 a 88 impostiamo un metodo per ognuno dei tre strumenti che interagiscono con la mappa. Quando uno strumento è attivato premendo il suo pulsante nella barra degli strumenti, è attivato il metodo corrispondente che 'dice' al canvas della mappa che quello è lo strumento attivo. Lo strumento attivo governa cosa avviene quando il mouse è premuto nel canvas.

Lo strumento `zoom to full extent` non è uno strumento mappa—svolge il suo lavoro senza richiedere un click sulla mappa. Quando è attivato, viene chiamato il metodo del canvas della mappa `zoomFullExtent` (line 92). Questo completa l'implementazione di tutti i nostri strumenti eccetto uno—lo strumento `Add Layer`. Analizziamolo:

```

93  # Add an OGR layer to the map
94  def addLayer(self):
95      file = QFileDialog.getOpenFileName(self, "Open Shapefile", ".", "Shapefiles
```

```
96     (*.shp)")
97     fileInfo = QFileInfo(file)
98
99     # Add the layer
100     layer = QgsVectorLayer(file, fileInfo.fileName(), "ogr")
101
102     if not layer.isValid():
103         return
104
105     # Change the color of the layer to gray
106     symbols = layer.renderer().symbols()
107     symbol = symbols[0]
108     symbol.setFillColor(QColor.fromRgb(192,192,192))
109
110     # Add layer to the registry
111     QgsMapLayerRegistry.instance().addMapLayer(layer);
112
113     # Set extent to the extent of our layer
114     self.canvas.setExtent(layer.extent())
115
116     # Set up the map canvas layer set
117     cl = QgsMapCanvasLayer(layer)
118     layers = [cl]
119     self.canvas.setLayerSet(layers)
```

Nel metodo *addLayer* si usa **QFileDialog** per trovare il nome del file da caricare. Questo è fatto alla linea 96. Va notato che si specifica un 'filtro' così la finestra di dialogo mostrerà solamente file di tipo *.shp*.

Di seguito nella 97 si crea un oggetto **QFileInfo** dal percorso del file shape. Ora il layer è pronto per essere creato nella linea 100. Usando l'oggetto **QFileInfo** per arrivare al file dal percorso abbiamo specificato il nome del layer quando esso viene creato. Per essere sicuri che il layer sia valido e non causi alcun problem quando ccaricato, controlliamolo nella linea 102. Se non va bene, lo abbandoniamo e non lo aggiungiamo al canvas della mappa.

Normalmente i layer sono aggiunti con un colore random. Ora si vogliono modificare i colori del layer per rendere la vista più piacevole. Inoltre sappiamo che si deve aggiungere il layer *world_borders* alla mappa e questo lo renderà più bello sul nostro sfondo blu. Per cambiare il colore, è necessario prendere il simbolo usato per la rappresentazione grafica e usarlo per impostare un nuovo colore di riempimento. Questo è fatto dalla linea 106 alla 108.

Tutto ciò che rimane ora è aggiungere il layer al registro e alcune alltre voci di manutenzione (linee da 111 a 119). Questa roba è standard nell'aggiungere un layer ed il risultato finale sono i bordi del

mondo su uno sfondo celeste. La sola cosa che si potrebbe non voler impostare è l'estensione del layer, se si deve aggiungere più di un layer nell'applicazione.

Questo è il cuore dell'applicazione e completa la classe **MainWindow**.

17.3 Conclusione

Il resto del codice scritto sotto crea l'oggetto *QgsApplication*, imposta il percorso all'installazione di QGIS, imposta il metodo *main* e infine avvia l'applicazione. L'unica altra cosa da notare è che noi muoviamo la finestra dell'applicazione all'angolo superiore sinistro dello schermi. Potremmo anche essere fantasiosi e usare il Qt API per centrarla sullo schermo.

```
120 def main(argv):
121     # create Qt application
122     app = QApplication(argv)
123
124     # Initialize qgis libraries
125     QgsApplication.setPrefixPath(qgis_prefix, True)
126     QgsApplication.initQgis()
127
128     # create main window
129     wnd = MainWindow()
130     # Move the app window to upper left
131     wnd.move(100,100)
132     wnd.show()
133
134     # run!
135     retval = app.exec_()
136
137     # exit
138     QgsApplication.exitQgis()
139     sys.exit(retval)
140
141
142 if __name__ == "__main__":
143     main(sys.argv)
```

17.4 Lanciare l'applicazione

Ora si può lanciare l'applicazione e vedere cosa succede. Naturalmente se si è come molti sviluppatori, la si è testata man mano che la si scriveva.

Prima di lanciare l'applicazione, si devono impostare alcune variabili dell'ambiente.





```
export LD_LIBRARY_PATH=$HOME/qgis/lib%$
export PYTHONPATH=$HOME/qgis/share/qgis/python
export QGISHOME=$HOME/qgis%$
```



```
set PATH=C:\qgis;%PATH%
set PYTHONPATH=C:\qgis\python
set QGISHOME=C:\qgis
```

Si assume che

-  QGIS sia installato nella home directory in qgis.
-  QGIS sia installato in C:\qgis.

Quando l'applicazione si avvia, appare così:

Per aggiungere il layer `world_borders`, premere lo strumento Add Layer e navigare alla directory dei dati. Selezionare il file shape e premere **Open** per aggiungerlo alla mappa. Il nostro color di riempimento personalizzato è aggiunto e il risultato è:

Creare un'applicazione PyQGIS è veramente molto facile. In meno di 150 linee di codice avete un'applicazione che può caricare un file shapes e navigare la mappa. Se giocate un po' con la mappa, noterete che funzionano anche alcune delle caratteristiche integrate del canvas, incluso lo scrolling del mouse e lo spostamento premendo e tenendo premuta la barra **Space** e contemporaneamente muovendo il mouse.

Alcune applicazioni sofisticate sono state create con PyQGIS e molte sono in lavorazione. Questo è impressionante, se si considera che questo sviluppo è avvenuto anche prima del rilascio ufficiale di QGIS 1.0.

Tip 45 DOCUMENTAZIONE PER PYQGIS

Se state scrivendo un plugin o un'applicazione PyQGIS, avrete bisogno di far riferimento alle documentazioni QGIS API (<http://doc.qgis.org>) e alla Guida di riferimento di PyQt Python (<http://www.riverbankcomputing.com/Docs/PyQt4/pyqt4ref.html>). Questi documenti forniscono informazioni sulle classi e sui metodi che saranno usati per portare la vostra creazione in Python alla vita.

18 Aiuto e Supporto

18.1 Mailinglist

QGIS è attivamente in sviluppo e come tale non sempre funziona come ci si aspetterebbe. La maniera migliore di avere aiuto è registrarsi in una mailing list di utenti QGIS: qgis-users.

qgis-users

Le tue domande raggiungerà un pubblico più ampio e della risposta beneficeranno anche altri. Ci si può registrare alla mailing list qgis-users visitando il sito:

<http://lists.osgeo.org/mailman/listinfo/qgis-user>

qgis-developer

Se si è sviluppatori alle prese con problemi di natura più tecnica, si può registrarsi alla mailing list qgis-developer:

<http://lists.osgeo.org/mailman/listinfo/qgis-developer>

qgis-commit

Ogni volta che viene fatto un commit all'archivio codici QGIS, viene postata una e-mail in questa lista. Se si vuole essere informati di ogni cambiamento alla base codici attuale, si può registrarsi:

<http://lists.osgeo.org/mailman/listinfo/qgis-commit>

qgis-trac

Questa lista fornisce notifiche e-mail in relazione alla gestione del progetto, inclusi rapporti di mal-funzionamenti, obiettivi, e richieste di funzioni e caratteristiche. Per registrarsi a questa lista:

<http://lists.osgeo.org/mailman/listinfo/qgis-trac>

qgis-community-team

Questa lista si occupa di argomenti come la documentazione, aiuto contestuale, guida utente, esperienza online incluso siti web, blog, mailing list, forum, e impegno di traduzione. Se si vuole anche lavorare su una guida utente, questa lista è un buon punto di partenza per far domande. si può registrarsi a:

<http://lists.osgeo.org/mailman/listinfo/qgis-community-team>

qgis-release-team

Questa lista si occupa di argomenti come il processo di rilascio, la creazione di pacchetti dei codici binari per i diversi sistemi operativi e l'annuncio al mondo del rilascio di nuove versioni. Si può

sottoscrivere a:

<http://lists.osgeo.org/mailman/listinfo/qgis-release-team>

qgis-psc

Questa lista è usata per discutere questioni di pertinenza della Commissione Direttiva, cioè inerenti la generale gestione e direzione di Quantum GIS. Si può sottoscrivere a:

<http://lists.osgeo.org/mailman/listinfo/qgis-psc>

Siete i benvenuti a registrarvi in queste liste. Ricordate di contribuire alle liste fornendo risposte e condividendo le vostre esperienze. Tenete presente che qgis-commit e qgis-trac sono progettate come mezzo di notifica e non per accogliere post degli utenti.

18.2 IRC

Siamo anche presenti su IRC - visitateci registrandovi al canale #qgis su irc.freenode.net. Si prega di attendere un po' per le risposte, dato che molte persone sul canale fanno anche altre cose e quindi ci può volere un po' di tempo per notare la vostra domanda. E' disponibile anche un supporto commerciale per QGIS. Vedere questo sito <http://qgis.org/content/view/90/91> per ulteriori informazioni.

Se vi siete persi una discussione su IRC, non è un problema! Viene mantenuto registro di tutte le discussioni, così è più facile stare al passo. Basta andare su <http://logs.qgis.org> e leggere i log IRC.

18.3 Tracciatore di malfunzionamenti (Bug Tracker)

Mentre la mailing list qgis-users è utile per generiche domande 'come fare xyz in QGIS', potreste voler notificare noi per eventuali malfunzionamenti di QGIS. Potete farlo usando il tracciatore di malfunzionamenti a <https://trac.osgeo.org/qgis/>. Quando si crea un nuovo ticket per un malfunzionamento, fornire un indirizzo e-mail al quale noi possiamo richiedere informazioni aggiuntive.

Per favore ricordate che i vostri malfunzionamenti non sempre hanno la priorità che sperate (a seconda della gravità). Alcuni possono richiedere un significativo sforzo di sviluppo per rimediare e la forza lavoro non è sempre disponibile per questo.

Anche la richiesta di funzionalità può essere inoltrata usando lo stesso sistema di ticket come per i malfunzionamenti. Si prega di assicurarsi di selezionare il tipo *enhancement*.

Se avete trovato un malfunzionamento e l'avete risolto da soli, potete inoltrare anche questa riparazione. Di nuovo, il simpatico sistema di ticket a <https://trac.osgeo.org/qgis/> ha anche questa

tipologa. Selezionare `patch` dal menu dei tipi. Qualcuno degli sviluppatori ne farà una revisione e lo applicherà in QGIS.

Per favore, non allarmatevi se la vostra riparazione non viene applicata subito - gli sviluppatori possono essere impegnati in altri lavori.

18.4 Blog

La comunità QGIS mantiene anche un weblog (BLOG) a <http://blog.qgis.org> che ha alcuni articoli interessanti per gli utenti come per gli sviluppatori. Siete invitati a contribuire al blog dopo esservi registrati!

18.5 Wiki

Infine, manteniamo anche un sito web WIKI a <http://wiki.qgis.org> dove potete trovare una varietà di utili informazioni correlate allo sviluppo di QGIS, ai piani di rilascio di nuove versioni, collegamenti ai siti da cui scaricare, consiglio di traduzione del messaggio e così via. Esplorateli, ci sono cose interessanti dentro!

A Formati supportati

A.1 Formati OGR supportati

Alla data di stesura di questo documento, i seguenti formati sono supportati dalla libreria OGR. I formati che sono supportati anche da QGIS sono indicati in **grassetto**.

- **Arc/Info Binary Coverage**
- Comma Separated Value (.csv)
- DODS/OPeNDAP
- **ESRI Shapefile**
- FMEObjects Gateway
- GML
- IHO S-57 (ENC)
- **Mapinfo File**
- Microstation DGN
- OGDl Vectors
- ODBC
- Oracle Spatial
- PostgreSQL¹³
- **SDTS**
- SQLite
- UK .NTF
- U.S. Census TIGER/Line
- VRT - Virtual Datasource

A.2 Formati raster GDAL supportati

Alla data di stesura di questo documento, la libreria GDAL supporta i seguenti formati. Si noti che non è detto che tutti funzionino in QGIS per varie ragioni. Per esempio, alcuni formati necessitano di librerie esterne con licenza commerciale e quindi non liberamente distribuibili con QGIS. Nella lista dei formati visibile quando si carica un raster in QGIS sono presenti solo quei formati che sono stati ben testati. Altri formati possono essere caricati selezionando il filtro *Tutti gli altri files (*)* filter. I formati dei quali è nonto il funzionamento il QGIS è indicato in **grassetto**.

¹³QGIS implementa le proprie funzioni PostgreSQL. OGR dovrebbe essere compilato con il supporto a PostgreSQL

- **Arc/Info ASCII Grid**
- **Arc/Info Binary Grid (.adf)**
- Microsoft Windows Device Independent Bitmap (.bmp)
- BSB Nautical Chart Format (.kap)
- VTP Binary Terrain Format (.bt)
- CEOS (Spot for instance)
- First Generation USGS DOQ (.doq)
- New Labelled USGS DOQ (.doq)
- Military Elevation Data (.dt0, .dt1)
- ERMapper Compressed Wavelets (.ecw)
- ESRI .hdr Labelled
- ENVI .hdr Labelled Raster
- Envisat Image Product (.n1)
- EOSAT FAST Format
- FITS (.fits)
- Graphics Interchange Format (.gif)
- **GRASS Rasters¹⁴**
- **TIFF / GeoTIFF (.tif)**
- Hierarchical Data Format Release 4 (HDF4)
- **Erdas Imagine (.img)**
- Atlantis MFF2e
- Japanese DEM (.mem)
- **JPEG JFIF (.jpg)**
- JPEG2000 (.jp2, .j2k)
- JPEG2000 (.jp2, .j2k)
- NOAA Polar Orbiter Level 1b Data Set (AVHRR)
- Erdas 7.x .LAN and .GIS
- In Memory Raster
- Atlantis MFF
- Multi-resolution Seamless Image Database MrSID
- NITF

¹⁴Il supporto ai raster GRASS in QGIS è fornito dall'apposito plugin

- NetCDF
- OGD Bridge
- PCI .aux Labelled
- PCI Geomatics Database File
- Portable Network Graphics (.png)
- Netpbm (.ppm,.pgm)
- **USGS SDTS DEM (*CATD.DDF)**
- SAR CEOS
- **USGS ASCII DEM (.dem)**
- X11 Pixmap (.xpm)

B Moduli degli strumenti GRASS

La shell di GRASS avviabile dagli strumenti GRASS fornisce accesso a praticamente tutti (oltre 300) moduli GRASS in modalità riga di comando. Per offrire un ambiente di lavoro più confortevole per l'utente, all'incirca 200 di questi moduli e funzionalità sono disponibili con interfaccia grafica.

B.1 Moduli negli strumenti GRASS per l'importazione e l'esportazione di dati

Questa sezione elenca tutti i moduli con interfaccia grafica presenti tra gli strumenti GRASS per importare ed esportare dati nella location e mapset impostato.

B.2 Moduli negli strumenti GRASS per la conversione tra tipi di dato

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per convertire dati raster a vettoriali e viceversa nella location e mapset GRASS selezionati.

B.3 Moduli negli strumenti GRASS per la definizione della regione e l'impostazione della proiezione

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per la gestione e la modifica della regione del mapset attuale e per configurare la proiezione.

Tabella 8: Strumenti GRASS: Moduli per l'importazione di dati

Moduli per l'importazione di dati tra gli strumenti GRASS	
Nome del modulo	Scopo
r.in.arc	Converte un file raster in formato ESRI ARC/INFO ascii (GRID) in un livello raster (binario)
r.in.ascii	Converte un file raster in formato testo ASCII in un livello raster (binario)
r.in.aster	Georeferenzia, rettifica, e importa immagini Terra-ASTER imagery e relativi DEM usando il comando gdalwarp
r.in.gdal	Importa raster supportati da GDAL in formato raster binario GRASS
r.in.gdal.loc	Importa raster supportati da GDAL in formato raster binario GRASS e crea una location su misura per la visualizzazione del dato
r.in.gridatb	Importa file GRIDATB.FOR (TOPMODEL) in formato raster GRASS
r.in.mat	Importa file binari MAT-File(v4) in formato raster GRASS
r.in.poly	Crea mappe raster da file ascii descriventi poligoni/linee presenti nella directory attuale
r.in.srtm	Importa file SRTM HGT in GRASS
i.in.spotvgt	Importa file SPOT VGT NDVI in mappe raster
v.in.dxf	Importa livelli vettoriali in formato DXF
v.in.e00	Importa livelli vettoriali ESRI E00
v.in.garmin	Importa vettori da unità/formati gps usando il comando gpstrans
v.in.gpsbabel	Importa vettori da unità/formati gps usando il comando gpsbabel
v.in.mapgen	Importa vettoriali in formato MapGen or MatLab in GRASS
v.in.ogr	Importa livelli vettoriali OGR/PostGIS
v.in.ogr.loc	Importa livelli vettoriali OGR/PostGIS e crea una location su misura per la visualizzazione dei dati
v.in.ogr.all	Importa tutti i livelli vettoriali OGR/PostGIS presenti in una certa sorgente dati
v.in.ogr.all.loc	Importa tutti i livelli vettoriali OGR/PostGIS presenti in una certa sorgente dati e crea una location su misura per la visualizzazione del dato

Tabella 9: Strumenti GRASS: Moduli per per l'esportazione di dati

Moduli per l'esportazione di dati tra gli strumenti GRASS	
Nome modulo	Scopo
r.out.gdal.gtiff	Esporta livelli raster in formato Geo TIFF
r.out.arc	Converte un raster in un file ESRI ARCGRID
r.gridatb	Esporta raster GRASS in file GRIDATB.FOR (TOPMODEL)
r.out.mat	Esporta un raster GRASS verso il formato binario MAT-File
r.out.bin	Esporta un raster GRASS in una matrice binaria
r.out.png	Esporta raster GRASS come immagini non georeferenziate in formato immagine PNG
r.out.ppm	Converte una mappa raster GRASS in formato immagine PPM alla risoluzione impostata per la region in GRASS
r.out.ppm3	Converte 3 livelli raster (R,G,B) GRASS in formato immagine PPM alla risoluzione impostata per la region in GRASS
r.out.pov	Converte un raster in un campo di altezze per POV-Ray
r.out.tiff	Esporta una mappa raster GRASS verso un'immagine in formato TIFF a 8/24bit alla risoluzione impostata per la region in GRASS
r.out.vrml	Esporta una mappa raster in formato Virtual Reality Modeling Language (VRML)
v.out.ogr	Esporta layer vettoriali in vari formati supportati dalla libreria OGR
v.out.ogr.gml	Esporta layer vettoriali verso il formato GML
v.out.ogr.postgis	Esporta layer vettoriali verso vari formati (tramite la libreria OGR)
v.out.ogr.mapinfo	Esporta livelli vettoriali verso il formato Mapinfo
v.out.ascii	Converte un raster binario di GRASS verso il formato vettoriale GRASS ASCII
v.out.dxf	Converte un vettoriale GRASS verso il formato DXF

Tabella 10: Strumenti GRASS: moduli per la conversione tra tipi di dato

Moduli per la conversione tra tipi di dato tra gli strumenti GRASS	
Nome modulo	Scopo
r.to.vect.point	Converte un raster in un vettoriale di punti
r.to.vect.line	Converte un raster in un vettoriale di linee
r.to.vect.area	Converte un raster in un vettoriale di aree
v.to.rast.constant	Converte un vettore ad un raster usando un valore costante per le categorie
v.to.rast.attr	Converte un vettore in un raster usando gli attributi per le categorie

Tabella 11: Strumenti GRASS: moduli per la gestione della regione e della proiezione

Moduli per la gestione della regione e della proiezione tra gli strumenti GRASS	
Nome modulo	Scopo
g.region.save	Salva la regione corrente con nome
g.region.zoom	Restringi la regione corrente fino a quando non incontra celle non vuote di una data mappa raster
g.region.multiple.raster	Imposta la regione in modo che combaci con l'estensione di più raster
g.region.multiple.vector	Imposta la regione in modo che combaci con l'estensione di più vettoriali
g.proj.print	Stampa le informazioni di proiezione per la location corrente
g.proj.geo	Stampa le informazioni di proiezione di un file georeferenziato (raster, vettore o immagine)
g.proj.ascii.new	Stampa le informazioni di proiezione di un file ASCII georeferenziato contenente la descrizione della proiezione in formato WKT
g.proj.proj	Stampa le informazioni di proiezione da un file descrittivo di proiezione in formato PROJ.4
g.proj.ascii.new	Stampa informazioni di proiezione da un file ASCII georeferenziato contenente la descrizione della proiezione in formato WKT e crea una nuova location basata su queste informazioni e sull'estensione di questo file
g.proj.geo.new	Stampa le informazioni di proiezione di un file georeferenziato (raster, vettore o immagine) e crea una nuova location basata su queste informazioni e sull'estensione di questo file
g.proj.proj.new	Stampa le informazioni di proiezione da un file descrittivo di proiezione in formato PROJ.4 e crea una nuova location basata su queste informazioni e sull'estensione di questo file
m.cogo	Una semplice utility per convertire misure di direzione e distanze in coordinate e viceversa. Viene assunto un sistema di coordinate cartesiano

B.4 Moduli negli strumenti GRASS per le operazioni su dati raster

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per l'effettuazione di operazioni su dati raster salvati nella location e nel mapset selezionato.

Tabella 12: Strumenti GRASS: moduli per le operazioni su dati raster

Moduli per le operazioni su dati raster tra gli strumenti GRASS	
Nome modulo	Scopo
r.compress	Comprime e decompime mappe raster
r.region.region	Imposta i limiti della regione di un raster a quella di default
r.region.raster	Imposta i limiti della regione di un raster a quelli dell'estensione di una mappa raster esistente
r.region.vector	Imposta i limiti della regione di un raster a quelli di una mappa vettoriale esistente
r.region.edge	Imposta i limiti della regione di un raster ai valori impostati (n-s-e-w)
r.region.alignTo	Imposta la regione del raster in modo che abbia la risoluzione spaziale e i limiti di una mappa di riferimento indicata
r.null.val	Trasforma il valore di cella specificato in valore nullo
r.null.to	Assegna il valore specificato alle celle nulle
r.quant	Algoritmo per produrre il file dei quantili di una mappa con valori a virgola mobile
r.resamp.stats	Ricampiona una mappa raster usando il metodo dell'aggregazione statistica
r.resamp.interp	Ricampiona una mappa raster per mezzo dell'interpolazione dei valori
r.resample	Ricampiona una mappa raster verso una nuova risoluzione spaziale precedentemente impostata
r.resamp.rst	Reinterpola e esegue l'analisi topografica per mezzo del metodo regularized spline con i parametri tension e smoothing
r.support	Crea, rigenera o modifica i file di supporto di una mappa raster
r.support.stats	Aggiorna le statistiche della mappa raster
r.proj	Riproietta una mappa raster dalla location originale indicata a quella corrente

Tabella 13: Strumenti GRASS: moduli per la gestione del colore dei dati raster

Moduli per la gestione del colore dei dati raster tra gli strumenti GRASS	
Nome modulo	Scopo
r.colors.table	Imposta la tabella colori del raster a quella selezionata tra quelle predefinite
r.colors.rules	Imposta la tabella colori del raster in base alle regole impostate
r.colors.rast	Imposta la tabella colori del raster usando le impostazioni di un raster esistente
r.blend	Miscela le componenti di colore di due mappe raster del valore impostato
r.composite	Miscela le componenti di colore rosso, verde e blu per ottenere un singolo file raster a colori
r.his	Genera mappe raster separate per le componenti rosso, verde e blu combinando i valori di tonalità (hue), intensità (intensity), e saturazione (saturation) (metodo his) di una mappa raster in input specificata dall'utente

Tabella 14: Strumenti GRASS: moduli per le operazioni di processamento spaziale di dati raster

Moduli per l'analisi geospaziale di dati raster tra gli strumenti di GRASS	
Nome modulo	Scopo
r.buffer	Crea un buffer di dimensione impostata sui valori del raster
r.mask	Crea una maschera (MASK) per limitare l'estensione spaziale sulla quale eseguire le operazioni sul raster
r.mapcalc	Esegue operazioni geospaziali sul raster in modo algebrico
r.mapcalculator	Procedura guidata e semplificata per l'esecuzione di operazioni algebriche geospaziali su mappe raster
r.neighbors	Analisi raster di tipo neighbors, con creazione di una nuova mappa raster i cui valori in una certa posizione sono funzione (media, mediana, moda, minimo, massimo...) di un certo numero di valori (size) scelti dall'utente nelle vicinanze di quella posizione
v.neighbors	Creazione di una mappa raster in cui i valori di una certa cella sono funzione dei valori degli attributi assegnati a punti o centroidi presenti nelle vicinanze della medesima entro un certo raggio di ricerca impostato dall'utente
r.cross	Creazione di una nuova mappa raster che contenga un numero di categorie pari alle singole combinazioni di categorie dei raster in input
r.series	Creazione di una mappa raster in cui ogni cella assume il valore stabilito dalla funzione impostata che opera sui valori di cella delle mappe scelte dall'utente
r.patch	Creazione di una singola mappa raster mediante collage di altre mappe raster singole
r.statistics	Fornisce le statistiche di un layer cover sulla base delle sue relazioni con un layer base stabilite dalla funzione specificata
r.cost	Creazione di una mappa raster contenente il costo cumulativo conseguente lo spostamento tra diverse posizioni geografiche su una mappa raster i cui valori di cella rappresentino costi
r.drain	Crea una mappa raster contenente una linea di flusso sulla base dei valori contenuti in una mappa di elevazione
r.shaded.relief	Crea mappe ombreggiate
r.slope.aspect.slope	Genera mappe di acclività da modelli digitali di elevazione
r.slope.aspect.aspect	Genera mappe di orientazione dei versanti da modelli digitali di elevazione
r.param.scale	Estrae i parametri morfometrici di un'area da un modello digitale di elevazione
r.texture	Genera una mappa raster con le caratteristiche tessiturali di una mappa raster (prima serie di indici)
r.texture.bis	Genera una mappa raster con le caratteristiche tessiturali di una mappa raster (seconda serie di indici)
r.los	Analisi di visibilità su mappe raster
r.clump	Raggruppa celle contigue in un'unica categoria
r.grow	Genera una mappa raster in cui le aree contigue vengono accresciute di una cella rispetto alla mappa di partenza
r.thin	Assottiglia gli elementi lineari di una mappa raster

Tabella 15: Strumenti GRASS: moduli per la gestione di superfici

Moduli per la gestione di superfici tra gli strumenti di GRASS	
Nome modulo	Scopo
r.random	Crea una mappa di punti vettoriali campionando una mappa raster in modo casuale
r.random.cells	Genera celle casuali legate tra loro da relazioni spaziali
v.kernel	Genera una mappa raster di densità sulla base di un vettoriale di punti usando un Gaussian kernel
r.contour	Produce una mappa vettoriale delle isoipse a partire da una mappa raster con l'intervallo specificato
r.contour2	Produce una mappa vettoriale delle isoipse elencate dall'utente a partire da una mappa raster
r.surf.fractal	Crea una superficie frattale con dimensione del frattale specificata
r.surf.gauss	Crea una mappa raster della deviazione gaussiana la cui media e deviazione standard è impostata dall'utente
r.surf.random	Genera una mappa raster delle deviazioni random il cui intervallo è impostato dall'utente
r.bilinear	Interpolazione bilineare di mappe raster
v.surf.bispline	Interpolazione bicubica o bilinear spline con regolarizzazione di Tykhonov
r.surf.idw	Interpolazione di superfici 3d con metodo idw (inverse distance weighted)
r.surf.idw2	Altra opzione di generazione di superfici 3d con metodo idw
r.surf.contour	Generazione di superfici 3d da isoipse raster
v.surf.idw	Generazione di superfici 3d per interpolazione di valori di un vettoriale (metodo idw)
v.surf.rst	Generazione di superfici 3d per interpolazione di valori di un vettoriale (metodo Regularized Spline Tension, RST)
r.fillnulls	Riempie le aree prive di dati in un raster usando l'interpolazione con il modulo v.surf.rst

Tabella 16: Strumenti GRASS: moduli per la modifica delle categorie e per le etichette di dati raster

Moduli per le categorie e le etichette tra gli strumenti GRASS	
Nome modulo	Scopo
r.reclass.area.greater	Riclassifica una mappa raster con parcelle di estensioni in ettari superiori al valore stabilito dall'utente
r.reclass.area.lessen	Riclassifica una mappa raster con parcelle di estensioni in ettari inferiori al valore stabilito dall'utente
r.reclass	Riclassifica un raster usando un file di regole preformattato dall'utente
r.recode	Ricodifica una mappa raster
r.rescale	Riscalda gli intervalli dei valori di categoria di una mappa raster

Tabella 17: Strumenti GRASS: moduli per la modellazione idrologica

Moduli per la modellazione idrologica tra gli strumenti GRASS	
Nome modulo	Scopo
r.carve	Trasforma un file vettoriale dei percorsi dei corsi d'acqua in un raster di determinata larghezza e altezza e lo sottrae al DEM di base per dare un DEM in uscita
r.fill.dir	Filtra e genera una mappa di elevazione priva di depressioni e una mappa delle direzioni di flusso a partire da una certa mappa di elevazione
r.lake.xy	Riempie i laghi a partire da un punto di alimentazione di coordinate specificate xy fino ad un livello determinato
r.lake.seed	Riempie i laghi a partire da una mappa raster contenente il punto di alimentazione fino ad un livello determinato
r.topidx	Crea una mappa di volume 3D sulla base di una mappa di elevazione 2D e di una mappa raster di valori
r.basins.fill	Genera una mappa raster dei bacini imbriferi
r.water.outlet	Programma per l'individuazione di bacini imbriferi

Tabella 18: Strumenti GRASS: moduli per la generazione di report e l'analisi statistica

Moduli per la generazione di report e l'analisi statistica tra gli strumenti di GRASS	
Nome modulo	Scopo
r.category	Stampa le categorie e le etichette descrittive delle stesse di una mappa raster specificata dall'utente
r.sum	Somma i valori delle celle raster
r.report	Fornisce le statistiche di una mappa raster
r.average	Individua la media dei valori in una mappa di copertura entro le aree alle quali sia stata assegnata la stessa categoria di una mappa di base specificata dall'utente
r.median	Individua la mediana dei valori in una mappa di copertura entro le aree alle quali sia stata assegnata la stessa categoria di una mappa di base specificata dall'utente
r.mode	Individua la moda dei valori in una mappa di copertura entro le aree alle quali sia stata assegnata la stessa categoria di una mappa di base specificata dall'utente
r.volume	Calcola il volume di gruppi di dati e produce una mappa di punti vettoriale contenente i centroidi di questi raggruppamenti
r.surf.area	Calcola l'area di una rete di punti triangolare 3D regolarmente distribuiti rappresentanti i centri di celle raster
r.univar	Calcola le statistiche univariate di celle non nulle in un raster
r.covar	Fornisce una matrice di covarianza/correlazione a partire da mappe raster specificate dall'utente
r.regression.line	Calcola la regressione lineare tra due mappe raster secondo la formula: $y = a + b * x$
r.coin	Tabella la ricorrenza reciproca (coincidenza) di categorie per due mappe raster

B.5 Moduli negli strumenti GRASS per le operazioni su dati vettoriali

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per l'effettuazione di operazioni su dati vettoriali salvati nella location e nel mapset selezionato.

Tabella 19: Strumenti GRASS: moduli per la modifica di dati vettoriali

Moduli per la modifica di dati vettoriali tra gli strumenti di GRASS	
Nome modulo	Scopo
v.build.all	Ricostruisce la topologia di tutti i vettori del mapset
v.clean.break	Spezza le linee ad ogni intersezione
v.clean.snap	Aggancia le linee ai vertici entro una distanza impostata
v.clean.rmdangles	Rimuove i dangles
v.clean.chdangles	Cambia il dangle dal tipo boundary al tipo linea
v.clean.rmbridge	Rimuove i bridges che collegano un'area e un'isola o due isole
v.clean.chbridge	Cambia il bridge che collega un'area e un'isola o due isole da un tipo ad un altro
v.clean.rmdupl	Rimuove le linee doppie (fare attenzione alle categorie!)
v.clean.rmdac	Rimuove i centroidi doppi in un'area
v.clean.bpol	Spezza i limiti (boundaries) di un poligono in ogni punto condiviso tra due o più poligoni sul quale convergono segmenti con angolo diverso
v.clean.prune	Rimuove i vertici entro un valore impostato da linee e boundaries
v.clean.rmarea	Rimuove aree piccole (rimuove il più lungo boundary con area adiacente)
v.clean.rmline	Rimuove tutte le linee o boundaries di lunghezza zero
v.clean.rmsa	Rimuove piccoli angoli tra linee e nodi
v.type.lb	Converte linee in boundaries
v.type.bl	Converte boundaries in linee
v.type.pc	Converte punti in centroidi
v.type.cp	Converte centroidi in punti
v.centroids	Aggiunge centroidi mancanti a boundaries chiusi
v.build.polylines	Costruisce polilinee da segmenti di linee
v.segment	Crea punti/segmenti da in layer vettoriali di linee e posizioni in input
v.to.points	Crea punti lungo un vettoriale di linee in input
v.parallel	Crea linee parallele a quelle in input
v.dissolve	Dissolve limiti tra aree adiacenti
v.drape	Converte vettori 2D in vettori 3D campionando un raster di elevazione
v.transform	Esegue una trasformazione affine su una mappa vettoriale
v.proj	Consente di riproiettare layer vettoriale
v.support	Aggiorna i metadati di una mappa vettoriale
v.generalize	Generalizzazione di mappe vettoriali

Tabella 20: Strumenti GRASS: moduli per la gestione del collegamento ai database

Moduli per la gestione del collegamento ai database tra gli strumenti di GRASS	
Nome modulo	Scopo
v.db.connect	Connette un vettoriale ad un database
v.db.sconnect	Disconnette un vettoriale da un database
v.db.what.connect	Imposta/Mostra la connessione al database per un certo vettoriale

Tabella 21: Strumenti GRASS: moduli per la modifica delle categorie di un vettoriale

Moduli per la modifica delle categorie di un vettoriale tra gli strumenti di GRASS	
Nome modulo	Scopo
v.category.add	Aggiunge elementi al livello (TUTTI gli elementi del livello selezionato!)
v.category.del	Cancella valori di categoria
v.category.sum	Aggiunge un valore alle categorie esistenti
v.reclass.file	Riclassifica i valori di categoria usando un file di regole
v.reclass.attr	Riclassifica i valori di categoria usando una colonna attributi (di tipo intero positivo)

Tabella 22: Strumenti GRASS: moduli per lavorare con vettoriali di punti

Moduli per lavorare con vettoriali di punti tra gli strumenti di GRASS	
Nome modulo	Scopo
v.in.region	Crea una nuova mappa vettoriale con l'estensione della regione corrente
v.mkgrid.region	Crea una griglia nella regione corrente
v.in.db	Importa punti da una tabella database che ne contiene le coordinate
v.random	Genera una mappa vettoriale GRASS di punti casuali 2D/3D
v.kcv	Raggruppa casualmente punti in set di test
v.outlier	Rimuove estremi da dati vettoriali di punti
v.hull	Crea un poligono convesso
v.delaunay.line	Triangolazione Delaunay per punti (mediante linee)
v.delaunay.area	Triangolazione Delaunay per punti (mediante aree)
v.voronoi.line	Diagrammi Voronoi (mediante linee)
v.voronoi.area	Diagrammi Voronoi (mediante aree)

Tabella 23: Strumenti GRASS: Moduli per l'analisi spaziale di vettoriali e per l'analisi di reti

Moduli per l'analisi spaziale di vettoriali e per l'analisi di reti tra gli strumenti di GRASS	
Nome modulo	Scopo
v.extract.where	Seleziona elementi sulla base di attributi
v.extract.list	Estrae gli elementi selezionati in base alla lista
v.select.overlap	Seleziona gli elementi sovrapposti da quelli di un'altra mappa
v.buffer	Creazione di buffer su vettoriali
v.distance	Trova nel vettore 'to' l'elemento più vicino al vettore 'from'
v.net.nodes	Crea nodi su una rete
v.net.alloc	Alloca la rete
v.net.iso	Taglia la rete secondo isolinee di costo
v.net.salesman	Connette nodi lungo la strada più corta (commesso viaggiatore)
v.net.steiner	Connette i nodi selezionati lungo la diramazione più corta (Steiner tree)
v.patch	Crea una nuova mappa vettoriale combinando altre mappe
v.overlay.or	Unione logica di vettori
v.overlay.and	Intersezione logica di vettori
v.overlay.not	Sottrazione logica di vettori
v.overlay.xor	Non-intersezione logica di vettori

Tabella 24: Strumenti GRASS: moduli per l'aggiornamenti di vettoriali sulla base di altre mappe

Moduli per l'aggiornamenti di vettoriali sulla base di altre mappe tra gli strumenti di GRASS	
Nome modulo	Scopo
v.rast.stats	Calcola le statistiche univariate da una mappa raster GRASS sulla base di oggetti vettoriali
v.what.vect	Carica mappe delle quali editare la tabella attributi
v.what.rast	Carica i valori del raster nella posizione di punti vettoriali nella tabella attributi di questi ultimi
v.sample	Campiona un file raster nella posizione dei punti vettoriali

Tabella 25: Strumenti GRASS: moduli per i report e le statistiche su un vettoriale

Moduli per i report e le statistiche su un vettoriale tra gli strumenti di GRASS	
Nome modulo	Scopo
v.to.db	Inserisce le variabili geometriche nel database del vettoriale
v.report	Fornisce un report delle statistiche sulla geometria per i vettoriali
v.univar	Calcola le statistiche univariate di una colonna scelta nella tabella attributi di un vettoriale
v.normal	Test di normalità per punti vettoriali

B.6 Moduli negli strumenti GRASS per le operazioni su immagini

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per modificare ed analizzare immagini salvate nella location e nel mapset selezionato.

Tabella 26: Strumenti GRASS: moduli per l'analisi di immagini

Moduli per l'analisi di immagini tra gli strumenti di GRASS	
Nome modulo	Scopo
i.image.mosaik	Mosaica fino a 4 immagini
i.rgb.his	Funzione per la trasformazione del colore di un'immagine dallo spazio Red Green Blue (RGB) a quello Hue Intensity Saturation (HIS)
i.his.rgb	Funzione per la trasformazione del colore di un'immagine dallo spazio Hue Intensity Saturation (HIS) a quello Red Green Blue (RGB)
i.landsat.rgb	Auto-bilanciamento del colore per immagini LANDSAT
i.fusion.brovey	Trasformazione di Brovey per fondere canali multispettrali e canali pancromatici ad alta risoluzione
i.zc	Individuazione degli zero-crossing edge per il processamento di immagini
i.mfilter	
i.tasscap4	Trasformazione Tasseled Cap (Kauth Thomas) per dati LANDSAT-TM 4
i.tasscap5	Trasformazione Tasseled Cap (Kauth Thomas) per dati LANDSAT-TM 5
i.tasscap7	Trasformazione Tasseled Cap (Kauth Thomas) per dati LANDSAT-TM 7
i.fft	Trasformazione fast fourier (FFT) per il processamento di immagini
i.ifft	Trasformazioni inversa fast fourier per il processamento di immagini
r.describe	Stampa una lista chiara di valori di categoria trovati nella mappa raster indicata
r.bitpattern	Confronta bit patterns con una mappa raster
r.kappa	Calcola l'errore matriciale e il parametro kappa per stabilire l'accuratezza dei risultati di una classificazione
i.oif	Calcola il fattore indice ottimale per bande landsat tm

B.7 Moduli negli strumenti GRASS per la gestione dei database

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per gestire, collegare e lavorare con database interni ed esterni. Le operazioni su database esterni sono eseguite tramite OGR e non sono descritte in questa sezione.

Tabella 27: Strumenti GRASS: moduli per i database

Moduli per la gestione ed analisi di database tra gli strumenti di GRASS	
Nome modulo	Scopo
db.connect	Imposta la connessione al db generale per il mapset
db.connect.schema	Imposta la connessione generale al db per il mapset con uno schema
v.db.reconnect.all	Ricollega il vettoriale ad un nuovo database
db.login	Imposta user/password per il driver/database
db.in.ogr	Importa la tabella attributi in vari formati
v.db.addtable	Crea e aggiunge una nuova tabella ad un vettoriale
v.db.addcol	Aggiunge una o più colonne alla tabella attributi associata ad una certa mappa vettoriale
v.db.dropcol	Elimina una colonna dalla tabella attributi collegata ad una certa mappa vettoriale
v.db.renamecol	Rinomina una colonna della tabella attributi associata ad un vettoriale
v.db.update_const	Assegna un nuovo valore costante ad una colonna della tabella attributi di un vettoriale
v.db.update_query	Assegna un nuovo valore costante ad una colonna della tabella attributi di un vettoriale solo per gli elementi per i quali il risultato della query impostata è TRUE
v.db.update_op	Assegna un nuovo valore, risultante da operazioni su una o più colonne esistenti, ad una colonna della tabella attributi di un vettoriale
v.db.update_op_query	Assegna un nuovo valore, risultante da operazioni su una o più colonne esistenti, ad una colonna della tabella attributi di un vettoriale, solo per gli elementi per i quali il risultato della query impostata è TRUE
db.execute	Esegue un'espressione SQL
db.select	Stampa i risultati della selezione dal database secondo la richiesta SQL
v.db.select	Stampa gli attributi di una mappa vettoriale
v.db.select.where	Stampa gli attributi di una mappa vettoriale in base ad una richiesta SQL
v.db.join	Consente di unire una tabella a quella esistente di una mappa vettoriale
v.db.univar	Calcola la statistica univariata sulla colonna selezionata della tabella di un vettoriale

B.8 Moduli negli strumenti GRASS per il 3D

Questa Sezione elenca tutti i moduli con interfaccia grafica negli strumenti GRASS per lavorare con dati 3D. GRASS fornisce ulteriori moduli, attualmente disponibili solo tramite la shell.

Tabella 28: Strumenti GRASS: Visualizzazione 3D

Moduli per la visualizzazione e l'analisi 3D tra gli strumenti di GRASS	
Nome modulo	Scopo
nviz	Apri la vista tridimensionale in nviz

B.9 Moduli negli strumenti GRASS per l'aiuto in linea

Il GRASS GIS Reference Manual offre una panoramica completa di tutti i moduli GRASS disponibili, senza limitarsi a quelli implementati, a volte con funzioni ridotte, negli strumenti di GRASS in QGIS.

Tabella 29: Strumenti GRASS: Reference Manual

Moduli per consultare il Reference Manual tra gli strumenti di GRASS	
Nome modulo	Scopo
g.manual	Mostra le pagine del manuale GRASS in formato HTML

C Guida all'Installazione

I capitoli seguenti forniscono informazioni di compilazione ed installazione per QGIS Versione 1.0.0. Questo documento corrisponde più o meno ad una conversione \LaTeX del file INSTALL.t2t che si scarica dai sorgenti QGIS dal 16 dicembre 2008.

Una versione attuale è anche disponibile sul wiki, vedere: <http://wiki.qgis.org/qgiswiki/BuildingFromSource>

C.1 Note generali di compilazione

Dalla versione 0.8.1 QGIS non usa più gli strumenti automatici per compilarsi. QGIS, come un gran numero di progetti importanti (eg. KDE 4.0), usa ora cmake (<http://www.cmake.org>) per compilarsi dai sorgenti. Lo script di configurazione in questa directory semplicemente controlla che cmake esista e fornisce alcune indicazioni per compilare QGIS.

Per informazioni più complete, vedere wiki: http://wiki.qgis.org/qgiswiki/Building_with_CMake

C.2 Panoramica delle dipendenze richieste per la compilazione

Dipendenze richieste per la compilazione:

- CMake $\geq 2.4.3$
- Flex, Bison

Dipendenze richieste per runtime:

- Qt $\geq 4.3.0$
- Proj $\geq ?$ (noto funzionare con 4.4.x)
- GEOS ≥ 2.2 (3.0 è supportato, forse funziona anche 2.1.x)
- Sqlite3 $\geq ?$ (probabilmente 3.0.0)
- GDAL/OGR $\geq 1.4.x$

Dipendenze opzionali:

- per plugin GRASS - GRASS $\geq 6.0.0$
- per georeferenziatore - GSL $\geq ?$ (works with 1.8)
- per supporto postgis e plugin SPIT - PostgreSQL $\geq 8.0.x$
- per plugin gps - expat $\geq ?$ (1.95 is OK)

- per esportazione mapserver e PyQGIS - Python ≥ 2.3 (2.5+ preferita)
- per PyQGIS - SIP ≥ 4.5 , PyQt \geq deve accordarsi con la versione Qt

Dipendenze raccomandate per runtime:

- per plugin gps - gpsbabel

D Compilazione in ambiente windows usando msys

Nota: Per una descrizione accurata della compilazione di tutte le dipendenze da soli, visitare il sito web di Marco Pasetti's website:

<http://www.webalice.it/marco.pasetti/qgis+grass/BuildFromSource.html>

Leggere la parte sull'uso dell'approccio semplificato con librerie pre-costruite...

D.1 MSYS:

MSYS fornisce un ambiente di compilazione stile unix in windows. Abbiamo creato un archivio zip che contiene quasi tutte le dipendenze.

Scaricare questo:

<http://download.osgeo.org/qgis/win32/msys.zip>

e estrarlo in c:\msys

Se si vuole preparare l'ambiente msys da soli piuttosto che usarne uno già fatto, istruzioni dettagliate possono essere trovate altrove in questo documento.

D.2 Qt4.3

Scaricare qt4.3 opensource edizione precompilata e installare (incluso lo scaricamento e l'installazione di mingw) da qui:

<http://www.trolltech.com/developer/downloads/qt/windows>

Quando l'installatore chiede di MinGW, non è necessario scaricarlo e installarlo, solo indirizzare l'installatore a c:\msys\mingw

Quando l'installazione di Qt è completa:

Aprire C:\Qt\4.3.0\bin\qtvars.bat e aggiungere le seguenti linee:

```
set PATH=%PATH%;C:\msys\local\bin;c:\msys\local\lib  
set PATH=%PATH%;"C:\Program Files\Subversion\bin"
```

Suggerisco anche di aggiungere C:\Qt\4.3.0\bin\ al proprio percorso delle variabili dell'ambiente nella finestra delle preferenze di sistema.

Se si pianifica di fare debugging, sarà necessario compilare una versione debug di Qt:
C:\Qt\4.3.0\bin\qvars.bat compile_debug

Nota: C'è un problema nella compilazione della versione debug di Qt 4.3, lo script finisce con questo messaggio mingw32-make: *** No rule to make target 'debug'. Stop.. Per compilare la versione debug bisogna uscire dalla directory src ed eseguire il seguente comando:

```
c:\Qt\4.3.0 make
```

D.3 Flex and Bison

Nota: Penso che questa sezione possa essere rimossa dato che dovrebbe essere già installata nell'immagine msys.

Scaricare Flex

http://sourceforge.net/project/showfiles.php?group_id=23617&package_id=16424 (zip bin)
e estrarlo in c:\msys\mingw\bin

D.4 Materiale Python: (opzionale)

Seguire questa sezione nel caso in cui si voglia usare bindings Python per QGIS. Per riuscire a compilare i bindings, bisogna compilare SIP e PyQt4 dai sorgenti dato che il loro installatore non include alcuni file di sviluppo che sono necessari.

D.4.1 Scaricare e installare Python - uso dell'installatore di Windows

(Non importa in quale cartella lo si installi)

<http://python.org/download/>

D.4.2 Scaricare i sorgenti SIP and PyQt4

<http://www.riverbankcomputing.com/software/sip/download>

<http://www.riverbankcomputing.com/software/pyqt/download>

Estrarre ognuno dei file zip qui sopra in una directory temporanea. Assicurarsi di prendere versioni che si adattino alla propria versione Qt istallata.

D.4.3 Compilare SIP

```
c:\Qt\4.3.0\bin\qtvars.bat
python configure.py -p win32-g++
make
make install
```

D.4.4 Compilare PyQt

```
c:\Qt\4.3.0\bin\qtvars.bat
python configure.py
make
make install
```

D.4.5 Note finali python

Nota: Si possono eliminare le directory in cui si sono estratti i sorgenti SIP and PyQt4 dopo un'istallazione andata a buon fine, non sono più necessarie.

D.5 Subversioni:

Per ottenere i sorgenti QGIS dall'archivio, è necessaria la Subversione client. Questo istallotaore dovrebbe funzionare bene:

<http://subversion.tigris.org/files/documents/15/36797/svn-1.4.3-setup.exe>

D.6 CMake:

CMake è il sistema di compilazione usato da Quantum GIS. Si scarica qui:

<http://www.cmake.org/files/v2.4/cmake-2.4.6-win32-x86.exe>

D.7 QGIS:

Eseguire una finestra cmd.exe (Inizio -> Esegui -> cmd.exe) Creare la directory di sviluppo e spostarla in essa

```
md c:\dev\cpp
cd c:\dev\cpp
```

Ottenere i sorgenti per SVN Per svn head:

```
svn co https://svn.osgeo.org/qgis/trunk/qgis
```

Per svn 0.8 branch

```
svn co https://svn.osgeo.org/qgis/branches/Release-0_8_0 qgis0.8
```

D.8 Compilare:

Come inizio leggere le note generiche di compilazione con CMake alla fine di questo documento.

Eseguire una finestra cmd.exe (Inizio -> Esegui -> cmd.exe) se non se ne ha già una. Scrivere il percorso per il compilatore e il nostro ambiente MSYS:

```
c:\Qt\4.3.0\bin\qtvars.bat
```

Per facilità d'uso aggiungere c:\Qt\4.3.0\bin\ alle proprie proprietà di sistema così si può soltanto scrivere qtvars.bat quando si apre la console cmd. Creare la directory di compilazione e impostarla come directory attuale:

```
cd c:\dev\cpp\qgis
md build
cd build
```

D.9 Configurazione

```
cmakesetup ..
```

Nota: Si deve includere il '..' qui sopra.

Premere il pulsante 'Configura' button. Quando richiesto, si deve scegliere 'MinGW Makefiles' come generatore.

C'è un problema con MinGW Makefiles in ambiente Win2K. Se si compila da questa piattaforma, usare invece il generatore 'MSYS Makefiles'.

Tutte le dipendenze dovrebbero essere colte automaticamente, se si è impostato correttamente il percorso. La sola cosa da cambiare è la destinazione dell'installazione (CMAKE_INSTALL_PREFIX) e/o impostare 'Debug'.

Per la compatibilità con il pacchetto di script NSIS si raccomanda di lasciare il prefisso di installazione al suo default c:\program files\

Quando la configurazione è completata, premere 'OK' per uscire dall'utility di impostazione.

D.10 Compilazione and installazione

```
make make install
```

D.11 Eseguire qgis.exe dalla directory dove è installato (CMAKE_INSTALL_PREFIX)

Assicurarsi di copiare tutti i .dll necessari nella stessa directory dove si trova il binario qgis.exe, se non lo si è già fatto, altrimenti all'avvio QGIS si lamenterà della mancanza di librerie.

Il modo migliore per far questo è scaricare il pacchetto di installazione della versione attuale di QGIS da <http://qgis.org/uploadfiles/testbuilds/> e installarlo. Ora si copia la directory di installazione da C:\Program Files\Quantum GIS a c:\Program Files\qgis-0.8.1 (o qualsiasi sia la versione attuale. Il nome dovrebbe coincidere con il numero della versione.) Dopo aver fatto questa copia, si può disinstallare la versione rilasciata di QGIS dalla directory c:\Program Files usando il disinstallatore fornito. Controllare poi bene che la directory Quantum GIS dir sia completamente scomparsa da programmi.

Un'altra possibilità è eseguire qgis.exe quando il percorso contiene le directory c:\msys\local\bin e c:\msys\local\lib, così i DLLs verranno usati da quella posizione.

D.12 Creare il pacchetto d'installazione: (opzionale)

Scaricare e installare NSIS da (http://nsis.sourceforge.net/Main_Page)

Ora usando windows explorer, entrare nella directory the win_build nel tuo albero sorgente QGIS. Leggere il file README che si trova lì e seguire le istruzioni. Poi fare click con il tasto destro del mouse su qgis.nsi e scegliere l'opzione 'Compile NSIS Script'.

E Compilare su Mac OSX usando frameworks e cmake (QGIS > 0.8)

In questo approccio cercherò di evitare per quanto possibile di costruire dipendenze dal sorgente e piuttosto usare frameworks quando possibile.

Sono incluse alcune note per compilare su Mac OS X 10.5 (Leopard).

E.1 Installare XCODE

Raccomando di scaricare la più recente immagine xcode dmg dal sito web Apple XDC. Installare XCODE dopo che il download ~941mb è completo.

Nota: Può essere necessario creare certi symlinks dopo l'installazione di XCODE SDK (in particolare se si sta usando XCODE 2.5 su tiger):

```
cd /Developer/SDKs/MacOSX10.4u.sdk/usr/  
sudo mv local/ local_  
sudo ln -s /usr/local/ local
```

E.2 Installare Qt4 from .dmg

E' necessario almeno Qt4.3.0. Suggestisco di scaricare l'ultima versione (almeno al momento della scrittura di questo manuale).

```
ftp://ftp.trolltech.com/qt/source/qt-mac-opensource-4.3.2.dmg
```

Se si vogliono librerie di debug, Qt fornisce anche un'immagine dmg che le contiene:

```
ftp://ftp.trolltech.com/qt/source/qt-mac-opensource-4.3.2-debug-libs.dmg
```

Sto usando soltanto librari di rilascio a questo stadio dato che scaricare l'immagine debug dmg è sostanzialmente più lungo. Se però si pianifica di fare debugging, probabilmente è meglio prendere l'immagine dmg con le librerie di debug. Una volta che si sia scaricata, aprire l'immagine dmg e lanciare l'installatore.

Nota: è necessario accesso di amministratore per installare.

Dopo l'installazione si devono fare due piccoli cambiamenti:

Primo, aprire `/Library/Frameworks/QtCore.framework/Headers/qconfig.h` e cambiare

Nota: questo non sembra essere necessario dalla versione 4.2.3

`QT_EDITION_Unknown` a `QT_EDITION_OPENSOURCE`

Secondo, cambiare il mkspec symlink di default che manda a `macx-g++`:

```
cd /usr/local/Qt4.3/mkspecs/  
sudo rm default  
sudo ln -sf macx-g++ default
```

E.3 Installare i frameworks di sviluppo per le dipendenze QGIS

Scaricare l'eccellente framework tutto in uno che include proj, gdal, sqlite3 etc di William Kyngesburye

<http://www.kyngchaos.com/wiki/software:frameworks>

Una volta scaricato, aprire e installare i frameworks.

William fornisce un pacchetto d'installazione addizionale per PostgreSQL/PostGIS disponibile qui:

<http://www.kyngchaos.com/wiki/software:postgres>

Ci sono alcune dipendenze addizionali che al momento di scrivere non sono fornite come frameworks così abbiamo bisogno di compilarle dal sorgente.

E.3.1 Dipendenze addizionali: GSL

Scaricare la Library Scientific Gnu da

```
curl -O ftp://ftp.gnu.org/gnu/gsl/gsl-1.8.tar.gz
```

Quindi estrarla e compilarla ad un prefisso di /usr/local:

```
tar xvfz gsl-1.8.tar.gz
cd gsl-1.8
./configure --prefix=/usr/local
make
sudo make install
cd ..
```

E.3.2 Dipendenze aggiuntive: Expat

Scaricare i sorgenti expat:

http://sourceforge.net/project/showfiles.php?group_id=10127

```
tar xvfz expat-2.0.0.tar.gz
cd expat-2.0.0
./configure --prefix=/usr/local
make
sudo make install
cd ..
```

E.3.3 Dipendenze aggiuntive: SIP

Assicurarsi di avere l'ultima versione di Python da

<http://www.python.org/download/mac/>

Nota riguardo a Leopard: Leopard include una versione Python 2.5 utilizzabile. Comunque si può installare Python da python.org se si preferisce.

Scaricare il toolkit SIP per i binding python da

<http://www.riverbankcomputing.com/software/sip/download>

Quindi estrarlo e compilarlo (questo si installa automaticamente nel framework Python):

```
tar xvfz sip-<version number>.tar.gz
cd sip-<version number>
```

```
python configure.py
make
sudo make install
cd ..
```

Note riguardo a Leopard:

Se si compila in Leopard, usando il Python incluso in Leopard, SIP vuole installare nel percorso di sistema – questa non è una buona idea. Usare questo comando di configurazione al posto delle configurazione base di sopra:

```
python configure.py -d /Library/Python/2.5/site-packages -b \
/usr/local/bin -e /usr/local/include -v /usr/local/share/sip
```

E.3.4 Dipendenze aggiuntive: PyQt

Se si incontrano problemi compilando PyQt seguendo le istruzioni qui sotto, sai può provare anche aggiungendo python dalla propria directory di frameworks esplicitamente al proprio percorso, ad es.

```
export PATH=/Library/Frameworks/Python.framework/Versions/Current/bin:$PATH$
```

Scaricare il toolkit per Qt per i binding python da

<http://www.riverbankcomputing.com/software/pyqt/download>

Quindi estrarlo e compilarlo (si installa automaticamente nel framework Python):

```
tar xvfz PyQt-mac<version number here>
cd PyQt-mac<version number here>
export QTDIR=/Developer/Applications/Qt
python configure.py
yes
make
sudo make install
cd ..
```

Note riguardo a Leopard

Se si compila in Leopard, usando il Python incluso in Leopard, PyQt vuole installarsi nel percorso di sistema – questa non è una buona idea. Usare questo comando di configurazione invece della configurazione base sopra:

```
python configure.py -d /Library/Python/2.5/site-packages -b /usr/local/bin
```

Può esserci un problema con simboli indefiniti in QtOpenGL sotto Leopard. Aprire QtOpenGL/makefile e aggiungere -undefined dynamic_lookup a LFLAGS.

E.3.5 Dipendenze aggiuntive: Bison

Nota riguardo a Leopard: Leopard include Bison 2.3, quindi questo passaggio può essere saltato in Leopard.

La versione di bison disponibile di default in Mac OSX è troppo vecchia quindi è necessario procurarsi una versione più recente per il proprio sistema:

```
curl -O http://ftp.gnu.org/gnu/bison/bison-2.3.tar.gz
```

Ora compilarla e installarla in un prefisso di /usr/local :

```
tar xvfz bison-2.3.tar.gz
cd bison-2.3
./configure --prefix=/usr/local
make
sudo make install
cd ..
```

E.4 Installare CMAKE per OSX

Procurarsi l'ultima versione rilasciata:

```
http://www.cmake.org/HTML/Download.html
```

Al tempo della scrittura di questo manuale ho preso:

```
curl -O http://www.cmake.org/files/v2.4/cmake-2.4.6-Darwin-universal.dmg
```

Una volta scaricata, aprire l'immagine dmg ed eseguire l'installatore

E.5 Installare subversioni per OSX

Nota riguardo a Leopard: Leopard include SVN, così questo passaggio può essere saltato in Leopard.

Il progetto <http://sourceforge.net/projects/macsvn/> ha una versione compilata di svn scaricabile. Volendo si può anche reperire il loro client GUI. Reperire la linea di comando del client qui:

```
curl -O http://ufpr.dl.sourceforge.net/sourceforge/macsvn/Subversion_1.4.2.zip
```

Una volta scaricato, aprire il file zip ed eseguire l'installatore.

Può essere necessario installare anche BerkleyDB, disponibile allo stesso indirizzo <http://sourceforge.net/projects/macsvn/>. Al momento della scrittura di questo manuale era qui:

```
curl -O http://ufpr.dl.sourceforge.net/sourceforge/macsvn/Berkeley_DB_4.5.20.zip
```

Ancora una volta aprire il file zip eed eseguire l'installatore. Infine dobbiamo assicurarci che la linea di comando svn eseguibile sia nel percorso. aggiungere la seguente linea alla fine di /etc/bashrc usando sudo:

```
sudo vim /etc/bashrc
```

E aggiungere questa line in fondo prima di salvare e chiudere:

```
export PATH=/usr/local/bin:$PATH:/usr/local/pgsql/bin
```

/usr/local/bin deve essere primo nel percorso cosicchè il più recente bison (che sarà compilato dal sorgente più avanti) venga trovato prima del bison installato da MacOSX, che è molto vecchio.

Ora chiudere e riaprire la propria shell per avere le versioni aggiornate.

E.6 Check out QGIS da SVN

Ora facciamo il check out del sorgente per for QGIS. Prima si crea una directory per lavorarci:

```
mkdir -p ~/dev/cpp cd ~/dev/cpp
```

Ora si fa il check out del sorgente:

Trunk:

```
svn co https://svn.osgeo.org/qgis/trunk/qgis qgis
```

Per la branch svn 0.8

```
svn co https://svn.osgeo.org/qgis/branches/Release-0_8_0 qgis0.8
```

Per la branch svn 0.9

```
svn co https://svn.qgis.org/qgis/branches/Release-0_9_0 qgis0.9
```

la prima volta che si fa check out del sorgente QGIS sourcessi avrà probabilmente un messaggio di questo tipo:

```
Error validating server certificate for 'https://svn.qgis.org:443':
- The certificate is not issued by a trusted authority. Use the fingerprint to
  validate the certificate manually!  Certificate information:
- Hostname: svn.qgis.org
- Valid: from Apr  1 00:30:47 2006 GMT until Mar 21 00:30:47 2008 GMT
- Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US
- Fingerprint: 2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b
  (R) eject, accept (t)emporarily or accept (p)ermanently?
```

Suggerisco di premere 'p' per accettare la chiave in modo permanente.

E.7 Configurare la versione compilata

CMake supporta la compilazione fuori dal sorgente così possiamo creare una directory 'di compilazione' per il processo di compilazione. Per convenzione io compilo il mio software in una directory chiamata 'apps' nella mia directory home. Se si hanno i corretti permessi si può voler compilare direttamente nella cartella /Applications. Le istruzioni sotto assumono che si stia compilando in una preesistente directory \${HOME}/apps ...

```
cd qgis
mkdir build
cd build
cmake -D CMAKE_INSTALL_PREFIX=$HOME/apps/ -D CMAKE_BUILD_TYPE=Release ..
```

Nota riguardo a Leopard: Per trovare un'installazione personalizzata di SIP in Leopard, aggiungere "-D SIP_BINARY_PATH=/usr/local/bin/sip" al comando cmake sopra, prima di .. alla fine, per esempio:

```
cmake -D CMAKE_INSTALL_PREFIX=$HOME/apps/ -D CMAKE_BUILD_TYPE=Release -  
D SIP_BINARY_PATH=/usr/local/bin/sip ..
```

Per usare la versione compilata di GRASS su OSX, si può altrimenti eseguire cmake nella maniera seguente (richiesto almeno GRASS 6.3, sostituire la versione GRASS come richiesto):

```
cmake -D CMAKE_INSTALL_PREFIX=${HOME}/apps/ \  
-D GRASS_INCLUDE_DIR=/Applications/GRASS-6.3.app/Contents/MacOS/  
include \  
-D GRASS_PREFIX=/Applications/GRASS-6.3.app/Contents/MacOS \  
-D CMAKE_BUILD_TYPE=Release \  
..
```

Oppure, per usare una versione di Grass Unix-style, si può eseguire cmake nella maniera seguente (richiesta almeno una versione di GRASS come nei requisiti Qgis, sostituire il percorso GRASS e la versione come richiesto):

```
cmake -D CMAKE_INSTALL_PREFIX=${HOME}/apps/ \  
-D GRASS_INCLUDE_DIR=/user/local/grass-6.3.0/include \  
-D GRASS_PREFIX=/user/local/grass-6.3.0 \  
-D CMAKE_BUILD_TYPE=Release \  
..
```

E.8 Compilazione

Ora cominciamo il processo di compilazione:

```
make
```

se tutto compila senza errori si può passare all'installazione:

```
make install
```

F Compilare sotto GNU/Linux

F.1 Compilare QGIS con Qt4.x

Requisiti: Ubuntu Hardy / distro derivati Debian

Queste note sono attuali per Ubuntu 7.10 - altre versioni e distro derivati Debian potrebbero richiedere piccole variazioni nei nomi dei pacchetti.

Queste note sono per chi vuole compilare QGIS dal sorgente. Uno dei principali scopi qui è mostrare come questo si possa fare usando pacchetti binari per dipendenze ***all*** - compilando solo la parte core di QGIS dal sorgente. Preferisco questo approccio perchè significa che si può lasciare l'affare di gestire i pacchetti di sistema ad apt e preoccuparsi soltanto del codice QGIS!

Questo documento assume che si abbia un'installazione recente e un sistema 'pulito'. Queste istruzioni dovrebbero funzionare se questo è un sistema che è già stato in uso per un po', si potrebbe aver bisogno solo di saltare queipassaggi che sono irrilevanti per noi.

F.2 Preparare apt

Il pacchetto qgis dipende dalla versione disponibile nel componente universe di Ubuntu. Questo non è attivo di default, quindi bisogna ttivarlo:

1. Aprire il proprio file /etc/apt/sources.list 2. Decomentare tutte le linee che cominciano con deb

Sarà anche necessario eseguire (K)Ubuntu 'edgy' o superiore per incontrare tutte le dipendenze.

Ora aggiornare il database dei propri sorgente locali:

```
sudo apt-get update
```

F.3 Installare Qt4

```
sudo apt-get install libqt4-core libqt4-debug \
libqt4-dev libqt4-gui libqt4-qt3support libqt4-sql lsb-qt4 qt4-designer \
qt4-dev-tools qt4-doc qt4-qtconfig uim-qt gcc libapt-pkg-perl resolvconf
```

Nota speciale: Se si sta seguendo questo set di istruzioni su un sistema che già ha gli strumenti di sviluppo Qt3 installati, ci sarà un conflitto tra strumenti Qt3 e Qt4. Per esempio, qmake punterà alla versione Qt3 e non alla Qt4. In Ubuntu i pacchetti Qt4 e Qt3 sono fatti in modo da poter convivere. Questo significa per esempio che se si hanno installati entrambi si dovranno fare tre eseguibili:

```
/usr/bin/qmake -> /etc/alternatives/qmake
/usr/bin/qmake-qt3
/usr/bin/qmake-qt4
```

Lo stesso vale per tutti gli altri Qt binari. Avrete notato sopra che il canonico 'qmake' è gestito dalle alternative apt, quindi prima di cominciare a compilare QGIS, è necessario rendere Qt4 il predefinito. Per far poi tornare Qt3 a predefinito si può usare lo stesso procedimento.

si possono usare le alternative apt per correggere questo, così la versione Qt4 dell'applicazione verrà usata in tutti i casi:

```
sudo update-alternatives --config qmake
sudo update-alternatives --config uic
sudo update-alternatives --config designer
sudo update-alternatives --config assistant
sudo update-alternatives --config qtconfig
sudo update-alternatives --config moc
sudo update-alternatives --config lupdate
sudo update-alternatives --config lrelease
sudo update-alternatives --config linguist
```

Usare la semplice finestra della linea di comando che appare dopo aver eseguito ognuno dei comandi sopra per selezionare la versione Qt4 dell'applicazione in questione.

F.4 Installare dipendenze software additionali richieste da QGIS

```
sudo apt-get install gdal-bin libgdal1-dev libgeos-dev proj \
libgdal-doc libhdf4g-dev libhdf4g-run python-dev \
libgsl0-dev g++ libjasper-dev libtiff4-dev subversion \
libsqlite3-dev sqlite3 ccache make libpq-dev flex bison cmake txt2tags \
python-qt4 python-qt4-dev python-sip4 sip4 python-sip4-dev
```

Nota: Per gli utenti Debian dovrebbero usare libgdal-dev sopra piuttosto

Nota: Per i bindings in linguaggio python SIP ≥ 4.5 e PyQt4 ≥ 4.1 è necessario! Alcune distribuzioni stabili GNU/Linux (ad es. Debian o SuSE) forniscono soltanto SIP < 4.5 e PyQt4 < 4.1 . Per includere il support per i bindings in linguaggio python può essere necessario compilare e installare quei pacchetti dal sorgente.

Se non si ha già cmake installato:

```
sudo apt-get install cmake
```

F.5 Passaggi specifici GRASS

Nota: Se non si ha bisogno di compilare con il supporto GRASS, si può saltare questa sezione.

Ora si può installare grass dal dapper:

```
sudo apt-get install grass libgrass-dev libgdal1-1.4.0-grass
```

!/\ Può essere necessario specificare l'esatta versione grass, ad es. libgdal1-1.3.2-grass

F.6 Impostare ccache (Opzionale)

Si dovrebbe anche impostare ccache per velocizzare i tempi di compilazione:

```
cd /usr/local/bin
sudo ln -s /usr/bin/ccache gcc
sudo ln -s /usr/bin/ccache g++
```

F.7 Preparare il proprio ambiente di sviluppo

Per convenzione svolgo tutto il mio lavoro di sviluppo in \$HOME/dev/<language>, così in questo caso creeremo un ambiente di lavoro per lo sviluppo in C++ come this:

```
mkdir -p ${HOME}/dev/cpp
cd ${HOME}/dev/cpp
```

Questo percorso alla directory sarà assunto anche per tutte le istruzioni seguenti.

F.8 Check out del codice sorgente QGIS

Ci sono due modi per fare il check out del sorgente. Usare il metodo anonimo se non si hanno i privilegi per l'archivio dei sorgenti QGIS, o usare il check out da sviluppatore se si hanno i permessi per fare commit di cambiamenti al codice sorgente.

1. Checkout Anonimo

```
cd ${HOME}/dev/cpp
svn co https://svn.osgeo.org/qgis/trunk/qgis qgis
```

2. Checkout Sviluppatore

```
cd ${HOME}/dev/cpp
svn co --username <yourusername> https://svn.osgeo.org/qgis/trunk/qgis qgis
```

La prima volta che si fa check out del sorgente verrà richiesto di accettare il certificato qgis.org. Premere 'p' per accettarlo permanentemente:

```
Error validating server certificate for 'https://svn.qgis.org:443':
- The certificate is not issued by a trusted authority. Use the
  fingerprint to validate the certificate manually! Certificate
  information:
- Hostname: svn.qgis.org
- Valid: from Apr  1 00:30:47 2006 GMT until Mar 21 00:30:47 2008 GMT
- Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US
- Fingerprint:
  2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b (R)eject,
  accept (t)emporarily or accept (p)ermanently?
```

F.9 Cominciare a compilare

Nota: La prossima sezione descrive come compilare i pacchetti debian

Io compilo le mie versioni di sviluppo di QGIS nella directory ~/apps per evitare che confliggano con i pacchetti Ubuntu che possono essere in /usr. In questo modo per esempio si può usare il pacchetto binario di QGIS sul proprio sistema insieme con la propria versione di sviluppo. Suggesto che facciate qualcosa di simile:

```
mkdir -p ${HOME}/apps
```

Ora si crea una directory di compilazione e si esegue cmake:

```
cd qgis
mkdir build
cd build
cmake ..
```

Quando si esegue cmake (notare che il .. è richiesto!), un menu apparirà dove si possono configurare i vari aspetti della compilazione. Se non si ha accesso alla root o non si vuole sovrascrivere installazioni QGIS preesistenti (per mezzo del packagemanager per esempio), impostare il

CMAKE_BUILD_PREFIX per una posizione dove si hanno i privilegi di scrittura (di solito uso /home/timlinux/apps). Ora premere 'c' per configurare, 'e' per ignorare qualsiasi messaggio d'errore che possa apparire e 'g' per generare il file make. **Nota:** a volte 'c' deve essere premuto più volte prima che l'opzione 'g' divenga disponibile. Dopo che la generazione 'g' è completa, premere 'q' per uscire dalla finestra di dialogo interattiva di ccmake.

Ora avanti con la compilazione:

```
make
make install
```

Ci può volere un po' mentre si costruiscono le dipendenze sulla propria piattaforma.

F.10 Compilare i pacchetti Debian

Invece di creare un'installazione personale come nel precedente passaggio si può anche creare un pacchetto debian. Questo si fa dalla directory root di qgis, dove si trova una directory debian.

Prima è necessario installare gli strumenti di impacchettamento debian:

```
apt-get install build-essential
```

I pacchetti QGIS saranno creati con:

```
dpkg-buildpackage -us -us -b
```

Nota: Se dpkg-buildpackage si lamenta riguardo dipendenze che non sono state incontrate, queste si possono installare usando apt-get e rieseguire il comando.

Nota: Se si ha libqgis1-dev installato, bisogna rimuoverlo prima usando dpkg -r libqgis1-dev. Altrimenti dpkg-buildpackage si lamenterà riguardo un conflitto di compilazione.

I pacchetti vengono creati nella directory di origine (cioè un livello superiore). Installarli usando dpkg. Ad esempio:

```
sudo dpkg -i \
../qgis_1.0preview16_amd64.deb \
../libqgis-gui1_1.0preview16_amd64.deb \
../libqgis-core1_1.0preview16_amd64.deb \
../qgis-plugin-grass_1.0preview16_amd64.deb \
../python-qgis_1.0preview16_amd64.deb
```


F.11 Eseguire QGIS

Ora si può provare ad eseguire QGIS:

```
$HOME/apps/bin/qgis
```

Se tutto ha funzionato in modo giusto l'applicazione QGIS dovrebbe avviarsi e apparire sul vostro schermo.

G Creation of MSYS environment for compilation of Quantum GIS

G.1 Initial setup

G.1.1 MSYS

This is the environment that supplies many utilities from UNIX world in Windows and is needed by many dependencies to be able to compile.

Download from here:

<http://puzzle.dl.sourceforge.net/sourceforge/mingw/MSYS-1.0.11-2004.04.30-1.exe>

Install to `c:\msys`

All stuff we're going to compile is going to get to this directory (resp. its subdirs).

G.1.2 MinGW

Download from here:

<http://puzzle.dl.sourceforge.net/sourceforge/mingw/MinGW-5.1.3.exe>

Install to `c:\msys\mingw`

It suffices to download and install only `g++` and `mingw-make` components.

G.1.3 Flex and Bison

Flex and Bison are tools for generation of parsers, they're needed for GRASS and also QGIS compilation.

Download the following packages:

<http://gnuwin32.sourceforge.net/downlinks/flex-bin-zip.php>

<http://gnuwin32.sourceforge.net/downlinks/bison-bin-zip.php>

<http://gnuwin32.sourceforge.net/downlinks/bison-dep-zip.php>

Unpack them all to `c:\msys\local`

G.2 Installing dependencies

G.2.1 Getting ready

Paul Kelly did a great job and prepared a package of precompiled libraries for GRASS. The package currently includes:

- `zlib-1.2.3`
- `libpng-1.2.16-noconfig`
- `xdr-4.0-mingw2`
- `freetype-2.3.4`
- `fftw-2.1.5`
- `PDCurses-3.1`
- `proj-4.5.0`
- `gdal-1.4.1`

It's available for download here:

<http://www.stjohnspoint.co.uk/grass/wingrass-extralibs.tar.gz>

Moreover he also left the notes how to compile it (for those interested):

<http://www.stjohnspoint.co.uk/grass/README.extralibs>

Unpack the whole package to `c:\msys\local`

G.2.2 GDAL level one

Since Quantum GIS needs GDAL with GRASS support, we need to compile GDAL from source - Paul Kelly's package doesn't include GRASS support in GDAL. The idea is following:

1. compile GDAL without GRASS

2. compile GRASS

3. compile GDAL with GRASS

So, start with downloading GDAL sources:

<http://download.osgeo.org/gdal/gdal141.zip>

Unpack it to some directory, preferably `c:\msys\local\src`.

Start MSYS console, go to `gdal-1.4.1` directory and run the commands below. You can put them all to a script, e.g. `build-gdal.sh` and run them at once. The recipe is taken from Paul Kelly's instructions - basically they just make sure that the library will be created as DLL and the utility programs will be dynamically linked to it...

```
CFLAGS="-O2 -s" CXXFLAGS="-O2 -s" LDFLAGS=-s ./configure --without-libtool \
--prefix=/usr/local --enable-shared --disable-static --with-libz=/usr/local \
--with-png=/usr/local
make
make install
rm /usr/local/lib/libgdal.a
g++ -s -shared -o ./libgdal.dll -L/usr/local/lib -lz -lpng ./frmts/o/*.o ./gcore/*.o \
./port/*.o ./alg/*.o ./ogr/ogrsf_frmts/o/*.o ./ogr/ogrgeometryfactory.o \
./ogr/ogrpoint.o ./ogr/ogrcurve.o ./ogr/ogrlinestring.o ./ogr/ogrlinearring.o \
./ogr/ogrpolygon.o ./ogr/ogrutils.o ./ogr/ogrgeometry.o ./ogr/ogrgeometrycollection.o \
./ogr/ogrmultipolygon.o ./ogr/ogrsurface.o ./ogr/ogrmultipoint.o \
./ogr/ogrmultilinestring.o ./ogr/ogr_api.o ./ogr/ogrfeature.o ./ogr/ogrfeaturedefn.o \
./ogr/ogrfeaturequery.o ./ogr/ogrfeaturestyle.o ./ogr/ogrfielddefn.o \
./ogr/ogrspatialreference.o ./ogr/ogr_srsnode.o ./ogr/ogr_srs_proj4.o \
./ogr/ogr_fromepsg.o ./ogr/ogrct.o ./ogr/ogr_opt.o ./ogr/ogr_srs_esri.o \
./ogr/ogr_srs_pci.o ./ogr/ogr_srs_usgs.o ./ogr/ogr_srs_dict.o ./ogr/ogr_srs_panorama.o \
./ogr/swq.o ./ogr/ogr_srs_validate.o ./ogr/ogr_srs_xml.o ./ogr/ograssemblepolygon.o \
./ogr/ogr2gmlgeometry.o ./ogr/gml2ogrgeometry.o
install libgdal.dll /usr/local/lib
cd ogr
g++ -s ogrinfo.o -o ogrinfo.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s ogr2ogr.o -o ogr2ogr.exe -lgdal -L/usr/local/lib -lpng -lz -lgdal
g++ -s ogrtindex.o -o ogrtindex.exe -lgdal -L/usr/local/lib -lpng -lz -lgdal
install ogrinfo.exe ogr2ogr.exe ogrtindex.exe /usr/local/bin
cd ../apps
g++ -s gdalinfo.o -o gdalinfo.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s gdal_translate.o -o gdal_translate.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s gdaladdo.o -o gdaladdo.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s gdalwarp.o -o gdalwarp.exe -L/usr/local/lib -lpng -lz -lgdal
```

```
g++ -s gdal_contour.o -o gdal_contour.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s gdaltindex.o -o gdaltindex.exe -L/usr/local/lib -lpng -lz -lgdal
g++ -s gdal_rasterize.o -o gdal_rasterize.exe -L/usr/local/lib -lpng -lz -lgdal
install gdalinfo.exe gdal_translate.exe gdaladdo.exe gdalwarp.exe gdal_contour.exe \
gdaltindex.exe gdal_rasterize.exe /usr/local/bin
```

Finally, manually edit `gdal-config` in `c:\msys\local\bin` to replace the static library reference with `-lgdal`:

```
CONFIG_LIBS="-L/usr/local/lib -lpng -lz -lgdal"
```

GDAL build procedure can be greatly simplified to use `libtool` with a `libtool` line patch: configure `gdal` as below: `./configure --with-ngpython --with-xerces=/local/ --with-jasper=/local/ --with-grass=/local/grass-6.3.cvs/ --with-pg=/local/pgsql/bin/pg_config.exe`

Then fix `libtool` with: `mv libtool libtool.orig cat libtool.orig | sed 's/max_cmd_len=8192/max_cmd_len=32768/g' > libtool`

`Libtool` on windows assumes a line length limit of 8192 for some reason and tries to page the linking and fails miserably. This is a work around.

Make and make install should be hassle free after this.

G.2.3 GRASS

Grab sources from CVS or use a weekly snapshot, see:

<http://grass.itc.it/devel/cvs.php>

In `MSYS` console go to the directory where you've unpacked or checked out sources (e.g. `c:\msys\local\src\grass-6.3.cvs`)

Run these commands:

```
export PATH="/usr/local/bin:/usr/local/lib:$PATH"
./configure --prefix=/usr/local --bindir=/usr/local --with-includes=/usr/local/include \
--with-libs=/usr/local/lib --with-cxx --without-jpeg --without-tiff --with-postgres=yes \
--with-postgres-includes=/local/pgsql/include --with-pgsql-libs=/local/pgsql/lib \
--with-opengl=windows --with-fftw --with-freetype \
--with-freetype-includes=/mingw/include/freetype2 \
--without-x --without-tcltk \
```

```
--enable-x11=no --enable-shared=yes --with-proj-share=/usr/local/share/proj
make
make install
```

It should get installed to `c:\msys\local\grass-6.3.cvs`

By the way, these pages might be useful:

- http://grass.gdf-hannover.de/wiki/WinGRASS_Current_Status
- <http://geni.ath.cx/grass.html>

G.2.4 GDAL level two

At this stage, we'll use GDAL sources we've used before, only the compilation will be a bit different.

But first in order to be able to compile GDAL sources with current GRASS CVS, you need to patch them, here's what you need to change:

http://trac.osgeo.org/gdal/attachment/ticket/1587/plugin_patch_grass63.diff

(you can patch it by hand or use `patch.exe` in `c:\msys\bin`)

Now in MSYS console go to the GDAL sources directory and run the same commands as in level one, only with these differences:

1) when running `./configure` add this argument:

```
--with-grass=/usr/local/grass-6.3.cvs
```

2) when calling `g++` on line 5 (which creates `libgdal.dll`), add these arguments:

```
-L/usr/local/grass-6.3.cvs/lib -lgrass\_vect -lgrass\_dig2 -lgrass\_dgl -lgrass\_rtree \
-lgrass\_linkm -lgrass\_dbmiclient -lgrass\_dbmibase -lgrass\_I -lgrass\_gproj \
-lgrass\_vask -lgrass\_gmath -lgrass\_gis -lgrass\_datetime}
```

Then again, edit `gdal-config` and change line with `CONFIG_LIBS`

```
CONFIG_LIBS="-L/usr/local/lib -lpng -L/usr/local/grass-6.3.cvs/lib -lgrass\_vect \
-lgrass\_dig2 -lgrass\_dgl -lgrass\_rtree -lgrass\_linkm -lgrass\_dbmiclient \
-lgrass\_dbmibase -lgrass\_I -lgrass\_gproj -lgrass\_vask -lgrass\_gmath -lgrass\_gis \
-lgrass\_datetime -lz -L/usr/local/lib -lgdal"
```

Now, GDAL should be able to work also with GRASS raster layers.

G.2.5 GEOS

Download the sources:

<http://geos.refractions.net/geos-2.2.3.tar.bz2>

Unpack to e.g. `c:\msys\local\src`

To compile, I had to patch the sources: in file `source/headers/timeval.h` line 13. Change it from:

```
#ifdef _WIN32
```

to:

```
#if defined(_WIN32) && defined(_MSC_VER)
```

Now, in MSYS console, go to the source directory and run:

```
./configure --prefix=/usr/local  
make  
make install
```

G.2.6 SQLITE

You can use precompiled DLL, no need to compile from source:

Download this archive:

http://www.sqlite.org/sqlitedll-3_3_17.zip

and copy `sqlite3.dll` from it to `c:\msys\local\lib`

Then download this archive:

http://www.sqlite.org/sqlite-source-3_3_17.zip

and copy `sqlite3.h` to `c:\msys\local\include`

G.2.7 GSL

Download sources:

<ftp://ftp.gnu.org/gnu/gsl/gsl-1.9.tar.gz>

Unpack to `c:\msys\local\src`

Run from MSYS console in the source directory:

```
./configure
make
make install
```

G.2.8 EXPAT

Download sources:

<http://dfn.dl.sourceforge.net/sourceforge/expat/expat-2.0.0.tar.gz>

Unpack to `c:\msys\local\src`

Run from MSYS console in the source directory:

```
./configure
make
make install
```

G.2.9 POSTGRES

We're going to use precompiled binaries. Use the link below for download:

<http://wwwmaster.postgresql.org/download/mirrors-ftp?file=%2Fbinary%2Fv8.2.4%2Fwin32%2Fpostgresql-8.2.4-1-binaries-no-installer.zip>

copy contents of `pgsql` directory from the archive to `c:\msys\local`

G.3 Cleanup

We're done with preparation of MSYS environment. Now you can delete all stuff in `c:\msys\local\src` - it takes quite a lot of space and it's not necessary at all.

H Building with MS Visual Studio

! \ This section describes a process where you build all dependencies yourself. See the section after this for a simpler procedure where we have all the dependencies you need pre-packaged and we focus just on getting Visual Studio Express set up and building QGIS.

Note: that this does not currently include GRASS or Python plugins.

H.1 Setup Visual Studio

This section describes the setup required to allow Visual Studio to be used to build QGIS.

H.1.1 Express Edition

The free Express Edition lacks the platform SDK which contains headers and so on that are needed when building QGIS. The platform SDK can be installed as described here:

<http://msdn.microsoft.com/vstudio/express/visualc/usingpsdk/>

Once this is done, you will need to edit the <vsinstalldir>\Common7\Tools\vsvars file as follows:

```
Add %PlatformSDKDir%\Include\atl and %PlatformSDKDir%\Include\mfcc to the
@set INCLUDE entry.
```

This will add more headers to the system INCLUDE path. **Note:** that this will only work when you use the Visual Studio command prompt when building. Most of the dependencies will be built with this. You will also need to perform the edits described here to remove the need for a library that Visual Studio Express lacks:

<http://www.codeproject.com/wtl/WTLEExpress.asp>

H.1.2 All Editions

You will need stdint.h and unistd.h. unistd.h comes with GnuWin32 version of flex & bison binaries (see later). stdint.h can be found here:

<http://www.azillionmonkeys.com/qed/pstdint.h>

Copy both of these to <vsinstalldir>\VC\include.

H.2 Download/Install Dependencies

This section describes the downloading and installation of the various QGIS dependencies.

H.2.1 Flex and Bison

Flex and Bison are tools for generation of parsers, they're needed for GRASS and also QGIS compilation.

Download the following packages and run the installers:

<http://gnuwin32.sourceforge.net/downloads/flex.php>

<http://gnuwin32.sourceforge.net/downloads/bison.php>

H.2.2 To include PostgreSQL support in Qt

If you want to build Qt with PostgreSQL support you need to download PostgreSQL, install it and create a library you can later link with Qt.

Download from `.../binary/v8.2.5/win32/postgresql-8.2.5-1.zip` from an PostgreSQL.org Mirror and install.

PostgreSQL is currently build with MinGW and comes with headers and libraries for MinGW. The headers can be used with Visual C++ out of the box, but the library is only shipped in DLL and archive (.a) form and therefore cannot be used with Visual C++ directly.

To create a library copy following sed script to the file `mkdef.sed` in PostgreSQL lib directory:

```
/Dump of file / {
s/Dump of file \([^\ ]*\)$ /LIBRARY \1/p
a\
EXPORTS
}
/[ ]*ordinal hint/,/^[ ]*Summary/ {
/^[ ]*\+[0-9]\+/ {
s/^[ ]*\+[0-9]\+[ ]*\+[0-9A-Fa-f]\+[ ]*\+[0-9A-Fa-f]\+[ ]*\+([^\ ]*=[^\ ]*)\.*$/ \1/p
}
}
```

and process execute in the Visual Studio C++ command line (from Programs menu):

```
cd c:\Program Files\PostgreSQL\8.2\bin
dumpbin /exports ..\bin\libpq.dll | sed -nf ../lib/mkdef.sed >..\lib\libpq.def
cd ..\lib
lib /def:libpq.def /machine:x86
```

You'll need an sed for that to work in your path (e.g. from cygwin or msys).

That's almost it. You only need to the include and lib path to INCLUDE and LIB in vcvars.bat respectively.

H.2.3 Qt

Build Qt following the instructions here:

http://wiki.qgis.org/qgiswiki/Building_QT_4_with_Visual_C%2B%2B_2005

H.2.4 Proj.4

Get proj.4 source from here:

<http://proj.maptools.org/>

Using the Visual Studio command prompt (ensures the environment is setup properly), run the following in the src directory:

```
nmake -f makefile.vc
```

Install by running the following in the top level directory setting PROJ_DIR as appropriate:

```
set PROJ_DIR=c:\lib\proj
```

```
mkdir %PROJ_DIR%\bin
mkdir %PROJ_DIR%\include
mkdir %PROJ_DIR%\lib
```

```
copy src\*.dll %PROJ_DIR%\bin
copy src\*.exe %PROJ_DIR%\bin
copy src\*.h %PROJ_DIR%\include
copy src\*.lib %PROJ_DIR%\lib
```

This can also be added to a batch file.

H.2.5 GSL

Get gsl source from here:

<http://david.geldreich.free.fr/downloads/gsl-1.9-windows-sources.zip>

Build using the gsl.sln file

H.2.6 GEOS

Get geos from svn (svn checkout <http://svn.refractor.net/geos/trunk> geos). Edit geos\source\makefile.vc as follows:

Uncomment lines 333 and 334 to allow the copying of version.h.vc to version.h.

Uncomment lines 338 and 339.

Rename geos_c.h.vc to geos_c.h.in on lines 338 and 339 to allow the copying of geos_c.h.in to geos_c.h.

Using the Visual Studio command prompt (ensures the environment is setup properly), run the following in the top level directory:

```
nmake -f makefile.vc
```

Run the following in top level directory, setting GEOS_DIR as appropriate:

```
set GEOS_DIR="c:\lib\geos"
```

```
mkdir %GEOS_DIR%\include
```

```
mkdir %GEOS_DIR%\lib
```

```
mkdir %GEOS_DIR%\bin
```

```
xcopy /S/Y source\headers\*.h %GEOS_DIR%\include
```

```
copy /Y capi\*.h %GEOS_DIR%\include
```

```
copy /Y source\*.lib %GEOS_DIR%\lib
```

```
copy /Y source\*.dll %GEOS_DIR%\bin
```

This can also be added to a batch file.

H.2.7 GDAL

Get gdal from svn (svn checkout <https://svn.osgeo.org/gdal/branches/1.4/gdal> gdal).

Edit nmake.opt to suit, it's pretty well commented.

Using the Visual Studio command prompt (ensures the environment is setup properly), run the following in the top level directory:

```
nmake -f makefile.vc
```

and

```
nmake -f makefile.vc devinstall
```

H.2.8 PostGIS

Get PostGIS and the Windows version of PostgreSQL from here:

<http://postgis.refractory.net/download/>

Note: the warning about not installing the version of PostGIS that comes with the PostgreSQL installer. Simply run the installers.

H.2.9 Expat

Get expat from here:

http://sourceforge.net/project/showfiles.php?group_id=10127

You'll need expat-win32bin-2.0.1.exe.

Simply run the executable to install expat.

H.2.10 CMake

Get CMake from here:

<http://www.cmake.org/HTML/Download.html>

You'll need cmake-<version>-win32-x86.exe. Simply run this to install CMake.

H.3 Building QGIS with CMAKE

Get QGIS source from svn (svn co <https://svn.osgeo.org/qgis/trunk/qgis> qgis).

Create a 'Build' directory in the top level QGIS directory. This will be where all the build output will be generated.

Run Start->All Programs->CMake->CMake.

In the 'Where is the source code:' box, browse to the top level QGIS directory.

In the 'Where to build the binaries:' box, browse to the 'Build' directory you created in the top level QGIS directory.

Fill in the various *_INCLUDE_DIR and *_LIBRARY entries in the 'Cache Values' list.

Click the Configure button. You will be prompted for the type of makefile that will be generated. Select Visual Studio 8 2005 and click OK.

All being well, configuration should complete without errors. If there are errors, it is usually due to an incorrect path to a header or library directory. Failed items will be shown in red in the list.

Once configuration completes without error, click OK to generate the solution and project files.

With Visual Studio 2005, open the qgis.sln file that will have been created in the Build directory you created earlier.

Build the ALL_BUILD project. This will build all the QGIS binaries along with all the plugins.

Install QGIS by building the INSTALL project. By default this will install to c:\Program Files\qgis<version> (this can be changed by changing the CMAKE_INSTALL_PREFIX variable in CMake).

You will also either need to add all the dependency dlls to the QGIS install directory or add their respective directories to your PATH.

I Building under Windows using MSVC Express

Note:: Building under MSVC is still a work in progress. In particular the following dont work yet: python, grass, postgis connections.

/!\ This section of the document is in draft form and is not ready to be used yet.

Tim Sutton, 2007

I.1 System preparation

I started with a clean XP install with Service Pack 2 and all patches applied. I have already compiled all the dependencies you need for gdal, expat etc, so this tutorial wont cover compiling those from source too. Since compiling these dependencies was a somewhat painful task I hope my pre-compiled libs will be adequate. If not I suggest you consult the individual projects for specific build documentation and support. Lets go over the process in a nutshell before we begin:

- * Install XP (I used a Parallels virtual machine)
- * Install the premade libraries archive I have made for you
- * Install Visual Studio Express 2005 sp1
- * Install the Microsoft Platform SDK
- * Install command line subversion client
- * Install library dependencies bundle
- * Install Qt 4.3.2
- * Check out QGIS sources
- * Compile QGIS
- * Create setup.exe installer for QGIS

I.2 Install the libraries archive

Half of the point of this section of the MSVC setup procedure is to make things as simple as possible for you. To that end I have prepared an archive that includes all dependencies needed to build QGIS except Qt (which we will build further down). Fetch the archive from:

http://qgis.org/uploadfiles/msvc/qgis_msvc_deps_except_qt4.zip

Create the following directory structure:

```
c:\dev\cpp\
```

And then extract the libraries archive into a subdirectory of the above directory so that you end up with:

```
c:\dev\cpp\qgislibs-release
```

Note: that you are not obliged to use this directory layout, but you should adjust any instructions that follow if you plan to do things differently.

I.3 Install Visual Studio Express 2005

First thing we need to get is MSVC Express from here:

<http://msdn2.microsoft.com/en-us/express/aa975050.aspx>

The page is really confusing so don't feel bad if you can't actually find the download at first! There are six coloured blocks on the page for the various studio family members (vb / c# / j# etc). Simply choose your language under the 'select your language' combo under the yellow C++ block, and your download will begin. Under internet explorer I had to disable popup blocking for the download to be able to commence.

Once the setup commences you will be prompted with various options. Here is what I chose :

* Send usage information to Microsoft (No) * Install options: * Graphical IDE (Yes) * Microsoft MSDN Express Edition (No) * Microsoft SQL Server Express Edition (No) * Install to folder: C:\Program Files\Microsoft Visual Studio 8\ (default)

It will need to download around 90mb of installation files and reports that the install will consume 554mb of disk space.

I.4 Install Microsoft Platform SDK2

Go to this page:

<http://msdn2.microsoft.com/en-us/express/aa700755.aspx>

Start by using the link provided on the above page to download and install the platform SDK2.

The actual SDK download page is once again a bit confusing since the links for downloading are hidden amongst a bunch of other links. Basically look for these three links with their associated 'Download' buttons and choose the correct link for your platform:

PSDK-amd64.exe	1.2 MB	Download
PSDK-ia64.exe	1.3 MB	Download
PSDK-x86.exe	1.2 MB	Download

When you install make sure to choose 'custom install'. These instructions assume you are installing into the default path of:

C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\

We will go for the minimal install that will give us a working environment, so on the custom installation screen I made the following choices:

Configuration Options

+ Register Environmental Variables (Yes)

Microsoft Windows Core SDK

+ Tools	(Yes)
+ Tools (AMD 64 Bit)	(No unless this applies)
+ Tools (Intel 64 Bit)	(No unless this applies)
+ Build Environment	
+ Build Environment (AMD 64 Bit)	(No unless this applies)
+ Build Environment (Intel 64 Bit)	(No unless this applies)
+ Build Environment (x86 32 Bit)	(Yes)
+ Documentation	(No)
+ Redistributable Components	(Yes)
+ Sample Code	(No)
+ Source Code	(No)
+ AMD 64 Source	(No)
+ Intel 64 Source	(No)
Microsoft Web Workshop	(Yes) (needed for shlwapi.h)
+ Build Environment	(Yes)
+ Documentation	(No)
+ Sample Code	(No)
+ Tools	(No)
Microsoft Internet Information Server (IIS) SDK	(No)
Microsoft Data Access Services (MDAC) SDK	(Yes) (needed by GDAL for odbc)
+ Tools	
+ Tools (AMD 64 Bit)	(No)
+ Tools (AMD 64 Bit)	(No)
+ Tools (x86 32 Bit)	(Yes)
+ Build Environment	
+ Tools (AMD 64 Bit)	(No)
+ Tools (AMD 64 Bit)	(No)
+ Tools (x86 32 Bit)	(Yes)
+ Documentation	(No)
+ Sample Code	(No)
Microsoft Installer SDK	(No)
Microsoft Table PC SDK	(No)
Microsoft Windows Management Instrumentation	(No)
Microsoft DirectShow SDK	(No)
Microsoft Media Services SDK	(No)
Debuggin Tools for Windows	(Yes)

Note: that you can always come back later to add extra bits if you like.

Note: that installing the SDK requires validation with the Microsoft Genuine Advantage application. Some people have a philosophical objection to installing this software on their computers. If you are one of them you should probably consider using the MINGW build instructions described elsewhere

in this document.

The SDK installs a directory called

```
C:\Office10
```

Which you can safely remove.

After the SDK is installed, follow the remaining notes on the page link above to get your MSVC Express environment configured correctly. For your convenience, these are summarised again below, and I have added a couple more paths that I discovered were needed:

- 1) open Visual Studio Express IDE
- 2) Tools -> Options -> Projects and Solutions -> VC++ Directories
- 3) Add:

Executable files:

```
C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Bin
```

Include files:

```
C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Include
```

```
C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Include\atl
```

```
C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Include\mf
```

Library files: C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Lib

- 4) Close MSVC Express IDE

- 5) Open the following file with notepad:

```
C:\Program Files\Microsoft Visual Studio 8\VC\VCProjectDefaults\corewin_express.vsprops
```

and change the property:

```
AdditionalDependencies="kernel32.lib"
```

To read:

```
AdditionalDependencies="kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib  
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib"
```

The notes go on to show how to build a mswin32 application which you can try if you like - I'm not going to recover that here.

I.5 Edit your vsvars

Backup your vsvars32.bat file in

C:\Program Files\Microsoft Visual Studio 8\Common7\Tools

and replace it with this one:

```
@SET VSINSTALLDIR=C:\Program Files\Microsoft Visual Studio 8
@SET VCINSTALLDIR=C:\Program Files\Microsoft Visual Studio 8\VC
@SET FrameworkDir=C:\WINDOWS\Microsoft.NET\Framework
@SET FrameworkVersion=v2.0.50727
@SET FrameworkSDKDir=C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0
@if "%VSINSTALLDIR%"==" " goto error_no_VSINSTALLDIR
@if "%VCINSTALLDIR%"==" " goto error_no_VCINSTALLDIR

@echo Setting environment for using Microsoft Visual Studio 2005 x86 tools.

@rem
@rem Root of Visual Studio IDE installed files.
@rem
@set DevEnvDir=C:\Program Files\Microsoft Visual Studio 8\Common7\IDE

@set PATH=C:\Program Files\Microsoft Visual Studio 8\Common7\IDE;C:\Program \
Files\Microsoft Visual Studio 8\VC\BIN;C:\Program Files\Microsoft Visual Studio 8\ \
Common7\Tools;C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\bin; \
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;C:\Program Files\Microsoft Visual \
Studio 8\VC\VCPackages;%PATH%
@rem added by Tim
@set PATH=C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Bin;%PATH%
@set INCLUDE=C:\Program Files\Microsoft Visual Studio 8\VC\INCLUDE; \
%INCLUDE%
@rem added by Tim
@set INCLUDE=C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\ \
Include;%INCLUDE%
@set INCLUDE=C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\ \
Include\mf;C:\dev\cpp\qgislibs-release\include\postgresql
@set LIB=C:\Program Files\Microsoft Visual Studio 8\ \
VC\LIB;C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\lib;%LIB%
@rem added by Tim
```

```
@set LIB=C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\Lib;%LIB%
@set LIB=%LIB%;C:\dev\cpp\qgislibs-release\lib
@set LIBPATH=C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727

@goto end

:error_no_VSINSTALLDIR
@echo ERROR: VSINSTALLDIR variable is not set.
@goto end

:error_no_VCINSTALLDIR
@echo ERROR: VCINSTALLDIR variable is not set.
@goto end

:end
```

I.6 Environment Variables

Right click on 'My computer' then select the 'Advanced' tab. Click environment variables and create or augment the following "System" variables (if they dont already exist):

Variable Name:	Value:

EDITOR	vim
INCLUDE	C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2 \
	\Include\.
LIB	C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2 \
	\Lib\.
LIB_DIR	C:\dev\cpp\qgislibs-release
PATH	C:\Program Files\CMake 2.4\bin;
	%SystemRoot%\system32;
	%SystemRoot%;
	%SystemRoot%\System32\Wbem;
	C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2 \
	\Bin\.;
	C:\Program Files\Microsoft Platform SDK for Windows Server 2003 R2\ \
	\Bin\WinNT\;
	C:\Program Files\svn\bin;C:\Program Files\Microsoft Visual Studio 8 \
	\VC\bin;
	C:\Program Files\Microsoft Visual Studio 8\Common7\IDE;

```
"c:\Program Files\Microsoft Visual Studio 8\Common7\Tools";  
c:\Qt\4.3.2\bin;  
"C:\Program Files\PuTTY"  
QTDIR      c:\Qt\4.3.2  
SVN_SSH     "C:\\Program Files\\PuTTY\\plink.exe"
```

I.7 Building Qt4.3.2

You need a minimum of Qt 4.3.2 here since this is the first version to officially support building the open source version of Qt for windows under MSVC.

Download Qt 4.x.x source for windows from

<http://www.trolltech.com>

Unpack the source to

c:\Qt\4.x.x\

I.7.1 Compile Qt

Open the Visual Studio C++ command line and cd to c:\Qt\4.x.x where you extracted the source and enter:

```
configure -platform win32-msvc2005  
nmake  
nmake install
```

Add -qt-sql-odbc -qt-sql-psql to the configure line if your want odbc and PostgreSQL support build into Qt.

Note: For me in some cases I got a build error on qsscreenshot.pro. If you are only interested in having the libraries needed for building Qt apps, you can probably ignore that. Just check in c:\Qt\4.3.2\bin to check all dlls and helper apps (assistant etc) have been made.

I.7.2 Configure Visual C++ to use Qt

After building configure the Visual Studio Express IDE to use Qt:

- 1) open Visual Studio Express IDE
- 2) Tools -> Options -> Projects and Solutions -> VC++ Directories
- 3) Add:

Executable files:

`$(QTDIR)\bin`

Include files:

`$(QTDIR)\include`

`$(QTDIR)\include\Qt`

`$(QTDIR)\include\QtCore`

`$(QTDIR)\include\QtGui`

`$(QTDIR)\include\QtNetwork`

`$(QTDIR)\include\QtSvg`

`$(QTDIR)\include\QtXml`

`$(QTDIR)\include\Qt3Support`

`$(LIB_DIR)\include` (needed during qgis compile to find stdint.h and unistd.h)

Library files:

`$(QTDIR)\lib`

Source Files:

`$(QTDIR)\src`

Hint: You can also add

`QString = t=<d->data, su>, size=<d->size, i>`

to AutoExp.DAT in C:\Program Files\Microsoft Visual Studio 8\Common7\Packages\Debugger before

[Visualizer]

That way the Debugger will show the contents of QString when you point at or watch a variable in the debugger. There are probably much more additions - feel free to add some - I just needed QString and took the first hit in google I could find.

I.8 Install Python

Download <http://python.org/ftp/python/2.5.1/python-2.5.1.msi> and install it.

I.9 Install SIP

Download <http://www.riverbankcomputing.com/Downloads/sip4/sip-4.7.1.zip> and extract it into your c:\dev\cpp directory. From a Visual C++ command line cd to the directory where you extract SIP and run:

```
c:\python25\python configure.py -p win32-msvc2005
nmake
nmake install
```

I.10 Install PyQt4

Download <http://www.riverbankcomputing.com/Downloads/PyQt4/GPL/PyQt-win-gpl-4.3.1.zip> and extract it into your c:\dev\cpp directory. From a Visual C++ command line cd to the directory where you extracted PyQt4 and run:

```
c:\python25\python configure.py -p win32-msvc2005
nmake
nmake install
```

I.11 Install CMake

Download and install cmake 2.4.7 or better, making sure to enable the option: Update path for all users

I.12 Install Subversion

You "must" install the command line version if you want the CMake svn scripts to work. Its a bit tricky to find the correct version on the subversion download site as they have som misleadingly named similar downloads. Easiest is to just get this file:

<http://subversion.tigris.org/downloads/1.4.5-win32/apache-2.2/svn-win32-1.4.5.zip>

Extract the zip file to

C:\Program Files\svn

And then add

C:\Program Files\svn\bin

To your path.

I.13 Initial SVN Check out

Open a cmd.exe window and do:

```
cd \  
cd dev  
cd cpp  
svn co https://svn.osgeo.org/qgis/trunk/qgis
```

At this point you will probably get a message like this:

```
C:\dev\cpp>svn co https://svn.osgeo.org/qgis/trunk/qgis  
Error validating server certificate for 'https://svn.qgis.org:443':  
- The certificate is not issued by a trusted authority. Use the  
  fingerprint to validate the certificate manually!  
Certificate information:  
- Hostname: svn.qgis.org  
- Valid: from Sat, 01 Apr 2006 03:30:47 GMT until Fri, 21 Mar 2008 03:30:47 GMT  
- Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US  
- Fingerprint: 2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b  
(R)eject, accept (t)emporarily or accept (p)ermanently?
```

Press 'p' to accept and the svn checkout will commence.

I.14 Create Makefiles using cmakesetup.exe

I won't be giving a detailed description of the build process, because the process is explained in the first section (where you manually build all dependencies) of the windows build notes in this document. Just skip past the parts where you need to build GDAL etc, since this simplified install process does all the dependency provisioning for you.

```
cd qgis  
mkdir build  
cd build  
cmakesetup ..
```

Cmakesetup should find all dependencies for you automatically (it uses the LIB_DIR environment to find them all in c:\dev\cpp\qgislibs-release). Press configure again after the cmakesetup gui appears and when all the red fields are gone, and you have made any personalisations to the setup, press ok to close the cmake gui.

Now open Visual Studio Express and do: File -> Open -> Project / Solution

Now open the cmake generated QGIS solution which should be in :

```
c:\dev\cpp\qgis\build\qgisX.X.X.sln
```

Where X.X.X represents the current version number of QGIS. Currently I have only made release built dependencies for QGIS (debug versions will follow in future), so you need to be sure to select 'Release' from the solution configurations toolbar. Next right click on ALL_BUILD in the solution browser, and then choose build. Once the build completes right click on INSTALL in the solution browser and choose build. This will by default install qgis into c:\program files\qgisX.X.X.

I.15 Running and packaging

To run QGIS you need to at the minimum copy the dlls from c:\dev\cpp\qgislibs-release\bin into the c:\program files\qgisX.X.X directory.

J QGIS Coding Standards

The following chapters provide coding information for QGIS Version 1.0.0. This document corresponds almost to a \LaTeX conversion of the CODING.t2t file coming with the QGIS sources from December, 16th 2008.

These standards should be followed by all QGIS developers. Current information about QGIS Coding Standards are also available from wiki at:

<http://wiki.qgis.org/qgiswiki/CodingGuidelines>
<http://wiki.qgis.org/qgiswiki/CodingStandards>
<http://wiki.qgis.org/qgiswiki/UsingSubversion>
<http://wiki.qgis.org/qgiswiki/DebuggingPlugins>
<http://wiki.qgis.org/qgiswiki/DevelopmentInBranches>
<http://wiki.qgis.org/qgiswiki/SubmittingPatchesAndSvnAccess>

J.1 Classes

J.1.1 Names

Class in QGIS begin with Qgs and are formed using mixed case.

Examples:

QgsPoint
QgsMapCanvas
QgsRasterLayer

J.1.2 Members

Class member names begin with a lower case *m* and are formed using mixed case.

mMapCanvas
mCurrentExtent

All class members should be private. **Public class members are STRONGLY discouraged**

J.1.3 Accessor Functions

Class member values should be obtained through accessor functions. The function should be named without a *get* prefix. Accessor functions for the two private members above would be:

```
mapCanvas()  
currentExtent()
```

J.1.4 Functions

Function names begin with a lowercase letter and are formed using mixed case. The function name should convey something about the purpose of the function.

```
updateMapExtent()  
setUserOptions()
```

J.2 Qt Designer

J.2.1 Generated Classes

QGIS classes that are generated from Qt Designer (ui) files should have a *Base* suffix. This identifies the class as a generated base class.

Examples:
QgsPluginMangerBase
QgsUserOptionsBase

J.2.2 Dialogs

All dialogs should implement the following:

- * Tooltip help for all toolbar icons and other relevant widgets
- * WhatsThis help for **all** widgets on the dialog
- * An optional (though highly recommended) context sensitive *Help* button that directs the user to the appropriate help page by launching their web browser

J.3 C++ Files

J.3.1 Names

C++ implementation and header files should have a .cpp and .h extension respectively. Filename should be all lowercase and, in the case of classes, match the class name.

Example:

Class QgsFeatureAttribute source files are
qgsfeatureattribute.cpp and qgsfeatureattribute.h

J.3.2 Standard Header and License

Each source file should contain a header section patterned after the following example:

```
/*
*****
qgsfield.cpp - Describes a field in a layer or table
-----
Date           : 01-Jan-2004
Copyright      : (C) 2004 by Gary E.Sherman
Email          : sherman at mrcc.com
*****
*
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*
*****
*/
```

J.3.3 CVS Keyword

Each source file should contain the \$Id\$ keyword. This will be expanded by CVS to contain useful information about the file, revision, last committer, and date/time of last checkin.

Place the keyword right after the standard header/license that is found at the top of each source file:

```
/* $Id$ */
```

J.4 Variable Names

Variable names begin with a lower case letter and are formed using mixed case.

Examples:

mapCanvas

currentExtent

J.5 Enumerated Types

Enumerated types should be named in CamelCase with a leading capital e.g.:

```
enum UnitType
{
    Meters,
    Feet,
    Degrees,
    UnknownUnit
} ;
```

Do not use generic type names that will conflict with other types. e.g. use UnkownUnit rather than Unknown

J.6 Global Constants

Global constants should be written in upper case underscore separated e.g.:

```
const long GEOCRS_ID = 3344;
```

J.7 Editing

Any text editor/IDE can be used to edit QGIS code, providing the following requirements are met.

J.7.1 Tabs

Set your editor to emulate tabs with spaces. Tab spacing should be set to 2 spaces.

J.7.2 Indentation

Source code should be indented to improve readability. There is a .indent.pro file in the QGIS src directory that contains the switches to be used when indenting code using the GNU indent program. If you don't use GNU indent, you should emulate these settings.

J.7.3 Braces

Braces should start on the line following the expression:

```
if(foo == 1)
{
    // do stuff
    ...
}else
{
    // do something else
    ...
}
```

J.8 API Compatibility

From QGIS 1.0 we will provide a stable, backwards compatible API. This will provide a stable basis for people to develop against, knowing their code will work against any of the 1.x QGIS releases (although recompiling may be required). Cleanups to the API should be done in a manner similar to the Trolltech developers e.g.

```
class Foo
{
    public:
        /** This method will be deprecated, you are encouraged to use
            doSomethingBetter() rather.
            @see doSomethingBetter()
            */
        bool doSomething();

        /** Does something a better way.
            @note This method was introduced in QGIS version 1.1
            */
}
```

```
bool doSomethingBetter();  
  
}
```

J.9 Coding Style

Here are described some programming hints and tips that will hopefully reduce errors, development time, and maintenance.

J.9.1 Where-ever Possible Generalize Code

If you are cut-n-pasting code, or otherwise writing the same thing more than once, consider consolidating the code into a single function.

This will allow changes to be made in one location instead of in multiple places

- help prevent code bloat
- make it more difficult for multiple copies to evolve differences over time, thus making it harder to understand and maintain for others

J.9.2 Prefer Having Constants First in Predicates

Prefer to put constants first in predicates.

```
"0 == value" instead of "value == 0"
```

This will help prevent programmers from accidentally using = when they meant to use ==, which can introduce very subtle logic bugs. The compiler will generate an error if you accidentally use = instead of == for comparisons since constants inherently cannot be assigned values.

J.9.3 Whitespace Can Be Your Friend

Adding spaces between operators, statements, and functions makes it easier for humans to parse code.

Which is easier to read, this:

```
if (!a&& b)
```

or this:

```
if ( ! a && b )
```

J.9.4 Add Trailing Identifying Comments

Adding comments at the end of function, struct and class implementations makes it easier to find them later.

Consider that you're at the bottom of a source file and need to find a very long function – without these kinds of trailing comments you will have to page up past the body of the function to find its name. Of course this is ok if you wanted to find the beginning of the function; but what if you were interested at code near its end? You'd have to page up and then back down again to the desired part.

E.g.,

```
void foo::bar()
{
    // ... imagine a lot of code here
} // foo::bar()
```

J.9.5 Use Braces Even for Single Line Statements

Using braces for code in if/then blocks or similar code structures even for single line statements means that adding another statement is less likely to generate broken code.

Consider:

```
if (foo)
    bar();
else
    baz();
```

Adding code after `bar()` or `baz()` without adding enclosing braces would create broken code. Though most programmers would naturally do that, some may forget to do so in haste.

So, prefer this:

```
if (foo)
{
```

```
    bar();  
}  
else  
{  
    baz();  
}
```

J.9.6 Book recommendations

Effective C++ <http://www.awprofessional.com/title/0321334876>

More Effective C++ <http://www.awprofessional.com/bookstore/product.asp?isbn=020163371X&rl=1>

Effective STL <http://www.awprofessional.com/title/0201749629>

Design Patterns <http://www.awprofessional.com/title/0201634988>

You should also really read this article from Qt Quarterly on designing Qt style
<http://doc.trolltech.com/qq/qq13-apis.html>

K SVN Access

This page describes how to get started using the QGIS Subversion repository

K.1 Accessing the Repository

To check out QGIS HEAD:

```
svn --username [your user name] co https://svn.qgis.org/repos/qgis/trunk/qgis
```

K.2 Anonymous Access

You can use the following commands to perform an anonymous checkout from the QGIS Subversion repository. Note we recommend checking out the trunk (unless you are a developer or really HAVE to have the latest changes and dont mind lots of crashing!).

You must have a subversion client installed prior to checking out the code. See the Subversion website for more information. The Links page contains a good selection of SVN clients for various platforms.

To check out a branch


```
svn co https://svn.qgis.org/repos/qgis/branches/<branch name>
```

To check out SVN stable trunk:

```
svn co https://svn.qgis.org/repos/qgis/trunk/qgis qgis_unstable
```

/!\ **Note:** If you are behind a proxy server, edit your `~/subversion/servers` file to specify your proxy settings first!

/!\ **Note:** In QGIS we keep our most stable code in trunk. Periodically we will tag a release off trunk, and then continue stabilisation and selective incorporation of new features into trunk.

See the INSTALL file in the source tree for specific instructions on building development versions.

K.3 QGIS documentation sources

If you're interested in checking out Quantum GIS documentation sources:

```
svn co https://svn.qgis.org/repos/qgis_docs/trunk qgis_docs
```

You can also take a look at DocumentationWritersCorner for more information.

K.4 Documentation

The repository is organized as follows:

<http://wiki.qgis.org/images/repo.png>

See the Subversion book <http://svnbook.red-bean.com> for information on becoming a SVN master.

K.5 Development in branches

K.5.1 Purpose

The complexity of the QGIS source code has increased considerably during the last years. Therefore it is hard to anticipate the side effects that the addition of a feature will have. In the past, the QGIS project had very long release cycles because it was a lot of work to reestablish the stability of the software system after new features were added. To overcome these problems, QGIS switched to a development model where new features are coded in svn branches first and merged to trunk (the main branch) when they are finished and stable. This section describes the procedure for branching and merging in the QGIS project.

K.5.2 Procedure

* Initial announcement on mailing list Before starting, make an announcement on the developer mailing list to see if another developer is already working on the same feature. Also contact the technical advisor of the project steering committee (PSC). If the new feature requires any changes to the QGIS architecture, a request for comment (RFC) is needed. * Create a branch Create a new svn branch for the development of the new feature (see UsingSubversion for the svn syntax). Now you can start developing. * Merge from trunk regularly It is recommended to merge the changes in trunk to the branch on a regular basis. This makes it easier to merge the branch back to trunk later. * Documentation on wiki It is also recommended to document the intended changes and the current status of the work on a wiki page. * Testing before merging back to trunk When you are finished with the new feature and happy with the stability, make an announcement on the developer list. Before merging back, the changes will be tested by developers and users. Binary packages (especially for OSX and Windows) will be generated to also involve non-developers. In trac, a new Component will be opened to file tickets against. Once there are no remaining issues left, the technical advisor of the PSC merges the changes into trunk.

K.5.3 Creating a branch

We prefer that new feature developments happen out of trunk so that trunk remains in a stable state. To create a branch use the following command:

```
svn copy https://svn.qgis.org/repos/qgis/trunk/qgis \  
https://svn.qgis.org/repos/qgis/branches/qgis_newfeature  
svn commit -m "New feature branch"
```

K.5.4 Merge regularly from trunk to branch

When working in a branch you should regularly merge trunk into it so that your branch does not diverge more than necessary. In the top level dir of your branch, first type 'svn info' to determine the revision numbers of your branch which will produce output something like this:

```
timlinux@timlinux-desktop:~/dev/cpp/qgis_raster_transparency_branch$ svn info  
Caminho: .  
URL: https://svn.qgis.org/repos/qgis/branches/raster_transparency_branch  
Raiz do Repositorio: https://svn.qgis.org/repos/qgis  
UUID do repositorio: c8812cc2-4d05-0410-92ff-de0c093fc19c  
Revisao: 6546  
Tipo de No: diretorio
```

Agendado: normal

Autor da Ultima Mudanca: timlinux

Revisao da Ultima Mudanca: 6495

Data da Ultima Mudanca: 2007-02-02 09:29:47 -0200 (Sex, 02 Feb 2007)

Propriedades da Ultima Mudanca: 2007-01-09 11:32:55 -0200 (Ter, 09 Jan 2007)

The second revision number shows the revision number of the start revision of your branch and the first the current revision. You can do a dry run of the merge like this:

```
svn merge --dry-run -r 6495:6546 https://svn.qgis.org/repos/qgis/trunk/qgis
```

After you are happy with the changes that will be made do the merge for real like this:

```
svn merge -r 6495:6546 https://svn.qgis.org/repos/qgis/trunk/qgis
svn commit -m "Merged upstream changes from trunk to my branch"
```

K.6 Submitting Patches

There are a few guidelines that will help you to get your patches into QGIS easily, and help us deal with the patches that are sent to use easily.

K.6.1 Patch file naming

If the patch is a fix for a specific bug, please name the file with the bug number in it e.g. **bug777fix.diff**, and attach it to the original bug report in trac (<https://trac.osgeo.org/qgis/>).

If the bug is an enhancement or new feature, its usually a good idea to create a ticket in trac (<https://trac.osgeo.org/qgis/>) first and then attach you

K.6.2 Create your patch in the top level QGIS source dir

This makes it easier for us to apply the patches since we don't need to navigate to a specific place in the source tree to apply the patch. Also when I receive patches I usually evaluate them using kompare, and having the patch from the top level dir makes this much easier. Below is an example of you you can include multiple changed files into your patch from the top level directory:

```
cd qgis
svn diff src/ui/somefile.ui src/app/somefile2.cpp > bug872fix.diff
```

K.6.3 Including non version controlled files in your patch

If your improvements include new files that don't yet exist in the repository, you should indicate to svn that they need to be added before generating your patch e.g.

```
cd qgis
svn add src/lib/somenewfile.cpp
svn diff > bug7887fix.diff
```

K.6.4 Getting your patch noticed

QGIS developers are busy folk. We do scan the incoming patches on bug reports but sometimes we miss things. Don't be offended or alarmed. Try to identify a developer to help you - using the [Project Organigram] and contact them asking them if they can look at your patch. If you don't get any response, you can escalate your query to one of the Project Steering Committee members (contact details also available on the [Project Organigram]).

K.6.5 Due Diligence

QGIS is licensed under the GPL. You should make every effort to ensure you only submit patches which are unencumbered by conflicting intellectual property rights. Also do not submit code that you are not happy to have made available under the GPL.

K.7 Obtaining SVN Write Access

Write access to QGIS source tree is by invitation. Typically when a person submits several (there is no fixed number here) substantial patches that demonstrate basic competence and understanding of C++ and QGIS coding conventions, one of the PSC members or other existing developers can nominate that person to the PSC for granting of write access. The nominator should give a basic promotional paragraph of why they think that person should gain write access. In some cases we will grant write access to non C++ developers e.g. for translators and documentors. In these cases, the person should still have demonstrated ability to submit patches and should ideally have submitted several substantial patches that demonstrate their understanding of modifying the code base without breaking things, etc.

K.7.1 Procedure once you have access

Checkout the sources:

```
svn co https://svn.qgis.org/repos/qgis/trunk/qgis qgis
```

Build the sources (see INSTALL document for proper detailed instructions)

```
cd qgis
mkdir build
ccmake ..      (set your preferred options)
make
make install   (maybe you need to do with sudo / root perms)
```

Make your edits

```
cd ..
```

Make your changes in sources. Always check that everything compiles before making any commits. Try to be aware of possible breakages your commits may cause for people building on other platforms and with older / newer versions of libraries.

Add files (if you added any new files). The svn status command can be used to quickly see if you have added new files.

```
svn status src/plugin/grass/modules
```

Files listed with ? in front are not in SVN and possibly need to be added by you:

```
svn add src/plugin/grass/modules/foo.xml
```

Commit your changes

```
svn commit src/plugin/grass/modules/foo.xml
```

Your editor (as defined in \$EDITOR environment variable) will appear and you should make a comment at the top of the file (above the area that says 'dont change this'. Put a descriptive comment and rather do several small commits if the changes across a number of files are unrelated. Conversely we prefer you to group related changes into a single commit.

Save and close in your editor. The first time you do this, you should be prompted to put in your username and password. Just use the same ones as your trac account.

L Unit Testing

As of November 2007 we require all new features going into trunk to be accompanied with a unit test. Initially we have limited this requirement to `qgis_core`, and we will extend this requirement to other parts of the code base once people are familiar with the procedures for unit testing explained in the sections that follow.

L.1 The QGIS testing framework - an overview

Unit testing is carried out using a combination of `QTestLib` (the Qt testing library) and `CTest` (a framework for compiling and running tests as part of the CMake build process). Lets take an overview of the process before I delve into the details:

- **There is some code you want to test**, e.g. a class or function. Extreme programming advocates suggest that the code should not even be written yet when you start building your tests, and then as you implement your code you can immediately validate each new functional part you add with your test. In practice you will probably need to write tests for pre-existing code in QGIS since we are starting with a testing framework well after much application logic has already been implemented.
- **You create a unit test**. This happens under `<QGIS Source Dir>/tests/src/core` in the case of the core lib. The test is basically a client that creates an instance of a class and calls some methods on that class. It will check the return from each method to make sure it matches the expected value. If any one of the calls fails, the unit will fail.
- **You include `QtTestLib` macros in your test class**. This macro is processed by the Qt meta object compiler (`moc`) and expands your test class into a runnable application.
- **You add a section to the `CMakeLists.txt` in your tests directory** that will build your test.
- **You ensure you have `ENABLE_TESTING` enabled in `ccmake` / `cmakesetup`**. This will ensure your tests actually get compiled when you type `make`.
- **You optionally add test data to `<QGIS Source Dir>/tests/testdata`** if your test is data driven (e.g. needs to load a shapefile). These test data should be as small as possible and wherever possible you should use the existing datasets already there. Your tests should never modify this data in situ, but rather may a temporary copy somewhere if needed.
- **You compile your sources and install**. Do this using normal `make` && `(sudo) make install` procedure.
- **You run your tests**. This is normally done simply by doing `make test` after the `make install` step, though I will explain other approaches that offer more fine grained control over running tests.

Right with that overview in mind, I will delve into a bit of detail. I've already done much of the configuration for you in CMake and other places in the source tree so all you need to do are the easy bits - writing unit tests!

L.2 Creating a unit test

Creating a unit test is easy - typically you will do this by just creating a single .cpp file (not .h file is used) and implement all your test methods as public methods that return void. I'll use a simple test class for QgsRasterLayer throughout the section that follows to illustrate. By convention we will name our test with the same name as the class they are testing but prefixed with 'Test'. So our test implementation goes in a file called testqgsrasterlayer.cpp and the class itself will be TestQgsRasterLayer. First we add our standard copyright banner:

```

/*****
    testqgsvectorfilewriter.cpp
    -----
    Date           : Frida Nov 23 2007
    Copyright      : (C) 2007 by Tim Sutton
    Email          : tim@linfiniti.com
    *****/
*
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*   *****/

```

Next we use start our includes needed for the tests we plan to run. There is one special include all tests should have:

```
#include <QtTest>
```

Beyond that you just continue implementing your class as per normal, pulling in whatever headers you may need:

```

//Qt includes...
#include <QObject>
#include <QString>
#include <QObject>
#include <QApplication>
#include <QFileInfo>
#include <QDir>

//qgis includes...
#include <qgsrasterlayer.h>
#include <qgsrasterbandstats.h>
#include <qgsapplication.h>

```

Since we are combining both class declaration and implementation in a single file the class declaration comes next. We start with our doxygen documentation. Every test case should be properly documented. We use the doxygen **ingroup** directive so that all the UnitTests appear as a module in the generated Doxygen documentation. After that comes a short description of the unit test:


```
/** \ingroup UnitTests
 * This is a unit test for the QgsRasterLayer class.
 */
```

The class **must** inherit from `QObject` and include the `Q_OBJECT` macro.

```
class TestQgsRasterLayer: public QObject
{
    Q_OBJECT;
```

All our test methods are implemented as **private slots**. The `QtTest` framework will sequentially call each private slot method in the test class. There are four 'special' methods which if implemented will be called at the start of the unit test (**`initTestCase`**), at the end of the unit test (**`cleanupTestCase`**). Before each test method is called, the **`init()`** method will be called and after each test method is called the **`cleanup()`** method is called. These methods are handy in that they allow you to allocate and cleanup resources prior to running each test, and the test unit as a whole.

```
private slots:
    // will be called before the first testfunction is executed.
    void initTestCase();
    // will be called after the last testfunction was executed.
    void cleanupTestCase(){};
    // will be called before each testfunction is executed.
    void init(){};
    // will be called after every testfunction.
    void cleanup();
```

Then come your test methods, all of which should take **no parameters** and should **return void**. The methods will be called in order of declaration. I am implementing two methods here which illustrates to types of testing. In the first case I want to generally test the various parts of the class are working, I can use a **functional testing** approach. Once again, extreme programmers would advocate writing these tests **before** implementing the class. Then as you work your way through your class implementation you iteratively run your unit tests. More and more test functions should complete successfully as your class implementation work progresses, and when the whole unit test passes, your new class is done and is now complete with a repeatable way to validate it.

Typically your unit tests would only cover the **public API** of your class, and normally you do not need to write tests for accessors and mutators. If it should happen that an accessor or mutator is not working as expected you would normally implement a **regression** test to check for this (see lower down).

```
//  
// Functional Testing  
//  
  
/** Check if a raster is valid. */  
void isValid();  
  
// more functional tests here ...
```

Next we implement our **regression tests**. Regression tests should be implemented to replicate the conditions of a particular bug. For example I recently received a report by email that the cell count by rasters was off by 1, throwing off all the statistics for the raster bands. I opened a bug (ticket #832) and then created a regression test that replicated the bug using a small test dataset (a 10x10 raster). Then I ran the test and ran it, verifying that it did indeed fail (the cell count was 99 instead of 100). Then I went to fix the bug and reran the unit test and the regression test passed. I committed the regression test along with the bug fix. Now if anybody breaks this in the source code again in the future, we can immediately identify that the code has regressed. Better yet before committing any changes in the future, running our tests will ensure our changes don't have unexpected side effects - like breaking existing functionality.

There is one more benefit to regression tests - they can save you time. If you ever fixed a bug that involved making changes to the source, and then running the application and performing a series of convoluted steps to replicate the issue, it will be immediately apparent that simply implementing your regression test **before** fixing the bug will let you automate the testing for bug resolution in an efficient manner.

To implement your regression test, you should follow the naming convention of regression<TicketID> for your test functions. If no trac ticket exists for the regression, you should create one first. Using this approach allows the person running a failed regression test easily go and find out more information.

```
//  
// Regression Testing  
//  
  
/** This is our second test case...to check if a raster  
    reports its dimensions properly. It is a regression test  
    for ticket #832 which was fixed with change r7650.  
    */  
void regression832();  
  
// more regression tests go here ...
```

Finally in our test class declaration you can declare privately any data members and helper methods your unit test may need. In our case I will declare a `QgsRasterLayer *` which can be used by any of our test methods. The raster layer will be created in the `initTestCase()` function which is run before any other tests, and then destroyed using `cleanupTestCase()` which is run after all tests. By declaring helper methods (which may be called by various test functions) privately, you can ensure that they won't be automatically run by the `QTest` executable that is created when we compile our test.

```
private:
    // Here we have any data structures that may need to
    // be used in many test cases.
    QgsRasterLayer * mpLayer;
};
```

That ends our class declaration. The implementation is simply inlined in the same file lower down. First our init and cleanup functions:

```
void TestQgsRasterLayer::initTestCase()
{
    // init QGIS's paths - true means that all path will be init from prefix
    QString qgisPath = QCoreApplication::applicationDirPath ();
    QgsApplication::setPrefixPath(qgisPath, TRUE);
#ifdef Q_OS_LINUX
    QgsApplication::setPkgDataPath(qgisPath + "/../share/qgis");
#endif
    //create some objects that will be used in all tests...

    std::cout << "Prefix  PATH: " << QgsApplication::prefixPath().toLocal8Bit().data() \
    << std::endl;
    std::cout << "Plugin  PATH: " << QgsApplication::pluginPath().toLocal8Bit().data() \
    << std::endl;
    std::cout << "PkgData PATH: " << QgsApplication::pkgDataPath().toLocal8Bit().data() \
    << std::endl;
    std::cout << "User DB PATH: " << QgsApplication::qgisUserDbFilePath().toLocal8Bit() \
    .data() << std::endl;

    //create a raster layer that will be used in all tests...
    QString myFileName (TEST_DATA_DIR); //defined in CmakeLists.txt
    myFileName = myFileName + QDir::separator() + "tenbytenraster.asc";
    QFileInfo myRasterFileInfo ( myFileName );
    mpLayer = new QgsRasterLayer ( myRasterFileInfo.filePath(),
```

```
        myRasterFileInfo.completeBaseName() );
}

void TestQgsRasterLayer::cleanupTestCase()
{
    delete mpLayer;
}
```

The above init function illustrates a couple of interesting things.

1. I needed to manually set the QGIS application data path so that resources such as srs.db can be found properly. 2. Secondly, this is a data driven test so we needed to provide a way to generically locate the 'tenbytenraster.asc' file. This was achieved by using the compiler define **TEST_DATA_PATH**. The define is created in the CMakeLists.txt configuration file under <QGIS Source Root>/tests/CMakeLists.txt and is available to all QGIS unit tests. If you need test data for your test, commit it under <QGIS Source Root>/tests/testdata. You should only commit very small datasets here. If your test needs to modify the test data, it should make a copy of it first.

Qt also provides some other interesting mechanisms for data driven testing, so if you are interested to know more on the topic, consult the Qt documentation.

Next let's look at our functional test. The isValid() test simply checks the raster layer was correctly loaded in the initTestCase. QVERIFY is a Qt macro that you can use to evaluate a test condition. There are a few other use macros Qt provide for use in your tests including:

```
QCOMPARE ( actual, expected )
QEXPECT_FAIL ( dataIndex, comment, mode )
QFAIL ( message )
QFETCH ( type, name )
QSKIP ( description, mode )
QTEST ( actual, testElement )
QTEST_APPLESS_MAIN ( TestClass )
QTEST_MAIN ( TestClass )
QTEST_NOOP_MAIN ()
QVERIFY2 ( condition, message )
QVERIFY ( condition )
QWARN ( message )
```

Some of these macros are useful only when using the Qt framework for data driven testing (see the Qt docs for more detail).

```
void TestQgsRasterLayer::isValid()
```

L UNIT TESTING

```
{  
    QVERIFY ( mpLayer->isValid() );  
}
```

Normally your functional tests would cover all the range of functionality of your classes public API where feasible. With our functional tests out the way, we can look at our regression test example.

Since the issue in bug #832 is a misreported cell count, writing our test is simply a matter of using QVERIFY to check that the cell count meets the expected value:

```
void TestQgsRasterLayer::regression832()  
{  
    QVERIFY ( mpLayer->getRasterXDim() == 10 );  
    QVERIFY ( mpLayer->getRasterYDim() == 10 );  
    // regression check for ticket #832  
    // note getRasterBandStats call is base 1  
    QVERIFY ( mpLayer->getRasterBandStats(1).elementCountInt == 100 );  
}
```

With all the unit test functions implemented, there one final thing we need to add to our test class:

```
QTEST_MAIN(TestQgsRasterLayer)  
#include "moc_testqgsrasterlayer.cxx"
```

The purpose of these two lines is to signal to Qt's moc that this is a QTest (it will generate a main method that in turn calls each test function). The last line is the include for the MOC generated sources. You should replace 'testqgsrasterlayer' with the name of your class in lower case.

L.3 Adding your unit test to CMakeLists.txt

Adding your unit test to the build system is simply a matter of editing the CMakeLists.txt in the test directory, cloning one of the existing test blocks, and then search and replacing your test class name into it. For example:

```
#  
# QgsRasterLayer test  
#  
SET(qgis_rasterlayertest_SRCS testqgsrasterlayer.cpp)  
SET(qgis_rasterlayertest_MOC_CPPS testqgsrasterlayer.cpp)
```

```
QT4_WRAP_CPP(qgis_rasterlayertest_MOC_SRCS ${qgis_rasterlayertest_MOC_CPPS})
ADD_CUSTOM_TARGET(qgis_rasterlayertestmoc ALL DEPENDS ${qgis_rasterlayertest_MOC_SRCS})
ADD_EXECUTABLE(qgis_rasterlayertest ${qgis_rasterlayertest_SRCS})
ADD_DEPENDENCIES(qgis_rasterlayertest qgis_rasterlayertestmoc)
TARGET_LINK_LIBRARIES(qgis_rasterlayertest ${QT_LIBRARIES} qgis_core)
INSTALL(TARGETS qgis_rasterlayertest RUNTIME DESTINATION ${QGIS_BIN_DIR})
ADD_TEST(qgis_rasterlayertest ${QGIS_BIN_DIR}/qgis_rasterlayertest)
```

I'll run through these lines briefly to explain what they do, but if you are not interested, just clone the block, search and replace e.g.

```
: '<,'>s/rasterlayer/mynewtest/g
```

Lets look a little more in detail at the individual lines. First we define the list of sources for our test. Since we have only one source file (following the methodology I described above where class declaration and definition are in the same file) its a simple statement:

```
SET(qgis_rasterlayertest_SRCS testqgsrasterlayer.cpp)
```

Since our test class needs to be run through the Qt meta object compiler (moc) we need to provide a couple of lines to make that happen too:

```
SET(qgis_rasterlayertest_MOC_CPPS testqgsrasterlayer.cpp)
QT4_WRAP_CPP(qgis_rasterlayertest_MOC_SRCS ${qgis_rasterlayertest_MOC_CPPS})
ADD_CUSTOM_TARGET(qgis_rasterlayertestmoc ALL DEPENDS ${qgis_rasterlayertest_MOC_SRCS})
```

Next we tell cmake that it must make an executeable from the test class. Remember in the previous section on the last line of the class implementation I included the moc outputs directly into our test class, so that will give it (among other things) a main method so the class can be compiled as an executeable:

```
ADD_EXECUTABLE(qgis_rasterlayertest ${qgis_rasterlayertest_SRCS})
ADD_DEPENDENCIES(qgis_rasterlayertest qgis_rasterlayertestmoc)
```

Next we need to specify any library dependencies. At the moment classes have been implemented with a catch-all QT_LIBRARIES dependency, but I will be working to replace that with the specific Qt libraries that each class needs only. Of course you also need to link to the relevant qgis libraries as required by your unit test.

L UNIT TESTING

```
TARGET_LINK_LIBRARIES(qgis_rasterlayertest ${QT_LIBRARIES} qgis_core)
```

Next I tell cmake to the same place as the qgis binaries itself. This is something I plan to remove in the future so that the tests can run directly from inside the source tree.

```
INSTALL(TARGETS qgis_rasterlayertest RUNTIME DESTINATION ${QGIS_BIN_DIR})
```

Finally here is where the best magic happens - we register the class with ctest. If you recall in the overview I gave in the beginning of this section we are using both QTest and CTest together. To recap, **QTest** adds a main method to your test unit and handles calling your test methods within the class. It also provides some macros like QVERIFY that you can use as to test for failure of the tests using conditions. The output from a QTest unit test is an executable which you can run from the command line. However when you have a suite of tests and you want to run each executable in turn, and better yet integrate running tests into the build process, the **CTest** is what we use. The next line registers the unit test with CMake / CTest.

```
ADD_TEST(qgis_rasterlayertest ${QGIS_BIN_DIR}/qgis_rasterlayertest)
```

The last thing I should add is that if your test requires optional parts of the build process (e.g. PostgreSQL support, GSL libs, GRASS etc.), you should take care to enclose your test block inside a IF () block in the CMakeLists.txt file.

L.4 Building your unit test

To build the unit test you need only to make sure that ENABLE_TESTS=true in the cmake configuration. There are two ways to do this:

1. Run cmake .. (cmakesetup .. under windows) and interactively set the ENABLE_TESTS flag to ON.
1. Add a command line flag to cmake e.g. cmake -DENABLE_TESTS=true ..

Other than that, just build QGIS as per normal and the tests should build too.

L.5 Run your tests

The simplest way to run the tests is as part of your normal build process:

```
make && make install && make test
```

The make test command will invoke CTest which will run each test that was registered using the ADD_TEST CMake directive described above. Typical output from make test will look like this:

Running tests...

Start processing tests

Test project /Users/tim/dev/cpp/qgis/build

```
1/ 3 Testing qgis_applicationtest      ***Exception: Other
2/ 3 Testing qgis_filewritertest      *** Passed
3/ 3 Testing qgis_rasterlayertest      *** Passed
```

0% tests passed, 3 tests failed out of 3

The following tests FAILED:

1 - qgis_applicationtest (OTHER_FAULT)

Errors while running CTest

make: *** [test] Error 8

If a test fails, you can use the `ctest` command to examine more closely why it failed. Use the `-R` option to specify a regex for which tests you want to run and `-V` to get verbose output:

```
[build] ctest -R appl -V
```

Start processing tests

Test project /Users/tim/dev/cpp/qgis/build

Constructing a list of tests

Done constructing a list of tests

Changing directory into /Users/tim/dev/cpp/qgis/build/tests/src/core

```
1/ 3 Testing qgis_applicationtest
```

Test command: /Users/tim/dev/cpp/qgis/build/tests/src/core/qgis_applicationtest

***** Start testing of TestQgsApplication *****

Config: Using QTest library 4.3.0, Qt 4.3.0

PASS : TestQgsApplication::initTestCase()

Prefix PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/..

Plugin PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/../../lib/qgis

PkgData PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/../../share/qgis

User DB PATH: /Users/tim/.qgis/qgis.db

PASS : TestQgsApplication::getPaths()

Prefix PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/..

Plugin PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/../../lib/qgis

PkgData PATH: /Users/tim/dev/cpp/qgis/build/tests/src/core/../../share/qgis

User DB PATH: /Users/tim/.qgis/qgis.db

QDEBUG : TestQgsApplication::checkTheme() Checking if a theme icon exists:

QDEBUG : TestQgsApplication::checkTheme()

/Users/tim/dev/cpp/qgis/build/tests/src/core/.. \

//share/qgis/themes/default/mIconProjectionDisabled.png


```
FAIL!   : TestQgsApplication::checkTheme() '!myPixmap.isNull()' returned FALSE. ()
Loc: [/Users/tim/dev/cpp/qgis/tests/src/core/testqgsapplication.cpp(59)]
PASS    : TestQgsApplication::cleanupTestCase()
Totals: 3 passed, 1 failed, 0 skipped
***** Finished testing of TestQgsApplication *****
-- Process completed
***Failed

0% tests passed, 1 tests failed out of 1

The following tests FAILED:
1 - qgis_applicationtest (Failed)
Errors while running CTest
```

Well that concludes this section on writing unit tests in QGIS. We hope you will get into the habit of writing test to test new functionality and to check for regressions. Some aspects of the test system (in particular the CMakeLists.txt parts) are still being worked on so that the testing framework works in a truly platform way. I will update this document as things progress.

M HIG (Human Interface Guidelines)

In order for all graphical user interface elements to appear consistent and to all the user to instinctively use dialogs, it is important that the following guidelines are followed in layout and design of GUIs.

1. Group related elements using group boxes: Try to identify elements that can be grouped together and then use group boxes with a label to identify the topic of that group. Avoid using group boxes with only a single widget / item inside.
2. Capitalise first letter only in labels: Labels (and group box labels) should be written as a phrase with leading capital letter, and all remaining words written with lower case first letters
3. Do not end labels for widgets or group boxes with a colon: Adding a colon causes visual noise and does not impart additional meaning, so don't use them. An exception to this rule is when you have two labels next to each other e.g.: Label1 **Plugin** Label2 [/path/to/plugins]
4. Keep harmful actions away from harmless ones: If you have actions for 'delete', 'remove' etc, try to impose adequate space between the harmful action and innocuous actions so that the users is less likely to inadvertently click on the harmful action.
5. Always use a QDialogBox for 'OK', 'Cancel' etc buttons: Using a button box will ensure that the order of 'OK' and 'Cancel' etc, buttons is consistent with the operating system / locale / desktop environment that the user is using.

N GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The Program, below, refers to any such program or work, and a work based on the Program means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term modification.) Each licensee is addressed as you.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under

the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from

distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and any later version, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM - AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PRO-

GRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

N.1 Quantum GIS Qt exception for GPL

In addition, as a special exception, the QGIS Development Team gives permission to link the code of this program with the Qt library, including but not limited to the following versions (both free and commercial): Qt/Non-commercial Windows, Qt/Windows, Qt/X11, Qt/Mac, and Qt/Embedded (or with modified versions of Qt that use the same license as Qt), and distribute linked combinations including the two. You must obey the GNU General Public License in all respects for all of the code used other than Qt. If you modify this file, you may extend this exception to your version of the file, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

Literature

- [1] T. Mitchell. Web mapping illustrated, published by o'reilly, 2005.
- [2] M. Neteler and H. Mitasova. Open source gis: A grass gis approach. 3. edition, springer, new york, 2008.

Web-References

- [3] GRASS GIS. <http://grass.osgeo.org>, 2008.
- [4] PostGIS. <http://postgis.refractory.net/>, 2006.
- [5] Web Map Service (1.1.1) Implementation Specification. <http://portal.opengeospatial.org>, 2002.
- [6] Web Map Service (1.3.0) Implementation Specification. <http://portal.opengeospatial.org>, 2004.

Indice analitico

%%, [44](#)

actions, [44](#)

 defining, [44](#)

 examples, [45](#)

 using, [45](#)

Attiva/disattiva modifica, [53](#)

attributes, [49](#)

bookmarks, [29](#)

command line options, [15](#)

coordinate reference system, [75](#)

crashes, [109](#)

CRS, [75](#)

data

 sample, [11](#)

data providers, [113](#)

delimited text, [31](#)

documentation, [1](#)

editing, [50](#)

 an existing layer, [52](#)

 copying features, [57](#)

 creating a new layer, [59](#)

 cutting features, [57](#)

 icons, [53](#)

 pasting features, [57](#)

 saving changes, [59](#)

 snap, [59](#)

EPSG, [80](#)

ESRI

 shapefiles, [31](#)

GDAL

 supported formats, [193](#)

GRASS, [85](#)

 attribute linkage, [92](#)

 attribute storage, [92](#)

 category settings, [93](#)

digitizing, [91](#)

digitizing tools, [93](#), [94](#)

display results, [99](#)

edit permissions, [97](#)

loading data, [86](#)

region, [97](#)

 display, [97](#)

 editing, [97](#)

snapping tolerance, [94](#)

starting QGIS, [85](#)

symbology settings, [95](#)

table editing, [96](#)

toolbox, [97](#), [98](#)

 Browser, [100](#)

 customize, [101](#)

 modules, [196](#)

topology, [91](#)

vector data model, [91](#)

identify

 WMS, [75](#)

installation, [11](#)

layer

 visibility, [20](#)

layers

 initial visibility, [25](#)

layout

 toolbars, [20](#)

legend, [20](#)

license

 exception, [286](#)

 GPL, [281](#)

main window, [17](#)

map

 overview, [23](#)

 view, [22](#)

MapInfo

 MIF files, [31](#)

- TAB files, [31](#)
- measure, [26](#)
- measure:areas, [26](#)
- measure:line length, [26](#)
- menus, [18](#)
- MIF files, [31](#)
- OGC
 - Authentication, [77](#)
 - coordinate reference system, [75](#)
 - CRS, [75](#)
 - introduction, [71](#)
 - WMS
 - client, [71](#)
- OGR, [31](#)
 - supported formats, [193](#)
- pan
 - arrow keys, [23](#)
- plugin
 - georeferencer, [114](#)
 - grass toolbox, [114](#)
- plugins, [109](#), [147](#)
 - coordinate capture, [114](#)
 - copyright, [114](#)
 - core, [114](#)
 - delimited text, [114](#)
 - DXF2Shape, [114](#)
 - gps, [114](#)
 - graticule, [114](#)
 - installing, [110](#)
 - Interpolation, [114](#)
 - manager, [109](#), [120](#)
 - managing, [109](#)
 - MapServer Export, [114](#)
 - north arrow, [114](#)
 - OGR converter, [114](#)
 - Plugin Installer, [114](#)
 - Python Plugin Installer, [110](#)
 - quick print, [114](#)
 - scalebar, [114](#)
 - spit, [114](#)
 - types, [109](#)
 - upgrading, [110](#)
 - Zoom To Point, [147](#)
- plugins settings, [114](#)
- PostGIS, [31](#), [80](#)
 - Exporting, [37](#)
 - layers, [34](#)
 - query builder, [62](#)
 - spatial index, [38](#)
 - GiST, [38](#)
 - SPIT, [37](#)
 - editing field names, [38](#)
 - importing data, [37](#)
 - loading, [37](#)
 - reserved words, [38](#)
- PostgreSQL
 - connection, [34](#), [35](#)
 - testing, [35](#)
 - connection manager, [35](#)
 - connection parameters, [35](#)
 - database, [35](#)
 - host, [35](#)
 - layer details, [36](#)
 - loading layers, [34](#), [36](#)
 - password, [35](#)
 - port, [35](#)
 - PostGIS, [31](#)
 - query builder, [62](#)
 - username, [35](#)
- Print composer
 - tools, [102](#)
- Projections
 - coordinate reference system, [75](#)
 - CRS, [75](#)
 - custom, [83](#)
 - enabling, [83](#)
 - specifying, [80](#)
 - SRS, [75](#)
 - WMS, [75](#)
 - working with, [80](#)
- projects, [26](#)

- Query Builder, 61
 - adding fields, 61
 - changing layer definitions, 62
 - generating sample list, 61
 - getting all values, 61
 - testing queries, 62
- query builder
 - PostGIS, 62
 - PostgreSQL, 62
- raster layer
 - classify, 68
- raster layers, 63
 - context menu, 64
 - data formats, 63
 - definition, 63
 - GDAL implementation, 63
 - georeferenced, 63
 - histogram, 69
 - loading, 64
 - metadata), 70
 - properties, 68
 - pyramids, 69
 - resolution pyramids, 69
 - standard deviation, 66
 - statistics, 70
 - supported channels, 66
 - supported formats, 193
 - transparency, 67
- rasters
 - metadata, 75
 - properties, 75
 - WMS, 71
- rendering, 23
 - options, 25
 - quality, 25
 - scale dependent, 24
 - suspending, 24
 - update during drawing, 25
- scale, 24
 - calculate, 24
- security, 35
- settings, 35
- shapefile
 - format, 31
 - loading, 31
 - specification, 31
- shapefiles, 31
- SHP files, 31
- spatial bookmarks,
 - seebookmarks29
- spatial index
 - shapefiles, 33
- SRS, 75
- symbology
 - changing, 41
- TAB files, 31
- Toggle Editing, 53
- toolbars, 20
- vector layers, 31–62
 - add
 - island, 57
 - ring, 57
 - adding
 - feature, 54
 - vertex, 56
 - ArcInfo Coverage, 34
 - copy
 - feature, 57
 - cut
 - feature, 57
 - deleting
 - feature, 58
 - vertex, 57
 - editing, 52
 - vertex, 56
 - ESRI shapefiles, 31
 - MapInfo, 34
 - move
 - feature, 55
 - moving

- vertex, [56](#)
 - paste
 - feature, [57](#)
 - PostGIS, *vedi* PostGIS
 - properties dialog, [39](#)
 - renderers
 - continuous color, [41](#)
 - graduated symbol, [41](#)
 - single symbol, [40](#)
 - unique value, [41](#)
 - split
 - feature, [56](#)
 - styles, [41](#)
 - symbolology, [40](#)
 - transparency, [42](#)
- WFS
- remote server, [78](#)
- WKT, [80](#)
- WMS
- capabilities, [75](#)
 - client, [71](#)
 - about, [71](#)
 - connection parameters, [72](#)
 - layers, [73](#)
 - limits, [77](#)
 - coordinate reference system, [75](#)
 - CRS, [75](#)
 - GetFeatureInfo, [75](#)
 - identify, [75](#)
 - image encoding, [73](#)
 - layer settings
 - editing, [77](#)
 - layer transparency, [74](#)
 - metadata, [75](#)
 - properties, [75](#)
 - remote server
 - authentication, [77](#)
 - layer ordering, [74](#)
 - selection, [72](#)
 - URL, [73](#)
 - secured layers, [77](#)
- URL, [72](#)
 - zoom
 - mouse wheel, [22](#)