

# Quantum GIS Design Document

QGIS Core Design Team\*

11th March 2003

---

\* Gary E Sherman, Mark Coletti, Denis Antipov

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Definitions</b>	<b>4</b>
<b>3</b>	<b>History</b>	<b>4</b>
<b>4</b>	<b>Goals</b>	<b>5</b>
4.1	List of Goals . . . . .	5
<b>5</b>	<b>Requirements</b>	<b>5</b>
5.1	User Interface . . . . .	6
5.2	Standards . . . . .	6
5.3	Data Formats . . . . .	6
5.4	Platform Support . . . . .	6
5.5	Toolkits . . . . .	7
<b>6</b>	<b>Use Cases</b>	<b>7</b>
6.1	Actors . . . . .	7
6.2	Primary Use Cases . . . . .	7
6.3	Case Descriptions . . . . .	8
6.3.1	Load Data . . . . .	9
6.3.2	Browse Data . . . . .	10
6.3.3	Load Plugin . . . . .	11
6.3.4	Find Feature . . . . .	12
6.3.5	Identify Feature . . . . .	13
6.3.6	Get Help . . . . .	14
6.3.7	Customize . . . . .	15
6.3.8	Save Project . . . . .	16
6.3.9	Restore Project . . . . .	17
6.3.10	Navigate . . . . .	18
6.3.11	Save Image . . . . .	19
6.3.12	Print . . . . .	20
6.3.13	Write Script . . . . .	21
6.3.14	Run Script . . . . .	22
6.3.15	Edit Data . . . . .	23
6.3.16	Digitize . . . . .	24
6.3.17	Buffer Feature . . . . .	25
6.3.18	Process Data - Union . . . . .	26
6.3.19	Process Data - Intersect . . . . .	27
6.3.20	Process Data - Merge . . . . .	28
6.3.21	Change projection . . . . .	29
6.3.22	Import Data . . . . .	30
6.3.23	Export Data . . . . .	31
6.3.24	Export Selected Set . . . . .	32
6.3.25	Select Spatially . . . . .	33
6.3.26	Select by Attribute . . . . .	34

<b>7</b>	<b>Core Architecture</b>	<b>34</b>
7.1	Main Window . . . . .	34
7.2	Plugins . . . . .	35
7.3	Class Diagram . . . . .	35
7.4	Load Sequence . . . . .	35

# 1 Introduction

This document describes the requirements and design for Quantum GIS (QGIS), a desktop GIS application for Linux and Unix. This document presents use cases, high-level class diagrams, and additional information about the design and implementation of QGIS.

QGIS is hosted on SourceForge at <http://qgis.sourceforge.net>. The current release of QGIS is a viewer with a minimal feature set, including loading data, browsing, and identifying features.

The design outlined in this document represents the next phase of QGIS development, which will move the application to a flexible and extensible platform for working with spatial data.

Note that it's presumed that the reader is familiar with C++, object-oriented design, and UML.

# 2 Definitions

The following is a list of definitions for terms used in this document. Some of these definitions are taken from the Association for Geographic Information and the University Of Edinburgh Department of Geography online dictionary of GIS terms available at <http://www.geo.ed.ac.uk/agidict/welcome.html>.

**Attribute:** A fact describing an entity in a relational data model, equivalent to the column in a relational table.

**Feature:** A set of points, lines or polygons in a spatial database that represent a real-world entity. The terms feature and object are often used synonymously.

**GIS:** Geographic Information System. A software system that provides the ability to view and analyze spatial data and its attributes.

**Layer:** A dataset that has a spatial context and can be drawn on a map canvas. Layers may have associated attributes.

**Plugin:** A compiled library that can be dynamically loaded into an application to provide new functionality.

**Projection:** A method of representing the earth's three-dimensional surface as a flat two-dimensional surface. This normally involves a mathematical model that transforms the locations of features on the earth's surface to locations on a two-dimensional surface. Because the earth is three-dimensional, some method must be used to depict the map in two dimensions. Therefore such representations distort some parameter of the earth's surface, be it distance, area, shape, or direction.

# 3 History

The QGIS project was registered with SourceForge on June 15, 2002. Since that time, QGIS has developed into a minimally functional viewer with support for shapefiles<sup>1</sup> and vector data stored in a PostgreSQL<sup>2</sup> database using the PostGIS<sup>3</sup> extensions.

The development thus far has been useful in developing an understanding of the challenges involved in developing a more robust Open Source desktop GIS application. In March 2003, planning began to restructure QGIS in order to make it more extensible and to provide a means to add advanced functionality through the use of plugins.

---

<sup>1</sup>ESRI format for file-based spatial data.

<sup>2</sup>PostgreSQL Relational Database - <http://www.postgresql.org>

<sup>3</sup>PostGIS extension - <http://postgis.refrains.net>

The current version of QGIS (0.0.9) is still usable as a simple GIS viewer for shapefiles and PostGIS layers. Only minor maintenance is being done on the current code base at this time.

## 4 Goals

There are already other Open Source GIS projects available today. Many have asked what purpose QGIS will serve:

- Will QGIS be a complete desktop GIS application?
- Will it compete with commercial products?
- Why are you developing another GIS application?

The answers to these questions are not clear-cut. A list of high-level goals for the QGIS project are enumerated below. The reader can perhaps use this information to answer these and other questions related to the project:

### 4.1 List of Goals

1. Provide an easy to use desktop GIS application
2. Provide an easy to install application for users with minimum system experience
3. Support common data formats
4. Provide the foundation for more advanced tools (plugins)
5. Become a tool for spatial data collection
6. Support data analysis and conversion
7. Print/plot maps
8. Integrate with Internet mapping technologies

Section 5 provides details about the functional requirements.

## 5 Requirements

This section describes the functional requirements of Quantum GIS. These functional requirements drive the use cases discussed in Section 6.

QGIS will provide GIS functionality somewhere between a simple viewer and an industrial strength application. The ultimate nature of QGIS is only limited by the talent of those software engineers who will provide advanced capabilities through plugins.

Some of the major design requirements include:

- Extensible architecture using plugins
- Internationalization
- Integrated scripting language

- Projection on-the-fly
- Flexible symbology for all feature classes
- Ability to render and browse data in many formats:
  - Spatio-temporal data using a feature-centric model
  - Shapefiles
  - PostgreSQL / PostGIS layers
  - Raster
- Support for OpenGIS implementation Specifications
  - Geography Markup Language (GML)
  - Web Feature Service (WFS)

## 5.1 User Interface

QGIS will use the SDI (Single Document Interface). QGIS currently has (and will have) standard GUI interface elements such as menus, toolbar, and a statusbar. In addition, the standard QGIS interface will have a legend panel and a map canvas (or drawing area).

## 5.2 Standards

Development of QGIS will proceed with adoption of applicable OpenGIS standards. This will include support for GML and possibly WFS.

## 5.3 Data Formats

Any data format could be supported by the development of a plugin that reads and renders the data store. The “standard” formats that will be included in the core QGIS are similar to those currently supported:

- Shapefiles
- PostgreSQL/PostGIS
- Rasters

Format support will be provided by plugins.

## 5.4 Platform Support

Initially, QGIS will target Linux and other Unix operating systems supported by the Qt toolkit. Coding during development of QGIS and plugins will be done in a way so as to not introduce any platform dependencies. This will provide to possibility of a Windows version at some point in the future.

## 5.5 Toolkits

QGIS will leverage existing libraries and toolkits to support the GUI, GIS data stores, and GIS processing algorithms. At present the identified toolkits include:

- Qt (<http://www.trolltech.com>)
- GDAL and OGR (<http://www.remotesensing.org/gdal>)
- Proj.4 (<http://www.remotesensing.org/proj/>)

## 6 Use Cases

Use cases provide a means to identify and visualize the major goal oriented tasks the application should address.

### 6.1 Actors

Actors are really persons (or physical entities) that use a system to achieve a specific goal (these goals are expressed as use cases). A number of actors could be defined for QGIS, however at this point the simple approach is taken. The actors are:

- Casual GIS User
- Professional GIS User

These two actors are sufficient to frame the development and discussion of QGIS use cases.

### 6.2 Primary Use Cases

The primary use cases for QGIS are listed below in no particular order of importance:

- Load data
- Browse data
- Install plugin
- Find feature
- Identify feature
- Get help
- Customize
- Save project
- Restore project
- Navigate map
  - Pan
  - Zoom
- Save image
- Print
  - Print image
  - Print metadata
  - Print feature information
- Write script
- Run script
- Edit data
  - Digitize
- Buffer feature
- Process data
  - Union
  - Merge
  - Intersect
- Convert data
  - Change projection
- Import data
- Export data
  - Export selected set
- Select features
  - Select by attribute
  - Select spatially

### 6.3 Case Descriptions

In the sections that follow, the use cases are presented in no particular order.



### 6.3.1 Load Data

**Use Case:** Load Data

**Goal in Context:** Load a data set from a data store and display it

**Scope & Level:** Primary task

**Preconditions:** Application is running

**Success End Condition:** Data set is loaded and displayed

**Failed End Condition:** Application is in pre-load state

**Primary Actor(s):** Casual, Professional user

**Trigger:** User wants to view data

**Description of Steps:**

1. Decision is made to view a dataset
2. Navigate to the location of the dataset
3. Select dataset
4. Dataset is displayed

### 6.3.2 Browse Data

<b>Use Case:</b>	Browse Data
<b>Goal in Context:</b>	Browse through a loaded data set and its features
<b>Scope &amp; Level:</b>	Primary task
<b>Preconditions:</b>	Application is running, data set loaded
<b>Success End Condition:</b>	Ability to browse, look at the needed information about loaded data
<b>Failed End Condition:</b>	Unable to browse and look at the loaded data information
<b>Primary Actor(s):</b>	Casual, Professional user
<b>Trigger:</b>	User wants to browse the features and other information about the data set
<b>Description of Steps:</b>	<ol style="list-style-type: none"><li>1. User decides to view the features and other info</li><li>2. User sees a list of available items in the data set</li><li>3. User chooses the desired item in the browse set</li><li>4. The details about the item are displayed</li></ol>

### 6.3.3 Load Plugin

- Use Case:** Load plugin
- Goal in Context:** Load a plugin into the 'core' system from the available list of installed plugins
- Scope & Level:** Primary task
- Preconditions:** Application is running
- Success End Condition:** Plugin loaded and the the new tools corresponding to the plugin appear available
- Failed End Condition:** Application is in the initial or the previous plugin state
- Primary Actor(s):** Casual, Professional user
- Trigger:** User wants to view data with a different plugin, or wants to add certain functionality to the existing state
- Description of Steps:**
1. User decides to load a new plugin
  2. User inspects all the installed plugins
  3. User selects a plugin
  4. A plugin is loaded and new functionality added

#### 6.3.4 Find Feature

<b>Use Case:</b>	Find feature
<b>Goal in Context:</b>	Find a feature on a map or loaded data set
<b>Scope &amp; Level:</b>	Primary task
<b>Preconditions:</b>	Application is running and a data set loaded
<b>Success End Condition:</b>	A feature that corresponds to the search condition found if such exists and not found if not exists
<b>Failed End Condition:</b>	A feature that exists and corresponds to the search conditions not found
<b>Primary Actor(s):</b>	Casual, Professional user
<b>Trigger:</b>	User wants to find a feature on a map or data set
<b>Description of Steps:</b>	<ol style="list-style-type: none"><li>1. User decides to find a feature in a loaded data set</li><li>2. User submits a search query that would identify somehow the feature</li><li>3. A feature is displayed if found or a message notifying that it was not found</li></ol>

### 6.3.5 Identify Feature

**Use Case:** Identify Feature

**Goal in Context:** After selecting a feature on a map, display its attributes

**Scope & Level:** Primary task

**Preconditions:** Application is running and a data set loaded

**Success End Condition:** Attributes of a feature displayed

**Failed End Condition:** Nothing displayed when a feature selected

**Primary Actor(s):** Casual, Professional user

**Trigger:** User wants to view attributes of a feature

**Description of Steps:**

1. User decides to view attributes of a particular feature
2. User chooses a feature on the map
3. User selects a feature
4. Attributes are displayed

### 6.3.6 Get Help

- Use Case:** Get Help
- Goal in Context:** Load help system
- Scope & Level:** Primary task
- Preconditions:** Application is running
- Success End Condition:** Help system loaded and the menu of help topics displayed
- Failed End Condition:** Application is in 'helpless' state
- Primary Actor(s):** Casual, Professional user
- Trigger:** User needs help with an application or one of its plugins
- Description of Steps:**
1. User finds out that they don't know as much about QGIS as they thought they did
  2. User launches help system
  3. User selects one of the topics/subtopics of interest
  4. User is enlightened about rich QGIS functionality and how to use it

### 6.3.7 Customize

- Use Case:** Customize
- Goal in Context:** Customize application user interface
- Scope & Level:** Primary task
- Preconditions:** Application is running a plugin that is being customized loaded
- Success End Condition:** Interface changes according to customization
- Failed End Condition:** No changes occur after customization
- Primary Actor(s):** Casual, Professional user
- Trigger:** User wants to change the interface
- Description of Steps:**
1. User decides to change the interface
  2. User launches the customization system
  3. User adjusts the interface according to the customizationsystem
  4. Changes in the interface take effect

### 6.3.8 Save Project

**Use Case:** Save Project

**Goal in Context:** Save entire project in a specified file format

**Scope & Level:** Primary task

**Preconditions:** Application is running, data set(s) loaded

**Success End Condition:** Current project saved in a file

**Failed End Condition:** Current project not saved in a file

**Primary Actor(s):** Casual, Professional user

**Trigger:** User wants to save the layout of map, layers, etc.

**Description of Steps:**

1. User decides to save the current project
2. User chooses location and filename for the project
3. The project saved in a file



### 6.3.9 Restore Project

- Use Case:** Restore Project
- Goal in Context:** Load a project from a previously saved file
- Scope & Level:** Primary task
- Preconditions:** Application is running
- Success End Condition:** Project loaded and all the data sets restored in the saved condition
- Failed End Condition:** Project is not restored
- Primary Actor(s):** Casual, Professional user
- Trigger:** User wants to restore the project that was saved
- Description of Steps:**
1. User decides to restore a project
  2. User navigates to the location of the project
  3. User selects a project
  4. Project is restored

### 6.3.10 Navigate

**Use Case:** Navigate

**Goal in Context:** Pan and Zoom on a data set

**Scope & Level:** Primary task

**Preconditions:** Application is running, data set loaded and rendered

**Success End Condition:** Data set is zoomed or panned

**Failed End Condition:** No changes to the view of a data set

**Primary Actor(s):** Casual, Professional user

**Trigger:** User wants to zoom or pan on a data set

**Description of Steps:**

1. User decides to zoom or pan on a data set
2. User uses zoom or pan
3. The view is zoomed or panned

### 6.3.11 Save Image

- Use Case:** Save Image
- Goal in Context:** Save the current view of the data set in an image file
- Scope & Level:** Primary task
- Preconditions:** Application is running, data set loaded and rendered
- Success End Condition:** An image displaying the current view of a data set saved
- Failed End Condition:** No image saved
- Primary Actor(s):** Casual, Professional user
- Trigger:** User wants to save the current view of a data set as an image
- Description of Steps:**
1. User decides to save an image
  2. User chooses the location and filename of the image
  3. The image saved

### 6.3.12 Print

**Use Case:** Print

**Goal in Context:** Output on a printer the current data set as an image, metadata, and its set of features

**Scope & Level:** Primary task

**Preconditions:** Application is running, a dataset loaded and rendered

**Success End Condition:** The different types of data (image, metadata, feature set) being output on a printer

**Failed End Condition:** No attempt to output data on a printer

**Primary Actor(s):** Casual, Professional user

**Trigger:** User wants to output data on a printer

**Description of Steps:**

1. User decides to output data set on a printer
2. User chooses the type of data output (image, metadata, feature set)
3. User selects a printer
4. Dataset is output on a printer in a specified format

### 6.3.13 Write Script

**Use Case:** Write Script

**Goal in Context:** Write a script to perform some QGIS function in an automated manner

**Scope & Level:** Primary task

**Preconditions:** Application is running

**Success End Condition:** Script is completed and tested

**Failed End Condition:** Script is not saved and application is in pre-script state

**Primary Actor(s):** Professional user

**Trigger:** User needs a script to perform a repetitive or specialized operation

**Description of Steps:**

1. Identify need for the script
2. Open the script editor
3. Write and test script in iterative fashion
4. Save script for future use

### 6.3.14 Run Script

**Use Case:** Run Script

**Goal in Context:** Execute a stored script to perform a task

**Scope & Level:** Primary task

**Preconditions:** Application is running with appropriate or required data loaded

**Success End Condition:** Script performs desired function

**Failed End Condition:** Script exits gracefully

**Primary Actor(s):** Casual, Professional user

**Trigger:** User needs to perform a repetitive or complex task for which a script exists

**Description of Steps:**

1. Identify script that can be used to perform the desired task
2. Select the script
3. Additional requirements necessary to execute the script are met
4. Execute script
5. Result of script is reported to the user or displayed on the map canvas

### 6.3.15 Edit Data

**Use Case:** Edit Data

**Goal in Context:** Edit data to change the attributes, spatial location, or shape of a feature or features

**Scope & Level:** Primary task

**Preconditions:** Data layer to be edited has been loaded

**Success End Condition:** Data is changed and saved to the data store

**Failed End Condition:** Data is not changed but remains in the state prior to beginning the edit operation

**Primary Actor(s):** Professional user

**Trigger:** User needs to modify data

**Description of Steps:**

1. Enter edit mode
2. Locate feature to be modified
3. Edit the feature
4. Save data
5. Exit edit mode

### 6.3.16 Digitize

**Use Case:** Digitize Data

**Goal in Context:** Create a new data layer or add to an existing layer by digitizing information from a raster image on the screen (heads-up digitizing)

**Scope & Level:** Primary task

**Preconditions:** Raster is loaded and displayed

**Success End Condition:** New features are added to the data layer and saved to the data store

**Failed End Condition:** Existing data is not changed but remains in the state prior to beginning the edit operation

**Primary Actor(s):** Professional user

**Trigger:** User needs to create new data from a raster

**Description of Steps:**

1. Loads or create the target layer
2. Enter edit mode
3. Digitize features using the underlying raster
4. Add attributes to each feature as it is completed
5. Save data
6. Exit edit mode



### 6.3.17 Buffer Feature

<b>Use Case:</b>	Buffer Feature
<b>Goal in Context:</b>	Create a new data layer that contains a buffer around one or more features on an existing layer
<b>Scope &amp; Level:</b>	Primary task
<b>Preconditions:</b>	Data layer containing features to buffer is loaded and displayed
<b>Success End Condition:</b>	New data layer containing polygon buffers is created
<b>Failed End Condition:</b>	No new data layer is created and application remains in state prior to beginning of buffer operation
<b>Primary Actor(s):</b>	Professional user
<b>Trigger:</b>	User needs to buffer features for the purpose of performing spatial analysis or answering a question about proximity of features
<b>Description of Steps:</b>	<ol style="list-style-type: none"><li>1. Select feature or features to buffer</li><li>2. Specify the buffer distance in appropriate map units</li><li>3. Create buffer</li><li>4. Save buffer layer to permanent store</li></ol>

### 6.3.18 Process Data - Union

- Use Case:** Process Data - Union
- Goal in Context:** Create a new data layer that contains the union of two existing layers
- Scope & Level:** Primary task
- Preconditions:** Data layers needed for operation are loaded and displayed
- Success End Condition:** New data layer containing union of source layers is created
- Failed End Condition:** No new data layer is created and application remains in state prior to beginning of union operation
- Primary Actor(s):** Professional user
- Trigger:** User needs to union features for the purpose of performing spatial analysis
- Description of Steps:**
1. Select layers to union
  2. Perform union operation
  3. Save resulting layer to permanent store

### 6.3.19 Process Data - Intersect

**Use Case:** Process Data - Intersect

**Goal in Context:** Create a new data layer that contains the intersection of two existing layers

**Scope & Level:** Primary task

**Preconditions:** Data layers needed for operation are loaded and displayed

**Success End Condition:** New data layer containing intersection of source layers is created

**Failed End Condition:** No new data layer is created and application remains in state prior to beginning of intersect operation

**Primary Actor(s):** Professional user

**Trigger:** User needs to create a new layer representing the intersection of features from two source layers

**Description of Steps:**

1. Select layers to intersect
2. Perform intersect operation
3. Save resulting layer to permanent store

### 6.3.20 Process Data - Merge

- Use Case:** Process Data - Merge
- Goal in Context:** Create a new data layer that contains the contents of two existing layers. The layers are “tiled” into one.
- Scope & Level:** Primary task
- Preconditions:** Data layers needed for operation are loaded and displayed
- Success End Condition:** New data layer containing merge of source layers is created
- Failed End Condition:** No new data layer is created and application remains in state prior to beginning of merge operation
- Primary Actor(s):** Professional user
- Trigger:** User needs to combine the features from two source layers into one
- Description of Steps:**
1. Select layers to merge
  2. Perform merge operation
  3. Save resulting layer to permanent store

### 6.3.21 Change projection

**Use Case:** Change Projection

**Goal in Context:** Project a source data layer into a new map projection to create a new data layer

**Scope & Level:** Primary task

**Preconditions:** Data layer needed for operation is loaded and displayed

**Success End Condition:** Data layer in desired projection is created from the source layer

**Failed End Condition:** No new data layer is created and application remains in state prior to beginning of merge operation

**Primary Actor(s):** Professional user

**Trigger:** User needs to a copy of a data layer in a different projection

**Description of Steps:**

1. Select layer to project
2. Specify the projection of the source layer
3. Specify the projection for the new layer
4. Project the data
5. Save resulting layer to permanent store

### 6.3.22 Import Data

**Use Case:** Import Data

**Goal in Context:** Create a new data layer by importing data from a source not directly usable with QGIS

**Scope & Level:** Primary task

**Preconditions:** Application is running

**Success End Condition:** New data layer is created

**Failed End Condition:** No new data layer is created and application remains in state prior to beginning of import operation

**Primary Actor(s):** Professional user

**Trigger:** User needs to work with data not currently in a format supported by QGIS

**Description of Steps:**

1. Select layer to project
2. Specify the projection of the source layer
3. Specify the projection for the new layer
4. Project the data
5. Save resulting layer to permanent store

### 6.3.23 Export Data

<b>Use Case:</b>	Export Data
<b>Goal in Context:</b>	Convert data from one format (source) to another (target) for the purpose of using it in another application.
<b>Scope &amp; Level:</b>	Primary task
<b>Preconditions:</b>	Data layer needed for operation is loaded and displayed
<b>Success End Condition:</b>	Export file is created
<b>Failed End Condition:</b>	Export file is not created and application remains in state prior to beginning of export operation
<b>Primary Actor(s):</b>	Professional user
<b>Trigger:</b>	User needs to export data for use by another application
<b>Description of Steps:</b>	<ol style="list-style-type: none"><li>1. Select data layer to export</li><li>2. Choose output location</li><li>3. Export the data</li></ol>

### 6.3.24 Export Selected Set

- Use Case:** Export Selected Set
- Goal in Context:** Convert a selected set of feature data from one format (source) to another (target) for the purpose of using it in another application.
- Scope & Level:** Primary task
- Preconditions:** Data layer needed for operation is loaded and displayed
- Success End Condition:** Export file is created
- Failed End Condition:** Export file is not created and application remains in state prior to beginning of export operation
- Primary Actor(s):** Professional user
- Trigger:** User needs to export data for use by another application
- Description of Steps:**
1. Select data layer to export
  2. Select features to be included in the export
  3. Choose output location
  4. Export the features



### 6.3.25 Select Spatially

- Use Case:** Select Spatially
- Goal in Context:** Select features on the map canvas for purpose of using in another function or operation
- Scope & Level:** Subfunction
- Preconditions:** Data layer needed for operation is loaded and displayed
- Success End Condition:** Features are selected and highlighted on map
- Failed End Condition:** No selection set is created and application remains in state prior to beginning of export operation
- Primary Actor(s):** Professional user
- Trigger:** User needs to select features for use in another function or operation
- Description of Steps:**
1. Select data layer containing features of interest
  2. Interactively draw a box around features
  3. Select

### 6.3.26 Select by Attribute

<b>Use Case:</b>	Select by Attribute
<b>Goal in Context:</b>	Select features on the map canvas for purpose of using in another function or operation
<b>Scope &amp; Level:</b>	Subfunction
<b>Preconditions:</b>	Data layer needed for operation is loaded and displayed
<b>Success End Condition:</b>	Features are selected and highlighted on map
<b>Failed End Condition:</b>	No selection set is created and application remains in state prior to beginning of export operation
<b>Primary Actor(s):</b>	Professional user
<b>Trigger:</b>	User needs to select features for use in another function or operation
<b>Description of Steps:</b>	<ol style="list-style-type: none"><li>1. Select data layer containing features of interest</li><li>2. Open the query builder to define the selection criteria</li><li>3. Execute the query to select features</li></ol>

## 7 Core Architecture

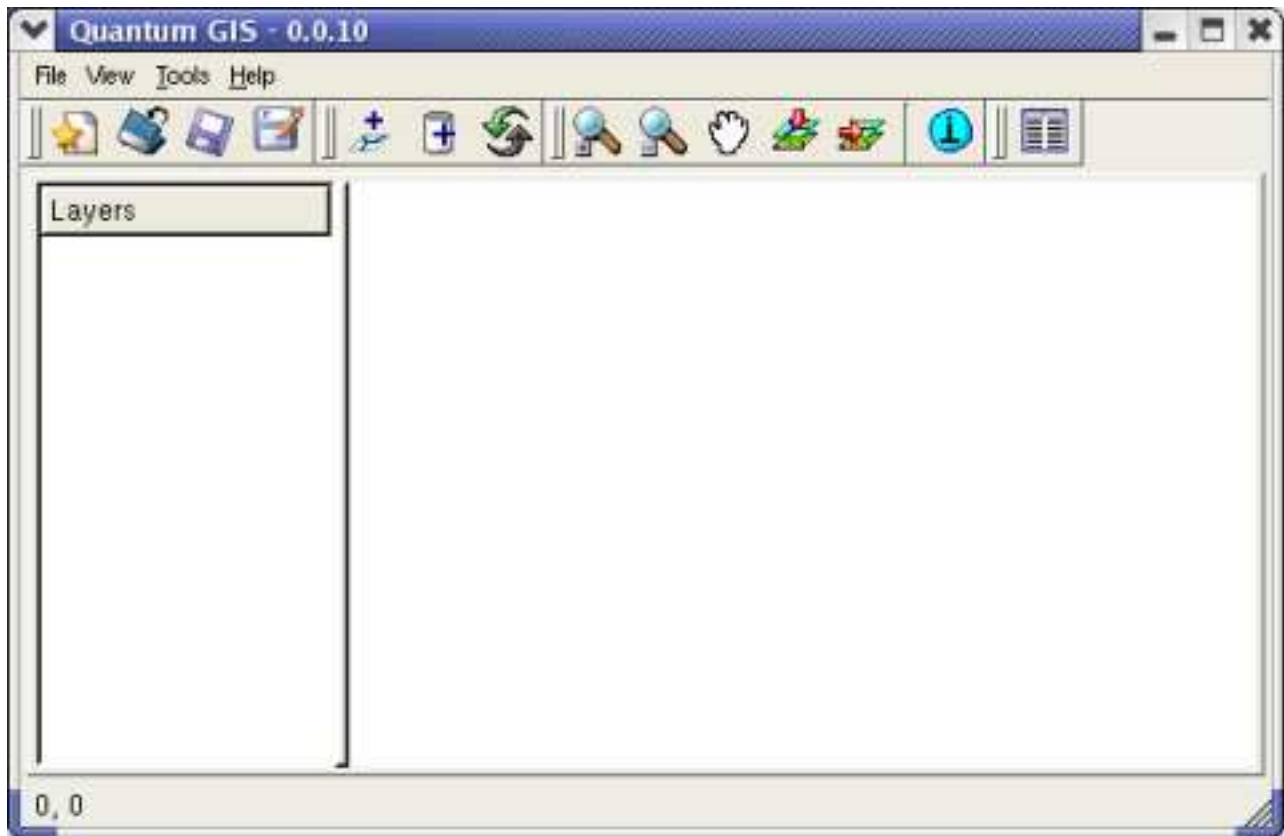
This section describes the functionality of the QGIS “core”. The core application contains minimal functionality. QGIS depends on plugins to implement all non-trivial functions. The components of the core are discussed in the following sections. Details with regard to physical implementation are included where appropriate.

### 7.1 Main Window

The QGIS main window consists of a Qt QMainWindow widget and includes docking areas, menus, toolbar(s), and status bar area. The window is further divided into a legend panel and canvas panel. The general layout of the main window is shown below is shown in Figure 1. The size of the main windows, as well as the location of each toolbar is saved on exit and restored on startup.

Plugins are free to implement additional GUI elements or windows to accomplish their task(s).

Figure 1: Sample QGIS Main Window Layout



## 7.2 Plugins

QGIS will rely heavily on plugins to implement all major functions. With a flexible plugin architecture, QGIS can be easily extended to support additional features and capabilities.

## 7.3 Class Diagram

The class diagram in Figure 2 provides a conceptual model of QGIS with regard to plugins.

## 7.4 Load Sequence

The main application class uses `QLibrary` to load and resolve plugin functions. In order for a plugin to be recognized, it must implement the methods defined in the `QgisPlugin` abstract class and also return a valid version string when queried by the main application. The sequence of events when a plugin is loaded is:

1. QGIS attempts to load the plugin's shared library using `QLibrary::load()`.
2. If load succeeds, the `classFactory` method is resolved using `QLibrary::resolve()`. The `classFactory` method is declared as extern "C" and is responsible for creating an instance of the plugin and returning a pointer to it.

3. If the classFactory method is resolved successfully, QGIS calls the method, passing a pointer to the one and only instance of QgisInterface.
4. Once the plugin instance is created, the plugin can do any initialization required, including adding menus and toolbars to the using the QgisInterface pointer.

Figure 2: Conceptual Diagram of Plugin Architecture

