ABSTRACT

Programming is the way through which we talk to computers and computer replies with an output. This output can be in textual form or graphical form. Humans are attached to the visual content more than he is attached to textual content. This is the reason that desktops have moved from Command Line Interface to Graphical User Interface. There's a lot of programming and brain involved behind getting these graphics on screen. I, with curiosity of knowing the process and hence learning the same. There are many programming languages out there, which can be used for the process. I choose one of the most trusted and used language C++. C++ is a Object Oriented Programming language and working on it enhanced my general knowledge about programming and programming language. As writers tend to read a lot, singers tend to listen to music a lot a good programmer reads code a lot. This basic necessity of every programmer is fulfilled by treasure of open source softwares which provide the source for us to read, understand and learn from the source. Open source softwares are like a luxurious gift for anyone who wants to learn programming.

To understand the real application of computer graphics, I choose Computer Aided Design softwares. In CAD softwares, I choose open source software called LibreCAD. LibreCAD is built in Qt (pronounced as cute) framework, which is a GUI framework built in C++ for cross platform desktop apps development. So I had to learn basics about Qt too. Qt allows developers to write applications once and deploy them across many desktop and embedded operating systems without rewriting the source code. Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded and mobile platforms.

With all the gathered knowledge. The experience which I gathered from making primitives in terminal and all the knowledge of C++ programming that I learnt during the six months training, I used them to add new features to LibreCAD. I added *Flower Pattern*, which is a special pattern made up of various circles. The second feature that I added is very useful in engineering drawing. I added a feature that makes top and side views of *Room.* I added these features in main LibreCAD source by understanding the existing code and adding new code to the main code. I get inspire to add more features in LibreCAD. I added Shapes like *Cardiod, Epicycoid, Hypocloid, Nephroid.* Then I got involved in community of LibreCAD, and gave a suggestion about another feature in LibreCAD which was accepted and decided to work upon. All credit goes to Free and Open Source methodology which allowed me to learn and practice.

By adding new features in LibreCAD, I learned how to read source code of a software. I explore LibreCAD code. Being an Open Source, its Code available. And also it is a free sotware, means that we have right of freedom to edit it, hack it, or make own the same one. It is released under GPLv2. You can download, install and distribute LibreCAD freely, with no fear of copyright infringement. I read its code, and came to know about how a large software is maintained, how the algorithms work behind these shapes/graphics. Working with the graphical shapes is ofcource of mine interest. I got this oppourtunity to contribute in open source software. I also intracted with LibreCAD developers via mailing list, IRC channel, personal mails. I made discussions there.

These are the features that I added to source code of LibreCAD. But working with this software, also inspires me to do *Reverse Engineering*. I explored DXF specifications 2000 of AutoCAD. DXF is a Drawing Exchange Format, also known as Drawing Interchange Format. By opening the dxf file in any text editor, it shows an ascii code. This code is hard to understand, But reading its Specification helps, to solve the problem. There are various group codes and it values behind these file formats. These files have nothing meaningful, except some ascii codes with value, which itself depicts some meaning if one understand its specifications. I created some primitives by doing reverse engineering process.

OBJECTIVES AND SCOPE

The main objective of doing this project was to understand source code and to contribute in such organisation. I created 6 new features in LibreCAD. This helps me to understand its soure code and learn OOPs concept. LibreCAD is me in Qt, which is a cross platform framework of c++.
This also helps me to explore Qt libraries and its inbuilt classes and functions.

Its scope is well defined by its name. It has is scope in field of engineering, designing, drafting, etc CAD drafting has achieved this position by continuous effort made by designers and developers. CAD drafting is a constructive solution i.e it allows an organization to create or alter a business solution as per its own distinctive requirements. High performance of CAD drafting can increase productivity and thus can satisfy new demands by applying its modern technology.CAD is a more creative field one needs to have a good amount of concentration and knowledge to accept the changing business trends and technology.

Some of the outsourcing benefits are:
a) Sensible and flexible drawing
b) Reliable output with minimum turnaround time
c) Helps your design more attractive, simple and sharable
d) Quick and secure output
e) Accurate results.
f) Increases Creativity in mind.
g) Provides Accuracy in measurements.

Chapter - 1
INTRODUCTION

The term computer graphics includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing. Here in our lab at the Program of Computer Graphics, we think of computer graphics as drawing pictures on computers, also called rendering. The pictures can be photographs, drawings, movies, or simulations -- pictures of things which do not yet exist and maybe could never exist. Or they may be pictures from places we cannot see directly, such as medical images from inside your body. Computer graphics is now used in various fields; for industrial, educational, medical and entertainment purposes. The aim of computer graphics is to visualize real objects and imaginary or other abstract items. In order to visualize various things, many technologies are necessary and they are mainly divided into two types in computer graphics: modeling and rendering technologies. This book covers the most advanced technologies for both types. It also includes some visualization techniques and applications for motion blur, virtual agents and historical textiles. This book provides useful insights for researchers in computer graphics.

Computer-aided design (CAD), also known as computer-aided drafting(CAD) or computer - aided design and drafting (CADD), is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design. Computer-aided drafting describes the process of creating a technical drawings with the use of computer software. CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic files for print or machining operations. CAD software uses either vector based graphics to depict the objects of traditional drafting, or may also produce raster graphics showing the overall appearance of designed objects.

Today there are very few aspects of our lives not affected by computers. Practically every cash or monetary transaction that takes place daily involves a computer. In many cases, the same is true of computer graphics. Whether you see them on television, in newspapers, in weather reports or while at the doctor’s surgery, computer images are all around you. “A picture is worth a thousand words” is a well-known saying and highlights the advantages and benefits of the visual presentation of our data. We are able to obtain a comprehensive overall view of our data and also study features and areas of particular interest. A well-chosen graph is able to transform a complex table of numbers into meaningful results. You know that such graphs are used to illustrate papers, reports and thesis, as well as providing the basis for presentation material in the form of slides and overhead transparencies. A range of tools and facilities are available to enable users to visualize their data, and this document provides a brief summary and overview. Computer graphics can be used in many disciplines. Charting, Presentations, Drawing, Painting and Design, Image Processing and Scientific Visualization are some among them. Computer graphics is concerned with all aspects of producing images using a computer. It concerns with the pictorial synthesis of real or imaginary objects from their computer-based models.

CAD often involves more than just shapes. As in the manual drafting of Technical and Engineering Drawings, the output of CAD must convey information, such as material, processes,dimensions and tolerances, according to application-specific conventions. CAD may be used to design curves and figures in two-Dimensional(2D) space; or curves, surfaces, and solids in three dimensional(3D) space.CAD is an important industrial art extensively used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design,prosthetic, and many more. CAD is also widely used to produce Computer animation for special Effects in movies,advertising and technical manuals. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by engineers of the 1960s. Because of its enormous economic importance, CAD has been a major driving force for research in computational geometry, computer graphics (both hardware and software), and discrete differential geometry.The design of geometric model for object shapes, in particular, is occasionally called computer-aided geometric design (CAGD).While the goal of automated CAD systems is to increase efficiency, they are not necessarily the best way to allow newcomers to understand the geometrical principles of Solid Modeling.

I explore LibreCad Source Code. LibreCAD is Free and Open Source CAD Software.LibreCAD is a fully comprehensive 2D CAD application that you can download and install for free. There is a large base of satisfied LibreCAD users worldwide, and it is available in more than 20 languages and for all major operating systems, including Microsoft Windows, Mac OS X and Linux (Debian, Ubuntu, Fedora, Mandriva, Suse ...). Librecad is an application for computer aided design (cad) in two dimensions (2d). with librecad you can create technical drawings such as plans for buildings, interiors, mechanical parts or schematics and diagrams.
The app is great for industrial designers, but anyone who wants to learn how to make 2D CAD drawings will like this program. For a free software, LibreCAD gives you a lot of tools to work with. New users will be able to create basic drawings, while advanced users can make engineering plans with

the software. Layers can be added, ideal for complex drawings. The provided tools are sufficient for producing high precision drawings. You can start drawings from scratch. But it is also easy to put in splines, ellipses, arcs, lines and circles. A single item can have several iterations. For instance, you have 4 modes for a rectangle parameter. The different shapes can be combined easily.
LibreCAD also has a powerful zoom tool that lets you look at models at different distances. This is essential for designers who are going to make life-size copies of a drawing. There are three tabs above the working area. The first tab is for changing color, useful for layer definition. The other tab is for changing size and the third for workspace customization.

LibreCAD also has grids which are extremely useful for those new to CAD. Once you have made the basic object, you can customize it in many ways. Scaling is particularly easy here. Also worth mentioning here is the "Explode text into letters" effect. It is a special feature that will come in handy presentations. LibreCAD allows you to put horizontal or vertical restrictions on completed models. Relative zeros may be locked, useful for ending and starting points. All in all, it is powerful, free CAD application. You can download, install and distribute LibreCAD freely, with no fear of copyright infringement.

1.1 LibreCAD's features

LibreCAD is a feature-packed 2D-CAD application with some really great advantages:

- It's free – no worry about license costs or annual fees.
- No language barriers – it's available in a large number of languages, with more being added continually.
- GPLv2 public license – you can use it, customize it, hack it and copy it with free user support and developer support from our active worldwide community and our experienced developer team.
- LibreCAD is an Open Source community-driven project: development is open to new talent and new ideas, and our software is tested and used daily by a large and devoted user community; you, too, can get involved and influence its future development.
- LibreCAD is an Application for Computer Aided Design (CAD) in two dimension (2D). With LibreCAD you can create technical drawings such as plans for building, interiors, mechanical parts or schematics and diagrams.

1.2 Introducion to Qt

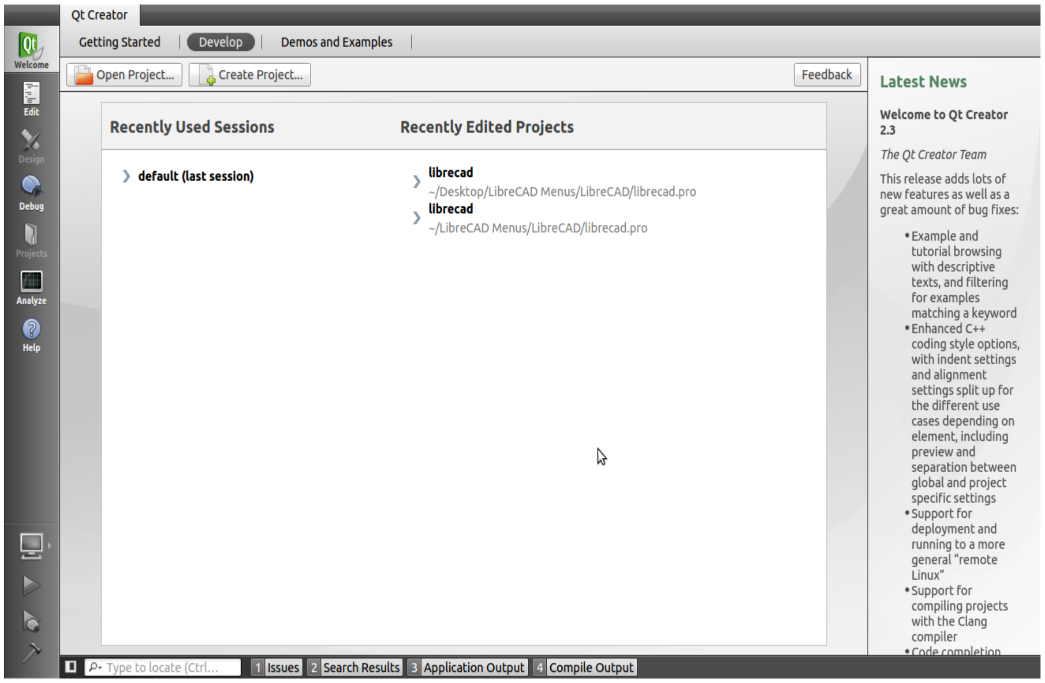
Qt Creator is a complete IDE for creating applications with Qt Quick and the Qt application framework. Qt is designed for developing applications and user interfaces once and deploying them across several desktop and mobile operating systems.

One of the major advantages of Qt Creator is that it allows a team of developers to share a project across different development platforms (Microsoft Windows, Mac OS X, and Linux) with a common tool for development and debugging. In addition, UI designers can join the team by using Qt Quick tools for creating fluid user interfaces in close cooperation with the developers.
The main goal for Qt Creator is meeting the development needs of Qt Quick developers who are looking for simplicity, usability, productivity, extendibility and openness, while aiming to lower the barrier of entry for newcomers to Qt Quick and Qt. The key features of Qt Creator allow UI designers and developers to accomplish the following tasks:

- Get started with Qt Quick application development quickly and easily with examples, tutorials, and project wizards.
- Design application user interface with the integrated editor, Qt Quick Designer, or use graphics software to design the user interface and use scripts to export the design to Qt Quick Designer.
- Develop applications with the advanced code editor that provides new powerful features for completing code snippets, refactoring code, and viewing the element hierarchy of QML files.
- Build and deploy Qt Quick applications that target multiple desktop and mobile platforms, such as Microsoft Windows, Mac OS X, Linux, Symbian, MeeGo, and Maemo.
- Debug JavaScript functions and execute JavaScript expressions in the current context, and inspect QML at runtime to explore the object structure, debug animations, and inspect colors.
- Profile your Qt Quick applications with the QML Profiler. You can inspect binding evaluations, signal handling, and painting operations when running QML code. This is useful for identifying potential bottlenecks, especially in the evaluation of bindings.
- Deploy applications to mobile devices and create application installation packages for Symbian and Maemo devices that can be published in the Ovi Store and other channels.
- Easily access information with the integrated context-sensitive Qt Help system.
- It has differents modes such as Welcome, edit debug, design,analyze and help

1.3 Working with Qt Creator

Qt Creator meets its design goals of simplicity, ease-of-use, and productivity by relying on the concept of modes. These adapt the user interface to the different application development tasks at hand. When developers start Qt Creator, it opens to the Welcome mode, where they can open tutorials and example projects or start the project wizard to create their own projects.



Latest News

Welcome to Qt Creator 2.3

The Qt Creator Team

This release adds lots of new features as well as a great amount of bug fixes:

- Example and tutorial browsing with descriptive texts, and filtering for examples matching a keyword
- Enhanced C++ coding style options, with indent settings and alignment settings split up for the different use cases depending on element, including preview and separation between global and project specific settings
- Support for deployment and running to a more general "remote Linux"
- Support for compiling projects with the Clang compiler
- Code completion

Figure 1.3.1: Welcome screen of Qt creator

Figure 1.3.2: Welcome mode

Each mode has its own view that shows only the information required for performing a given task, and provides only the most relevant features and functions related to it. As a result, the majority of the Qt Creator window area is always dedicated to actual application development tasks.
Users can employ the mode selector to switch to a Qt Creator mode. The following image displays an example application in Edit mode and Design mode.

Figure 1.3.3: Edit mode.

1.4 Creating Projects

To be able to build and run applications, Qt Creator needs the same information as a compiler would need. This information is specified in the project build and run settings. Setting up a new project in Qt Creator is aided by a wizard that guides the user step-by-step through the project creation process. In the first step, the user selects the type of project from the categories. When creating Qt Quick Projects, the user can select either Qt Quick UI or Qt Quick Application.

A Qt Quick UI project contains a single QML file that defines the main view of the application. UI designers can use it to create an application user interface and review it in the QML Viewer, without having to build the application. UI designers do not need to have the development environment installed on their computers to create and run this type of projects.

Developers can build Qt Quick applications and deploy them on mobile target platforms. For example, they can create signed Symbian Installation System (SIS) packages or Debian packages for this type of project. Developers can use ready-made Qt Quick Components for Symbian and MeeGo Harmattan that allow them to create applications with a native look and feel for the selected mobile platform. The components are delivered as part of Qt SDK. A Qt Quick UI project can be easily converted into a Qt Quick application by using the Qt Quick application wizard to import the main QML file in the Qt Quick UI project. The wizard prompts developers to enter the settings needed for a particular type of project.

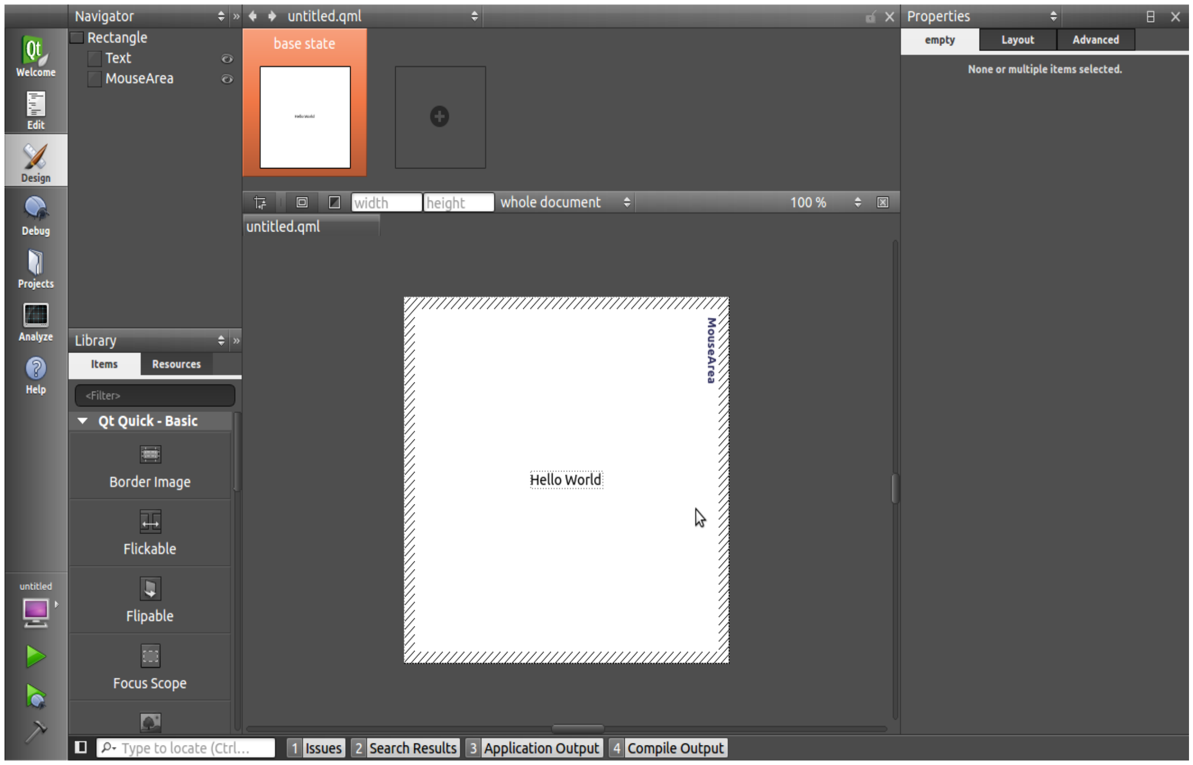


Figure 1.4: New Qt Quick Application project wizard.

When the steps have been completed, Qt Creator automatically generates the project with required headers, source files, user interface descriptions and project files, as defined by the wizard. Not only does the wizard help new users get up and running quickly, it also enables more experienced users to streamline their workflow for the creation of new projects. The convenient user interface makes it easier to ensure that a project begins with the correct configuration and dependencies. Specifically, the Qt Quick application wizard allows developers to create projects that they can deploy on mobile devices with a click of the run button.

1.5 Designing User Interfaces

Before Qt Quick, cooperation between developers and UI designers used to take time, because they used different tool sets. UI designers prefer graphics software that allows them to design visually striking user interfaces, which can be difficult or impossible to implement with the tools available for developers. This often lead to compromise and sub-optimal results. Qt Quick allows both UI designers and developers to use their preferred tools, by making the transfer of deliverables easier. UI designers can work in Photoshop or Gimp and use a QML export script to export their designs to Qt Creator. Developers can then add the necessary code to complete the application. If more changes are needed, UI designers can make them in Qt Quick Designer. Of course, it is also possible to design the user interface from start to finish in the Qt Quick Designer.

The center of the Qt Quick Designer view is used for the construction of the user interface, with the available building blocks (items and resources) kept in the Library on the left side of the window. Reusable elements that you copy to the project folder are automatically added to the Library. The other tools include the Navigator, which displays the QML elements in the current QML file; Properties, which organizes the properties of the selected QML element or QML component; and State, which displays the different states of the component. UI designers and developers can edit the QML files also in the code editor. Qt Quick Designer changes the QML files only in ways that allow users to switch between the Design and Edit modes. The Qt Quick Designer supports a subset of QML features that developers can implement in the code editor. Writing, editing and navigating in source code are core tasks in application development. Therefore, the code editor is one of the key components of Qt Creator. The code editor can be used in the Edit mode to write QML code.

The code editor offers a number of features that help developers maintain readability and coding style:

- Code completion for elements, properties, ids and code snippets. This is also supported for the user's own classes in the current project.
- Support for refactoring code to improve the internal quality of an application, its performance

and extendibility, and code readability and maintainability, as well as to simplify code structure.

- Qt Quick Toolbars for specifying properties of QML elements that are difficult to get right without visual tools.
- Checking code syntax and marking errors (with wavy underlining in red) while editing, to find typos and syntax errors.
- Syntax highlighting for keywords, symbols, and macros in QML files. In addition, generic highlighting is supported for other types of files.
- Incremental search that highlights the matching strings in the window while typing. Advanced search allows developers to search from currently open projects or files on the file system. In addition, developers can search for symbols when developers want to refactor code.

The code editor supports different keyboard shortcuts for faster editing. It is possible to work without using the mouse at all, allowing developers to keep their hands on the keyboard and work more productively.

1.6 Using Qt Quick Toolbars

When users edit QML code in the code editor, they specify the properties of QML components. For some properties, such as colors and font names, this is not a trivial task. For example, few people can visualize the color #18793f.
To easily edit these properties, users can employ the Qt Quick Toolbars. When a component is selected in the code and a toolbar is available, a light bulb icon appears. Users select the icon to open the toolbar.

Qt Quick Toolbar indicator and Qt Quick Toolbar for rectangles. It is used for inserting text, images, and animation. Qt Quick Toolbars are available for editing the properties of the following QML elements:

- Rectangles
- Text
- Images
- Animation

1.7 Building for Multiple Targets

Qt Quick UI projects do not have to be built before they can be run. However, to deploy Qt Quick applications to mobile devices, developers must create installation packages for them. Qt Creator provides support for building, running, and deploying Qt Quick applications for mobile devices (Symbian, Maemo, and MeeGo Harmattan).

Qt Creator allows developers to specify separate build settings for each development platform and to quickly switch between build targets. By default, shadow builds are used to keep the build specific files separate from the source. Developers can create separate versions of project files to keep platform-dependent code separate. They can use qmake scopes to select the file to process depending on which platform qmake is run on.

Developers can build the application for the Qt Simulator to test Qt Quick applications that are intended for mobile devices in an environment similar to that of the device. Developers can change the information that the device has about its configuration and environment. The Qt Simulator is installed as part of the Nokia Qt SDK. After it is installed, developers can select it as a build target in Qt Creator.

Debugging

Developers can use the Qt Creator Debug mode to inspect the state of the application while debugging JavaScript functions. They can set breakpoints, view call stack trace, and examine locals and watchers. When the application is interrupted by a breakpoint, developers can use the QML Script Console to execute JavaScript expressions in the current context. They can type JavaScript expressions and use them to get information about the state of the application, such as property values. If developers change property values or add properties in the code editor, the changes are updated in the running application when they are saved.

While the application is running, developers can use the QML Inspector view to explore the object structure, debug animations, and inspect colors. When debugging complex applications, developers can jump to the position in code where an element is defined.

Object structure displayed in the QML Inspector.

Analyzing Code

The memory available on mobile devices is limited and you should use it carefully. Qt Creator contains tools that you can use to analyze your code.

The QML Profiler allows you to profile your Qt Quick applications. You can inspect binding evaluations, signal handling, and painting operations when running QML code. This is useful for identifying potential bottlenecks, especially in the evaluation of bindings.

1.8 Deploying Applications to Mobile Devices

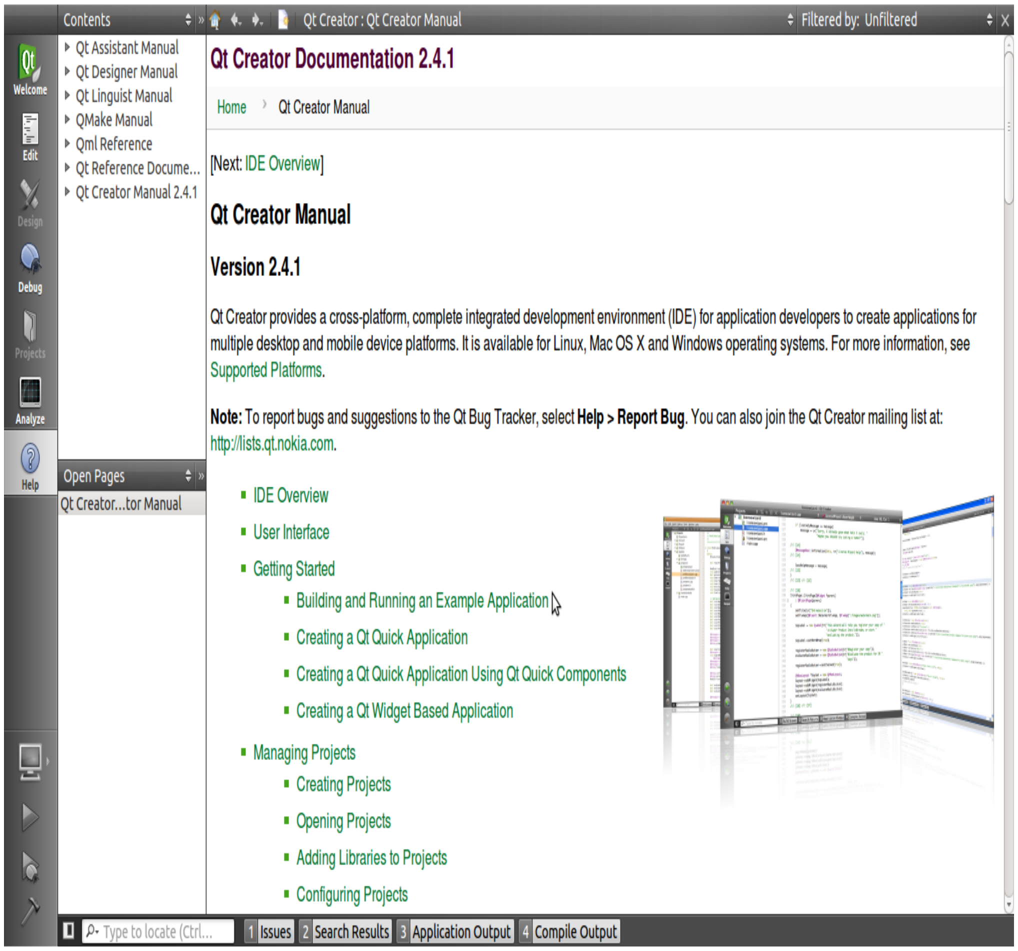
Qt Creator deploy configurations handle the packaging of the application as an executable and copying it to a location developers want to run the executable at. The files can be copied to a location in the file system of the development PC or a mobile device. To deploy files on mobile devices, developers must either connect the devices to the development PC or use the installation packages generated by Qt Creator. Qt Quick UI projects must be converted into Qt Quick applications for deployment on mobile devices.

Qt Creator allows developers to create installation packages for Symbian, MeeGo, and Maemo devices that are suitable for publishing on Ovi Store.

1.9 Getting Help

From time to time, developers may need further information about a certain QML element, Qt class, function, or other part of the Qt API. All the Qt documentation and examples are accessible via the Qt Help plugin in Qt Creator.

To view the documentation, the Help mode is used, where most of the window is devoted to the help text. While working with source code in Edit mode, the user can access context sensitive help by moving the text cursor to a Qt class or function and then press the F1 key. The documentation is displayed within a panel on the right side of the code editor, as shown in the following figure.



Qt Creator Documentation 2.4.1

Home › Qt Creator Manual

[Next: IDE Overview]

Qt Creator Manual

Version 2.4.1

Qt Creator provides a cross-platform, complete integrated development environment (IDE) for application developers to create applications for multiple desktop and mobile device platforms. It is available for Linux, Mac OS X and Windows operating systems. For more information, see Supported Platforms.

Note: To report bugs and suggestions to the Qt Bug Tracker, select **Help > Report Bug**. You can also join the Qt Creator mailing list at: http://lists.qt.nokia.com.

- IDE Overview
- User Interface
- Getting Started
  - Building and Running an Example Application
  - Creating a Qt Quick Application
  - Creating a Qt Quick Application Using Qt Quick Components
  - Creating a Qt Widget Based Application
- Managing Projects
  - Creating Projects
  - Opening Projects
  - Adding Libraries to Projects
  - Configuring Projects

Figure 1.9: Displaying context sensitive Qt Help information.

Qt Designer is a powerful cross-platform GUI layout and forms builder. It allows you to rapidly design and build widgets and dialogs using on-screen forms using the same widgets that will be used in your application. Forms created with Qt Designer are fully-functional, and they can be previewed so that you can ensure that they will look and feel exactly as you intended

Features & Benefits

- Design user interfaces quickly with drag and drop functionality
- Customize widgets or choose from library of standard widgets
- Instantly preview forms in native look and feel
- Generate C++ or Java code from your interface prototypes
- Use Qt Designer with Visual Studio or Eclipse IDEs
- Build fully-functional user interfaces with Qt's signals and slots

1.10 Introduction to CAD

Computer-aided design (CAD),also known as computer-aided drafting(CAD) or computer-aided design and drafting (CADD), is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design. Computer-aided drafting describes the process of creating a technical drawings with the use of computer software. CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic files for print or machining operations. CAD software uses either vector based graphics to depict the objects of traditional drafting, or may also produce raster graphics showing the overall appearance of designed objects.

CAD often involves more than just shapes. As in the manual drafting of Technical and Engineering Drawings, the output of CAD must convey information, such as material, processes,dimensions and tolerances, according to application-specific conventions. CAD may be used to design curves and figures in two-Dimensional(2D) space; or curves, surfaces, and solids in three dimensional(3D) space.CAD is an important industrial art extensively used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design,prosthetic, and many more. CAD is also widely used to produce Computer animation for special Effects in movies,advertising and technical manuals. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by engineers of the 1960s. Because of its enormous economic importance, CAD has been a major driving force for research in computational geometry, computer graphics (both hardware and software), and discrete differential geometry.The design of geometric model for object shapes, in particular, is occasionally called computer-aided geometric design (CAGD).While the goal of automated CAD systems is to increase efficiency, they are not necessarily the best way to allow newcomers to understand the geometrical principles of Solid Modeling.

1.10.1 Introduction to LibreCAD

LibreCAD is a fully comprehensive 2D CAD application that you can download and install for free. There is a large base of satisfied LibreCAD users worldwide, and it is available in more than 20 languages and for all major operating systems, including Microsoft Windows, Mac OS X and Linux (Debian, Ubuntu, Fedora, Mandriva, Suse ...). Librecad is an application for computer aided design (cad) in two dimensions (2d). with librecad you can create technical drawings such as plans for buildings, interiors, mechanical parts or schematics and diagrams.
The app is great for industrial designers, but anyone who wants to learn how to make 2D CAD drawings will like this program.
For a free software, LibreCAD gives you a lot of tools to work with. New users will be able to create basic drawings, while advanced users can make engineering plans with the software. Layers can be added, ideal for complex drawings. The provided tools are sufficient for producing high precision drawings. You can start drawings from scratch. But it is also easy to put in splines, ellipses, arcs, lines and circles. A single item can have several iterations. For instance, you have 4 modes for a rectangle parameter.
The different shapes can be combined easily. LibreCAD also has a powerful zoom tool that lets you look at models at different distances. This is essential for designers who are going to make life-size copies of a drawing. There are three tabs above the working area. The first tab is for changing color, useful for layer definition. The other tab is for changing size and the third for workspace customization.
LibreCAD also has grids which are extremely useful for those new to CAD. Once you have made the basic object, you can customize it in many ways. Scaling is particularly easy here. Also worth mentioning here is the "Explode text into letters" effect. It is a special feature that will come in handy presentations.
LibreCAD allows you to put horizontal or vertical restrictions on completed models. Relative zeros may be locked, useful for ending and starting points. All in all, it is powerful, free CAD application.
You can download, install and distribute LibreCAD freely, with no fear of copyright infringement.

LibreCAD's features:

LibreCAD is a feature-packed and mature 2D-CAD application with some really great advantages:

- It's free – no worry about license costs or annual fees.
- No language barriers – it's available in a large number of languages, with more being added continually.
- GPLv2 public license – you can use it, customize it, hack it and copy it with free user support

and developer support from our active worldwide community and our experienced developer team.

- LibreCAD is an Open Source community-driven project: development is open to new talent and new ideas, and our software is tested and used daily by a large and devoted user community; you, too, can get involved and influence its future development.
- LibreCAD is an Application for Computer Aided Design (CAD) in two dimension (2D). With LibreCAD you can create technical drawings such as plans for building, interiors, mechanical parts or schematics and diagrams.

How it started?

LibreCAD started as a project to build CAM capabilities into the community version of QCad for use with a Mechmate CNC router. LibreCAD is a version of QCad CE ported to Qt4. Since QCad CE was built around the outdated Qt3 library, it had to be ported to Qt4 before additional enhancements. This gave rise to CADuntu.

The project was known as CADuntu only for a couple of months before the community decided that the name was inappropriate. After some discussion within the community and research on existing names, CADuntu was renamed to LibreCAD.

Porting the rendering engine to Qt4 proved to be a large task, so LibreCAD initially still depended on the Qt3 support library. The Qt4 porting was completed eventually during the development of 2.0.0 series, thanks to our master developer Rallaz, and LibreCAD has become Qt3 free except in the 1.0.0 series. LibreCAD is available in the "Ubuntu Software Center" as "librecad" on Ubuntu 11.04 (Natty) and later. It will be automatically installed and configured for your system!

1.10.2 CAD concepts in LibreCAD:

Entities - Entities are graphical objects in a CAD system. Typical entities which are supported by most CAD systems are: points, lines and circular and elliptical arcs. More complex and CAD specific entities include polylines, texts, dimensioning, hatches and splines.

Attributes: Every entity has its attributes - such as its color, line type (continuous,dashed etc..) and line width.

Layers - A basic concept in computer aided drafting is the use of layers to organize a drawing. Every entity in a drawing is on exactly one layer and a layer can contain lots of entities. Typically entities with a common 'function' or common attributes are put on the same layer. E.g. you might want to put all axes in a drawing on a layer named 'axes'(see Figure 1). Layers can have their own attributes also (colour, line width, line style etc...). Each entity can have its own attributes or have its attributes defined by the layer it is placed on. In the latter case for example you can change the colour of all the entities on the "axes"

layer by setting the colour (red for example) for the layer 'axes'.
In traditional manual drafting, a similar approach was used. Whether for Engineering, Architectural or Construction drawing etc...Layers were used to show different aspects of a drawing -for example this could be a layer set up for showing centre lines on an engineering drawing or to show different building systems, such as wiring and air conditioning. The layers were often drawn on separate transparent sheets of paper. These sheets were then overlaid one on top of another to produce final drawings.

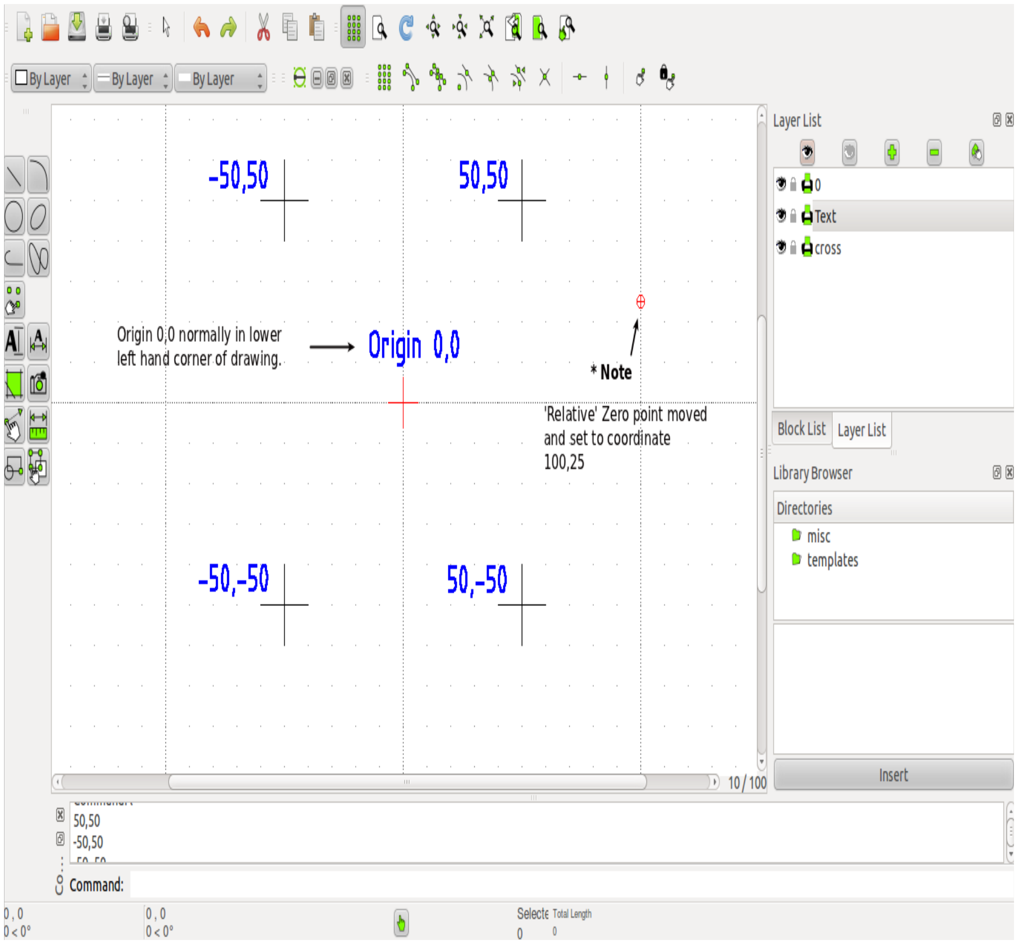
Blocks - A Block is a group of entities. Blocks can be inserted into the same drawing more than once with different Attributes and at different locations and at a different scale and rotation angle (see image below) . Such an instance of a block is usually called an Insert. Inserts have attributes just like entities and layers. An Entity that is part of an Insert can have its own attributes or share the attributes of the Insert. Once created, Inserts are still linked to the Block they instantiate. The power of inserts is, that you can modify the Block once and all Inserts will be updated accordingly.

Drafting - In many ways, CAD is similar to traditional manual drafting. For example when drawing a plan or a view of an object on paper, you would use tools such as a pencil and a ruler to draw lines. In CAD systems there is a variety of tools available to achieve the same goal. The big advantage 0,0) i of using a CAD system is the fact that you can change every entity of your drawing easily after you've created it, very often in seconds! This is probably the most difficult thing to get used to and learn when making the move from "manual" drafting on paper to using a CAD system.
When working with a CAD system you will very often create temporary construction lines, reference points etc... to aid your workflow, that do not necessarily need to be on the final printout. Some lines may not have the correct length and will need to be trimmed later - and so on.
A common mistake for CAD beginners is wanting to create the final drawing straight away. The preliminary auxiliary construction lines, reference points etc...Are an essential part of the process of working with a CAD system and are in fact encouraged! You will find in LibreCAD and most all CAD programs these modification tool sets to "clean" up and modify your drawing before the final printout.

The coordinate system - The coordinate system used in most CAD applications sets the main reference point (called the "origin" or "zero"; at coordinate n the lower-left corner of your worksheet. In LibreCAD this is represented by a red + mark, visible in the lower left corner of a new drawing.
You can set the unit of measure for the coordinate system points;
For the Current drawing:
From main menu -> Edit -> Current Drawing -> Units.
For the 'Default 'unit of measure (affecting all new drawings):
From main menu -> Edit -> Application Preferences -> Defaults -> Units.
Selecting the unit of measure tells LibreCAD how much "1 unit" is equal to (without defining this, +1 unit

to a line could mean "add 1 millimeter" or as easily as "adding 1 mile").

In order to get the best out of LibreCAD it is wise to have a good understanding of the coordinate system and how coordinates work. Everything that you draw in LibreCAD will be exact and precise and will be placed there accurately based on the X,Y coordinate system.



```
50,50
-50,50
```

Figure 1.10.2(a): The co-ordinate system in LibreCAD

The **absolute origin** or **Zero** point in your drawing is where the X and Y axes cross each other (represented by a Red cross), every entity you draw is located in relation to this origin.
In LibreCAD there is also the option to set the **Relative Zero Point** (small red circle).This Relative zero point can be temporarily set to a new location in a drawing so that all subsequent X and Y coordinates of entities drawn or blocks placed for example will be relative to this newly set **Relative Zero Point**.

In libreCAD`s 2D coordinate system all **X** units are measured horizontally and all **Y** units are measured vertically. Coordinates can also be shown as 'Positive' (+) or 'Negative'(-) values.

Basically there are two types of Coordinates Cartesian and Polar.

The Cartesian coordinate system is generally the standard system used in most CAD programs. A specific point in a drawing is located by exact distances from both the X and Y axes - for example a point in a drawing could be 60,45 (note the comma -, separates the two numbers.

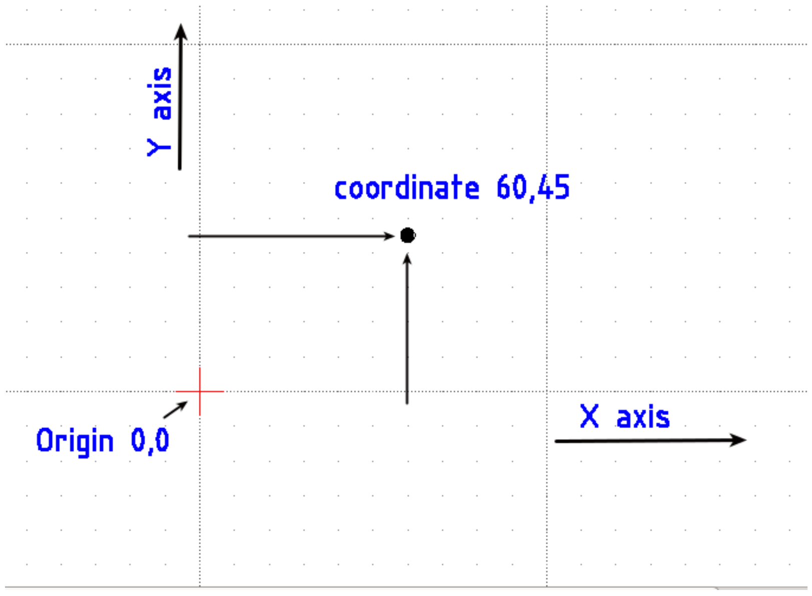


Figure1.10.2(b): Rectangular co-ordinate.

The Polar coordinate system uses one distance and one angle to define a point in a drawing -for example a point in a drawing could be 50 < 45, so 50 units long and at an angle of 45 degrees (note the < sign is used for the angle). see example image below.

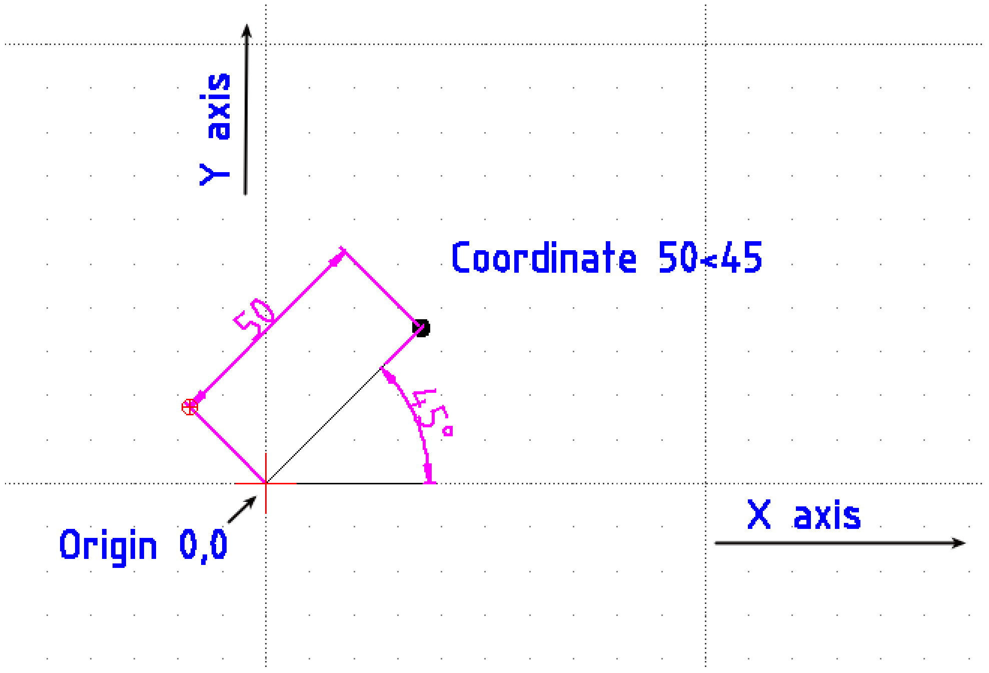


Figure 1.10.2(c): Polar coordinates

In LibrecAD lines,points, Arcs, Polylines, Circles and many more entities can be drawn and placed in a drawing using either Absolute or Relative coordinate input.
To input coordinate value points in LibreCAD you can 'type' your values in the command line or inside a 'text input box' (presented by tool options requiring distance,angle etc...).This method is 100% accurate.
Or
You can 'manually', move the mouse cursor around and visually pick a coordinate point, but obviously this method is less accurate but may be acceptable for some 'rough' sketch or freehand work!

Absolute coordinates - using this method,coordinate points are entered in direct relation to the Origin 0,0. To do this in LibreCAD just enter in the exact point e.g. 60,45.
Relative coordinates - using this method, coordinate points are entered in relation to the previous point entered (not the origin), so for example - if your first point is 20,45, to then enter your next point 'relative' to this - you would use the '@' symbol - e.g @50,50 would then enter the second point 50 units horizontally along the x axis and 50 units vertically along the Y axis to give this second point relative to your last point (20,45).

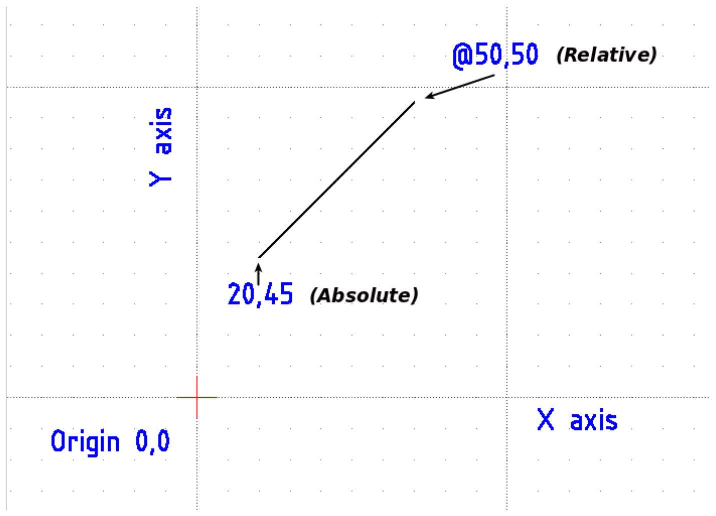


Figure 1.10.2(d): Absolute Zero and Relative Zero in Rectangular coordinate

Relative Polar coordinates - this is a very useful way of drawing entities of which you know the exact length and angle.

For example you could draw a 100mm long line from start point 50,50 (absolute coordinate) and specify your second point at 100<45 (relative 'polar' coordinate).

You can see from this example that the second point is based on our 'distance' of 100mm and at an angle of 45 degrees.

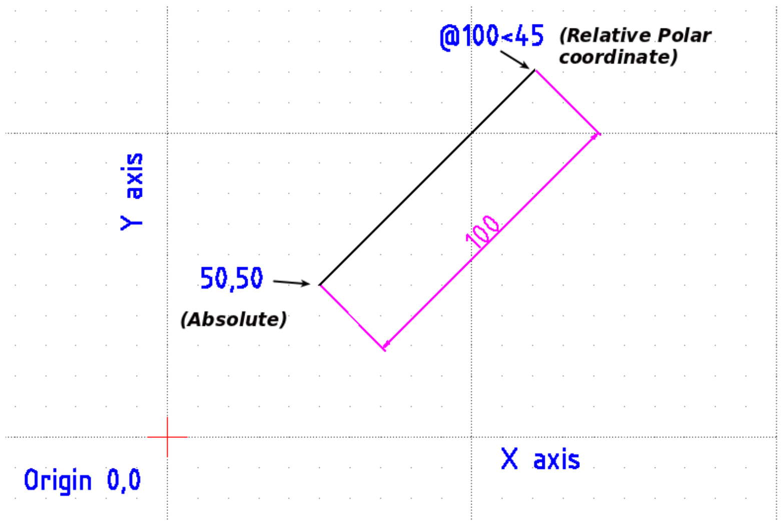


Figure 1.10.2(e): Absolute and Relative Point in Polar coordinate

Angles in LibreCAD- *It is worth mentioning here a brief explanation of how angles work in LibreCAD.All angles in LibreCAD are measured in 360 degrees in an anti-clockwise direction* *(see image below) beginning from 0 degrees (the 3 o'clock position). The < symbol is used before the angle - e.g.50<45.*

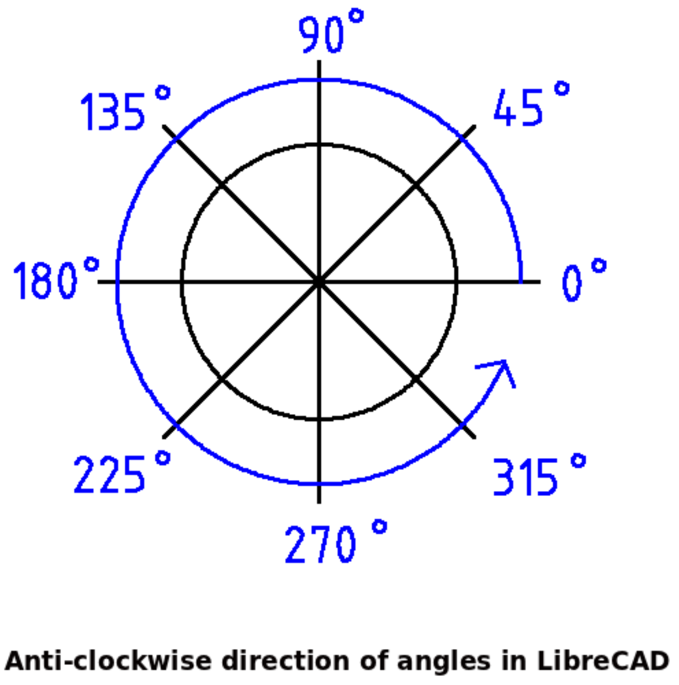


Figure 1.10.2(f): Anti-clockwise direcion of angles in LibreCAD

Object and Grid Snapping - When you want to specify and select a coordinate of an object/entity in LibreCAD, you can use the 'snap' tools which allow you to precisely 'snap to' grid points or significant points on existing objects: endpoints or midpoints of lines, center points, middle points and intersections etc...

Entity Selection - An entity must be selected before it can be deleted, duplicated, or transformed. Entity selection is one of the most basic of CAD operations. There are a wide variety of selection tools available to use in LibreCAD to quickly select groups of entities, entities within a range, connected entities, etc. (See image below).The selection tools are available from either the main menu - 'Select' or from the show toolbar 'Select' tools (no. 15 above).
Alternatively you can use the keycodes:
Ctrl-A - Select all
tn / Ctrl-K - Deselect all

Deletion - Deleting an entity means to totally remove it from the drawing. In LibreCAD The deletion options 'delete' and 'delete selected' are available via the main menu > modify. The 'delete' option is also available from the 'modify' toolbar and by entering the keycode 'er' in the command line.

Modification - Basic modifications (changes) of an entity in a CAD system include translation, rotation, reflection, and scaling. These operations do not alter the characteristic geometry of the selected entity - however there are some CAD modifications that do - such as break, trim, extend or stretching of selected entities.
The modification tools are available from either the main menu > Modify or from the modify toolbar or by entering keycodes in the command line

Dimension - Required sizes of objects within a drawing are conveyed through the use of dimensions. Dimension 'distances' may be shown with either of two standardized forms of dimension - Linear and ordinate. The linear dimensions use two parallel lines - called "extension lines," which are spaced at the 'required' distance between two given points.A line perpendicular to these extension lines is called a "dimension line," with arrows at its endpoints. The numerical indication of the distance is placed at the midpoint of the dimension line, adjacent to it or in a gap provided for it!

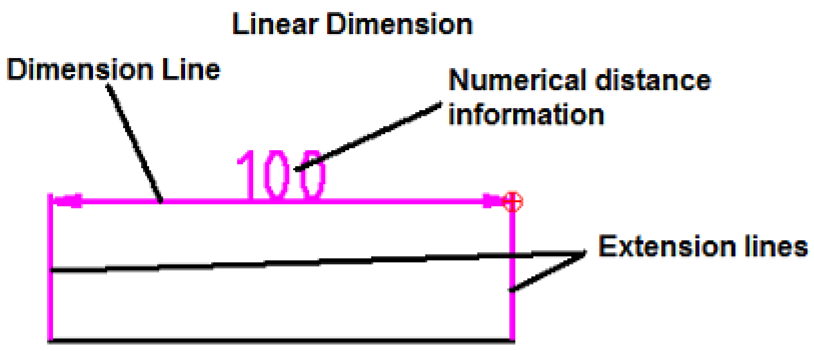


Figure 1.10.2(g): Linear Dimension

Ordinate dimensions use one horizontal and one vertical extension line to establish an origin for the entire view. The origin is identified with **0,0** placed at the ends of these extension lines. Distances along the **x** and **y** axes to other points on a drawing are indicated using additional extension lines with numerical information placed appropriately will help to make an accurate drawing.

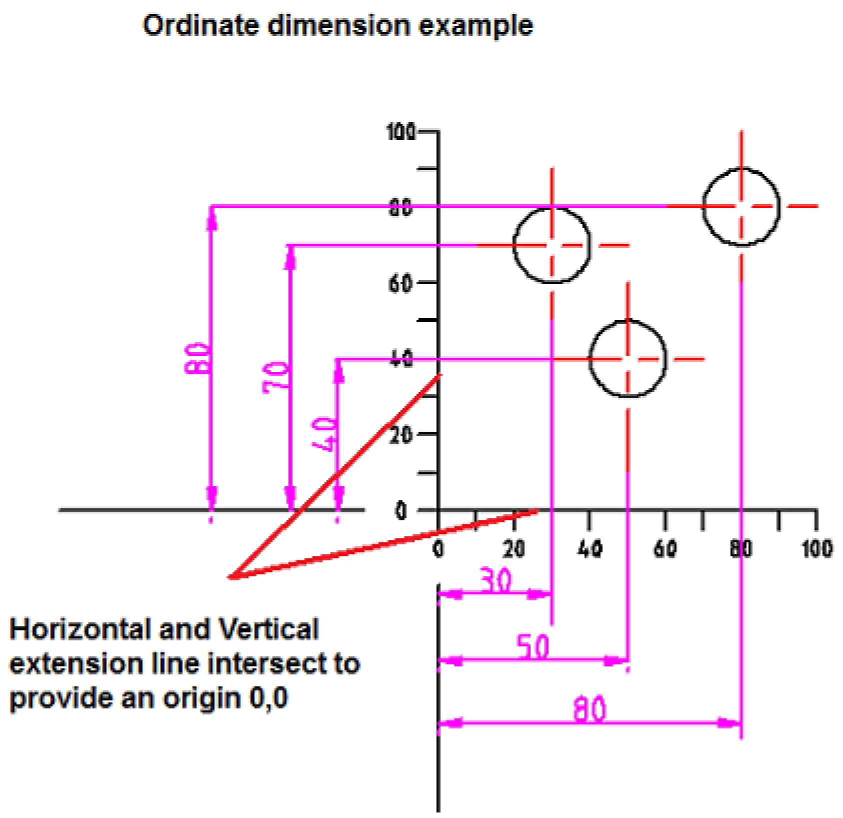

Ordinate dimension example
100
80
60
40
20
0
0
20
40
60
80
100
80
70
40
30
50
80
Horizontal and Vertical extension line intersect to provide an origin 0,0

Figure 1.10.2(h): Ordinate Dimension

Sizes of circular features are indicated using either diametral or radial dimensions. Radial dimensions use an "R" followed by the value for the radius; Diametral dimensions use a circle with forward-leaning diagonal line through it, called the diameter symbol, followed by the value for the diameter. A radially-aligned line with arrowhead pointing to the circular feature, called a leader, is used in conjunction with both diametral and radial dimensions. All types of dimensions are typically composed of two parts: the nominal value, which is the "ideal" size of the feature, and the tolerance, which specifies the amount that the value may vary above and below the nominal. Architectural Dimensions.

Scaling and Viewing - Unlike in manual drafting, there is no need when using a CAD program to first determine and work-out in advance the sheet size and drawing scale. When drafting with CAD, all sizes and distances are specified using their full-scale values,for instance drawing a 10 meter object in CAD is drawn as a 'real size' 10 meter object.It is only when it comes to printing that the drawing scale needs to be determined and adjusted based on sheet and drawing size.
On the screen (in model space) the user can adjust the current visible area of the drawing by using the 'Zoom' tools, for example: zooming in to view more detail or zooming out to view a wider extent!
Another important viewing operation in CAD is the use of 'panning' - panning allows us to to see another portion of the drawing without changing the display scale, a user pans to it by "moving" a rectangular

display window until it's over the desired spot. Zoom Panning, click and drag to pan zoom - a small hand icon will show when using this function.

First-angle projection

In **first-angle projection**, the object is conceptually located in quadrant **I**, i.e. it **floats above and before** the viewing planes, the planes are **opaque**, and each view is **pushed** through the object onto the plane furthest from it. (Mnemonic: an "actor on a stage".) Extending to the 6-sided box, each view of the object is projected in the direction (sense) of sight of the object, onto the (opaque) interior walls of the box; that is, each view of the object is drawn on the opposite side of the box. A two-dimensional representation of the object is then created by "unfolding" the box, to view all of the **interior** walls. This produces two plans and four elevations. A simpler way to visualize this is to place the object on top of an upside-down bowl. Sliding the object down the right edge of the bowl reveals the right side view.

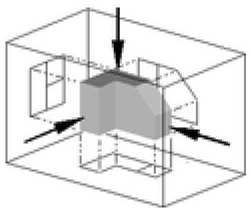


Figure 1.10.2(i): First angle of Projection(i)

● Image of object in box, with views of object projected in the direction of sight onto walls using first-angle projection.

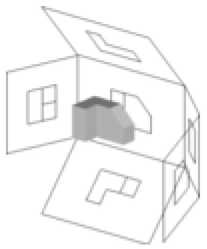


Figure 1.10.2(j): First angle of Projection(ii)

● Similar image showing the box unfolding from around the object.

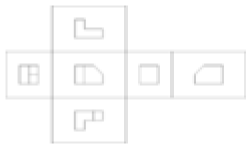


Figure 1.10.2 (k): First angle of Projection(iii)

● Image showing orthographic views located relative to each other in accordance with first-angle projection.

The Isometric Drawing should have the following:

1. Create a three dimensional figure out of cubes. The figure should be somewhat complicated, such as the one students just saw.
2. Use the View button to see a 3D view.
3. Uncheck the 3D "box" to see a Front-Right-Top view.
4. Check the 3D box to see the three-dimensional view once again.

As students are moving between views, they should observe how the views change and how they relate to the original three dimensional figure they created.
Going back to the original figure and view they created, students should delete one cube from their picture. Ask them to note what they observe about the Front-Top-Right view. Students should continue deleting cubes. Ask them to note the fewest number of cubes they can delete to make the front view look different, the right view look different, and the top view look different. If needed, students can try again using another figure they have created.

How it started?

LibreCAD started as a project to build CAM capabilities into the community version of QCad for use with a Mechmate CNC router. LibreCAD is a version of QCad CE ported to Qt4. Since QCad CE was built around the outdated Qt3 library, it had to be ported to Qt4 before additional enhancements. This gave rise to CADuntu.
The project was known as CADuntu only for a couple of months before the community decided that the name was inappropriate. After some discussion within the community and research on existing names, CADuntu was renamed to LibreCAD.
Porting the rendering engine to Qt4 proved to be a large task, so LibreCAD initially still depended on the Qt3 support library. The Qt4 porting was completed eventually during the development of 2.0.0 series, thanks to our master developer Rallaz, and LibreCAD has become Qt3 free except in the 1.0.0 series.

LibreCAD is available in the "Ubuntu Software Center" as "librecad" on Ubuntu 11.04 (Natty) and later. It will be automatically installed and configured for your system!

1.10.3 Downloading Source Code

Fired up the terminal because you need to install the qt4 development libraries, tools, compiler and git.

```
$ sudo apt-get install g++ gcc make git-core libqt4-dev
  qt4-qmake libqt4-help qt4-dev-tools libboost-all-dev
  libmuparser-dev libfreetype6-dev
$ sudo apt-get build-dep librecad
```

Now you have a development environment on your computer. You need to download the source of LibreCAD. The source code including the latest development of LibreCAD is available from the project page at github, https://github.com/LibreCAD/LibreCAD

Steps: create a folder librecad to make clone.You can use any name in place of librecad.

```
$mkdir librecad
$cd librecad
 $ git clone https://github.com/LibreCAD/LibreCAD.git
```

Compilation

Now you can run qmake (or qmake-qt4) to create a makefile and run make to compile LibreCAD.

```
$cd LibreCAD
$qmake librecad.pro
$make
```

librecad.pro is a project file. qmake creates a makefile. Make command will compile the project. Compiling LibreCAD might take awhile, depending on the speed of your computer, but just let it run until it finishes.

To finally run LibreCAD, execute the following commands:

```
$cd unix
$./librecad
```

LibreCAD started as a project to build CAM capabilities into the community version of QCad for use with a Mechmate CNC router. Since QCad CE was built around the outdated Qt3 library, it had to be ported to Qt4 before additional enhancements. This gave rise to CADuntu.

The project was known as CADuntu only for a couple of months before the community decided that the name was inappropriate. After some discussion within the community and research on existing names, CADuntu was renamed to LibreCAD.

Porting the rendering engine to Qt4 proved to be a large task, so LibreCAD initially still depended on the Qt3 support library. The Qt4 porting was completed eventually during the development of 2.0.0 series, thanks to our master developer Rallaz, and LibreCAD has become Qt3 free except in the 1.0.0 series.

LibreCAD is a fully comprehensive 2D CAD application that you can download and install for free. There is a large base of satisfied LibreCAD users worldwide, and it is available in more than 20 languages and for all major operating systems, including Microsoft Windows, Mac OS X and Linux (Debian, Ubuntu, Fedora, Mandriva, Suse ...).

You can download, install and distribute LibreCAD freely, with no fear of copyright infringement.

LibreCAD's features:
LibreCAD is a feature-packed and mature 2D-CAD application with some really great advantages:

It's free – no worry about license costs or annual fees.
No language barriers – it's available in a large number of languages, with more being added continually.
GPLv2 public license – you can use it, customize it, hack it and copy it with free user support and developer support from our active worldwide community and our experienced developer team.
LibreCAD is an Open Source community-driven project: development is open to new talent and new ideas, and our software is tested and used daily by a large and devoted user community; you, too, can get involved and influence its future development.

Chapter -2
ADDING NEW FEATURES IN LIBRECAD

As a major project, I choose to work upon open source, to learn from it and in return contribute to it. Depending upon my interests I chose LibreCAD which is written in C++ using Qt framework. It is a Qt application (Qt is a C++ library). With this project, I aimed at understanding C++ more clearly and deeply.
I added these two feature in LibreCAD
1) Flower Pattern
2) Room
3) Cardiod
4) Epicycloid
5) Hypocycloid
6) Nephroid

2.1.1 <u>Flower Pattren:</u>

In this, we need to draw inner circle only, the remaining 6 circles surrounded with inner one made itself on triggering the inner one.
I used this logic to make flower pattren.

```
for(int i=0; i<=5; i++)
 {
     double radian = i*60* (M_PI/180.0);
     dataNew.center.x = data.center.x + data.radius * cos(radian);
     dataNew.center.y = data.center.y + data.radius * sin(radian);

     RS_Circle* circleNew = new RS_Circle(container,
                              dataNew);
     container->addEntity(circleNew);

    dataNew.center.x=0;
    dataNew.center.y=0;
 }
```

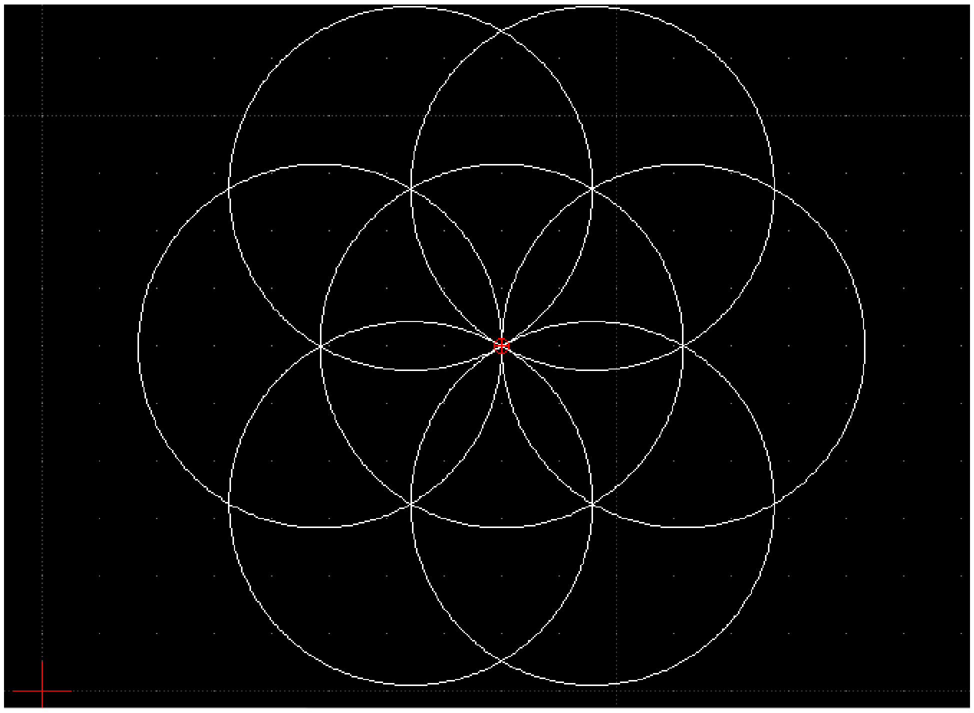


Figure 2.1.1: Flower Pattren

2.1.2 Room:

It shows TOP , SIDE, and FRONT VIEW of Room.
It is formed by using lines with different coordinate points. Qt provides us a function to draw line which is under class QPainter.

Function of Qt class to draw a line with point x1,y1 and x2,y2 are:

```
void drawLine(int x1, int y1, int x2, int y2)
```

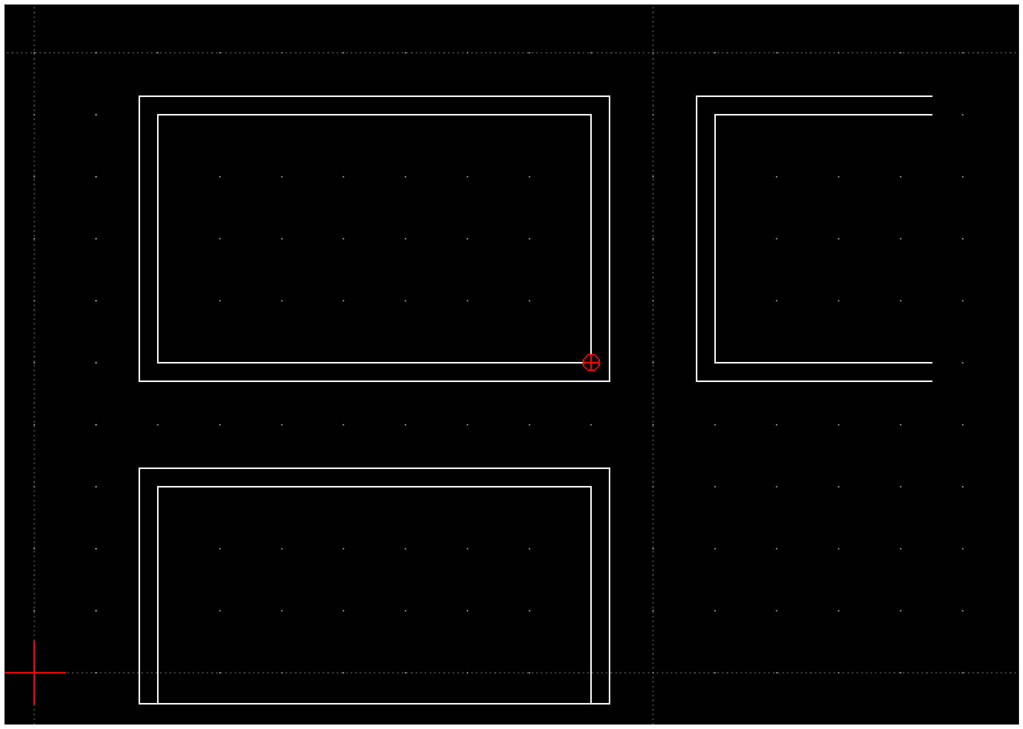


Figure 2.1.2: Top, Right, Front View Of Room

2.1.3 <u>Cardiod:</u>

A cardioid (from the Greek καρδία "heart") is a plane curve traced by a point on the perimeter of a circle that is rolling around a fixed circle of the same radius. It is therefore a type of limaçon and can also be defined as an epicycloid having a single cusp. It is also a type of sinusoidal spiral, and an inverse curve of the parabola with the focus as the center of inversion.

Logic for Cardiod:

```
    x = radius * (2.0*cos(angle) - cos(2.0*angle))+.5;
    y = radius * (2.0*sin(angle) - sin(2.0*angle))+.5;
```

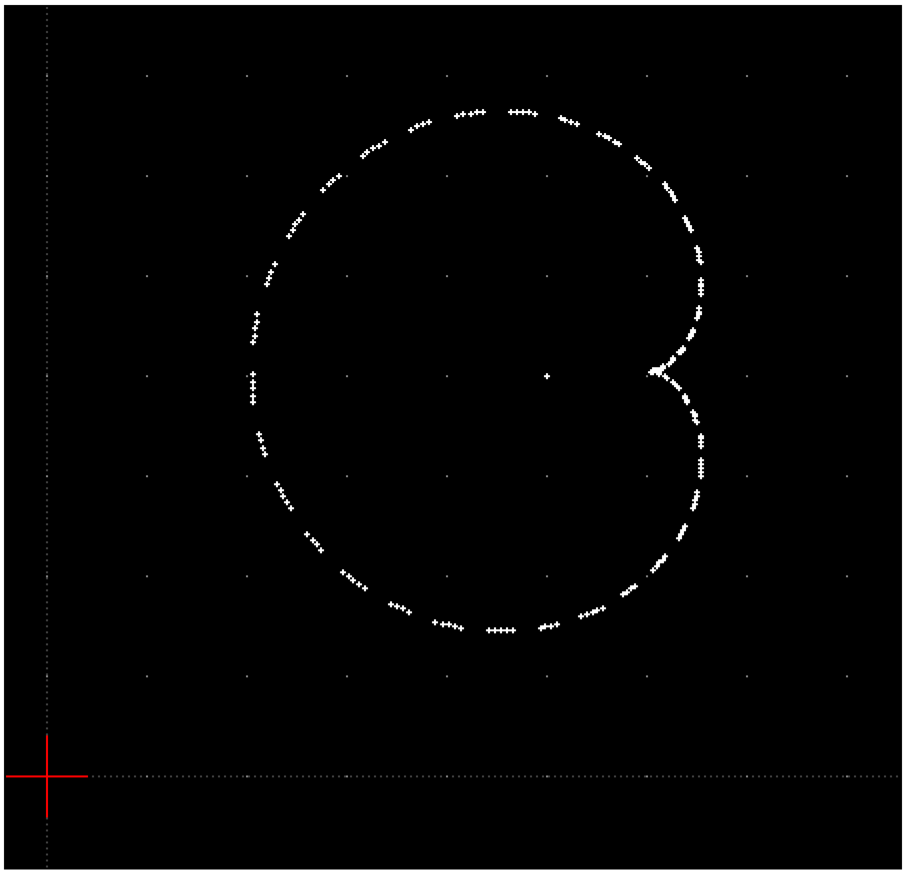


Figure 2.1.3: Cardiod

2.1.4 <u>Epicycloid:</u>

In geometry, an epicycloid is a plane curve produced by tracing the path of a chosen point of a circle — called an epicycle — which rolls without slipping around a fixed circle.

Logic used for Epicycloid:

```
    float a=pt.x;
    float b=pt.y;
    float radius2=5,radius1=4*radius2;
    for (float angle = 0;  angle <= 200; angle += 1)
    {
    pt.x = pt.x +  (radius1 + radius2)*cos(angle) - radius2*cos((radius1/radius2 + 1.0)*angle) +.5;
    pt.y = pt.y +  (radius1 + radius2)*sin(angle) - radius2*sin((radius1/radius2 + 1.0)*angle)+.5;

    RS_Point* point1 = new RS_Point(container, RS_PointData(pt));
    container->addEntity(point1);
          pt.x =a;
           pt.y=b;
}
```

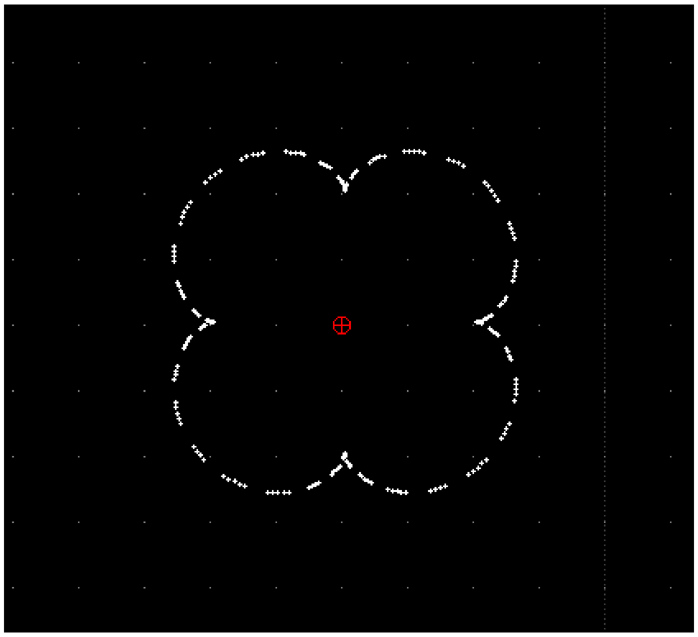


Figure 2.1.4: Epicycloid

2.1.5 <u>Hypocycloid:</u>

In geometry, a hypocycloid is a special plane curve generated by the trace of a fixed point on a small circle that rolls within a larger circle. It is comparable to the cycloid but instead of the circle rolling along a line, it rolls within a circle.

Logic used for Hypocycloid:;

```
float a=pt.x;
float b=pt.y;
float radius2=5,radius1=4*radius2;
for (float angle = 0;  angle <= 200; angle += 1)
{
 pt.x = pt.x + (((radius1- radius2) * cos(angle)) + (radius2 * cos((radius1/radius2 - 1.0)*angle)));
 pt.y = pt.y +  (((radius1- radius2) * sin(angle)) - (radius2 * sin((radius1/radius2 - 1.0)*angle)));
 RS_Point* point1 = new RS_Point(container, RS_PointData(pt));
 container->addEntity(point1);
 pt.x =a;
 pt.y=b; }
```

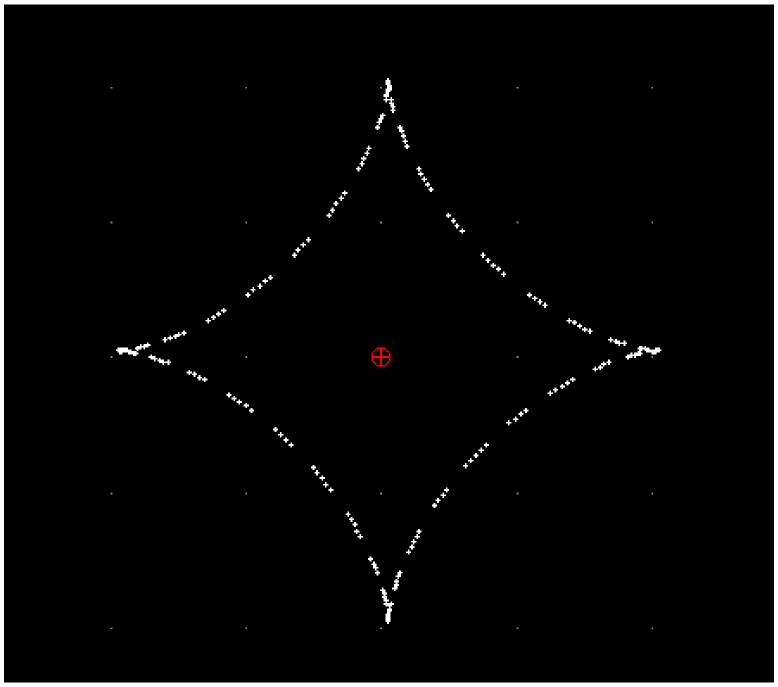


Figure 2.1.5: Hypocycloid

2.1.6 <u>Nephroid:</u>

The nephroid is a plane curve whose name means kidney-shaped (compare nephrology.)

Logic for nephroid:

```
    pt.x = pt.x + (radius * (3.0 * cos(angle) - cos(3.0*angle))+.5);
    pt.y = pt.y + (radius * (3.0 * sin(angle) - sin(3.0*angle))+.5);
```

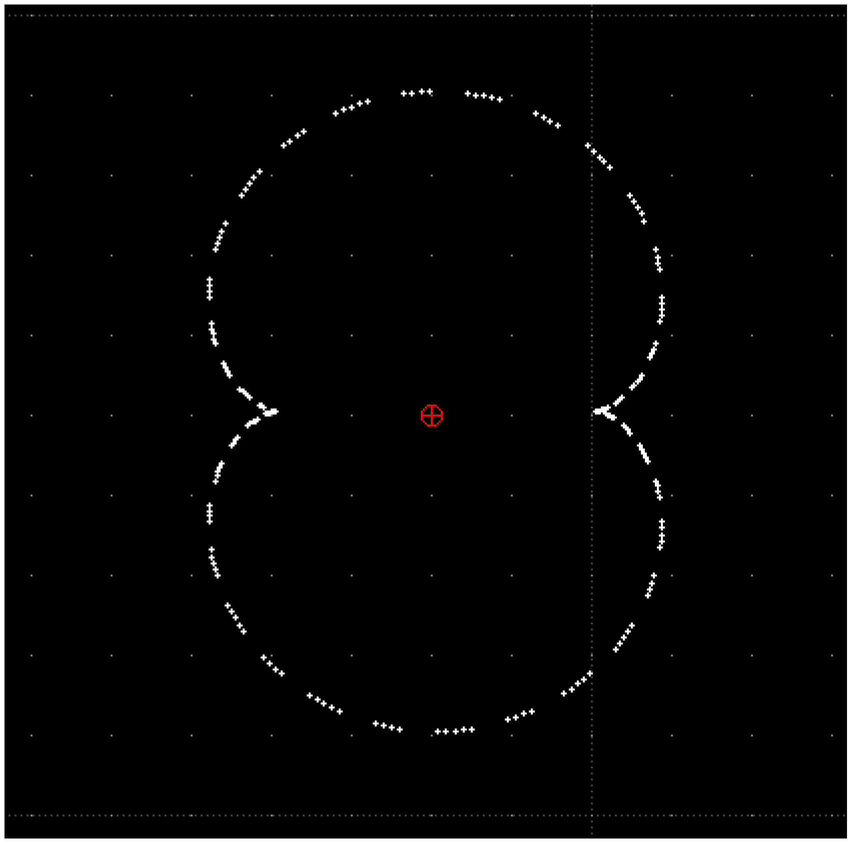


Figure 2.1.6: Nephroid

2.2 User's Documentation:

LibreCAD is an Application for Computer Aided Design (CAD) in two dimension (2D). With LibreCAD you can create technical drawings such as plans for building, interiors, mechanical parts or schematics and diagrams.

COMPILATION:

Now you can run qmake (or qmake-qt4) to create a makefile and run make to compile LibreCAD.

Commands to run:
Make sure that you are in the folder (Librecad).

```
$cd LibreCAD
$qmake librecad.pro
$make (or $time make)
```

librecad.pro is a project file. qmake creates a makefile. Make command will compile the project. Compiling LibreCAD might take a while, depending on the speed of your computer, but just let it run until it finishes.

```
$time make
```

Not down the time you take using ‘Time’ Command. My system took this time for compilation

```
real 11m40.172s
user 10m49.105s
sys 0m46.903s
```

To check librecad’s version, Goto Help menu in menu bar > About
Version: master
SCM Revision: 2.0.0beta1
Compiled on: Apr 21 2013

If error occurs:

1) If error is muparser error:rs_math: muparser not found, then install muparser library by giving command in terminal.

```
$ sudo apt-get install libmuparser-dev
```

After installing libraries replace:
#include <muParser.h> with #include “/usr/include/muParser/muParser.h” in the file librecad/src/lib/math/rs_math.cpp

2) If error is boost library error: boost library not found

```
$ sudo apt-get install libboost-dev
```

To finally run LibreCAD, execute the following commands:

```
$cd unix
$./librecad
```

How to use new features?

Open LibreCAD, There are new features added to toolbar, menubar, and from command line.

1) From menu,

goto menu>Draw>New Features> (Select any feature from 6 new features)

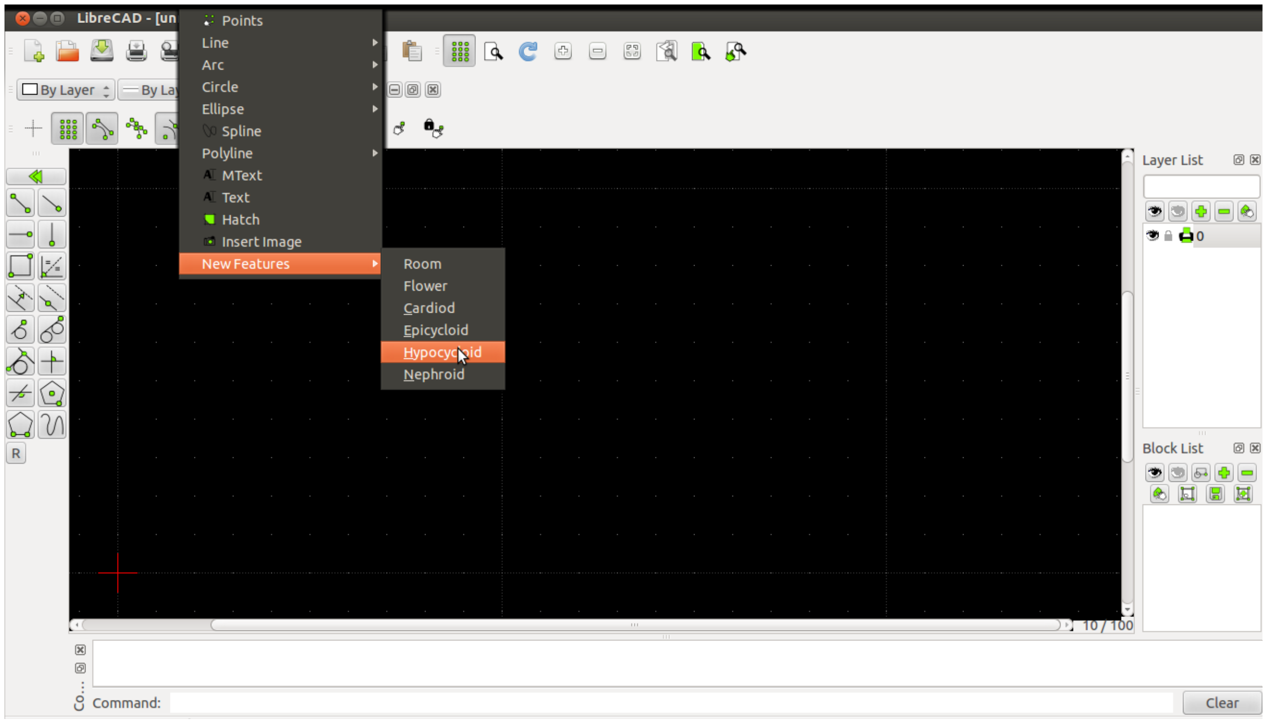


Figure 2.7: LibreCAD Window with all 6 new features

2) From Toolbar,

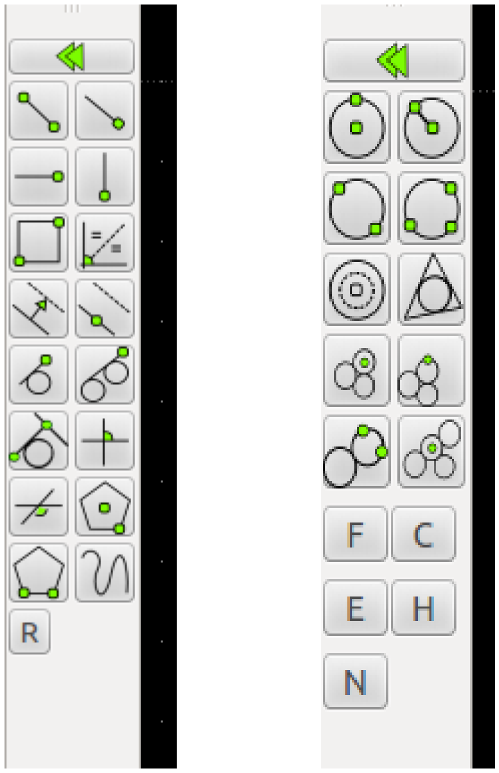


Figure 2.8: Toolbar (1) Line Toolbar (2) Circle Toolbar

3) From Command Line,
Use short hand commands in command line,

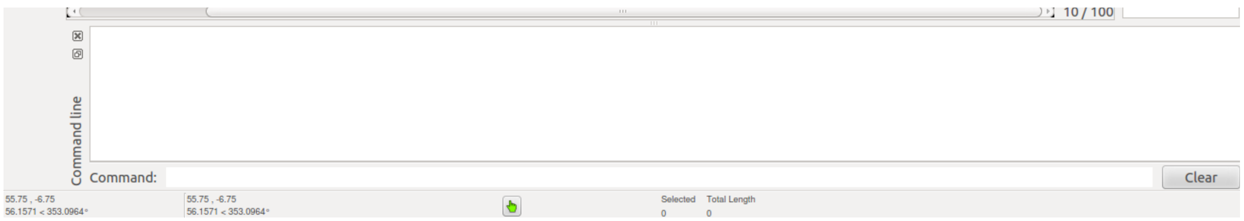


Figure 2.9: Command line Bar

Room ------------> room
Then Specify its first corner (x1,y1) say 60,60
Specify its second corner (x2,y2) say 30,30
note: X and Y should be separated by comma.

Flower-----------> fl

Specify center(x,y) say 30,30
Specify radius(r) say 40

Cardiod----------->crd
Specify location(x,y) say 30,30

Epicycloid-------->epi
Specify location(x,y) say 30,30

Hypocycloid----->hypo
Specify location(x,y) say 30,30

Nephroid--------->Neph/ Nephro
Specify location(x,y) say 30,30

Chapter - 3
REVERSE ENGINEERING

The reproduction of another manufacturer's product following detailed examination of its construction or composition.Reverse engineering, in computer programming,is a technique used to analyze software in order to identify and understand the parts it is composed of. The usual reasons for reverse engineering a piece of software are to recreate the program, to build something similar to it, to exploit its weaknesses or strengthen its defenses.

Because closed, proprietary software never comes with documentation that reveals the source code used to create it, people use reverse engineering whenever they want to understand the software's inner workings. Some hackers use reverse engineering to find weak points of programs which they can exploit.Other hackers use reverse engineering to locate weak points with the intention of strengthening the defenses there.

Software companies with competing products reverse engineer their competitors' programs to find out where and how improvements can be made on their own products. Some companies use reverse engineering when they don't have similar products yet, to create products of their own.

Those who intend to build their own product based on an existing one often prefer reverse engineering over creating from scratch because once the parts and the dependencies are identified, the process of reconstructing tends to be much easier.

Reverse engineering is taking apart an object to see how it works in order to duplicate or enhance the object. The practice, taken from older industries, is now frequently used on computer hardware and software. Software reverse engineering involves reversing a program's machine code (the string of 0s and 1s that are sent to the logic processor) back into the source code that it was written in, using program language statements.

3.1 AutoCAD DXF Specifications:

AutoCAD DXF (Drawing Interchange Format, or Drawing Exchange Format) is a CAD data file format

developed by Autodesk for enabling data interoperability between AutoCAD and other programs.
DXF was originally introduced in December 1982 as part of AutoCAD 1.0, and was intended to provide an exact representation of the data in the AutoCAD native file format, DWG (Drawing), for which Autodesk for many years did not publish specifications. Because of this, correct imports of DXF files have been difficult. Autodesk now publishes the DXF specifications on its website for versions of DXF dating from AutoCAD Release 13 to AutoCAD 2010.

The DXF format is a tagged data representation of all the information contained in an AutoCAD drawing file. *Tagged data* means that each data element in the file is preceded by an integer number that is called a *group code*. A group code's value indicates what type of data element follows. This value also indicates the meaning of a data element for a given object (or record) type. Virtually all user-specified information in a drawing file can be represented in DXF format.

ASCII versions of DXF can be read with a text-editor. The basic organization of a DXF file is as follows:

- **HEADER** section – General information about the drawing. Each parameter has a variable name and an associated value.
- **CLASSES** section – Holds the information for application-defined classes whose instances appear in the BLOCKS, ENTITIES, and OBJECTS sections of the database. Generally does not provide sufficient information to allow interoperability with other programs.
- **TABLES** section – This section contains definitions of named items.

Application ID (APPID) table

Block Record (BLOCK_RECORD) table

Dimension Style (DIMSTYPE) table

Layer (LAYER) table

Linetype (LTYPE) table

Text style (STYLE) table

User Coordinate System (UCS) table

View (VIEW) table

Viewport configuration (VPORT) table

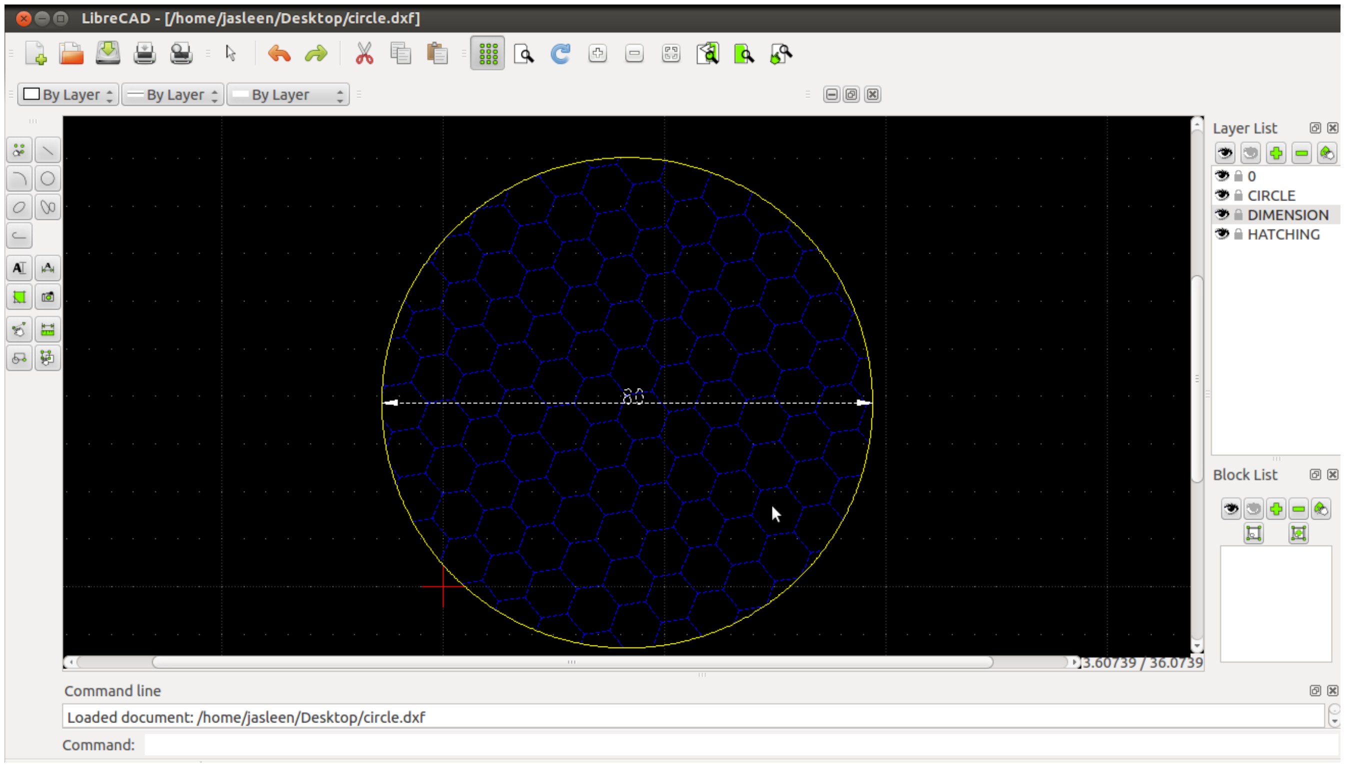
- **BLOCKS** section – This section contains Block Definition entities describing the entities comprising each Block in the drawing.
- **ENTITIES** section – This section contains the drawing entities, including any Block References.
- **OBJECTS** section – Contains the data that apply to nongraphical objects, used by AutoLISP and ObjectARX applications.
- **THUMBNAILIMAGE** section – Contains the preview image for the DXF file.
- **END OF FILE**

The data format of a DXF is called a "tagged data" format which "means that each data element in the file is preceded by an integer number that is called a group code. A group code's value indicates what type of data element follows. This value also indicates the meaning of a data element for a given object (or record) type. Virtually all user-specified information in a drawing file can be represented in DXF format."

Group codes and the associated values define a specific aspect of an object or entity. The line immediately following the group code is the associated value. This value can be a a string, an integer, or a floating-point value, such as the X coordinate of a point. The lines following the second line of the group, if any, are determined by the group definition and the data associated with the group.

Special group codes are used as file separators, such as markers for the beginning and end of sections, tables, and the end of the file itself. Entities, objects, classes, tables and table entries, and file separators are introduced with a 0 group code that is followed by a name describing the group.

The maximum DXF file string length is 256 characters. If your AutoCAD drawing contains strings that exceed this number, those strings are truncated during SAVE, SAVEAS, and WBLOCK. OPEN and INSERT fail if your DXF file contains strings that exceed this number.



Loaded document: /home/jasleen/Desktop/circle.dxf

Figure 3.1: Reverse Engineering Through DXF 2002

Chapter - 4
REQUIREMENT SPECIFICATIONS

Hardware Requirment:

Processor	intel core i5
Speed	2 GHz
RAM	250 MB
Hard Disk	40 GB

Software Requirement:

Platform	Linux
Operating System	Ubuntu12.04
Software	LibreCAD 2.* release
IDE	Qt Creator
Build tool	qmake
Libraries	Qt framework

Chapter - 5
METHODOLOGY

Hard work pays a lot. This Project takes a daily man hour of 3-4 hours. I took more than a month to study its code. Being an Open Source Software, its code is available to us. And its Free, we have right to modify its code. I created these 6 features with the aim of contributing in LibreCAD.

I discussed my project with LibreCAD developers and they help me in finding the solutions to these problems. I opened this source code using Qt creator IDE. It helps to provide ease to understand and modify the code.

I worked in two ways: 1) By engineering 2) By reverse engineering
1) The branch of science and technology concerned with the design, building, and use of engines, machines, and structures.
2) The reproduction of another manufacturer's product following detailed examination of its construction or composition.

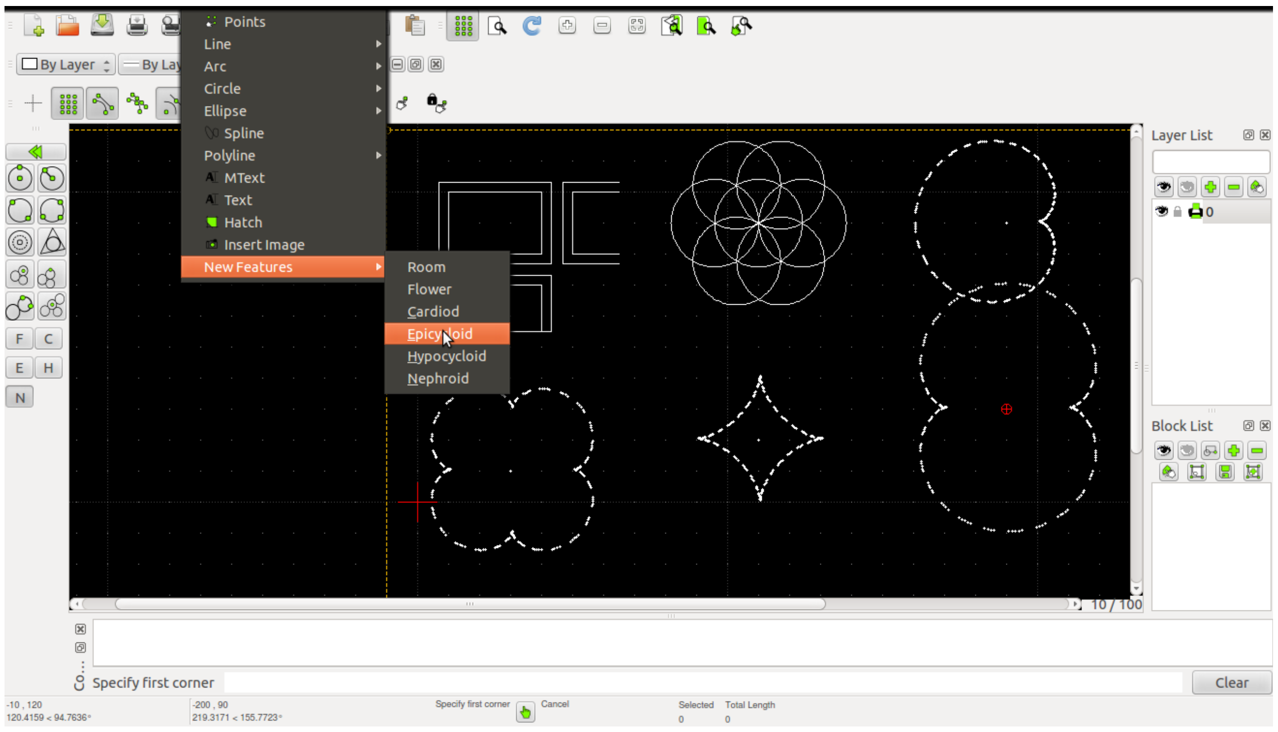


Figure 5.1: LibreCAD with 6 New Features

Chapter - 6
RESULTS AND DISCUSSION

Result

- Realising programming as art.
- Learning the basics of programming in C++.
- Learning the basics of computer graphics.
- Learn about programming in Qt.
- Learn the basics of making application made in qt in Android.
- Working in an open source community such as LibreCAD.
- Reading and understanding the existing code of LibreCAD
- Adding 6 new features in LibreCAD.

Discussion with My Mentor and LibreCAD developers

1) *Suggestion of Idea of adding layers concept :*

Coversation is at http://www.rvantwisk.nl/librecad-irc-log/2012/november/27.html
In this conversation, I share an idea to make a tree-like structure of layers (Sub-layers) . dli_ (Dongxu Li) talked about list structure, It is already there in LibreCAD, as all layers added are in a list.
He also told about exposing the layer number to control the layers.
Then sir said after reading this conversion to make Layers concept in LibreCAD. Which is under construction.

Chapter - 7
CONCLUSION AND FUTURE SCOPE

The future of GUI programming in my opinion is Qt programming, which involves the efficiency and low levelness of C++ as it's a C++ framework but also has all the qualities that a modern programming language has. It gives you a way to easily develop applications and GUIs which using C++ was earlier not possible. Open source is always growing field I have a vision that LibreCAD will someday be as famous as other CAD programs such as Autocad. This is the dawn of Open Source Softwares and we still have to grow a lot. I have added Six new features in LibreCAD many more may add many more features and by the time passes, the LibreCAD is going to improve. LibreCAD is completely scalable so I can also add as many features as I want. The Sine Wave and other graphic primitives that I made using characters in console would allow console based operating systems (such as on servers and embedded systems) to work and manipulate using these output primitives. For me they were a step that made me learn and have a taste of C++ programming. Future scope is bright as I believe that this is just the beginning and lot of more features will be added.

Now I am working on LibreCAD new features which was discussed by me, my mentor, and my developers. Layers will be having a tree like structure in LibreCAD. It will be given name and tags as suggested by my mentor.

Chapter - 8
BIBLIOGRAPHY

1) http://librecad.wordpress.com/
2) http://librecad.org/cms/home.html
3) http://en.wikipedia.org/wiki/Qt_(framework)
4) http://en.wikipedia.org/wiki/AutoCAD_DXF
5) http://qt.digia.com/