A Mid-Term Progress Report

On

Automated Building Drawings

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
(Computer Science and Engineering)

Submitted By:
Navdeep Singh (1243678)
Mandeep Singh (1243667)

Guru Nanak Dev Engineering College
Ludhiana - 141006

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Automated Building Drawing is a project for creating two-dimensional drawings (front-view, top-view, side-view etc.) from a three-dimensional model.

The main purpose or objective of the project is to make it usable even by the layman. The main target users are the Civil Engineers who want their plans to be printed on the sheets. As of now, they have to create the drawings separately with different views in any CAD software and the 3D model separately. So to automate converting a particular three-dimensional model to the print-ready drawings (with different views), this project will be beneficial. The interface should be easy to use and pretty intuitive. Because the interface is a thing that makes user experience better and to make the user use it.

The Drawing module allows you to put your 3D work on paper. That is, to put views of your models in a 2D window and to insert that window in a drawing, for example a sheet with a border, a title and your logo and finally print that sheet. The drawing may consist of different views like top, front, side and orthographic views.

1.2 Objectives

- To put views of your models in a 2D window and to insert that window in a drawing,.
- Automatically creates orthographic views of an object.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 System Requirements

2.1.1 Hardware Requirements

- Operating System: Ubuntu 14.04 (or any other Linux distribution)
- Processor Speed: 512KHz or more
- RAM: Minimum 256MB

2.1.2 Software Requirements

- Software: Any CAD software like LibreCAD, AutoCAD.
- Programming Language: C++.
- Framework: Qt

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1 Software Requirement Analysis

A Software Requirements Analysis for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation.

- **General Description**: The Drawing module allows you to put your 3D work on paper. That is, to put views of your models in a 2D window and to insert that window in a drawing, for example a sheet with a border, a title and your logo and finally print that sheet. The drawing may consist of different views like top, front, side and orthographic views. It is developed using FreeCad, Qt and C++.

- **Users of the System**: The main target users are the Civil Engineers who want their plans to be printed on the sheets. As of now, they have to create the drawings separately with different views in any CAD software and the 3D model separately. So to automate converting a particular three-dimensional model to the print-ready drawings (with different views), this project will be beneficial. So to decrease the efforts, time and cost, it would be really beneficial.

3.1.1 Functional Requiremets

- **Specific Requirements**: This phase covers the whole requirements for the system. After understanding the system we need the input data to the system then we watch the output and determine whether the output from the system is according to our requirements or not. So what we have to input and then what well get as output is given in this phase. This phase also describe the software and non-function requirements of the system.

- **Input Requirements of the System**

  1. Three Dimensional Model.
  2. Dimensions Of Different Views.

3. Scaling Parameters.
4. Name Of Particular Object.

- **Output Requirements of the System**

1. 2-dimensional Model.
2. Generation Of Front View.
3. Generation Of Top View.
4. Generation Of Side View.

- **Special User Requirements**

1. Exporting the 2D model in pdf or svg format.

- **Software Requirements**

1. Programming language: C++
2. Framework: Qt
3. Documentation: Doxygen 1.8.3
4. Text Editor: Gedit, Notepad++, Sublime
5. Operating System: Ubuntu 12.04 or up
6. Web Server: Apache 2.4

3.1.2 Non functional requirements

1. Scalability: System should be able to handle a number of users. For e.g., handling around hundred users at the same time.
2. Usability: Simple user interfaces that a layman can understand.
3. Speed: Speed of the system should be responsive i.e. Response to a particular action should be available in short period of time.

CHAPTER 4

SOFTWARE DESIGN

4.1 Product Perspective

Drawing automation refers to the varied computer machinery and software used to digitally create, collect, store, manipulate, and relay information needed for accomplishing basic drawing tasks. Creating two dimensional models, electronic transfer, and the management of automated models comprise the basic activities of an drawing automation system.Drawing automation helps in optimizing or automating existing drawing procedures.

Automated Building Drawing is a project for creating two-dimensional drawings (front-view, top-view, side-view etc.) from a three-dimensional model. The main purpose or objective of the project is to make it usable even by the layman. The main target users are the Civil Engineers who want their plans to be printed on the sheets. As of now, they have to create the drawings separately with different views in any CAD software and the 3D model separately. So to automate converting a particular three-dimensional model to the print- ready drawings (with different views), this project will be beneficial. The interface should be easy to use and pretty intuitive. Because the interface is a thing that makes user experience better and to make the user use it.

4.2 User Characteristics

The objective of this system is to provide an efficient and effective service to the students and teachers or any other persron related to drawing directly or indirectly. It is aimed to encourage people to use computers and internet instead of paper and pen for their daily work. The Drawing module allows user to put your 3D work on paper. That is, to put views of your models in a 2D window and to insert that window in a drawing, for example a sheet with a border, a title and your logo and finally print that sheet. The drawing may consist of different views like top, front, side and orthographic views.

4.3 System Design

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

- External design: External design consists of conceiving, planning out and specifying the externally observable characteristics of the software product. These characteristics include user displays or user interface forms and the report formats, external data sources and the functional characteristics, performance requirements etc. External design begins during the analysis phase and continues into the design phase.
- Logical design: The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, which involves a simplistic (and sometimes graphical) representation of an actual system. In the context of systems design, modeling can undertake the following forms, including:
  - Data flow diagrams
  - Entity Relationship Diagrams
- Physical design: The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed as output.

4.4 Technologies Used

4.4.1 C++

C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms. As an efficient compiler to native code, its application domains include systems software, application software, device drivers, embedded software, high- performance server and client applications, and entertainment software such as video games. Sev- eral groups provide both free and proprietary C++ compiler software, including the GNU Project, Microsoft, Intel and Embarcadero Technologies. C++ has greatly influenced many other popular programming languages, most notably C# and Java. Other successful languages such as Objective- C use a very different syntax and approach to adding classes to C.

Bjarne Stroustrup began his work on C with Classes in 1979. The idea of creating a new language originated from Stroustrups experience in programming for his Ph.D. thesis. Stroustrup found that Simula had features that were very helpful for large software development, but the language was too slow for practical use, while BCPL was fast but too low-level to be suitable for large software development. When Stroustrup started working in AT&T Bell Labs, he had the problem of analyzing the UNIX kernel with respect to distributed computing. Remembering his Ph.D. experience, Stroustrup set out to enhance the C language with Simula-like features. C was chosen because it was general-purpose, fast, portable and widely used. Besides C and Simula, some other languages that inspired him were ALGOL 68, Ada, CLU and ML. At first, the class, derived class, strong type checking, inlining, and default argument features were added to C via Stroustrups C++ to

C compiler, Cfront. The first commercial implementation of C++ was released on 14 October 1985.

4.4.2 Introduction 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 copleting 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 contextsensitive Qt Help system.
- It has differents modes such as Welcome, edit debug, design,analyze and help

4.4.3 Introduction to Github

GitHub is a Git repository web-based hosting service which offers all of the functionality of Git as well as adding many of its own features. Unlike Git which is strictly a command-line tool, Github provides a web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as wikis, task management, and bug tracking

and feature requests for every project.

GitHub offers both paid plans for private repto handle everything from small to very large projects with speed and efficiency. ositories, and free accounts, which are usually used to host open source software projects. As of 2014, Github reports having over 3.4 million users, making it the largest code host in the world.

GitHub has become such a staple amongst the open-source development community that many developers have begun considering it a replacement for a conventional resume and some employers require applications to provide a link to and have an active contributing GitHub account in order to qualify for a job.

4.4.4 What is Git?

Git is a distributed revision control and source code management (SCM) system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.

As with most other distributed revision control systems, and unlike most clientserver systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server. Like the Linux kernel, Git is free and open source software distributed under the terms of the GNU General Public License version 2 to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

4.4.4.1 Installation of Git

Installation of git is a very easy process. The current git version is: 2.0.4. Type the commands in the terminal:

$ sudo apt-get update

$ sudo apt-get install git

This will install the git on your pc or laptop.

4.4.4.2 Various Git Commands

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. The commonly used Git command line instructions are:-

4.4.4.3 Create Repositories

Start a new repository or obtain from an exiting URL

$ git init [project-name]
Creates a new local repository with the specified name

$ git clone [url]
Downloads a project and its entire version history

4.4.4.4 Make Changes

Review edits and craft a commit transaction

$ git status
Lists all new or modified files to be committed

$ git diff
Shows file differences not yet staged

$ git add [file]
Snapshots the file in preparation for versioning

$ git reset [file]
Unstages the file, but preserve its contents

$ git commit -m "[descriptive message "]
Records file snapshots permanently in version history

4.4.5 Doxygen

Figure 4.1: Doxygen logo

Doxygen is a documentation generator, a tool for writing software reference documentation. The documentation is written within code, and is thus relatively easy to keep up to date. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.

Doxygen supports multiple programming languages, especially C++, C, C#, Objective-C, Java, Python, IDL, VHDL, Fortran and PHP.[2] Doxygen is free software, released under the terms of the GNU General Public License.

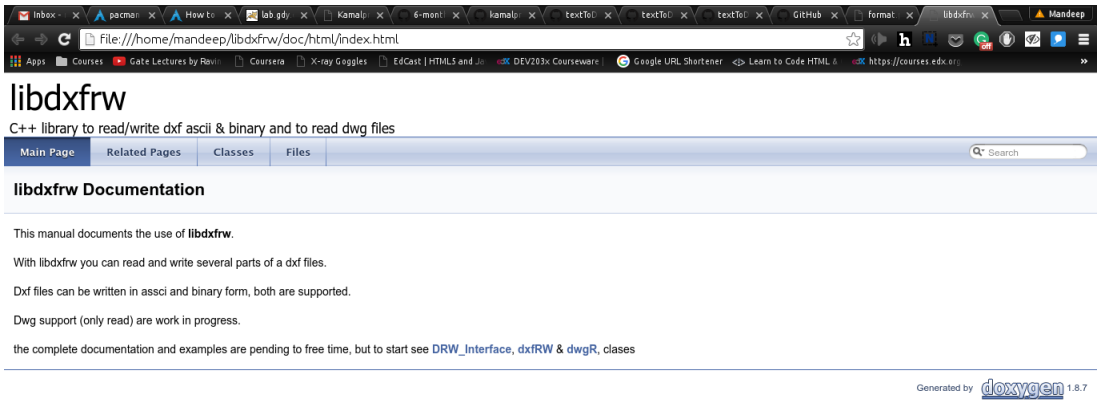

Figure 4.2: Documentation using Doxygen (main page)

4.4.5.1 Features of Doxygen

- Requires very little overhead from the writer of the documentation. Plain text will do, Markdown is support, and for more fancy or structured output HTML tags and/or some of doxygen's special commands can be used.
- Cross platform: Works on Windows and many Unix flavors (including Linux and Mac OS X).
- Comes with a GUI frontend (Doxywizard) to ease editing the options and run doxygen. The GUI is available on Windows, Linux, and Mac OS X.
- Automatically generates class and collaboration diagrams in HTML (as clickable image maps) and LaTeX (as Encapsulated PostScript images).
- Allows grouping of entities in modules and creating a hierarchy of modules.
- Doxygen can generate a layout which you can use and edit to change the layout of each page.
- Can cope with large projects easily.

4.4.5.2 Installation of Doxygen

Doxygen can be installed using following commands:

```
$ git clone https://github.com/doxygen/doxygen.git
$ cd doxygen
$ ./configure
$ make
```

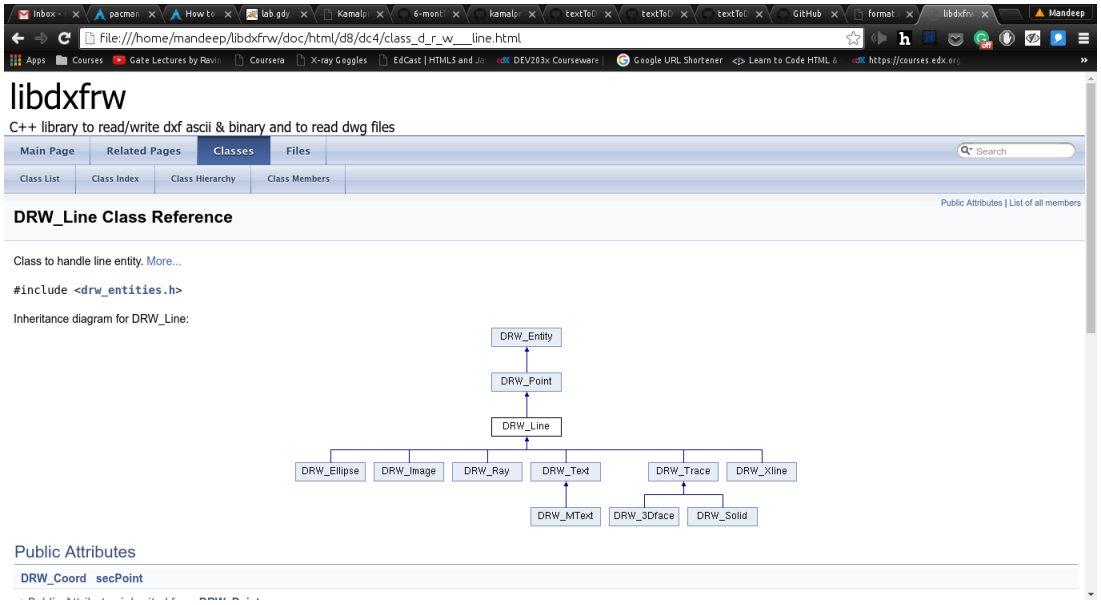

Figure 4.3: Doxygen documentation of a class

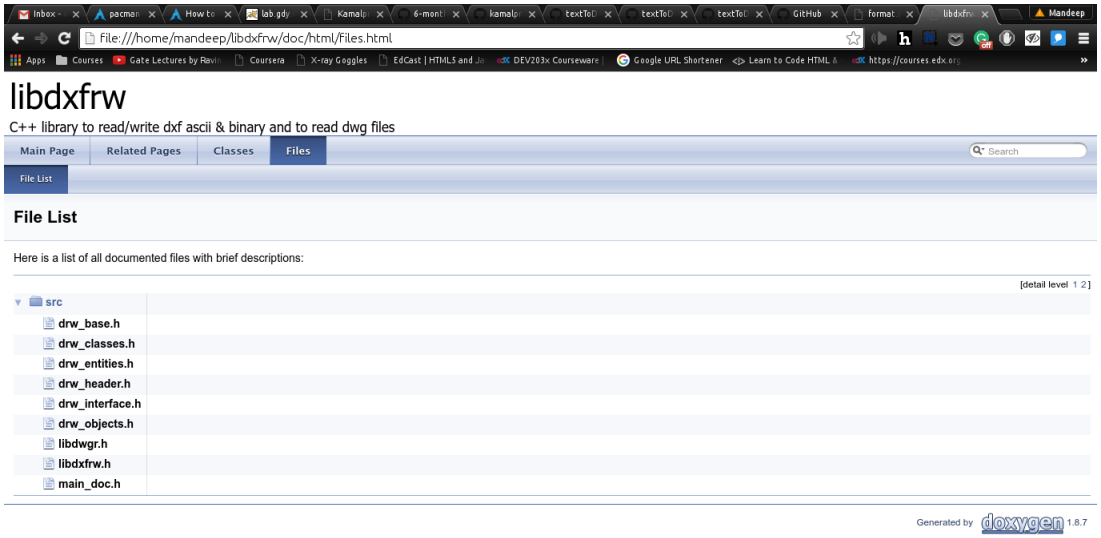

Figure 4.4: Documentation using Doxygen (list of files)

CHAPTER 5

CODING MODULE

The project uses C++ as the language and Qt-framework to make things easier and manageable. There are other technologies that have been used and are discussed further. Currently, no database have been used for this project. But we may incorporate database in our project to make it more robust and extensible.

After studying about some existing systems or libraries to read/write DXF files we came to a conclusion and choose a library 'dxflib'. It is an open-source library under development by QCAD. It supports reading and writing DXF files. The basic flow of the dxflib library is shown in the figure.

5.1 Coding Outline

The file structure of the project goes like this. There are mainly four files: a project.pro file, entity.cpp, entity.h and main.cpp.

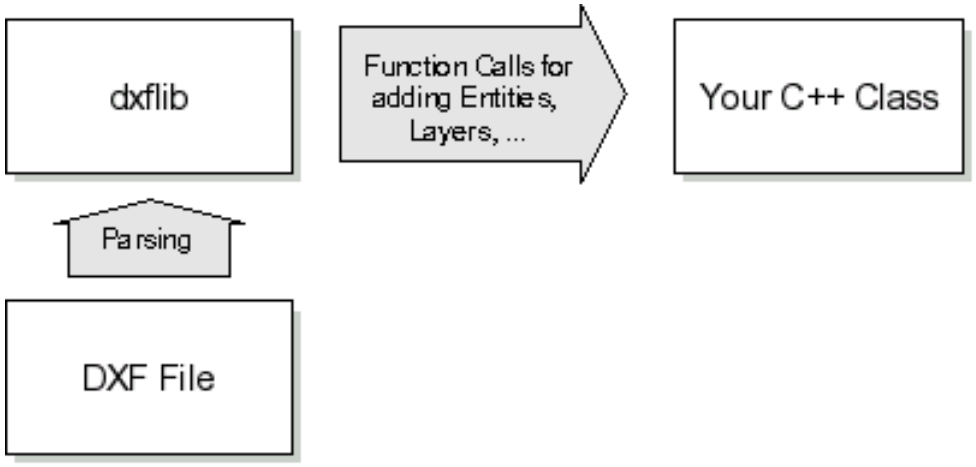

Figure 5.1: dxflib workflow

pro file

This is a Project file created for a Qt project. Project files contain all the information required by qmake to build your application, library, or plugin. The resources used by your project are generally specified using a series of declarations, but support for simple programming constructs allow you to describe different build processes for different platforms and environments.

entity.cpp

This is the file where entities are defined. As of now, there are two entities: createWall and createFlange. There will be more entities definition in the future like beam, column, stairs etc.

entity.h

It contains the function declaration for the entities like createWall etc. that have been used in the entity.cpp file later on.

main.cpp

This is the main file. First of all, we need an input file say "in.txt" which will contain the something like: wall(l=100,h=130,cx=10,cy=10). and it is then parsed by the main.cpp program using regular expressions to separate out the relevant information and stored in "out.txt" temporarily. Which is then parsed to create the entities using the library. If we talk about the input file. It contains a keyword "wall" which is pretty intuitive to guess that we are going to create a wall. Then there are four arguments where 'l' is for length of the wall, 'h' is for height of the wall and 'cx', 'cy' are the x and y coordinates of the starting point of the wall respectively.

Initially, the order in which the arguments are passed mattered. But now each of the following enitity will create the same drawing.

wall(l=100,h=30,bx=10,by=10)

wall(h=50,l=120,bx=10,by=40)

wall(bx=20,h=100,by=100,l=60)

There is another library 'libdxfrw' which we found recently that is under more rapid development than dxflib. It is also an open-source library created by LibreCAD (An open-source community-based CAD software). It have support for newer versions of DXF and it have also support to read the DWG files. We may incorporate this library into our project.

CHAPTER 6

USER INTERFACE

6.1 Workflow

The project is currently having a command-line workflow. Firstly we have to open the project in Qt. Then we will have to build the project by clicking the green play button on the bottom left (or through the menu). We also need an input file "in.txt" in which the entities with arguments are written. This file will be parsed and the output will be generated.

However, following are the screenshots to show the development process and workflow. These include the Qt-framework, the information to be parsed and the output files generated. The first figure 6.1 shows the interface of the Qt IDE.

After building the project, a new file "out.txt" is generated with the separated output as shown in figure 6.2. It will tokenize the line from the input file and separate all tokens with a new line.

If one have LibreCAD installed already on the system, then it will launch LibreCAD with the output file (DXF). Even if LibreCAD is not installed on the system, one may find the DXF file in a directory named something like Build-*. Also, one have to place the in.txt file in this directory. And it may be opened using some other CAD software of the choice of user. There are many other CAD softwares available. A popular and proprietary software AutoCAD, a free proprietary DraftSight, LibreCAD, FreeCAD, QCAD, OpenSCAD etc. Hence it depends completely on the user's choice to which software to use.

After the input being separated in "out.txt" as shown in figure 6.2, it's now time to view the output. The resulting dxf file is stored as myfile.dxf in the build directory in the current folder of the project. One of the output is shown in figure 6.4, where two walls are created using the createWall entity and a flange is created using the createFlange entity.

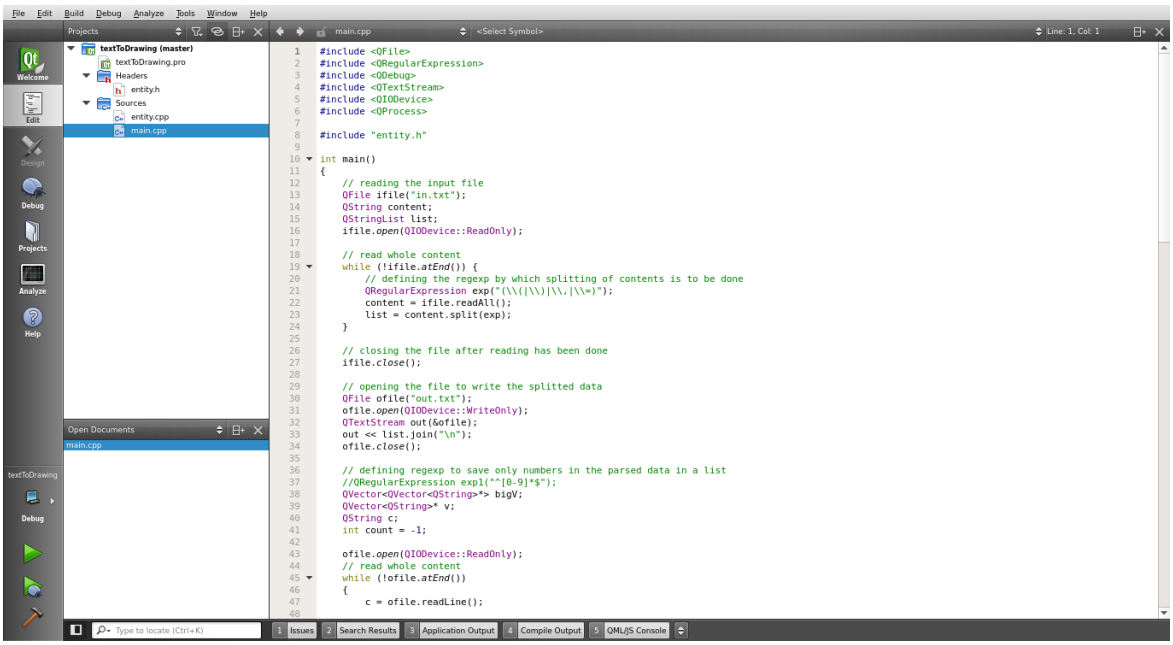

Figure 6.1: Project opened in Qt

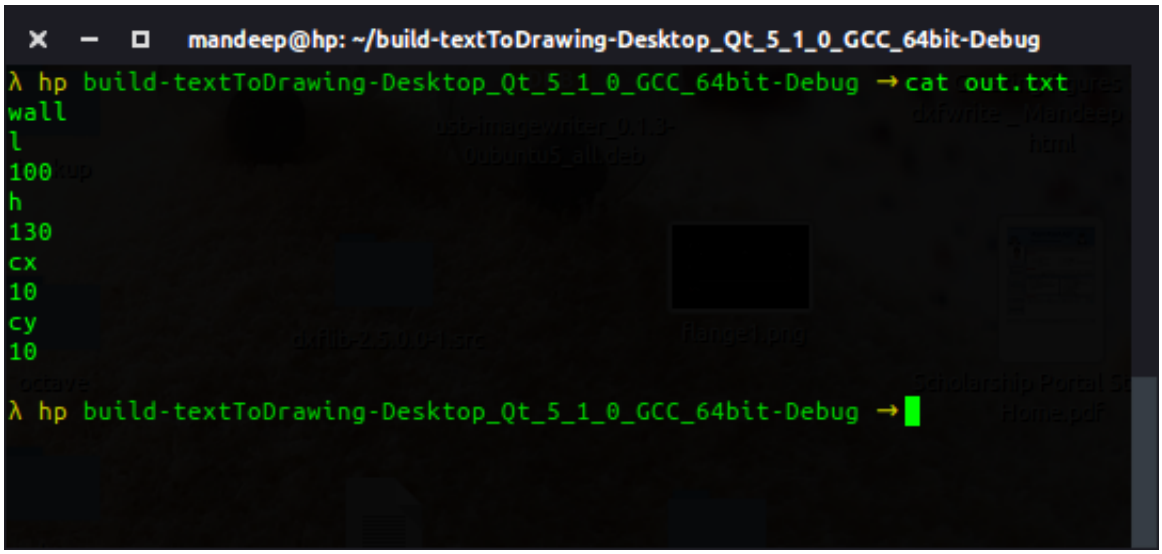

Figure 6.2: Output file after tokenizing

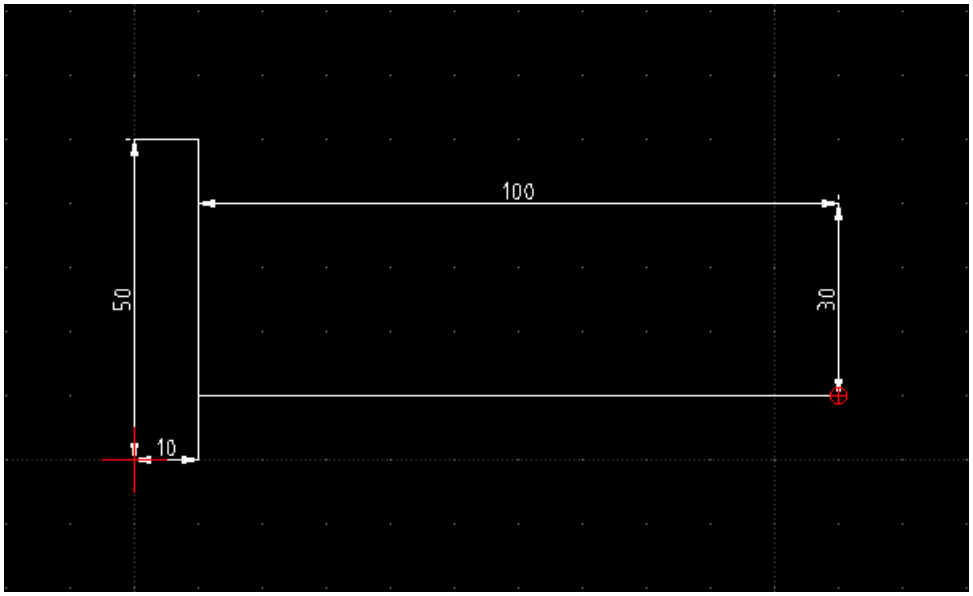

Figure 6.3: Dimensioning example

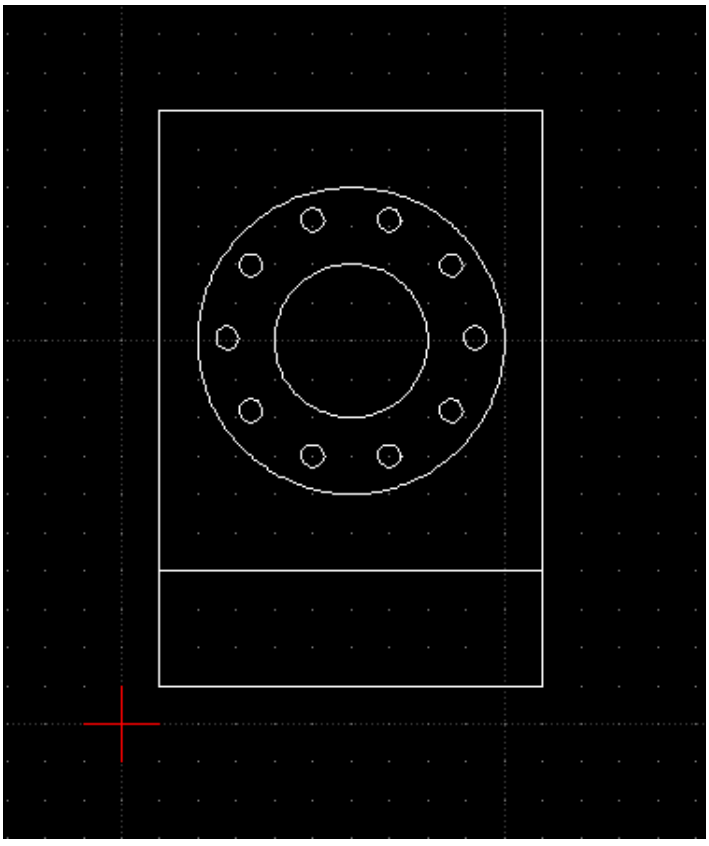

Figure 6.4: A wall and flange example

BIBLIOGRAPHY

[1] LibDXFrw, https://github.com/rvt/libdxfrw

[2] FreeCAD Code, https://github.com/FreeCAD/FreeCAD

[3] FreeCAD forum, http://forum.freecadweb.org

[4] Module, http://www.freecadweb.org/wiki/index.php?title=Drawing_Module

[5] Qt framework, http://www.qt.io/qt-framework/

[6] GitHub Repository, https://github.com/greatdevelopers/BuildD

[7] Wikipedia, http://en.wikipedia.org/wiki/