Automated Building Drawings

A major project submitted for the award of the degree of
BACHELORS OF TECHNOLOGY in
COMPUTER SCIENCE AND ENGINEERING

By
Navdeep Singh (1243678)
Mandeep Singh (1243667)

under the guidance of

Mrs. Sumeet Kaur Sehra
Assistant Professor, Department OF CSE

Guru Nanak Dev Engineering College, Ludhiana 141006

Department of Computer Science and Engineering

Guru Nanak Dev Engineering College Ludhiana 141006

Certificate

I hereby certify that the work which is being submitted in this project titled **”Automated Buiding Drawings”**, in partial fulfilment of the requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering submitted in Guru Nanak Dev Engineering College, Ludhiana, is an authentic record of my own work carried out under the supervision of Mrs. Sumeet Kaur Sehra.

(Navdeep Singh, Mandeep Singh)
125051, 125045
1243678, 1243667

This is to certify that the statements made above by the candidate are correct and true to the best of my knowledge.

(Mrs. Sumeet Kaur Sehra)
Assistant Professor
Computer Science and Engineering Department
Guru Nanak Dev Engineering College
Ludhiana-141006

Acknowledgement

The author is highly grateful to Dr. M.S. Saini Director, Guru Nanak Dev Engineering College Ludhiana for providing this opportunity to carry out the present project work.

The constant guidance and encouragement received from Dr. Parminder Singh Head of Department (Computer Science and Engineering) GNDEC, Ludhiana has been of great help in carrying out the present work and is acknowledged with reverential thanks.

I would also like to thank profusely to our project guide Mrs. Sumeet Kaur Sehra, Assistant Professor (Computer Science and Engineering) for her valuable, motivation and patience throughout the whole work. I would to thank all the staff members of Computer Science an Engineering Department for their intellectual support throughout the course of this work.

I am also thankful to all the authors whose work I have consulted and quoted in my project work. Last, but not the least I wish to thank my parents and friends who directly or indirectly have given me moral support and their relentless advice throughout the completion of this project work.

Abstract

Automated Building Drawing is a project for creating two-dimensional drawings. 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. 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.

Automated Building Drawings was made keeping in mind the various facts like people who spend several man hours, which often run into days, making the drawings using old traditions by the use of pencil and paper. This project will not only help them significantly reduce the time consumed in making the drawings, but will also make the overall process very simple and easy to use.

Also, this project is completely open source and is made using C++, libdxf, LibreCAD, and Qt and the entire code is available to the user as and when required. There is also a Complete Documentation as well as User manual along with it for making the developing and using the software a lot easier.

CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction To Project

Automated Building Drawing is a project for creating two-dimensional drawings. 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 with the use of paper and pencil. So to automate converting a particular drawing model to the print ready drawings, 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 work on computer. 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 wall and a circle and finally print that sheet.

1.2 Technologies Used

1.2.1 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. Several 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.

1.2.2 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

1.2.3 libdxfrw

libdxfrw is a free C++ library to read and write DXF files in both formats, ascii and binary form. It is licensed under the terms of the GNU General Public License version 2 (or at you option any later version).

Building and installing the library

```
mkdir build
cd build
cmake ..
make
sudo make install
```

For non-debug version

```
mkdir release
cd release
cmake -DCMAKE_BUILD_TYPE=Release ..
make
sudo make install
```

For UBUNTU/Mint Folks

```
mkdir release
cd release
cmake - DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX:PATH=/usr .. &&
make all
make
sudo make install
```

1.2.4 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.

1.2.5 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.

1.2.6 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.

1.2.7 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:-

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

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

1.3 Introduction to LaTeX

LaTeX

Figure 1.1: LaTeX Logo

LaTeX, I had never heard about this term before doing this project, but when I came to know about it's features, it is just excellent. LaTeX(pronounced /letk/, /letx/, /ltx/, or /ltk/) is a document markup language and document preparation system for the TeX typesetting program. Within the typesetting system, its name is styled as LaTeX.

Figure 1.2: Donald Knuth, Inventor Of TeX typesetting system

Within the typesetting system, its name is styled as LaTeX. The term LaTeX refers only to the language in which documents are written, not to the editor used to write those documents. In order to create a document in LaTeX, a .tex file must be created using some form of text editor. While most text editors can be used to create a LaTeX document, a number of editors have been created specifically for working with LaTeX.

LaTeX is most widely used by mathematicians, scientists, engineers, philosophers, linguists, economists and other scholars in academia. As a primary or intermediate format, e.g., translating DocBook and other XML-based formats to PDF, LaTeX is used because of the high quality of typesetting achievable by TeX. The typesetting system offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language that accesses the power of TeX. LaTeX essentially comprises a collection of TeX macros and a program to process LaTeXdocuments. Because the TeX formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

1.3.1 Typesetting

LaTeX is based on the idea that authors should be able to focus on the content of what they are writing without being distracted by its visual presentation. in preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. it therefore encourages the separation of layout from content while still allowing manual typesetting adjustments where needed.

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{\LaTeX}
\begin{document}
  \maketitle
  \LaTeX{} is a document preparation system
  for the \TeX{} typesetting program.
   \par
   $E=mc^2$
\end{document}
```

1.3.2 Installing LaTeX on System

Installation of LaTeX on personal system is quite easy. As i have used LaTeX on Ubuntu 13.04 so i am discussing the installation steps for Ubuntu 13.04 here:

- Go to terminal and type

  sudo apt-get install texlive-full

- Your Latex will be installed on your system and you can check for manual page by typing.

  man latex

  in terminal which gives manual for latex command.

- To do very next step now one should stick this to mind that the document which one is going to produce is written in any type of editor whether it may be your most common usable editor Gedit or you can use vim by installing first vim into your system using command.

  sudo apt-get install vim

- After you have written your document it is to be embedded with some set of commands that Latex uses so as to give a structure to your document. Note that whenever you wish your document to be looked into some other style just change these set of commands.
- When you have done all these things save your piece of code with .tex format say test.tex. Go to terminal and type

  latex path of the file test.tex Or pdflatex path of the file test.tex
  eg: pdflatex test.tex
  for producing pdf file simultaneously.
  After compiling it type command

  evince filename.pdf
  eg: evince test.pdf
  To see output pdf file.

1.3.3 Making Graphics in LaTeX

LaTeX s also know popularly for making complex graphics. One such example is shown below here:

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{calendar,shadings}
\renewcommand*{\familydefault}{\sfdefault}
\colorlet{winter}{blue}
\colorlet{spring}{green!60!black}
\colorlet{summer}{orange}
\colorlet{fall}{red}
\newcount\mycount
\begin{document}
\begin{tikzpicture}[transform shape,
every day/.style={anchor=mid,font=\tiny}]
\node[circle,shading=radial,outer color=blue!30,inner color=white,
minimum width=15cm] {\textcolor{blue!80!black}{\Huge\the\year}};
\foreach \month/\monthcolor in
{1/winter,2/winter,3/spring,4/spring,5/spring,6/summer,
7/summer,8/summer,9/fall,10/fall,11/fall,12/winter} {
\mycount=\month
\advance\mycount by -1
\multiply\mycount by 30
\advance\mycount by -90
\shadedraw[shading=radial,outer color=\monthcolor!30,middle color=white,
inner color=white,draw=none] (\the\mycount:5.4cm) circle(1.4cm);
\calendar at (\the\mycount:5.4cm) [
dates=\the\year-\month-01 to \the\year-\month-last]
if (day of month=1) {\large\color{\monthcolor!50!black}\tikzmonthcode}
if (Sunday) [red]
if (all) {
\mycount=1
\advance\mycount by -\pgfcalendarcurrentday
\multiply\mycount by 11
\advance\mycount by 90
\pgftransformshift{\pgfpointpolar{\mycount}{1.2cm}}};}
\end{tikzpicture}
\end{document}
```

LaTeX with just invoking few additional packages.

1.3.4 Pdfscreen LaTeX

There are some packages that can help to have unified document using LaTeX. Example of such a package is pdfscreen that let the user view its document in two forms-print and screen. Print for hard copy and screen for viewing your document on screen. Download this package from www.ctan.org/tex-archive/macros/latex/contrib/pdfscreen/.
Then install it using above mention method.

To test it the test code is given below:-
Just changing print to screen gives an entirely different view. But for working of pdfscreen another package required are comment and fancybox.

The fancybox package provides several different styles of boxes for framing and rotating content in your document. Fancybox provides commands that produce square-cornered boxes with single or double lines, boxes with shadows, and round-cornered boxes with normal or bold lines. You can box mathematics, floats, center, flushleft, and flushright, lists, and pages.

Whereas comments package selectively include/excludes portions of text. The comment package allows you to declare areas of a document to be included or excluded. One need to make these declarations in the preamble of your file. The package uses a method for exclusion that is pretty robust, and can cope with ill-formed bunches of text.

So these extra packages needed to be installed on system for the proper working of pdfscreen package.

1.4 Web based graphic generation using LaTeX

LaTeX is also useful when there is need of generating the graphics from browser. For example to draw a circle by just entering its radius in html input box. So this kind A of project can be conveniently handled using LaTeX. Basic idea behind this generation process is that when user clicks on submit button after entering radius a script will run that enter the radius in already made .tex file and recompiles it on server and makes its pdf and postscript file. After that user can view those files by clicking on link provided to view the files. See some screen shots of such a graphic generation project made by Dr. H.S. Rai:
So here in the above input page which is also the index page user can enter input for length of rectangle, breadth of rectangle and for radius of circle after that user can submit the values. After

the values get submitted a script get runs by php code at server side. This script first enters the dimensions of rectangle and circle that were selected by user in to an already existing .tex file and replace with the older dimensions there. After that script recompiles the the tex file and make it available for user.

In above figure it gets clear that .tex file has been compiled and pdf and postscript files are available to user and user can download the graphics so produced. Hence graphics can be generated in LaTeX through web interface.

1.5 Objectives

- To put views of your models in a 2D window and to insert that window in a drawing.
- To increase the productivity and hence efficiency.
- To encourage civil engineers to use the computer instead of drawing sheets.
- To create the drawings of various entities without using the mouse.
- To eliminate manual operations and thus saving the time and money by automating the entire drawing process.

CHAPTER 2 LITERATURE REVIEW

2.1 Overview

For the CAD drawings, there are plenty of softwares available there in the market including proprietary softwares such AutoCAD, STAAD.Pro and open-source ones like LibreCAD, OpenSCAD, FreeCAD etc. They are high-end softwares that require Designing knowledge and some computer usage knowledge. Hence, a layman that needs a small work to be done, will have to learn its usage effectively and some extra efforts for doing that. Hence this project will let them use it like they are making a wish for something to be done without any efforts.

2.2 Problems

It is difficult to operate high-end CAD software for a simple layman as he does not have much knowledge of the computer system because they are not connected to the internet in any ways. Giving training to these peoples is not easy as there is not so much manpower available to provide training and it will be expensive also. By not using the computer softwares instead of traditional drawing methods, Resources are being wasted on manually making the drawings at various levels. Also it is very difficult to make the drawing design using the traditional methods these days because of the evolving computer CAD softwares.

2.3 Solution Description

The layman is given a hand-held terminal working on latest technologies. Whenever a layman or any person related to Civil Engineering need to make a drawing of any building or other entity, then he/she is asked to give the entity or the drawing related details. On the basis of these details, the software will be able to draw the design of the specified architecture. The software prints the specified drawing and the drawing file is opened in the LibreCad. Moreover, this system will also enable the layman to know how the system works and the how the output is generated using the software and how the drawing is made.

2.4 Key Features

- Parse the input entity values into the system and produces a relavent output file with entity name and parameters.
- Export the output file in dxf format which can be imported into the CAD softwares like LibreCad.
- Enables simplification of drawing work for the civil engineers and the layman which otherwise may end up taking more time using the traditional drawing methods.

2.5 Benefits

- It enables the user to easily draw the two dimensional figures of supported entities in a very less time.
- Reduces the time required for the user to draw the designs of walls, circles and many other entities therefore helps in increasing efficiency.
- It has a very simple operating method which enables even a layman to use this system and draw the required designs using this software.
- Eliminates the manual operations used for drawings by automating the entire process.
- It reduces the time required to do a particular drawing task and thus leads to better efficiency or productivity.
- Enables the end user to create the drawings within few minutes.
- Provides an interface which can be used even by the peoples who are not so well proficint in computers.

CHAPTER 3 WORK CARRIED OUT

3.1 Problem Formulation

It is difficult to operate high end CAD software for a simple layman as he does not have much knowledge of the computer system because they are not connected to the internet in any ways. Giving training to these peoples is not easy as there is not so much manpower available to provide training and it will be expensive also. By not using the computer softwares instead of traditional drawing methods, Resources are being wasted on manually making the drawings at various levels. Also it is very difficult to make the drawing design using the traditional methods these days because of the evolving computer CAD softwares.

The pencil and paper work can be reduced to a great extent by doing the drawing work using Automated Building Drawings system making it environment friendly. Thus, making it automated process. Lot of time was wasted standing in the previous processes of making a drawing, but now with this system we have a facility to do it in an easy way. The previous process was quite difficult and time consuming. The laymen which were not quite familiar with the popular CAD softwars always found it difficult to make their design using these softwares which leads them to use the paper and pencil method for drawings. Now with the cration of this system they can easily make the desired designs and also can increase their productivity by reducing the time required.

3.2 Feasibility Study

Feasibility analysis involved a thorough assessment of the operational and technical aspects of the proposal. Feasibility study tested the system proposal and identified whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be cost effective from a business point of view and whether it can be developed with the most up to date technologies.

3.2.1 Operational Feasibility

Operational feasibility is a measure of how well a project solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. All the operations performed in the software are very quick and satisfy all the requirements.

3.2.2 Technical Feasibility

Technological feasibility is carried out to determine whether the project has the capability, in terms of software, hardware, personnel to handle and fulfill the user requirements. The assessment is based on an outline design of system requirements in terms of Input, Processes, Output and Procedures. Automated Building Drawings is technically feasible as it is built up using various open source technologies and it can run on any platform.

3.2.3 Economic Feasibility

Economic analysis is the most frequently used method to determine the cost/benefit factor for evalu- ating the effectiveness of a new system. In this analysis we determine whether the benefit is gained according to the cost invested to develop the project or not. If benefits outweigh costs, only then the decision is made to design and implement the system. It is important to identify cost and benefit factors, which can be categorized as follows:

- Development Cost
- Operation Cost

This System is also Economically feasible with 0 Development and Operating Charges as it is developed in Qt Framework and libdxfrw library which is open source technology and is available free of cost on the internet.

3.3 Facilities required for proposed work

3.3.1 Hardware Requirements

- Operating System: ubuntu 12.04 or windows 7
- Processor Speed: 512KHz or more
- RAM: Minimum 256MB

3.3.2 Software Requirements

- Software: LibreCad or FreeCad
- Programming Language: C++, Python, Qt
- Library: Dxflib, libdxfrw

3.4 Methodology

- Studying the current existing drawing automation systems and their problems.
- Proposing solutions for various problems in the existing systems.
- Implementing the solutions and keeping in mind the benefits of the Automated Building Drawings System.

3.5 Project Work

Studied Previous System:
Before starting the project, the previous challan system is studied such that we get an idea how to proceed towards the project and keeping in mind its limitations new objectives have been set.

Learn Qt:
Before starting with project, we have to go through the Qt Framework, such that there should not be any problem proceeding with project.

Get Familiar with libdxf:
We have gone through the libdxf library such that how to use this library in the proposed project.

LibreCad:
LibreCad is a CAD software used by a large number of users, the automated building drawings system will use LibreCad for showing the output of drawing.

Line Creation:
After the librecad has been installed next task was to create a simple line entity which enable the user to draw a line by given its specifications.

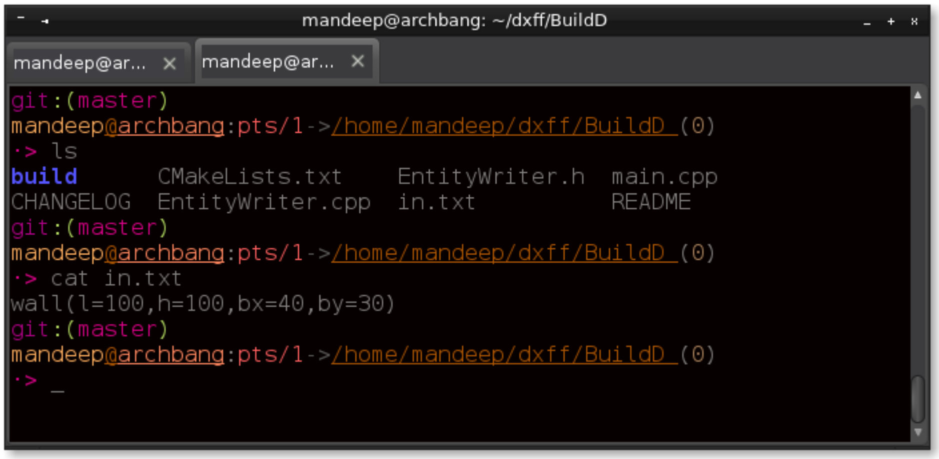
Entities Creation:
The other entities like wall, circle are crated after the line. The user has to only specify the parameters for example to draw a wall user has to only put the parameters of the wall after selecting the wall.

Creation of Input File:
An input file has been created in which user has to specify the entities to be drawn and their parameters. The required entities will be drawn automatically.

Parsing:
After getting the entities from the input file the entities are splitted and their values are stored in a output file which is later parsed into the drawing system to generate the required drawings.

3.6 Screenshots

Following are the screenshots of the workflow of the project.



```
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD (0)
·> ls
build      CMakeLists.txt    EntityWriter.h  main.cpp
CHANGELOG  EntityWriter.cpp  in.txt          README
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD (0)
·> cat in.txt
wall(l=100,h=100,bx=40,by=30)
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD (0)
·> _
```

Figure 3.1: Input file

As shown in the Figure 3.1, in the project root directory, there is a file called in.txt which is supposed to be input file that the user will dealt with for creating drawings. Currently it contains a single line that is saying to create a wall with specifications like length, height and x and y coordinates of the starting point of the wall in the Cartesian plane.

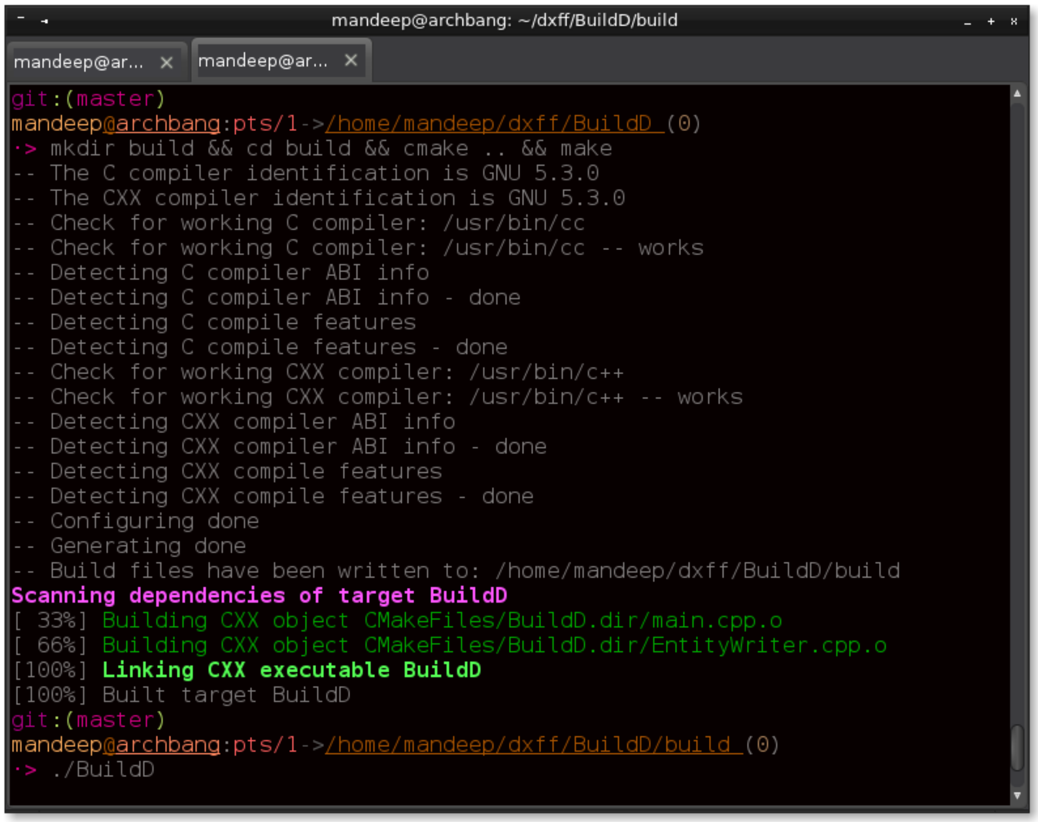
Then we will compile the project with the help of cmake and make that can be seen in Figure 3.2. It uses the cmake configuration file CMakeLists.txt that's present in the root directory of the project. We have used here the out-of-source build method. That separates the build directory to be separated from the actual source code. For that, a separate directory is created and cmake and make commands are executed there. After successful build, an executable file will be created with the name BuildD (as specified in the CMakeLists.txt). Hence, we can execute that executable file using ./BuildD.

Now let's look inside the project and think what's going on actually in there. First the in.txt file is parsed using regular expressions (using Qt libraries) and splitted and joined and stored as an semi-processed output in out.txt file as seen in the Figure 3.3. Basically input is divided into smaller tokens and stored in out.txt.

We can also work in Qt Creator. It is an IDE for C++ application development that's truly an helping aid with its nice and helping features. You can see the user interface of the Qt Creator in Figure 3.4. It contains a green play button at bottom-left side. One can click that button to build the project and execute the executable file with a single move.

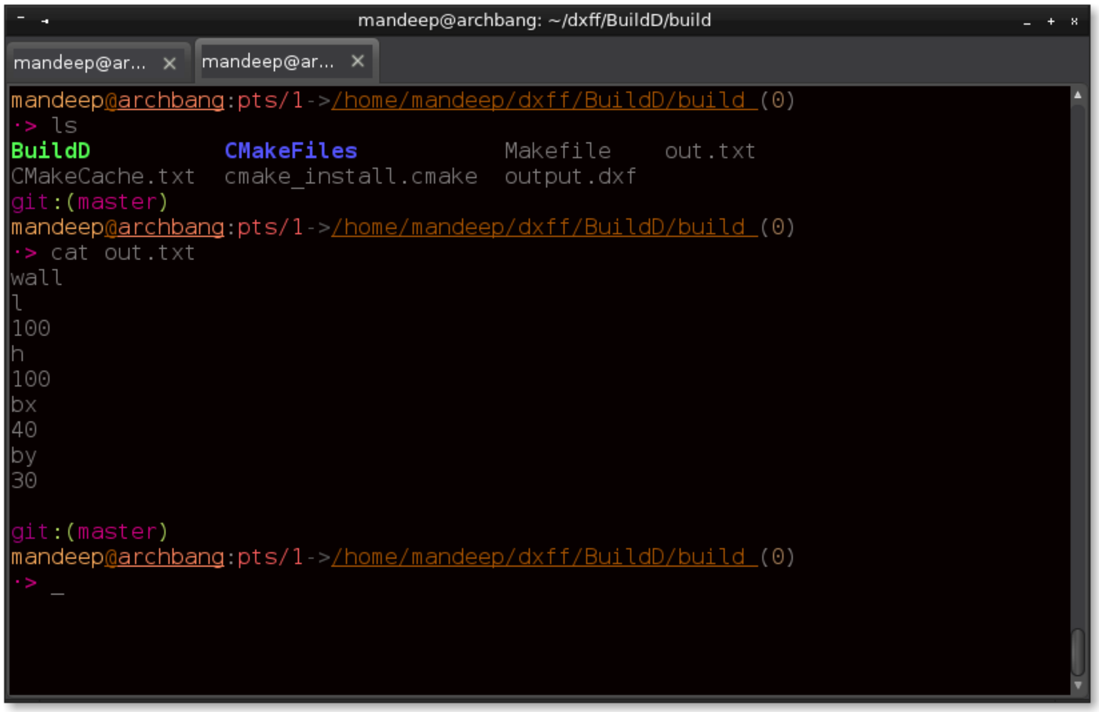
At the end, we have the final output that will be looking for: the DXF file. It's stored there in the build directory as seen in the Figure 3.5. It's named as output.dxf. We can open that file in LibreCAD directly from the command-line.

Now that our DXF file output.dxf is created and we have issued a command from command-line to open that file in LibreCAD. It can be seen in LibreCAD as shown in Figure 3.6. Although one might need to click the "Auto zoom" feature in LibreCAD to directly view the contents fit to the screen window.



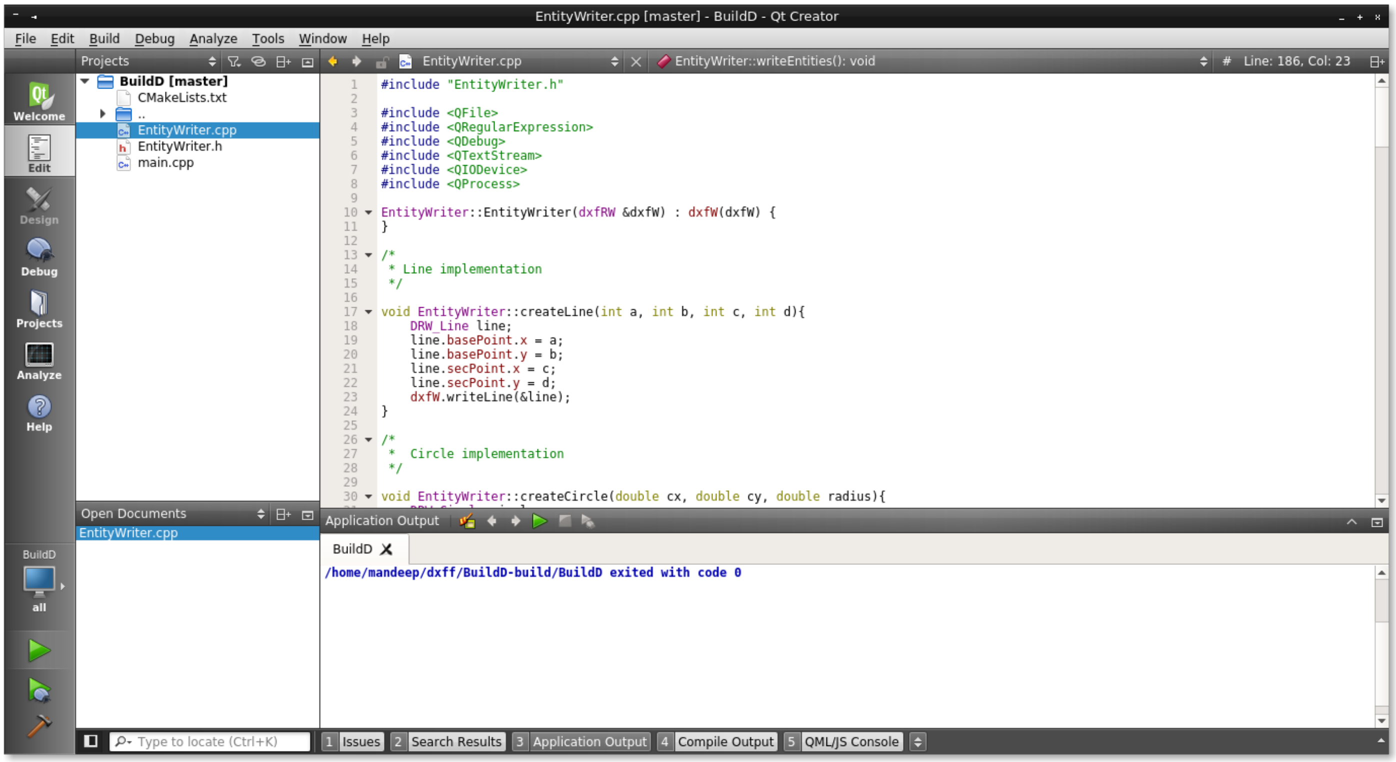
```
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD (0)
·> mkdir build && cd build && cmake .. && make
-- The C compiler identification is GNU 5.3.0
-- The CXX compiler identification is GNU 5.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mandeep/dxff/BuildD/build
Scanning dependencies of target BuildD
[ 33%] Building CXX object CMakeFiles/BuildD.dir/main.cpp.o
[ 66%] Building CXX object CMakeFiles/BuildD.dir/EntityWriter.cpp.o
[100%] Linking CXX executable BuildD
[100%] Built target BuildD
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (0)
·> ./BuildD
```

Figure 3.2: Command-line compilation



```
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (0)
·> ls
BuildD          CMakeFiles           Makefile    out.txt
CMakeCache.txt  cmake_install.cmake  output.dxf
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (0)
·> cat out.txt
wall
l
100
h
100
bx
40
by
30

git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (0)
·> _
```

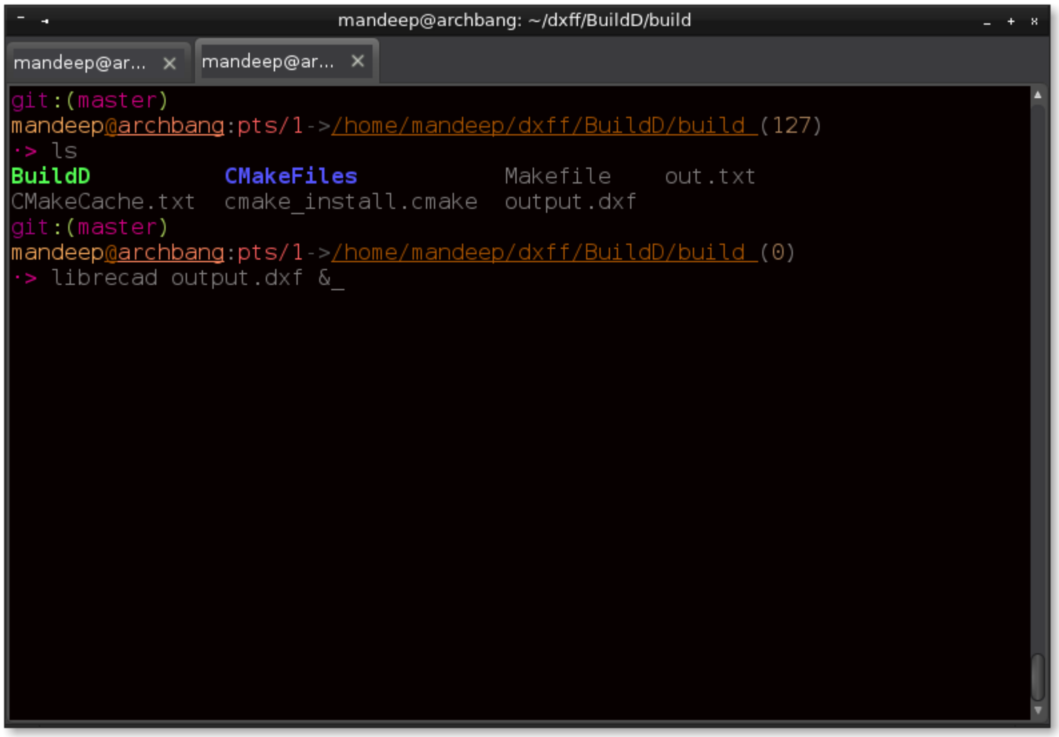
Figure 3.3: Tokenized intermediate file



```cpp
#include "EntityWriter.h"

#include <QFile>
#include <QRegularExpression>
#include <QDebug>
#include <QTextStream>
#include <QIODevice>
#include <QProcess>

EntityWriter::EntityWriter(dxfRW &dxfW) : dxfW(dxfW) {
}

/*
 * Line implementation
 */

void EntityWriter::createLine(int a, int b, int c, int d){
    DRW_Line line;
    line.basePoint.x = a;
    line.basePoint.y = b;
    line.secPoint.x = c;
    line.secPoint.y = d;
    dxfW.writeLine(&line);
}

/*
 *  Circle implementation
 */

void EntityWriter::createCircle(double cx, double cy, double radius){
```

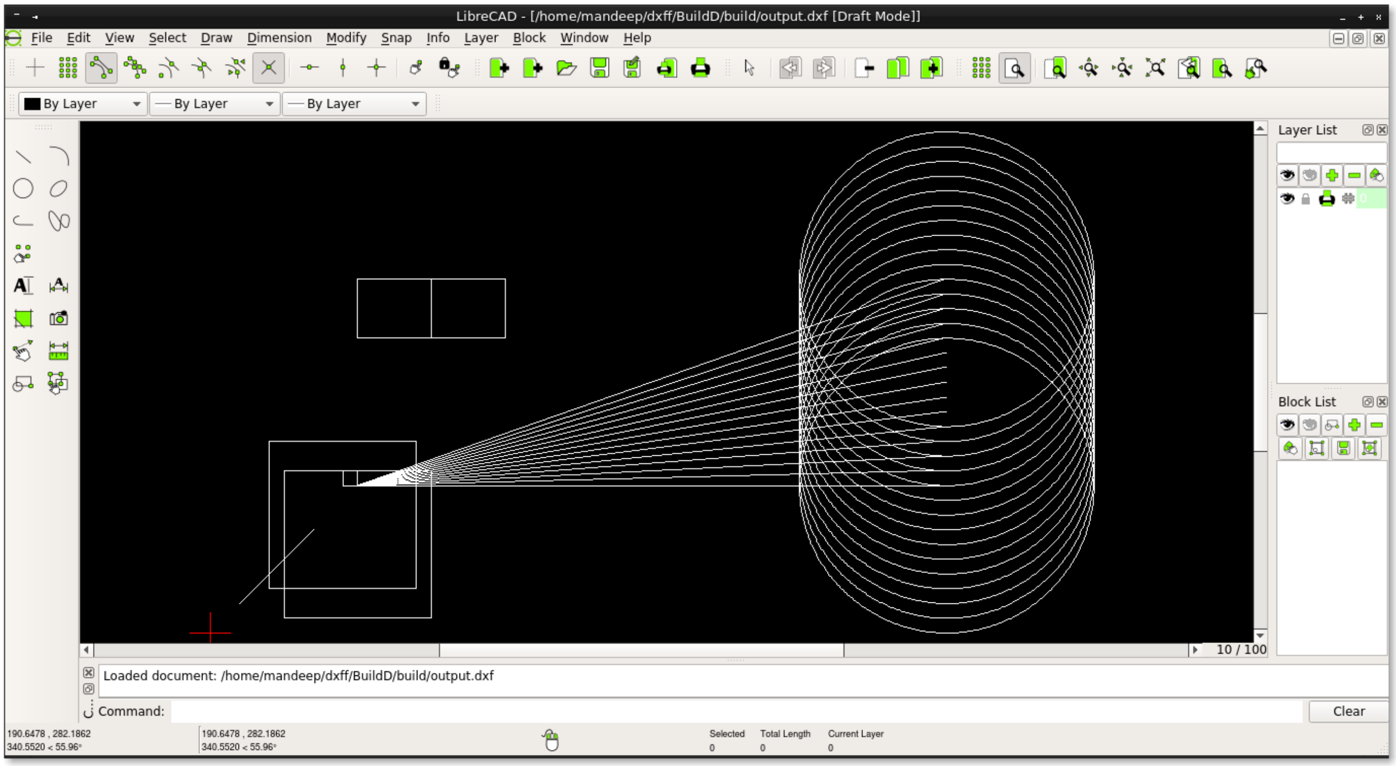
```
/home/mandeep/dxff/BuildD-build/BuildD exited with code 0
```

Figure 3.4: Qt interface



```
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (127)
·> ls
BuildD           CMakeFiles           Makefile    out.txt
CMakeCache.txt   cmake_install.cmake  output.dxf
git:(master)
mandeep@archbang:pts/1->/home/mandeep/dxff/BuildD/build (0)
·> librecad output.dxf &_
```

Figure 3.5: Opening DXF in LibreCAD from command-line


LibreCAD - [/home/mandeep/dxff/BuildD/build/output.dxf [Draft Mode]]
File Edit View Select Draw Dimension Modify Snap Info Layer Block Window Help
By Layer
By Layer
By Layer
Layer List
Block List
10 / 100
Command:
Clear
Selected
Total Length
Current Layer

```
Loaded document: /home/mandeep/dxff/BuildD/build/output.dxf
```

```
190.6478 , 282.1862     190.6478 , 282.1862     Selected  Total Length  Current Layer
340.5520 < 55.96°       340.5520 < 55.96°       0         0             0
```

Figure 3.6: DXF opened in LibreCAD

CHAPTER 4 EXPERIMENTAL RESULTS

Automated Building Drawings System is used to eliminate the previous manual process of creation of drawings using the traditional techniques by replacing it with the automated system, such that the end user can now easily make the required drawing design and the whole process can be made more easy and reliable for the users thus saving time. All the drawings can also be exported in some supported file format using this software thus making the system more useful and reliable.

User can easily maintain all the record of the previously created drawing designs by saving them into the computer memory. It eliminates the manual operations and thus increases productivity in the system by automating it. Entities can be crated just by giving their names and parameters thus the whole process becomes a very fast. The Drawing can be easily generated for a particular building by giving all the specifications such as length of the walls, height, width etc thus managing the system overall.

Firstly the user input the required specifications of the desired design such as length, height, the type of entity etc. Then these parameters and all other details are saved in a input file which is a txt file. Then this file is parsed into the system where the processing is done and the input file is splitted into the output file where its parameters are separated into the lines.

This file then parsed into the system from which the system reads the content that is specified by the user and then according to that information the Automated Building Drawings system produces a drawing which can be opened using the cad software LibreCad.

The traditional pencil and paper work can be reduced to a great extent by doing work using this software and saving paper thus making it environment friendly. Thus, making it automated process. Lot of time was wasted during the drawing of buildings using the pencil and paper. So with this software even the people who is not proficient in computer can easily take the benefit of cration of drawings using this software

CHAPTER 5 CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Automated Building Drawings System is a user friendly software system which can be used to produce an automated drawing for any entity. One can easily understand the software and use it for their convenience. Anyone even a person having less knowledge of computers can use this software to crate a two dimesional drawings of many entities and also the objects.

Firstly the user input the required specifications of the desired design such as length, height, the type of entity etc. Then these parameters and all other details are saved in a input file which is a txt file. Then this file is parsed into the system where the processing is done and the input file is splitted into the output file where its parameters are separated into the lines.

As computer is used by nearly all the organizations so it is very easy to use the softwares by the users as well as the officials and even the layman's. It would be very helpful to the people like layman and students, or civil engineering people who dont have the time to draw the drawings in a paper and spending hours in correcting the designs. Thus it will help all types of people with their drawing process.

5.2 Summary

Automated Building Drawings System is used to eliminate the previous manual process of preparing the drawings manually by replacing it with the automated system, such that the end user can now easily use this software without any inconvenience. Every person who is directly or indirectly related to the the civil engineering work can use this software for their better working processes.

Before starting the project, the previous system is studied such that we get an idea how to proceed towards the project and keeping in mind its limitations new objectives have been set. We have gone through the Qt tutorials such that the software can be made easily and in time by revising some concepts of existing libraries. Then the various cad and drawing libraries are studied and we got an idea to proceed toward our project goals.
Firstly the user input the required specifications of the desired design such as length, height, the

type of entity etc. Then these parameters and all other details are saved in a input file which is a txt file. Then this file is parsed into the system where the processing is done and the input file is splitted into the output file where its parameters are separated into the lines. After that the file is parsed into the system for the required output. The software processes the required parameters and produces an executable file. When that file is executed then we get a dxf file which is our required drawing output file. This file can be opened using any cad software which supports dxf format. We have used LibreCad in our project.

5.3 Future Scope

This project can be used for drawing automation in a particular area where the number of users are not well familiar with using the other softwares such as Freecad or Auto Cad. The drawing output records can be easily maintained by keeping the data without using the pen paper work, thus increases productivity in the system by automating it. Output drawing file can be saved into other formats too like Pdf or the formats which can be directly open in some Cad software which will allow the modifications in the previous output of drawing thus it will lead to the reusability of the existing designs, which will again increase the usefulness and efficiency of the manpower and it will also help in increasing the productivity.

BIBLIOGRAPHY

[1] Project code, https://gitlab.com/greatdeveloper/BuildD

[2] LibreCAD code, https://github.com/librecad/librecad

[3] FreeCAD code, https://github.com/FreeCAD/FreeCAD

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

[5] Libdxfrw code, https://github.com/rvt/libdxfrw

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

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