

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 Building 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.

1	Introduction	1
1.1	Overview	1
1.2	Introduction To Project	1
1.3	Objectives	2
1.4	Feasibility Study	2
1.4.1	Types of Feasibility	3
1.4.1.1	Technical Feasibility	3
1.4.1.2	Economic Feasibility	4
1.4.1.3	Behavioral Feasibility	5
1.5	Software Requirement Analysis	6
1.6	User Characteristics	6
1.7	Technologies Used	7
1.7.1	C++	7
1.7.2	Introduction To Qt	7
1.7.3	dxflib	8
1.7.3.1	Features	9
1.7.3.2	Compiling dxflib	9
1.7.3.3	Reading DXF Files	10
1.7.3.4	Implementing the Creation Interface	10
1.7.3.5	Writing DXF Files	11
1.7.3.6	Creating the Writer Object	11
1.7.3.7	Writing the DXF Header	11
1.7.3.8	Writing the Tables Section	12
1.7.3.9	Writing the Blocks Section	15
1.7.3.10	Writing the Entities Section	16
1.7.3.11	Writing the Objects Section	16
1.7.3.12	Ending and Closing the File	17
1.7.4	libdxfrw	17
1.7.5	Introduction to GitHub	17
1.7.6	What is Git?	18
1.7.7	Installation of Git	19
1.7.8	Various Git Commands	19

Create Repositories	19
Make Changes	20
1.8 FreeCAD	20
1.8.1 Installing	20
1.8.1.1 Install on Unix	21
1.9 Introduction to L ^A T _E X	26
1.9.1 Typesetting	27
1.9.2 Installing L ^A T _E X on System	27
1.9.3 Making Graphics in L ^A T _E X	28
1.9.4 Pdftscreen L ^A T _E X	29
1.10 Web based graphic generation using L ^A T _E X	29
1.11 Doxygen	30
1.11.1 Features of Doxygen	31
1.11.2 Installation of Doxygen	31
1.12 Implementation	32
2 Literature Review	34
2.1 Overview	34
2.2 Problems	34
2.3 Solution Description	34
2.4 Key Features	35
2.5 Benefits	35
3 Work Carried Out	36
3.1 Problem Formulation	36
3.2 Feasibility Study	36
3.2.1 Operational Feasibility	36
3.2.2 Technical Feasibility	37
3.2.3 Economic Feasibility	37
3.3 Facilities required for proposed work	37
3.3.1 Hardware Requirements	37
3.3.2 Software Requirements	37
3.4 Methodology	38
3.5 Project Work	38
3.6 Screenshots	38
4 Experimental Results	43
5 Conclusion and Future Scope	44
5.1 Conclusion	44
5.2 Summary	44
5.3 Future Scope	45

1.1	dxflib workflow	10
1.2	L ^A T _E X Logo	26
1.3	Donald Knuth, Inventor Of T _E X typesetting system	26
3.1	Input file	39
3.2	Command-line compilation	40
3.3	Tokenized intermediate file	40
3.4	Qt interface	41
3.5	Opening DXF in LibreCAD from command-line	41
3.6	DXF opened in LibreCAD	42

1.1 Overview

The drafting work can be automated and accelerated through the use of computer aided design systems. It may be applied for a wide variety of products in the field of automotive electronics, aerospace, naval, architecture, civil and other disciplines of engineering. CAD systems were originally used for automated drafting only. But now they also include three-dimensional modeling and computer simulated operations of the models. Sometimes, CAD is translated as computer assisted drafting, computer aided drafting or a similar phrase. Related acronyms are CADD which stands for Computer Aided design and Drafting, CAID for Computer Aided Industrial design, CAAD for Computer Aided Architectural design. All these terms are essentially synonymous, but there are some subtle differences in meaning and application.

CAD is used to design, develop and optimize products, which can be goods used by end consumers or intermediate good used in other products. CAD is also extensively used in the design of tools and equipment required in the manufacturing process, and in the drafting and design and design of all types of buildings, ranging from small residential houses to the largest commercial or Industrial complexes. CAD enables designers to layout and to develop their work on a computer screen, print and save it for future editing, thus saving a lot of time on their drawings. CAD is mainly used for detailed engineering of 3D models and/or 2D drawings of physical components, but it is also used throughout the engineering process, from conceptual design and layout of products to definition of manufacturing methods of components. Rather than building prototypes and changing components to determine the effects of tolerance ranges, engineers can use CAD systems to simulate operation to determine loads and stresses. The major benefits of such systems include lower product development costs and a greatly shortened design cycle. The CAD systems running on workstations and mainframe computers are increasingly integrated with computer-aided manufacturing systems.

1.2 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.

1.3 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.

1.4 Feasibility Study

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Carrying out a feasibility study involves information assessment, information collection and report writing. The information assessment phase identifies the information that is required to answer the three questions set out above. Once the information has been identified, you should question information sources to discover the answers to these questions Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

A feasibility study is designed to provide an overview of the primary issues related to a business idea. The purpose is to identify any make or break issues that would prevent your business from being successful in the marketplace. In other words, a feasibility study determines whether the business idea makes sense. A thorough feasibility analysis provides a lot of information necessary for the business plan. For example, a good market analysis is necessary in order to determine the projects feasibility. This information provides the basis for the market section of the business plan.

The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility

study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards.

Objectives of feasibility study are listed below.

- To analyze whether the software will meet organizational requirements
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule
- To determine whether the software can be integrated with other existing software.

1.4.1 Types of Feasibility

Various types of feasibility that are commonly considered include technical feasibility, operational feasibility, and economic feasibility.

1.4.1.1 Technical Feasibility

Technical feasibility is one of the first studies that must be conducted after the project has been identified. In large engineering projects consulting agencies that have large staffs of engineers and technicians conduct technical studies dealing with the projects. In individual agricultural projects financed by local agricultural credit corporations, the technical staff composed of specialized agricultural engineers, irrigation and construction engineers, and other technicians are responsible for conducting such feasibility studies. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system. This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed. Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyzes the technical skills and capabilities of the software development team members
- Determines whether the relevant technology is stable and established

- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within latest technology. Through the technology may become obsolete after some period of time, due to the fact that never version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

1.4.1.2 Economic Feasibility

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis.

Economic feasibility is the cost and logistical outlook for a business project or endeavor. Prior to embarking on a new venture, most businesses conduct an economic feasibility study, which is a study that analyzes data to determine whether the cost of the prospective new venture will ultimately be profitable to the company. Economic feasibility is sometimes determined within an organization, while other times companies hire an external company that specializes in conducting economic feasibility studies for them.

The purpose of business in a capitalist society is to turn a profit, or to earn positive income. While some ideas seem excellent when they are first presented, they are not always economically feasible. That is, that they are not always profitable or even possible within a company's budget. Since companies often determine their budget's several months in advance, it is necessary to know how much of the budget needs to be set aside for future projects. Economic feasibility helps companies determine what that dollar amount is before a project is ultimately approved. This allows companies to carefully manage their money to insure the most profitable projects are undertaken. Economic feasibility also helps companies determine whether or not revisions to a project that at first seems unfeasible will make it feasible.

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization.

- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
- Cost of hardware, software, development team, and training.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

1.4.1.3 Behavioral Feasibility

Behavioral feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. It is a measure of how well the solution of problems or a specific alternative solution will work in the organization. It is also measure of how people feel about the system. If the system is not easy to operate, than operational process would be difficult. The operator of the system should be given proper training. The system should be made such that the user can interface the system without any problem.

Operational feasibility is a measure of how well a proposed system 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. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviors are to be realized. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority.
- Determines whether the solution suggested by the software development team is acceptable.
- Analyzes whether users will adapt to a new software.

- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?
- The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

1.5 Software Requirement Analysis

Software requirement analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

1.6 User Characteristics

The objective of Automated Building Drawings is to give facility to the user to make the drawings of any particular building. User can choose any particular entities from the system. First of all user has to select an entity for example if he wants to draw a wall he has to specify the type of

entity to be drawn. Then later the user has to specify the parameters of the entity selected. User can select any of the entities supported by the system and can see the drawing of the same entity as the output on the screen. Also, the system has the feature to save the drawings in the dxf file format which can be opened later using the CAD software like LibreCAD. Thus, the Automated Building Drawings system allows the user to save all the drawings drawn into the computer thus user does not have to keep the old drawing sheets for future reference.

The basic workflow of using, by the user's point of view, will be to edit a normal text file. Then execution of the program will take place. The parsing will be done of the input file that user has just written and divided into small tokens. That can also be seen in a file named output.txt (just for debugging purposes).

1.7 Technologies Used

1.7.1 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.

1.7.2 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 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 contextsensitive Qt Help system.
- It has different modes such as Welcome, edit debug, design, analyze and help

1.7.3 dxflib

dxflib is an open source C++ library mainly for parsing DXFTM files. QCAD, CAM Expert and vec2web all use dxflib to import DXF files. dxflib can also write DXF files, but you need to have good knowledge of the DXF format to produce valid output.

dxflib is a C++ library for reading and writing DXF files. When reading DXF files, dxflib parses the file and calls functions that you define in your own C++ class for adding entities, layers, .. Please note that dxflib does not store any entities or other information for you. It only passes the supported entities and other objects found in the DXF file to your C++ class. Using dxflib to read DXF files doesn't require any knowledge of the DXF format. However, it's still an advantage to know the basics about entities, attributes, layers and blocks. To write DXF files with dxflib you definitely need an idea of how a DXF file is organized. dxflib does not depend on any exotic other libraries, just the standard C / C++ libraries.

1.7.3.1 Features

Supported DXF Format Sections

HEADER

Variables

TABLES

LAYER

BLOCKS

ENTITIES

3DFACE

ARC

CIRCLE

DIMENSION

Aligned

Linear

Radial

Diametric

Angular

Leader

ELLIPSE

HATCH

INSERT

LINE

LWPOLYLINE

POINT

POLYLINE

SPLINE

SOLID

TEXT

1.7.3.2 Compiling dxfli

Unix / Linux

Under Unix and Linux Systems, compiling dxfli is a simple standard procedure:

```
./configure  
make
```

This creates the file `./lib/dxfli.a`. To compile a dynamic version of dxfli, run `'make shared'` instead of `'make'`. This will create the file `./lib/libdxf.so.2.0.x.x` and the link `./lib/libdxf.so` that points to `./lib/libdxf.so.2.0.x.x`. Instead of running `'make install'`, you can also copy the header files into a header directory of your choice and copy the library files into a library directory of your choice.

Windows

There are many different ways to compile dxfli under Windows. You can use Microsoft's Visual C++ compiler, Borland's C++ command line tools, gcc and many other compilers. For this

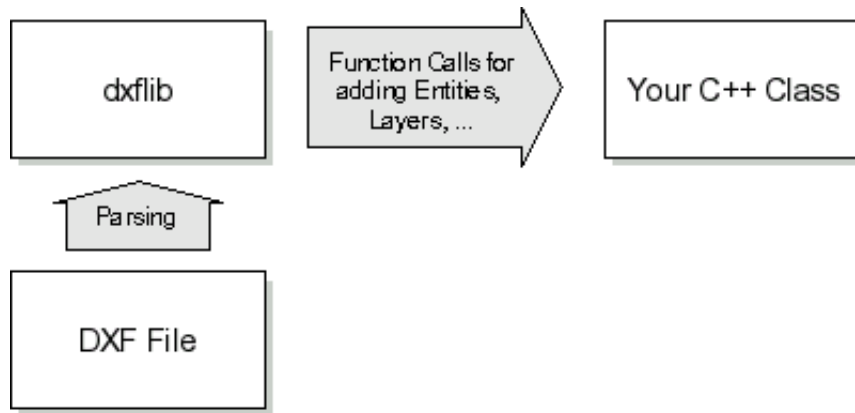


Figure 1.1: dxflib workflow

manual, only the process of compiling dxflib using cygwin [CYGWIN] and the gcc compiler from the MinGW32 package is shown:

```
./configure
MinGW32-make
```

1.7.3.3 Reading DXF Files

dxflib parses DXF files and calls functions in your class. In those functions you can for example add the entities to an entity container. The basic workflow can be seen in the Figure 1.1.

1.7.3.4 Implementing the Creation Interface

Your C++ class that handles DXF entities has to be derived from `DL_CreationInterface` or `DL_CreationAdapter`. In most cases `DL_CreationAdapter` is more convenient because it doesn't force you to implement all functions.

```
class MyDxfFilter : public DL_CreationAdapter {
    virtual void addLine(const DL_LineData& d);
    ...
}
```

The implementation of the functions in your class will typically add the entities to a container of entities or use the information in another way.

```
void MyDxfFilter::addLine(const DL_LineData& d) {
    std::cout << "Line: " << d.x1 << "/" << d.y1
    << " " << d.x2 << "/" << d.y2 << std::endl;
}
```

When reading a DXF file you simply pass on a reference to an object of your class to the parser.

```
MyDxfFilter f;
DL_Dxf* dxf = new DL_Dxf();
if (!dxf->in("drawing.dxf", &f)) {
    std::cerr << "drawing.dxf could not be opened.\n";
}
delete dxf;
```

1.7.3.5 Writing DXF Files

To write a DXF file, you need to wrap the entities, layers, blocks, .. you have into the wrapper classes of `dxflib`. Since `dxflib` does not store any entities, you need to iterate through your entities and call the write functions for each of them. Please note that you have to stick to the exact order in which you call the write functions of `dxflib`. Otherwise your DXF file will not be standard conform.

1.7.3.6 Creating the Writer Object

To create a DXF writer object you need to specify the file name as well as the DXF version you want to produce. At the time of writing only two DXF versions were supported: R12 and DXF 2000/2002. The `dxflib` codes for DXF version R12 is `DL_Codes::AC1009` and for DXF 2000/2002 `DL_Codes::AC1015`. There are two APIs you will need to write a DXF file. The API in `DL_WriterA` offers low level functions to write basic key/value tuples on which a DXF file is based. Creating a valid DXF file using only these functions would be very difficult and inconvenient. Therefore, there is a higher level API in the `DL_Dxf` class which allows you to write for example a whole line without knowing the key/value tuples that are needed for it.

1.7.3.7 Writing the DXF Header

Opening the DXF Header

The DXF header contains information about the DXF version. It has to be written before anything else with

```
dxflib.writeHeader(*dw);
```

The following list shows how a DXF header typically looks like:

```
999
dxflib 2.0.4.8
  0
SECTION
  2
HEADER
  9
$ACADVER
  1
AC1015
  9
$HANDSEED
  5
FFFF
```

As you can see, the `writeHeader()` function does not close the header. This is because you might want to store a set of variables into it. If you have to store variables, you have to do it now. If not, proceed with "Closing the Header".

Storing Additional Variables

Variables in the DXF header are used to store meta data for the drawing contained in the file. For a description of all supported variables, please refer to the DXF documentation [DXF]. The following code snippet shows examples for storing variables of different types. You can store as many variables as you need but you have to stick to the supported variable names and types in order to create a valid DXF file.

The following code creates and opens a file for a DXF 2000/2002 drawing:

```
DL_Dxf dxf;
DL_Codes::version exportVersion = DL_Codes::AC1015;
DL_WriterA* dw = dxf.out("myfile.dxf", exportVersion);
if (dw==NULL) {
    printf("Cannot open file 'myfile.dxf' \
          for writing.");
    // abort function e.g. with return
}

// int variable:
dw->dxfString(9, "$INSUNITS");
dw->dxfInt(70, 4);
// real (double, float) variable:
dw->dxfString(9, "$DIMEXE");
dw->dxfReal(40, 1.25);
// string variable:
dw->dxfString(9, "$TEXTSTYLE");
dw->dxfString(7, "Standard");
// vector variable:
dw->dxfString(9, "$LIMMIN");
dw->dxfReal(10, 0.0);
dw->dxfReal(20, 0.0);
```

Closing the Header

Use the following code to close the DXF header (end the current section):

```
dw->sectionEnd();
```

1.7.3.8 Writing the Tables Section

Opening the Tables Section

The tables section of a DXF file contains some tables defining viewports, linetypes, layers, etc. Open the tables section with the function:

```
dw->sectionTables();
```

Writing the Viewports

Viewports are not directly supported by dxfliib. However, they still need to be there in a valid DXF file. You can write the standard viewports using the function:

```
dxflib.writeVPort(*dw);
```

Writing the Linetypes

Only linetypes that are actually used need to be defined in the DXF file. For simplification, you might want to store all linetypes supported by dxfliib as shown below.

```
dw->tableLineTypes(25);
dxflib.writeLineType(*dw, DL_LineTypeData("BYBLOCK", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("BYLAYER", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("CONTINUOUS", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("ACAD_ISO02W100", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("ACAD_ISO03W100", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("ACAD_ISO04W100", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("ACAD_ISO05W100", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("BORDER", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("BORDER2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("BORDERX2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("CENTER", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("CENTER2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("CENTERX2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DASHDOT", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DASHDOT2", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("DASHDOTX2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DASHED", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DASHED2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DASHEDX2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DIVIDE", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DIVIDE2", 0));
dxflib.writeLineType(*dw,
    DL_LineTypeData("DIVIDEX2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DOT", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DOT2", 0));
dxflib.writeLineType(*dw, DL_LineTypeData("DOTX2", 0));
dw->tableEnd();
```

Writing the Layers

Layers are a substantial part of most DXF drawings. All layers that are used in the drawing need to be defined in this table section. The following example code writes three layers with names "0", "mainlayer" and "anotherlayer" to the DXF file. Note that before writing the layers, you need to specify how many layers there are in total. Layer "0" is the default layer. It cannot be omitted.

```
int numberOfLayers = 3;
dw->tableLayers(numberOfLayers);

dxf.writeLayer(*dw,
    DL_LayerData("0", 0),
    DL_Attributes(
        std::string(""), // leave empty
        DL_Codes::black, // default color
        100,             // default width
        "CONTINUOUS")); // default line style

dxf.writeLayer(*dw,
    DL_LayerData("mainlayer", 0),
    DL_Attributes(
        std::string(""),
        DL_Codes::red,
        100,
        "CONTINUOUS"));

dxf.writeLayer(*dw,
    DL_LayerData("anotherlayer", 0),
    DL_Attributes(
        std::string(""),
        DL_Codes::black,
        100,
        "CONTINUOUS"));

dw->tableEnd();
```

The default line width is given in 1/100mm. The color enum in namespace DL_Codes defines the most common colors. Writing Various Other Tables These tables are also needed. For more information, please refer to the DXF documentation [DXF].

```
dxf.writeStyle(*dw);
dxf.writeView(*dw);
dxf.writeUcs(*dw);

dw->tableAppid(1);
dw->tableAppidEntry(0x12);
dw->dxfString(2, "ACAD");
dw->dxfInt(70, 0);
```

```
dw->tableEnd();
```

Writing Dimension Styles

Dimension Styles define the look of dimensions.

```
dxfl.writeDimStyle(*dw,  
    arrowSize,  
    extensionLineExtension,  
    extensionLineOffset,  
    dimensionGap,  
    dimensionTextSize);
```

Writing Block Records

Block records define the names of available blocks in the DXF file. The following example declares the existence of two blocks with names "myblock1" and "myblock2". Note that the first call is also needed. It opens the blocks table and writes some standard blocks that might be required by the DXF version.

```
dxfl.writeBlockRecord(*dw);  
dxfl.writeBlockRecord(*dw, "myblock1");  
dxfl.writeBlockRecord(*dw, "myblock2");  
dw->tableEnd();
```

Ending the Tables Section

```
dw->sectionEnd();
```

1.7.3.9 Writing the Blocks Section

The blocks section defines the entities of each block.

```
dw->sectionBlocks();  
  
dxfl.writeBlock(*dw,  
    DL_BlockData("*Model_Space", 0, 0.0, 0.0, 0.0));  
dxfl.writeEndBlock(*dw, "*Model_Space");  
  
dxfl.writeBlock(*dw,  
    DL_BlockData("*Paper_Space", 0, 0.0, 0.0, 0.0));  
dxfl.writeEndBlock(*dw, "*Paper_Space");  
  
dxfl.writeBlock(*dw,  
    DL_BlockData("*Paper_Space0", 0, 0.0, 0.0, 0.0));  
dxfl.writeEndBlock(*dw, "*Paper_Space0");  
  
dxfl.writeBlock(*dw,  
    DL_BlockData("myblock1", 0, 0.0, 0.0, 0.0));  
// ...
```

```
// write block entities e.g. with dxf.writeLine(), ..
// ...
dxf.writeEndBlock(*dw, "myblock1");

dxf.writeBlock(*dw,
    DL_BlockData("myblock2", 0, 0.0, 0.0, 0.0));
// ...
// write block entities e.g. with dxf.writeLine(), ..
// ...
dxf.writeEndBlock(*dw, "myblock2");

dw->sectionEnd();
```

1.7.3.10 Writing the Entities Section

The entities section defines the entities of the drawing. The two entities in the following example use the attributes of their layer (256 = color by layer, -1 = line width by layer, "BYLAYER" = line style by layer).

```
dw->sectionEntities();

// write all your entities..
dxf.writePoint(
    *dw,
    DL_PointData(10.0,
        45.0,
        0.0),
    DL_Attributes("mainlayer", 256, -1, "BYLAYER"));

dxf.writeLine(
    *dw,
    DL_LineData(25.0,    // start point
        30.0,
        0.0,
        100.0,    // end point
        120.0,
        0.0),
    DL_Attributes("mainlayer", 256, -1, "BYLAYER"));

dw->sectionEnd();
```

1.7.3.11 Writing the Objects Section

```
dxf.writeObjects(*dw);
dxf.writeObjectsEnd(*dw);
```

1.7.3.12 Ending and Closing the File

```
dw->dxfeof();  
dw->close();  
delete dw;
```

1.7.4 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
```

1.7.5 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.

1.7.6 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.

Version control is the only reasonable way to keep track of changes in code, manuscripts, presentations, and data analysis projects. I used to make numbered tar.gz files for a project. But exploring the differences is difficult, to say the least. And if you use git properly, you'll have annotated each small change.

Merging collaborators' changes made easy. Have you ever had to deal with a collaborator sending you modifications distributed across many files, or had to deal with two people having made changes to the same file at the same time? Painful.git merge is the answer.

GitHub is essentially the place where you store the code that tells the baker how to make cookies. It is the best place to share code with friends, co-workers, classmates, and total strangers. But to really understand GitHub, you need to understand what a git is.

At the heart of GitHub is Git, an open source project started by Linux creator Linus Torvalds. Matthew McCullough, a trainer at GitHub, explains that Git, like other version control systems, manages and stores revisions of projects. Although it's mostly used for code, McCullough says Git could be used to manage any other type of file, such as Word documents or Final Cut projects. Think of it as a filing system for every draft of a document. Some of Git's predecessors, such as CVS and Subversion, have a central repository of all the files associated with a project. McCullough explains that when a developer makes changes, those changes are made directly to the central repository. With distributed version control systems like Git, if you want to make a change to a project you copy the whole repository to your own system. You make your changes on your local copy, then you check in the changes to the central server. McCullough says this encourages the sharing of more granular changes since you don't have to connect to the server every time you make a change.

So, what is Git in a nutshell? This is an important section to absorb, because if you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you. As you learn Git, try to clear your mind of the things you may know about other VCSs, such as Subversion and Perforce; doing so will help you avoid subtle confusion when using the tool. Git stores and thinks about information much differently than these other systems, even though the user interface is fairly similar, and understanding those differences will help prevent you from becoming confused while using it.

A git is a version control system. This system means that when you, the developer, create something and make changes to the code or release new versions, you'll be able to keep track of all the modifications in a central repository. It can even track off revisions others make so you save changes and improvements after collaborating. Developers can download a new version of the software, make changes, and upload the newest revision. And everyone can see these new changes,

download them, and contribute.

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes. In addition to being distributed, Git has been designed with performance, security and flexibility in mind.

As with most other distributed revision control systems, and unlike most clientserver systems, every Git working directory is a full-edged 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 workows.

1.7.7 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.

1.7.8 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

1.8 FreeCAD

FreeCAD is a general purpose feature-based, parametric 3D modeler for CAD, MCAD, CAx, CAE and PLM, aimed directly at mechanical engineering and product design but also fits a wider range of uses in engineering, such as architecture or other engineering specialties. It is 100% Open Source (LGPL2+ license) and extremely modular, allowing for very advanced extension and customization.

FreeCAD is based on OpenCasCade, a powerful geometry kernel, features an Open Inventor-compliant 3D scene representation model provided by the Coin 3D library, and a broad Python API. The interface is built with Qt. FreeCAD runs exactly the same way on Windows, Mac OSX, BSD and Linux platforms.

Home page: <http://www.freecadweb.org>

Documentation wiki: <http://www.freecadweb.org/wiki>

Forum: <http://forum.freecadweb.org>

Bug tracker: <http://www.freecadweb.org/tracker>

Git repository: <https://github.com/FreeCAD/FreeCAD>

1.8.1 Installing

Precompiled (installable) packages are available for Windows and Mac on the releases page at <https://github.com/FreeCAD/FreeCAD/releases>

On most Linux distributions, FreeCAD is directly installable from the software center application.

Other options are described at <http://www.freecadweb.org/wiki/index.php?title=Download>

1.8.1.1 Install on Unix

The installation of FreeCAD on the most well-known linux systems has been now endorsed by the community, and FreeCAD should be directly available via the package manager available on your distribution. The FreeCAD team also provides a couple of "official" packages when new releases are made, and a couple of experimental PPA repositories for testing bleeding-edge features.

- **Ubuntu and Ubuntu-based systems**

Many Linux distributions are based on Ubuntu and share its repositories. Besides official variants (Kubuntu, Lubuntu and Xubuntu), there are non official distros such as Linux Mint, Voyager and others. The installation options below should be compatible to these systems.

- **Official Ubuntu repository**

FreeCAD is available from Ubuntu repositories and can be installed via the Software Center or with this command in a terminal:

```
sudo apt-get install freecad
```

- **Debian and other debian-based systems**

Since Debian Lenny, FreeCAD is available directly from the Debian software repositories and can be installed via synaptic or simply with:

```
sudo apt-get install freecad
```

- **OpenSUSE**

FreeCAD is typically installed with:

```
zypper install FreeCAD
```

- **Gentoo**

FreeCAD can be built/installed simply by issuing:

```
emerge freecad
```

- **Compile from source**

On recent linux distributions, FreeCAD is generally easy to build, since all dependencies are usually provided by the package manager. It basically involves 3 steps:

- Getting the FreeCAD source code
- Getting the dependencies (packages FreeCAD depends upon)
- Configure with "cmake" and Compile with "make"

Below, you'll find detailed explanations of the whole process and particularities you might encounter. If you find anything wrong or out-of-date in the text below (Linux distributions change often), or if you use a distribution which is not listed, please help us correcting it.

- **Getting the source**

Before you can compile FreeCAD, you need the source code. There are 3 ways to get it:

- **Git**

The quickest and best way to get the code is to clone the read-only git repository now hosted on GitHub (you need the git package installed):

```
git clone https://github.com/FreeCAD/FreeCAD.git free-cad-code
```

This will place a copy of the latest version of the FreeCAD source code in a new directory called "free-cad-code".

- **Github**

The official FreeCAD repository is on Github: github.com/FreeCAD/FreeCAD

- **Source package**

Alternatively you can download a source package, but they could be already quite old so it's always better to get the latest sources via git or github.

Official FreeCAD source packages (distribution-independent): to be advised Getting the dependencies

To compile FreeCAD under Linux you have to install all libraries mentioned in Third Party Libraries first. Please note that the names and availability of the libraries will depend on your distribution. Note that if you don't use the most recent version of your distribution, some of the packages below might be missing from your repositories. In that case, look in the Older and non-conventional distributions section below.

- **Getting the dependencies**

To compile FreeCAD under Linux you have to install all libraries mentioned in Third Party Libraries first. Please note that the names and availability of the libraries will depend on your distribution. Note that if you don't use the most recent version of your distribution, some of the packages below might be missing from your repositories. In that case, look in the Older and non-conventional distributions section below.

- **Debian and Ubuntu**

On Debian-based systems (Debian, Ubuntu, Mint, etc...) it is quite easy to get all needed dependencies installed. Most of the libraries are available via apt-get or synaptic package manager.

```
build-essential
cmake
python
python-matplotlib
libtool
either:
```

libcoin60-dev (Debian Wheezy, Wheezy-backports, Ubuntu 13.04 and before)
or:

libcoin80-dev (Debian unstable(Jesse), testing, Ubuntu 13.10 and forward)
libsoqt4-dev
libxerces-c-dev
libboost-dev
libboost-filesystem-dev
libboost-regex-dev
libboost-program-options-dev
libboost-signals-dev
libboost-thread-dev
libboost-python-dev
libqt4-dev
libqt4-opengl-dev
qt4-dev-tools
python-dev
python-pyside
pyside-tools
either:

libopencascade-dev (official opencascade version)
or:

liboce*-dev (opencascade community edition)
oce-draw
libeigen3-dev
libqtwebkit-dev
libshiboken-dev
libpyside-dev
libode-dev
swig
libzipios++-dev
libfreetype6
libfreetype6-dev

Additional instruction for libcoin80-dev Debian wheezy-backports, unstable, testing, Ubuntu 13.10 and forward

Note that liboce*-dev includes the following libraries:

liboce-foundation-dev
liboce-modeling-dev
liboce-ocaf-dev
liboce-visualization-dev
liboce-ocaf-lite-dev

You may have to install these packages by individual name.

Optionally you can also install these extra packages:

`libimage-dev` (to make Coin to support additional image file formats)
`checkinstall` (to register your installed files into your system's package manager, so you can easily uninstall later)
`python-pivy` (needed for the 2D Drafting module)
`python-qt4` (needed for the 2D Drafting module)
`doxygen` and `libcoin60-doc` (if you intend to generate source code documentation)
`libspnav-dev` (for 3Dconnexion devices support like the Space Navigator or Space Pilot)

- **Compile FreeCAD**

- **Using cMake**

cMake is a newer build system which has the big advantage of being common for different target systems (Linux, Windows, MacOSX, etc). FreeCAD is now using the cMake system as its main building system. Compiling with cMake is usually very simple and happens in 2 steps. In the first step, cMake checks that every needed programs and libraries are present on your system and sets up all that's necessary for the subsequent compilation. You are given a few alternatives detailed below, but FreeCAD comes with sensible defaults. The second step is the compiling itself, which produces the FreeCAD executable. Changing any options for cmake away from their default values, is much easier with `cmake-gui` or other graphical cmake applications than with `cmake` on the command line, as the graphical applications will give you interactive feed back.

Since FreeCAD is a heavy application, compiling can take a bit of time (about 10 minutes on a fast machine, 30 minutes (or more) on a slow one)

- **In-source building**

If you are unsure then, due to its limitations, do not make an in-source build, create an out-of-source build as explained in the next section. However FreeCAD can be built in-source, which means that all the files resulting from the compilation stay in the same folder as the source code. This is fine if you are just looking at FreeCAD, and want to be able to remove it easily by just deleting that folder. But in case you are planning to compile it often, you are advised to make an out-of-source build, which offers many more advantages. The following commands will compile FreeCAD:

```
$ cd freecad (the folder where you cloned the freecad source)
```

If you want to use your system's copy of Pivy, which you most commonly will, then if not on Linux, set the compiler flag to use the correct pivy (via `FREECAD_USE_EXTERNAL_PIVY=`). Using external Pivy became the default for Linux, during development of FreeCAD 0.16, so it does not need to be manually set when compiling this version onwards, on Linux. Also, set the build type to Debug if you want a debug build or Release if not. A Release build will run much faster than a Debug build. Sketcher becomes very slow with complex sketches if your FreeCAD is a Debug build. (NOTE: the space and "." after the cmake flags are CRITICAL!):

- **For a Debug build**

```
$ cmake -DFREECAD_USE_EXTERNAL_PIVY=1 -DCMAKE_BUILD_TYPE=Debug .
$ make
```

Or for a Release build

```
$ cmake -DFREECAD_USE_EXTERNAL_PIVY=1 -DCMAKE_BUILD_TYPE=Release .
$ make
```

Your FreeCAD executable will then reside in the "bin" folder, and you can launch it with:

```
$ ./bin/FreeCAD
```

How to repair your source code directory after accidentally running an in-source build. This is a method, using Git, to repair your source code directory after accidentally running an in-source build.

- * delete everything in your source base directory EXCEPT the hidden .git folder
- * In terminal 'git reset --hard HEAD' //any remnants of an 'in source' build will be gone.
- * delete everything from your 'out of source' build directory and start over again with cmake and a full new clean build.

– Out-of-source build

If you intend to follow the fast evolution of FreeCAD, building in a separate folder is much more convenient. Every time you update the source code, cMake will then intelligently distinguish which files have changed, and recompile only what is needed. Out-of-source builds are specially handy when using the Git system, because you can easily try other branches without confusing the build system. To build out-of-source, simply create a build directory, distinct from your FreeCAD source folder, and, from the build folder, point cMake (or if using cmake-gui replace "cmake" in the code below with "cmake-gui") to the source folder:

```
mkdir freecad-build
cd freecad-build
cmake ../freecad (or whatever the path is to your FreeCAD source folder)
make
```

The FreeCAD executable will then reside in the "bin" directory (within your freecad-build directory).

– Configuration options

There are a number of experimental or unfinished modules you may have to build if you want to work on them. To do so, you need to set the proper options for the configuration phase. Do it either on the command line, passing -D `{var}:{type}={value}` options to cMake or using one of the availables gui-frontends (eg for Debian, packages cmake-qt-gui or cmake-curses-gui). Changing any options for cmake away from their default values, is much easier with cmake-gui or other graphical cmake applications than with cmake on the command line, as they will give you interactive feed back.

As an example, to configure FreeCAD with the Assembly module built just tick the box in a cmake gui application (e.g. cmake-gui) or on the command line issue:

```
cmake -D FREECAD_BUILD_ASSEMBLY:BOOL=ON ''path-to-freecad-root''
```

Possible options are listed in FreeCAD's root CmakeLists.txt file.

- **Qt designer plugin**

If you want to develop Qt stuff for FreeCAD, you'll need the Qt Designer plugin that provides all custom widgets of FreeCAD. Go to `freecad/src/Tools/plugins/widget` So far we don't provide a makefile – but calling `qmake plugin.pro` creates it. Once that's done, calling `make` will create the library `libFreeCAD_widgets.so`. To make this library known to Qt Designer you have to copy the file to `$QTDIR/plugin designer`

- **Doxygen**

If you feel bold enough to dive in the code, you could take advantage to build and consult Doxygen generated FreeCAD's Source documentation.

1.9 Introduction to L^AT_EX



Figure 1.2: L^AT_EX Logo

L^AT_EX, 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. L^AT_EX (pronounced /letk/, /letx/, /ltx/, or /ltk/) is a document markup language and document preparation system for the T_EX typesetting program. Within the typesetting system, its name is styled as L^AT_EX.



Figure 1.3: Donald Knuth, Inventor Of T_EX typesetting system

Within the typesetting system, its name is styled as L^AT_EX. The term L^AT_EX 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 L^AT_EX, a `.tex` file must be created using some form of text editor. While most text editors can be used to create a L^AT_EX document, a number of editors have been created specifically for working with L^AT_EX.

L^AT_EX 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, L^AT_EX 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 \LaTeX documents. Because the \TeX formatting commands are very low-level, it is usually much simpler for end-users to use \LaTeX .

1.9.1 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}
```

1.9.2 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 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.

1.9.3 Making Graphics in L^AT_EX

L^AT_EX 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}

```

L^AT_EX with just invoking few additional packages.

1.9.4 Pdfscreen L^AT_EX

There are some packages that can help to have unified document using L^AT_EX. 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.

1.10 Web based graphic generation using L^AT_EX

L^AT_EX 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 L^AT_EX. 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 L^AT_EX through web interface.

1.11 Doxygen



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.

Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent. Doxygen can help you in three ways:

- It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
- You can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
- You can also use doxygen for creating normal documentation (as I did for the doxygen user manual and web-site).

Doxygen looks at the files extension to determine how to parse a file. If a file has an .idl or .odl extension it is treated as an IDL file. If it has a .java extension it is treated as a file written in Java. Files ending with .cs are treated as C# files and the .py extension selects the Python parser. Finally, files with the extensions .php, .php4, .inc or .phtml are treated as PHP sources. Any other extension is parsed as if it is a C/C++ file, where files that end with .m are treated as Objective-C source files.

If you start using doxygen for an existing project (thus without any documentation that doxygen is aware of), you can still get an idea of what the structure is and how the documented result would look like. To do so, you must set the `EXTRACT ALL` tag in the configuration file to `YES`. Then, doxygen will pretend everything in your sources is documented. Please note that as a consequence warnings about undocumented members will not be generated as long as `EXTRACT ALL` is set to `YES`.

To analyse an existing piece of software it is useful to cross-reference a (documented) entity with its definition in the source files. Doxygen will generate such cross-references if you set the `SOURCE BROWSER` tag to `YES`. It can also include the sources directly into the documentation by setting `INLINE SOURCES` to `YES` (this can be handy for code reviews for instance).

Doxygen is developed under Mac OS X and Linux, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

1.11.1 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.

1.11.2 Installation of Doxygen

Doxygen can be installed using following commands:

```
$ git clone https://github.com/doxygen/doxygen.git
```

```
$ cd doxygen
```

```
$ ./configure
```

```
$ make
```

1.12 Implementation

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Training of the staff in the changeover phase.

The primary goal of implementation is to write the source code and the internal documentation so that conformance of the code to its specifications can easily be verified and so the debugging, modifications and testing are eased. This goal can be achieved by making the source code as clear and as straightforward as possible. Simplicity, Elegance and Clarity are the hallmarks of good programs whereas complexity are indications of inadequate design and misdirected thinking. The system implementation is a fairly complex and expensive task requiring numerous inter-dependent activities. It involves the effort of a number of groups of people: user and the programmers and the computer operating staff etc. This needs a proper planning to carry out the task successfully.

This project involves the following activities:

- Understand the working of the libdxfrw library and have some examples tested.
- Add some entities.
- Have an input file for the user input.
- Use parsing for extraction of information from the input file.
- Use the parsed information to create drawings.

The various functions provided by this project, that can be thought of as the features of this project, are listed as follows:

- To edit an input file in any text editor.
- Run one command and the drawings are created.
- No need to waste time creating drawings with mouse clicks (slower methods).
- Supports entities upto latest DXF version as per the DXF specifications.
- Outputs the DXF file.

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 relevant 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 proficient in computers.

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 softwares 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 iden-

tified 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 evaluating 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:

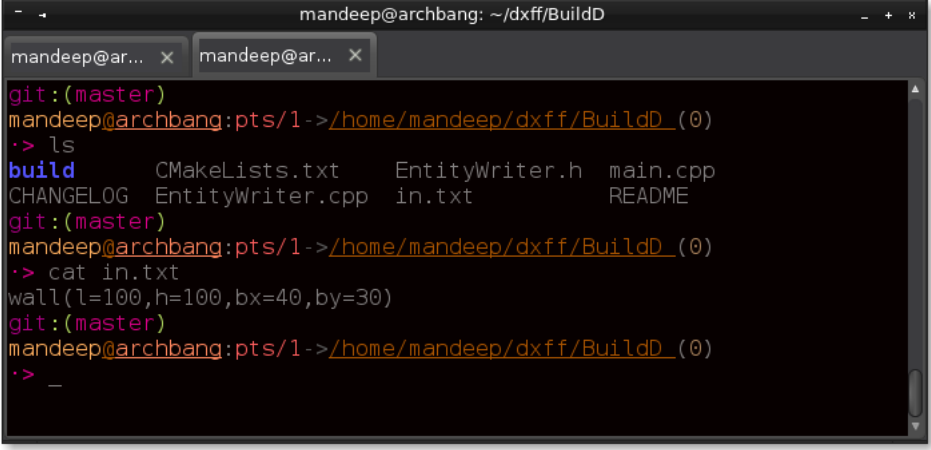
An input file has been created in which user has to specify the entities to be drawn and their parameters. The required entities will be drawn automatically.

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.

A terminal window titled 'mandeep@archbang: ~/dxff/BuildD' with two tabs. The first tab shows the command 'cat in.txt' and its output: 'wall(l=100,h=100,bx=40,by=30)'. The second tab shows the command 'ls' and its output: 'CMakeLists.txt EntityWriter.h main.cpp CHANGELOG EntityWriter.cpp in.txt README'.

```
mandeep@archbang: ~/dxff/BuildD
mandeep@ar... x mandeep@ar... x
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 deal 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.

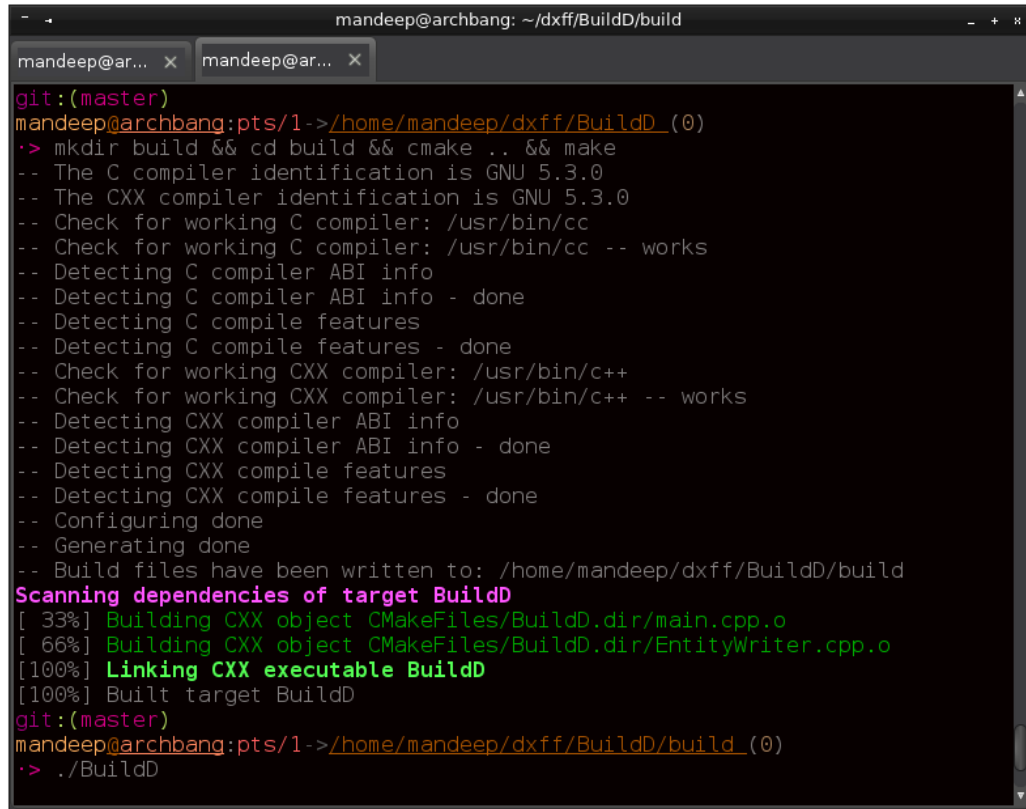
Then we will compile the project with the help of `cmake` and `make` that can be seen in Figure 3.2. It uses the `cmake` configuration file `CMakeLists.txt` that's present in the root directory of the project. We have used here the out-of-source build method. That separates the build directory to be 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.

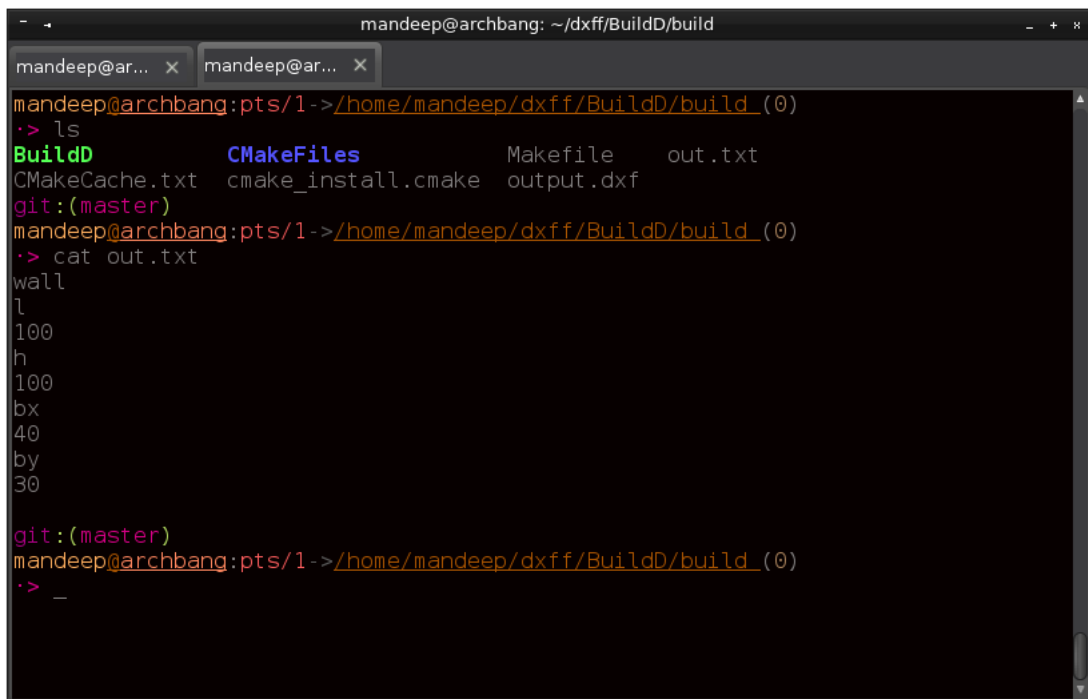
At the end, we have the final output that will be looking for: the DXF file. It's stored there in the build directory as seen in the Figure 3.5. It's named as `output.dxf`. We can open that file in LibreCAD directly from the command-line.

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.



```
mandeep@archbang: ~/dxff/BuildD/build
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: ~/dxff/BuildD/build
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

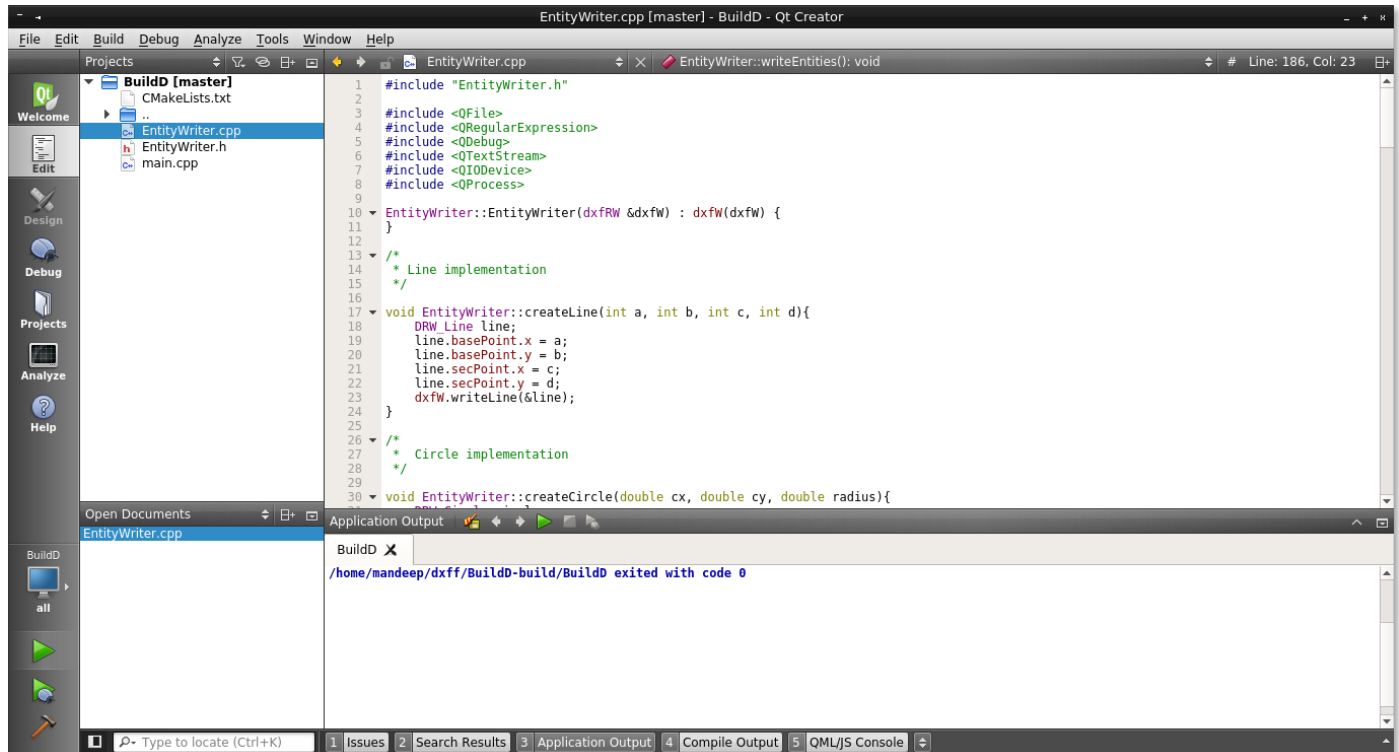


Figure 3.4: Qt interface

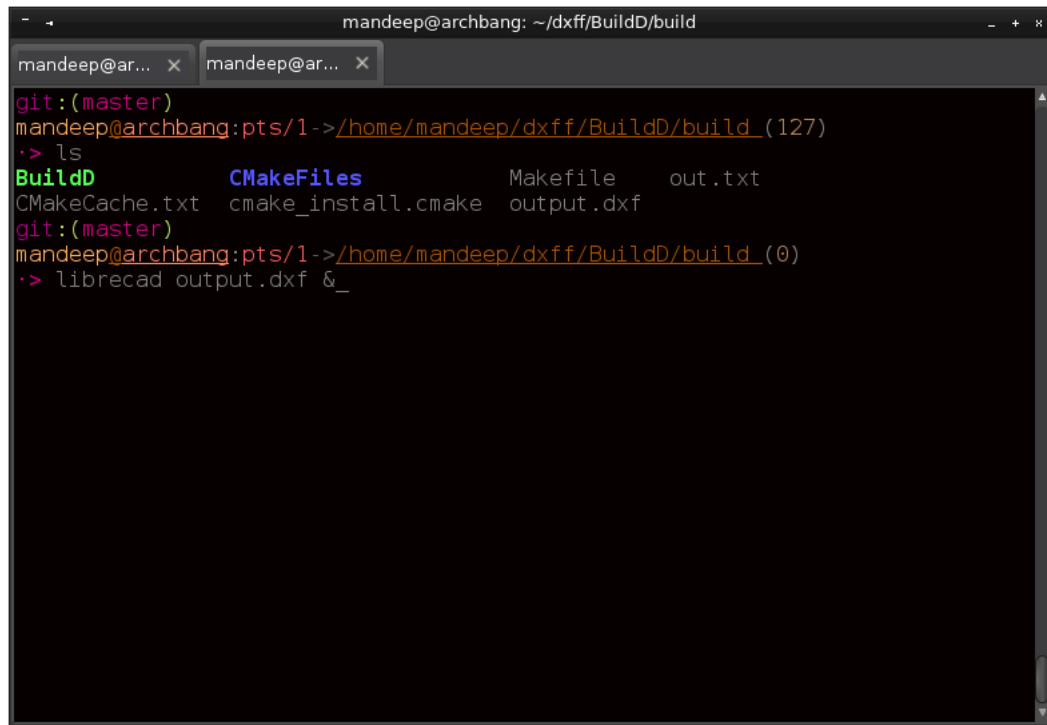


Figure 3.5: Opening DXF in LibreCAD from command-line

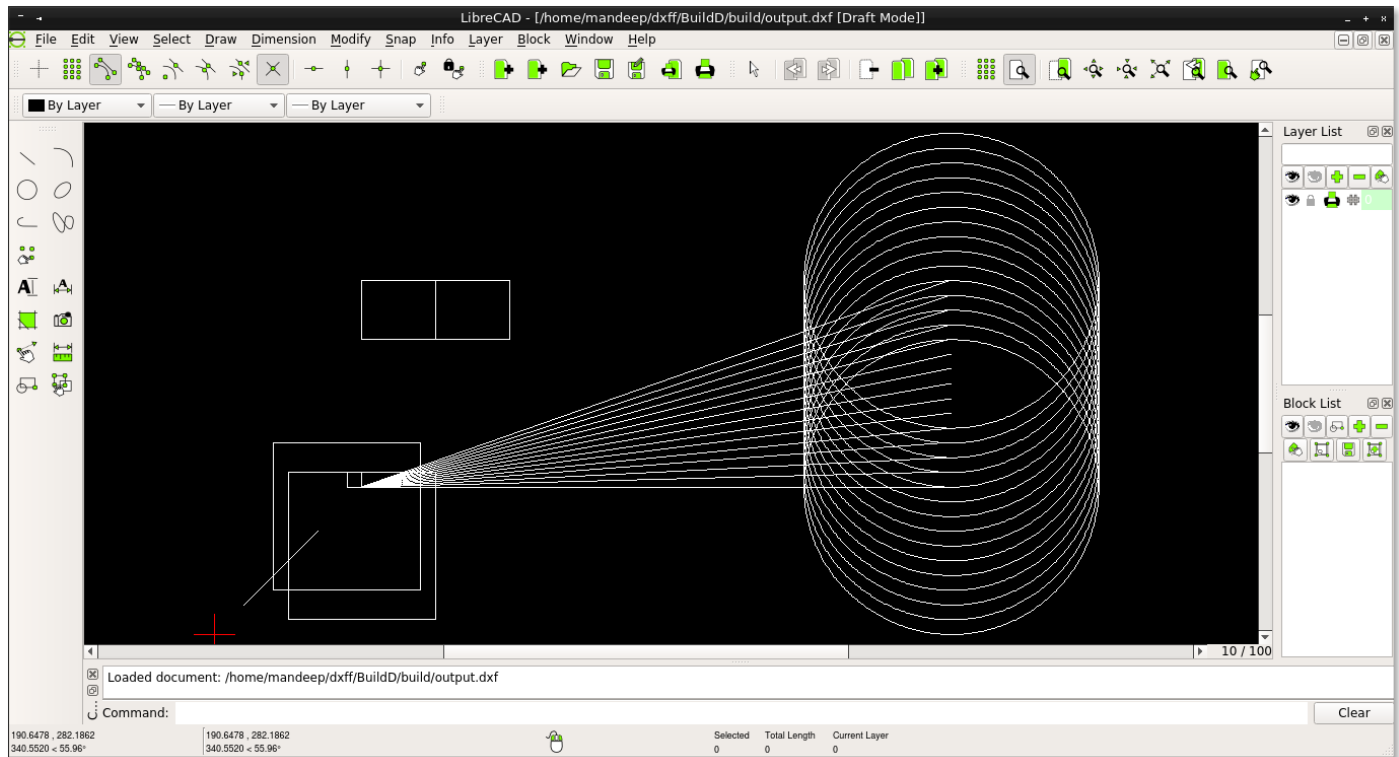


Figure 3.6: DXF opened in LibreCAD

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 created 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

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 create a two dimensional 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.

- [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/>
- [8] Autodesk DXF Reference, <http://www.autodesk.com/techpubs/autocad/acad2000/dxf>