VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Belagavi – 590 018

A
Project Report
On

"IMPLEMENTATION OF STACK AND QUEUE"

Submitted in partial fulfilment of Bachelor of Engineering Degree
In
COMPUTER SCIENCE AND ENGINEERING
VI Semester, **17CSL68 – Computer Graphics & Visualization Lab**

Submitted by

Gaurav Singh	1HK17CS047
Himanshu Sekhar Bhuyan	1HK17CS054
Junaid Ahmed Baig	1HK17CS060

Under the guidance of
Pushpa T
Assistant Professor, Department of Computer Science & Engineering
MAY 2020

Department of Computer Science and Engineering

HKBK COLLEGE *of* ENGINEERING
(Approved by AICTE & Affiliated to VTU)
Nagawara, Arabic College Post, Bangalore-45, Karnataka
Email: info@hkbk.edu.in URL: www.hkbk.edu.in

HKBK COLLEGE *of* ENGINEERING

Nagawara, Bangalore–560 045 Approved
by AICTE & Affiliated to VTU

Department of Computer Science and Engineering

Certificate

Certified that the Project Work entitled **"Implementation of Stack and Queue",** carried out by **Gaurav Singh (1HK17CS047), Himanshu Sekhar Bhuyan (1HK17CS054) and Junaid Ahmed Baig (1HK17CS060)** are bonafide students of **HKBK COLLEGE *of* ENGINEERING**, in partial fulfilment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University**, Belgaum, during the year 2019–120. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of **17CSL68– Computer Graphics & Visualization Lab** prescribed for the said Degree.

Pushpa T
Assistant Professor
Guide

Dr. Loganathan R
Professor & HOD

External Viva

Name of the Examiners	Signature with Date
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

I would like to express my regards and acknowledgement to all who helped me in completing this project successfully.

First of all I would take this opportunity to express my heartfelt gratitude to the personalities, **Mr. C M Ibrahim**, Chairman, HKBKGI and **Mr. C M Faiz Mohammed**, Director, HKBKGI for providing facilities throughout the course.

I express my sincere gratitude to the Principal, HKBKCE for his support and which inspired me towards the attainment of knowledge.

I consider it as great privilege to convey my sincere regards to **Dr. Loganathan. R.**, Professor and HOD, Department of CSE, HKBKCE for his constant encouragement throughout the course of the project.

I would specially like to thank my guide, **Prof. Pushpa T**, Assistant Professor, Department of CSE, HKBKCE for his vigilant supervision and his constant encouragement. He spent his precious time in reviewing the project work and provided many insightful comments and constructive criticism.

Finally, I thank Almighty, all the staff members of CSE Department, my family members and friends for their constant support and encouragement in carrying out the project work.

Gaurav Singh **[1HK17CS047]**

Himanshu Sekhar Bhuyan **[1HK17CS054]**

Junaid Ahmed Baig **[1HK17CS060]**

ABSTRACT

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Using computer graphics, we can create images by computers that are indistinguishable from photographs of real objects. We are implementing our project using a particular graphics software system that is, OPENGL. OpenGL is strictly defined as “a software interface to graphics hardware.” In essence, it is a 3D graphics and modelling library that is extremely portable and very fast. Using OpenGL, you can create elegant and beautiful 3D graphics with nearly the visual quality. OpenGL is intended for use with computer hardware that is designed and optimized for the display and manipulation of 3D graphics. OpenGL is used for a variety of purposes, from CAD engineering and architectural applications to computer-generated dinosaurs in blockbuster movies.

This aim of this project is to show the Implementation of Stack and Queue using a numbers as an element. The stack is a data structure which works on LIFO (Last In First Out) technique and queue is a data structure which works on FIFO (First In First Out). The numbers are pushed or popped in the stack while in the queue the numbers are inserted or deleted. When there is no element in the stack the message is shown as “Stack is Empty” and in queue it shows “No elements to display in queue”.

Table Of Contents

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Overview

The power and utility of computer graphics is widely recognized, and a broad range of graphics hardware and soft-ware systems are now available for applications in virtually all fields. Graphics capabilities for both two-dimensional and three- dimensional applications are now very common.

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

This report contains "Implementation of Stack and Queue" using a set of OpenGL functions. This project consists a view about the working of stack and queue. We are mainly using the mouse as interface to use the buttons to interact with the main window. The objects are drawn using GLUT functions. This project has been developed using Ubuntu 20.4 with OpenGL Package.

1.2 Content and Organization of the Report

There are six chapters included in this report, a brief idea behind these chapters can be summarized as follows:

Chapter-1: This chapter consists of introduction part of the report which gives brief idea about the concept of project along with a brief description of the all the chapters included in this report.

Chapter-2: This chapter specifies the requirements of the software and hardware required by the program to run along with a description of the environment where this project will work successful.

Chapter-3: This chapter gives the flow of program that is, how the functions are called and executed in the form of flow chart to specify the diagrammatic view.

Chapter-4: This chapter gives the overview of how the project is implemented and how the program will work with respect to the functions called. It specify the libraries used in the program along with the functions that form the program with its description in the form of comments.

Chapter-5: This chapter contains snapshots of the program output that is, the view of output what we will get when the program is executed.

Chapter-6: The last chapter contains the conclusion part of the report along the information about the future use or modifications of the project.

1.3 Objective

This project is basically a simulation of how the stack and queue works. The field of computer graphics has already got to a stage of maturity so that its applications are being widely used. Therefore, the underlying education in this field heavily depends on the goals and skills of the corresponding area of application. The inclusion of computer graphics education in Informatics Engineering courses is obviously mandatory. The results of some succeeded experiments are referred, along with the most relevant relationships with other disciplines. Important questions regarding the future of computer graphics education in the scope of Informatics Engineering are also considered.

CHAPTER 2
SYSTEM REQUIREMENTS AND SPECIFICATIONS

CHAPTER 2

SYSTEM REQUIREMENTS AND SPECIFICATIONS

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external 5ehavior of the system.

2.1 Functional Requirements

Functional Requirements define the internal working of the software. The following conditions must be taken care of: The ability to perform correct operations when corresponding keys are pressed.

2.2 Non-functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviours. This should be contrasted with functional requirements that specify specific 5ehavior or functions. Typical non-functional requirements are reliability and scalability. Non-functional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

Types of non-functional requirements

Volume: Volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.

Reliability: System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.

Security: The security of the system (it’s ability to resist attacks) is a complex property that cannot be easily measured. Attacks maybe devised that were not anticipated by the system designers and the may use default built-in safeguards.

Reparability: This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.

Usability: This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment

2.3 Software Requirements

Operating System	: Ubuntu
Front End	: Terminal
Coding Language	: C++
Library	: OpenGL

2.4 Hardware Requirements

System	: Pentium IV 2.4 GHz or above
Hard Disk	: 40 GB or above
Monitor	: 15 VGA color
RAM	: 256 MB or above

2.5 Introduction to Environment

OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data. In this situation, the computer on programs can work across a network even if the client and server are different kinds of computers. If an OpenGL program isn't running which your program runs and issues OpenGL drawing commands is called the client, and the computer that receives those commands and performs the drawing is called the server. The format for transmitting OpenGL commands (called the protocol) from the client to the server is always the same, so OpenGL across a network, then there's only one computer, and it is both the client and the server.

CHAPTER 3
DESIGN

CHAPTER 3

DESIGN

Requirement analysis encompasses all the tasks that go into the instigation, scoping and definition of a new or altered system. Requirement analysis is an important part of the design process. Here we identify the needs or requirements. Once the requirements have been identified the solution for the requirements can be designed. We a design a project with specific goals, tasks and outcomes. The more specific and the more closely aligned with traditional instructional objectives, the better.

The project “Implementation of Stack and Queue” is meant as a source to understand the working of the stack and queue. This project helps the students to get a clear idea how the stack and queue work.

3.1 High Level Design

High Level Design (HLD) is the overall system design – covering the system architecture and database design. It describes the relation between various modules and functions of the system. Data flow, flow charts and data structures are covered under HLD. OpenGLs basic operation is to accept primitives such a points, lines and polygons and convert them into pixels. This is done by graphics pipeline known as the OpenGL state machine.

The header files used are :

1. #include<stdlib.h>: This is C library function for standard input and output.

2. #include<GL/glut.h>: This header is included to read the glut.h, gl.h and glu.h

The different modules used in this project are as follows:

1. glutInit(int *argcp, char **argv);

argcp

A pointer to the program’s unmodified argc variable from main. Upon return, the value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library.

Argv

The program’s unmodified argv variable from main. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

2. glutInitDisplayMode(unsigned int mode);

mode: Display mode, normally the bitwise OR-ing of GLUT display mode bit masks. See values below:

GLUT_RGBA

Bit mask to select an RGBA mode window. This is the default if neither GLUT_RGBA nor GLUT_INDEX are specified.

GLUT_RGB

An alias for GLUT_RGBA.

GLUT_DOUBLE

Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.

GLUT_DEPTH

Bit mask to select a window with a depth buffer.

3. glutInitWindowSize(int width, int height);

Parameters

1. **width**: Width of future windows.
2. **height**: Height of future windows.

4. glutBitmapCharacter(void *font, int character);

Usage

1. **font**: Bitmap font to use.
2. **character**: Character to render (not confined to 8 bits).

glutBitmapCharacter renders a bitmap character using OpenGL.

5. glMatrixMode (Glenum mode);

mode: Specifies which matrix stack is the target for subsequent matrix operations. These values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE. The initial value is GL_MODELVIEW.

6. glLoadIdentity(void);

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix

3.2 User interface:

Mouse Interface

This project uses the mouse to click on the buttons. Mouse is used as interface between the user and the system. It helps in implementing the user input.

CHAPTER 4
IMPLEMENTATION

CHAPTER 4

IMPLEMENTATION

Implementation is the process of putting a decision into effect. It is an act or instance of implementing something that is, the process of making something active or effective. Implementation must follow any preliminary thinking in order for something to actually happen. In the context of information technology, implementation encompasses the process involved in getting new software or hardware to operate properly in its environment. This includes installation, configuration, running, testing and making necessary changes. The word deployment is sometime used to mean the same thing.

4.1 Libraries Used

OpenGL function names begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in Windows). The first is the **OpenGL Utility Library (GLU).** This library uses only GL functions but contains code for creating common objects, such as spheres, and other tasks that users prefer not to write repeatedly. The GLU library is available in all OpenGL implementations. The second library addresses the problems of interfacing with the window system. We use a readily available library called the **OpenGL Utility Toolkit (GLUT)** that provides the minimum functionality that should be expected in any modern windowing system. The GLX library provides the minimum “glue” between OpenGL and the Xwindow System. GLX is used by GLUT, and thus this library and various others are called from the OpenGL libraries, but the application program does not need to refer to these libraries directly. Strings such as GL_FILL and GL_POINTS are defined in header (.h) files. In most implementations, one of the “include” lines.

4.2 User Defined Functions

OpenGL function names begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in Windows). The first is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects, such as spheres, and other tasks that users prefer not to write repeatedly. The GLU library is available in all OpenGL implementations. The second library addresses the problems of interfacing with the window system. We use a readily available library called the OpenGL Utility Toolkit (GLUT) that provides the minimum functionality that should be expected in any modern windowing system. The GLX library provides the minimum “glue” between OpenGL and the Xwindow System. GLX is used by GLUT, and thus this library and various others are called from the OpenGL libraries, but the application program does not need to refer to these libraries directly. Strings such

as GL_FILL and GL_POINTS are defined in header (.h) files. In most implementations, one of the “include” lines.

4.2.1 Stack Function

4.2.1.1 Stack full

```
int stack::stfull()
{
        if (st.top >= size-1)
        return 1;
        else
        return 0;
}
```

4.2.1.2 Stack Empty

```
int stack::stempty()
{
        if (st.top == -1)
        return 1;
        else
        return 0;
}
```

4.2.1.3 Stack Push

```
void stack::push(int item)
{
        char str[10];
        snprintf(str, sizeof(str), “%d”, item);
        button btn(100,250+st.top*50,150,300+st.top*50,str);
        st.top++;
        st.s[st.top] = btn;
}
```

4.2.1.4 Stack Pop

```
button stack::pop()
```

```
{
        button item;
        item = st.s[st.top];
        st.top--;
        return (item);
}
```

4.2.1.5 Stack Display

```
void stack::displaystack()
{
        int I;
        if (st.stempty())
        drawstring(10,10,"Stack Is Empty!");
        else if (st.stfull())
        drawstring(10,10,"Stack Is Full!");
        else
        {
                for (I = st.top; I >= 0; i--)
                st.s[i].draw();
        }
}
```

4.2.2 Queue Functions

4.2.2.1 Queue Insert

```
void queue::insert_element()
{
        static int num=0;
        char str[10];
        snprintf(str, sizeof(str), "%d", num++);
        button btn(300,250+rear*50,350,300+rear*50,str);
        if(front==0 && rear==MAX-1)
        drawstring(10,10," Queue OverFlow I");
        else if(front==-1 && rear==-1)
        {
                front=rear=0;
```

```
                que[rear]=btn;
        }
        else if(rear==MAX-1 && front!=0)
        {
                rear=0;
                que[rear]=btn;
        }
        else
        {
                rear++;
                que[rear]=btn;
        }
}
```

4.2.2.2 Queue Delete

```
void queue::delete_element()
{
        button element;
        if(front==-1)
        {
                drawstring(300,10,"Underflow");
                return;
        }
        element=que[front];
        if(front==rear)
        front=rear=-1;
        else
        {
                if(front==MAX-1)
                front=0;
                else
                front++;
                //printf("\n The deleted element is: %s",element.str);
        }
}
```

4.2.2.3 Queue Display

```
void queue::displayqueue()
{
        int I;
        if(front==-1)
        {
                drawstring(300,10,"No elements to display in queue");
                return;
        }
        for(i=front;i<=rear;i++)
        {
                que[i].draw();
        }
        //printf("\n The queue elements are:\n ");
}
```

4.2.3 Displaying Stack and Queue

```
void displaystacknqueue()
{
        st.displaystack();
        q.displayqueue();
}
void display()
{
        glClearColor(1.0,1.0,1.0,0.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        btn1.draw();
        btn2.draw();
        btn3.draw();
        btn4.draw();
        displaystacknqueue();
        glFlush();
        glutSwapBuffers();
        //glutPostRedisplay();
}
```

```
void mouse(int btn, int state, int x, int y)
{
	static int itemno=0;
	y=600-y;
	if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		if(btn1.insidebutton(x,y))
		{
			btn1.togglestate();
			if(!st.stfull())
			st.push(itemno++);
		}
		if(btn2.insidebutton(x,y))
		{
			btn2.togglestate();
			if(!st.stempty())
			st.pop();
		}
		if(btn3.insidebutton(x,y))
		{
			btn3.togglestate();
			q.insert_element();
		}
		if(btn4.insidebutton(x,y))
		{
			btn4.togglestate();
			q.delete_element();
		}
	}
	if(btn==GLUT_LEFT_BUTTON && state == GLUT_UP)
	{
		if(btn1.insidebutton(x,y))
		{
			btn1.togglestate();
		}
```

```
            if(btn2.insidebutton(x,y))
            {
                btn2.togglestate();
            }
            if(btn3.insidebutton(x,y))
            {
                btn3.togglestate();
                        }
            if(btn4.insidebutton(x,y))
                        {
                            btn4.togglestate();
                        }
        }
        glutPostRedisplay();
}
```

4.3 Flow chart

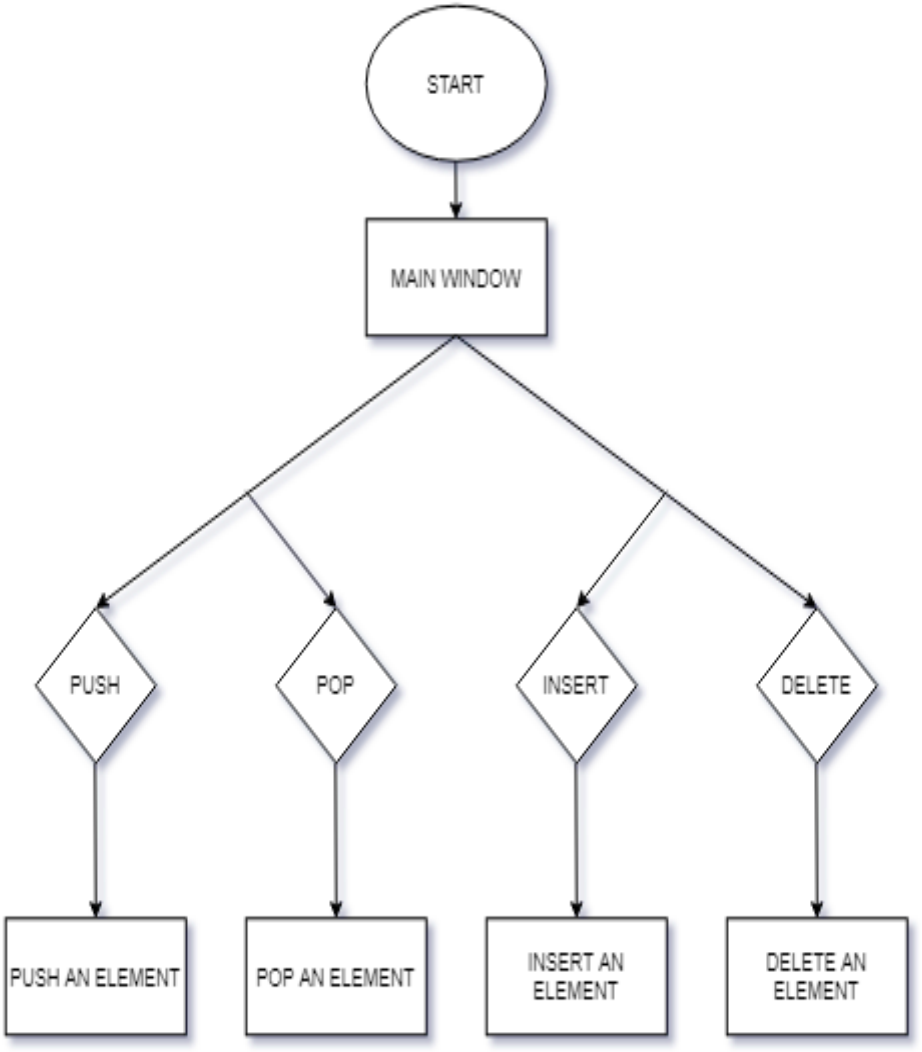

Figure 4.1 Flowchart of the Program

Chapter 5
SNAPSHOTS

CHAPTER 5

SNAPSHOTS

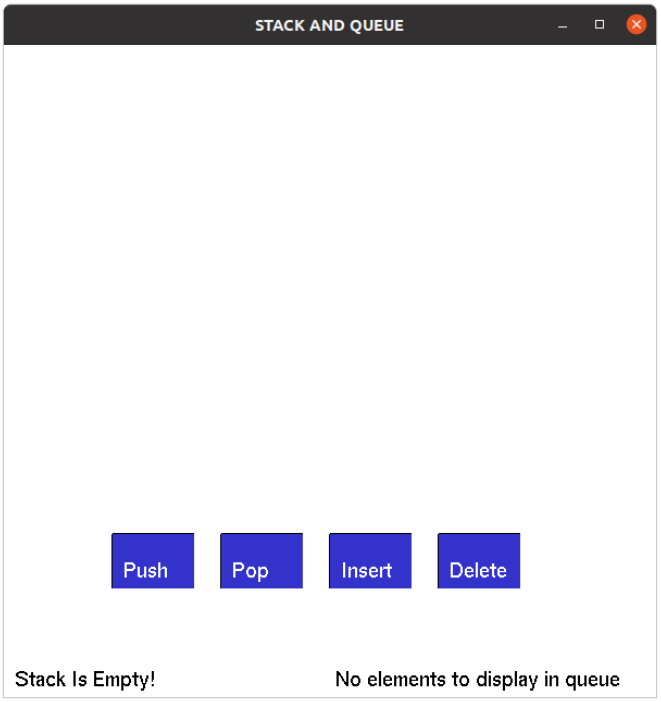

Figure 5.1 Main Window

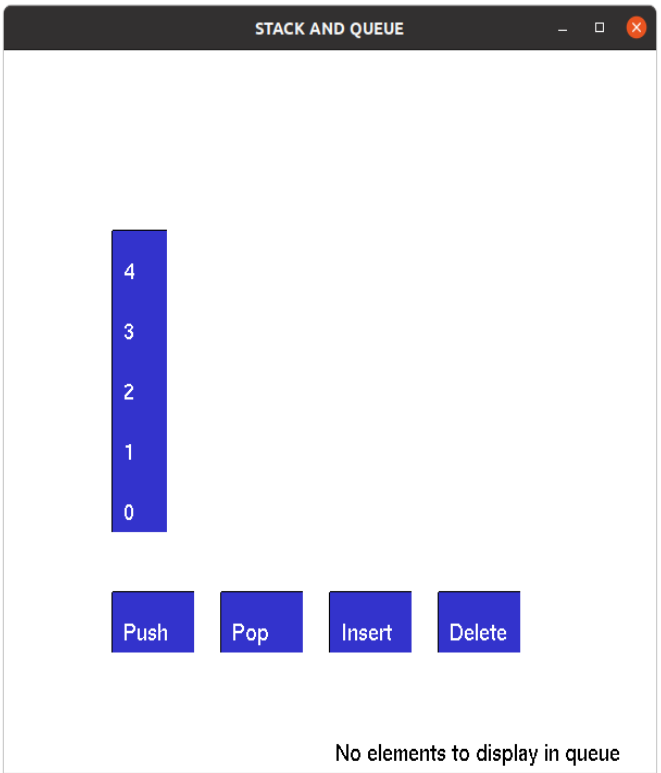

Figure 5.2 Pushing the elements in the Stack

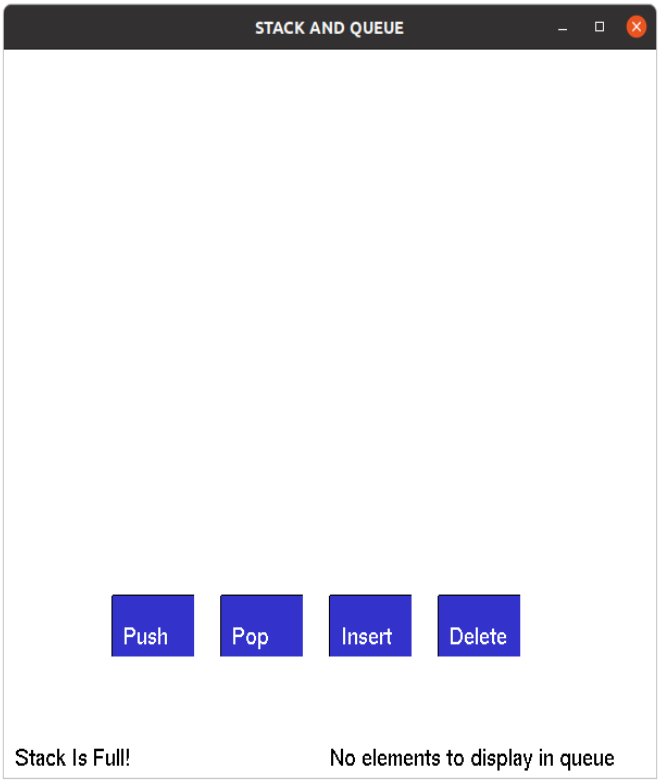

Figure 5.3 Stack is full

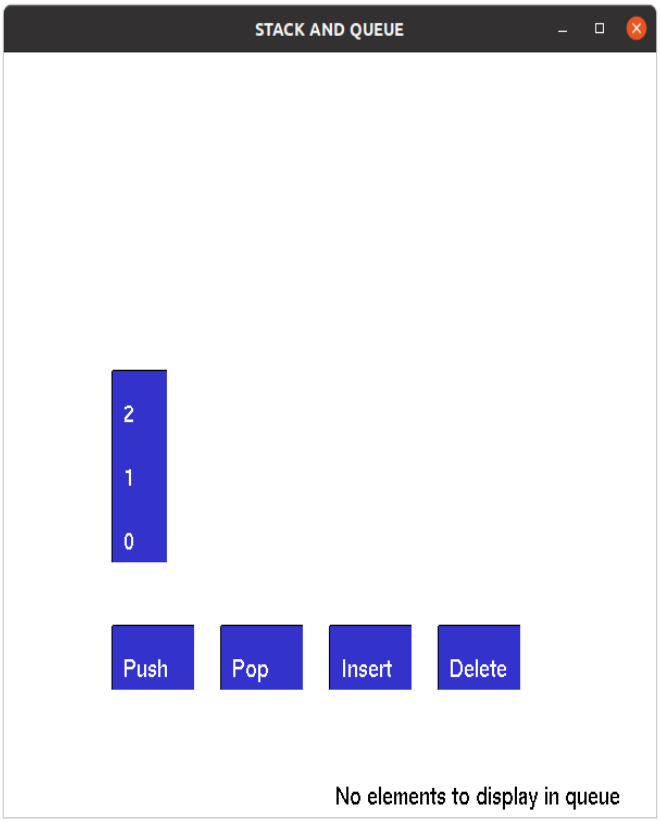

Figure 5.4 Popping out the elements from the Stack

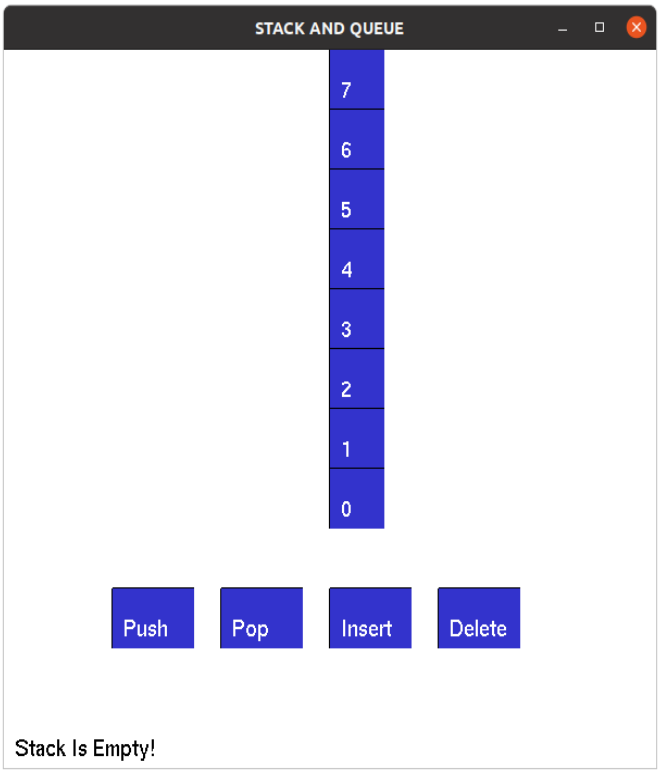

Figure 5.5 Inserting the elements in the Queue

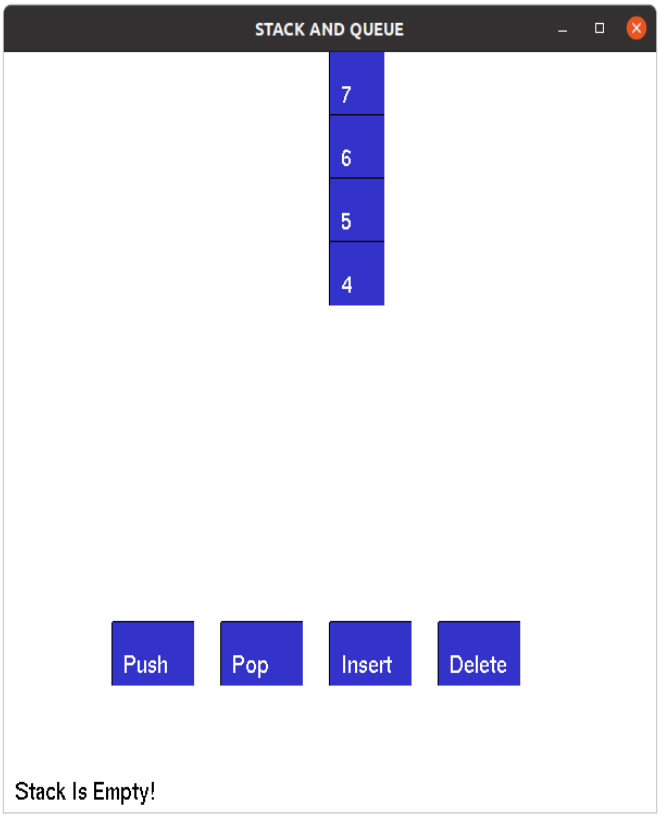

Figure 5.6 Deleting the elements from the Queue

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

Our project “Implementation of Stack and Queue” encompasses a broad range of OpenGL functionalities.

We have made use of multiple concepts of OpenGL that we have learnt theoretically.

This program uses Mouse events to implement all the options provided that is, the push, pop, insert and delete button provided in the main window.

The goal of this project was to get a first-hand experience of OpenGL programming. This was achieved by the successful completion of the project.

Thus this program was effectively implemented as required by the user software requirements. The program is user friendly and provides the necessary options to the user whenever required. The software is developed with a simple interface.

6.2 Scope for future enhancement

- This design can be used in a real-time scenario for further enhancements of the project.
- The design can be used as a teaching tool in educational institutes to understand the working of animation in OpenGL.
- The project can be further enhanced by implementing a three dimensional view.

BIBLIOGRAPHY

Books:

1. *Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3 / 4 Edition, Pearson Education,2011*
2. *Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008*
3. *James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education*
4. *Xiang, Plastock : Computer Graphics , sham's outline series, 2nd edition, TMG.*
5. *Kelvin Sung, Peter Shirley, steven Baer : Interactive Computer Graphics, concepts and applications, Cengage Learning*
6. *M M Raiker, Computer Graphics using OpenGL, Filip learning/Elsevier*

Websites:

- www.OpenGL.org/Redbook
- www.OpenGL.org/simpleexamples
- http://www.opengl.org/sdk/docs/tutorials/CodeColony
- www.Google.co.in
- www.youtube.com