ACKNOWLEDGEMENT

We dedicate this page to acknowledge and thank those responsible for the shaping of the project. Without their guidance and help, the experience while constructing the dissertation would not have been so smooth and efficient.

We are extremely thankful to our **Principal Dr Name.** for his support and encouragement.

We owe our profound gratitude to **Dr Name, Head of the Department**, Computer Science and Engineering, whose kind consent and guidance helped us to complete this work successfully.

We sincerely thank **Mrs Name, Mr. Name, Assistant Professor**, Computer Science and Engineering, for their guidance and valuable suggestions which helped us to fulfill the experiments prescribed by the university.

We would like to thank all our Computer Science and Engineering staff members who have always been with us extending their support, precious suggestions, guidance and encouragement through the project.

We also like to extend thanks to our friends and family members for their continuous support.

ABSTRACT

The main aim of the Circular linked list Computer Graphics Mini project is to illustrate the concepts and usage of pre-built functions in OpenGL.

Simulation of a circular linked list is being done using computer graphics. The circular linked list is a linked list in which the link field of the last node contains the address of first node. The operations that are performed on circular linked list are insertion, deletion, and display. In a circular linked list, it is possible to reach any node from any other node. Each node contains two parts fields: the data and a reference to the next node. The first part contains the information and the second part contains the address of the next node in the list. This is also called the link field. It is a linear data structure which contains a group of nodes and the nodes are sequentially arranged. Elements can be inserted anywhere in the linked list and any node can be deleted.

We are using OpenGL tools to implement this project in a visualized mode. To display the node two simple rectangle will be drawn. These are linked with the line to other elements in the list. Different menu will be given to user to choose the operation on the circular linked list. We have used input devices like mouse and keyboard to interact with the program.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics insensible. In today's life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures. Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

1.2 OpenGL

OpenGL (Open Graphics Library) is a cross-platform, hardware-accelerated, language independent, industrial standard API for producing 3D (including 2D) graphics. Modern computers have dedicated GPU (Graphics Processing Unit) with its own memory to speed up graphics rendering. OpenGL is the software interface to graphics hardware.

We use 3 sets of libraries in our OpenGL programs :

1. Core OpenGL (GL): consists of hundreds of commands, which begin with a prefix "gl" (e.g., glColor, glVertex, glTranslate, glRotate). The Core OpenGL models an object via a set of geometric primitives such as point, line and polygon.

2. OpenGL Utility Library (GLU): built on-top of the core OpenGL to provide important utilities (such as setting camera view and projection) and more building models (such as qradric surfaces and polygon tessellation). GLU commands start with a prefix "glu" (e.g., gluLookAt, gluPerspective).

3. OpenGL Utilities Toolkit (GLUT): OpenGL is designed to be independent of the windowing system or operating system. GLUT is needed to interact with the Operating System (such as creating a window, handling key and mouse inputs).

1.3 Overview of Project

This project is about the simulation of circular list visualizer. We are implementing it using different OpenGL primitives, OpenGL libraries and combining them together in a required manner. It highlights the key feature of a data structure and its high quality of efficiency that is obtained on its usage in the application program. This project consists of basic operations such as insertion, deletion, and display which are constructed by using graphics primitives available in OpenGL library. It illustrates the role of different callback functions that provides easier way to accomplish our project in an effective manner.

The basic operations that are performed on circular linked list :

1.**Insertion at beginning :** Inserting a new node as the first node. The next pointer of last will point to this node and this new node will point to the previous first node.

2.**Insertion at the end :** Inserting a new node as the last node. The next pointer of last will point to this node and this new node will point to the first node.

3.**Deletion at beginning :** Deleting the first node of linked list. For this, the next pointer of last will point to the second node of the linked list.

4.**Deletion at the end :** Deleting the last node of the linked list. For this, the second last node will point to the first node of the list.

5.**Display :** Displays all the elements in the linked list.

1.4 Problem Statement

In this project, we will write a program to represent the circular linked list and some operations that can be performed on them. The linked list uses dynamic memory, allocation. Elements can be inserted anywhere in the linked list and each node contains two parts. The left part of the node represents the information part of the node and the right part represents the next pointer field that contains the address of the next node. During insertion, the array elements are shifted into higher order memory locations and during deletion, elements are shifted into lower order memory locations. In addition, provides a menu to choose the operation and ask the user to type the input. This also uses input devices like mouse and keyboard to interact with the program. This project helps us to understand the concept of circular linked list.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification specifies the requirements required to run the given application. The detailed explanation of each type of requirement is given below.

2.1 Functional Requirement

- The system displays the information page with one message.
- User should right click on the message by using a mouse in order to perform the operations on circular linked list.
- The user should be able to understand the operations performed by the circular linked list.

2.2 Hardware Requirement

- PROCESSOR: Intel®core™2Duo
- SPEED: 2.10GHz
- RAM: 2.00GB Minimum
- SPACE ON DISC: 20GB Minimum

2.3 Software Requirement

- Operating System : Window 10
- C/C++ Platform : Code Blocks
- OpenGL Library : GLUT

2.3.1 Code Blocks

Through the aid of a compiler, a program written in a computer language, such as C++, is turned into machine code, which is executed on the computer. It may take several rewrites of code to get the program to work correctly. The purpose of this manual is to help the student develop the skills to organize program coding and develop sound techniques for finding and isolating errors. This is called debugging the program. Automatic tools have also been developed to help you trace programs that you have written and will be an important tool as your programs become more complex. This type of tool is called a debugger.

CHAPTER 3

SYSTEM DESIGN

Software design is the process by which an agent creates specification of software artifact, intended to accomplish goals, using the set of primitive components and subject to constraints.

3.1 Design of Circular Linked List

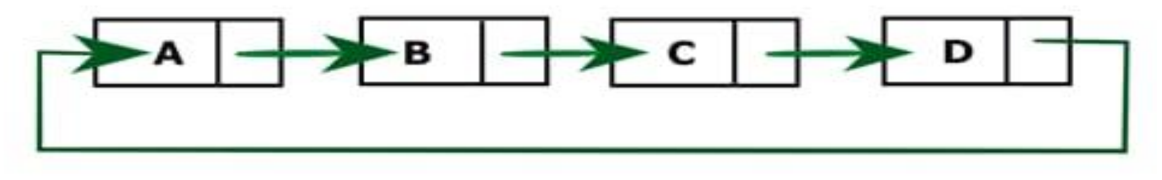

Fig 3.1: Design of circular linked list

The circular linked list is a kind of linked list. First thing first, the node is an element of the list, and it has two parts that are, data and next. Data represents the data stored in the node and next is the pointer that will point to next node. Head will point to the first element of the list, and tail will point to the last element in the list.

The diagram shown above depicts a circular linked list. Node A represents head and node D represents tail. So, in this list, A is pointing to B, B is pointing to C and C is pointing to D but what makes it circular is that node D is pointing back to node A.

3.2 Flow Chart of Circular Linked List

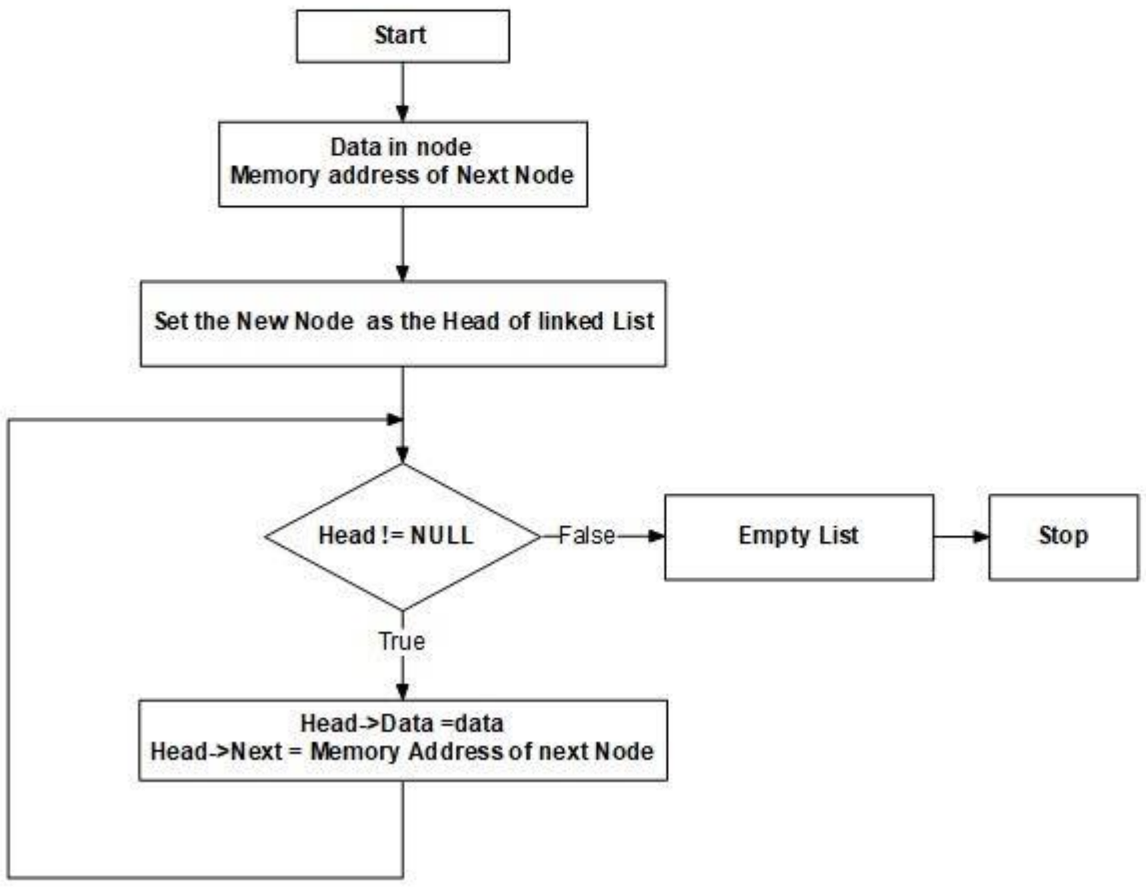
Start
Data in node
Memory address of Next Node
Set the New Node as the Head of linked List
Head != NULL
False
Empty List
Stop
True
Head->Data =data
Head->Next = Memory Address of next Node

Fig 3.2: Flow Chart

3.3Algorithm

- Include all the header files which are used in the program and declare all the user defined functions. Define a Node structure with two members data and next.
- Define a Node pointer 'head' and set it to NULL. It checks whether the head is null, then it will insert the node as head.
- If the head is not null, then the new node will be the new last, and the new last will point to the head as it is a circular linked list.
- Display will show all the nodes present in the list.
- Define a new node 'current' that will point to the head.
- Print current and data till current will points to head.
- Current will point to the next node in the list in each iteration.
- Implement the main method by displaying operations menu and make suitable function calls in the main method to perform user selected operation.

CHAPTER 4

IMPLEMENTATION

4.1 Description of Implementation Modules

In this project we have created “OpenGL” functional API. We have taken the help of built-in functions present in the header file. To provide functionality to our project we have written sub functions. These functions provide us the efficient way to design the project. In this chapter we are describing the functionality of our project using these functions.

4.2 List of Implementation Functions

void drawstring() :

This function is used to print strings on rendered window. It contains four arguments-x, y, z coordinates and a string, while the coordinate assumed to be the space where string to be placed and the string is the value to be display.

void cll_menu() :

This is the option menu used for operations.

void init() :

This function is used to initialize window properties.

glClearColor() :

This function call sets the present RGBA clear color used when clearing the color buffer. The RGB stands for Red, Green, and Blue.

OpenGL Tool Kit(GLUT) :

This provides support to interact with the Operating System (such as creating a window, handling key and mouse inputs) and more building models (such as sphere and torus).

glRasterPos() :

Specify the raster position for pixel operations.

glutBitmapCharacter() :

Automatically sets the OpenGL unpack pixel storage modes it needs appropriately and saves and restores the previous modes before returning. The generated call to glBitmap will adjust the current raster position based on the width of the character.

void myReshape ( ):

This function is used to change the position of an object.

void keys( ):

This function is used to set key to perform desired operations.

void mouse( ):

This function is used to set mouse to perform desired operations

4.3 Description of the functions

main():
The execution of the program starts from the main ().

glutInit ():

Initializes GLUT. The argument from main are passed in and can be used by the application.

glutInitWindowSize():

Specifies the initial height and width of the window in pixels.

glutInitWindowPosition() :

This function is used to specify the screen location for the upper left corner of window.

glutCreateWindow():

Creates the window on the display. The string can be used to label the window.

glClear() :

This function is used to clear the window. As the algorithm stores information depth buffer, so it is necessary to clear the buffer whenever we wish to redraw the display.

glutDisplayFunc():

Registers the display function that is executed when the window needs to redraw.

glutInitDisplayMode() :

This function should be called next as this function specifies the display mode for window.

glutMainLoop():

Causes the program to enter an event processing loop.

glTranslate() :

This function multiplies the current matrix by a matrix that translates an object by given x, y and z direction.

glutSwapBuffer() :

This function promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. Contents of back buffer then becomes undefined.

myinit():

This function is defined to initialize the window parameters.

4.4 Code snippets

4.4.1 Implementation of main window

```
int main(int argc, char** argv)
{
   printf("                    CIRCULAR LINKED LIST                    \n\n");
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(640, 480);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("Circular Linked List Visualizer");
   glutCreateMenu(cll_menu);
   glutAddMenuEntry("Insert at  Front", 1);
   glutAddMenuEntry("Insert  at Rear", 2);
   glutAddMenuEntry("Delete  from Front", 3);
   glutAddMenuEntry("Delete  from Rear", 4);
   glutAddMenuEntry("Exit", 5);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
   glutDisplayFunc(display);
   init();
   glutMainLoop();
   return 0;
}
```

4.4.2 Implementation of menu function

```
void cll_menu(int choice)
{
   switch(choice)
   {
      case 1: printf("Enter the item to be inserted: ");
              scanf("%d", &item);
              first = insert_front(item, first);
              count++;
              display();
              break;
```

```
        case 2: printf("Enter the item to be inserted: ");
                scanf("%d", &item);
                first = insert_rear(item, first);
                count++;
                display();
                break;
        case 3: succ = 1;
                first = delete_front(first);
                count--;
                display();
                break;
        case 4: succ = 1;
                first = delete_rear(first);
                count--;
                display();
                break;
        case 5: exit(0);
    }
}
```

4.4.3 Implementation of init function

```
void init()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(1.0, 1.0, 1.0, 0.0);
        glColor3f(0.0, 1.0, 1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, 640.0, 0.0, 480.0);
        glMatrixMode(GL_MODELVIEW);
}
```

4.4.4 Implementation of drawstring function

```
void drawstring(float x, float y, char *string, int col, int drawstringflag)
{
```

```
            char *c;
            glColor3fv(color[col]);
            glRasterPos2f(x, y);
            if (drawstringflag == 0)
            {
                        for(c = string; *c != '\0'; c++)
                        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
            }
            else if (drawstringflag == 1)
            {
                        for(c = string; *c != '\0'; c++)
                        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, *c);
             }
             else
            {
                       for(c = string; *c != '\0'; c++)
                       glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
             }
}
```

4.4.5 Implementation of display function

```
void display()
{
    if(titleflag)
    {
        title();
        titleflag = 0;
    }
    int i = 0, j = 0, k = 0, b[20];
    NODE temp;
    glClearColor(192.0, 192.0, 192.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.6);
    drawstring(200.0, 450.0, (char *)"CIRCULAR LINKED LIST", 1, 0);
```

```
if(first==NULL)
{
        if(prinrflag)
        {
                    printf("\nEmpty List\n");
                    printflag = 0;
        }
}
temp = first;
while(temp != NULL)
{
    for(k = 0;k < count;k++)
    {
        GLfloat x1 = 50, y1 = 200, x2 = 80;
        GLfloat x3 = 110, y3 = 250, x4 = 95, y4 = 225;
        GLfloat x5 = 160, y5 = 210, y6 = 240, x6 = 180;
        GLfloat y0=150,x0=30,x00=40;
        int pos = 130;
        if(temp->link == NULL)
        {
                glColor3f(0.8, 0.2, 0.70);
                glBegin(GL_POLYGON);
                glVertex2i(x1+k*pos, y1);
                glVertex2i(x1+k*pos, y3);
                glVertex2i(x2+k*pos, y3);
                glVertex2i(x2+k*pos, y1);
                glEnd();
                glColor3f(0.0, 0.60, 0.80);
                glBegin(GL_POLYGON);
                glVertex2i(x2+k*pos, y1);
                glVertex2i(x2+k*pos, y3);
                glVertex2i(x3+k*pos, y3);
                glVertex2i(x3+k*pos, y1);
                glEnd();
                glColor3f(1.0,0.0,0.0);
                glBegin(GL_LINES);
                glVertex2i(x4+k*pos, y4);
                glVertex2i(x5+k*pos, y4);
                glVertex2i(x5+k*pos, y4);
                glVertex2i(x5+k*pos, y0);
                glVertex2i(x5+k*pos, y0);
                glVertex2i(x0,y0);
                glVertex2i(x0, y0);
                glVertex2i(x0, y4);
                glVertex2i(x0, y4);
                glVertex2i(x1, y4);
                glEnd();
                glBegin(GL_LINES);
                glVertex2i(x00, y6);
                glVertex2i(x1, y4);
                glVertex2i(x1, y4);
                glVertex2i(x00, y5);
                glEnd();
        }
        else
        {
                glColor3f(1.0f, 0.0f, 0.0f);
                glBegin(GL_LINES);
                glVertex2i(x4+k*pos, y4);
                glVertex2i(x6+k*pos, y4);
                glEnd();
                glColor3f(1.0f, 0.0f, 0.0f);
                glBegin(GL_LINES);
                glVertex2i(x5+k*pos, y6);
                glVertex2i(x6+k*pos, y4);
                glVertex2i(x6+k*pos, y4);
                glVertex2i(x5+k*pos, y5);
                glEnd();
```

```
                glColor3f(0.0, 0.0, 0.70);
                glBegin(GL_POLYGON);
                glVertex2i(x1+k*pos, y1);
                glVertex2i(x1+k*pos, y3);
                glVertex2i(x2+k*pos, y3);
                glVertex2i(x2+k*pos, y1);
                glEnd();
                glColor3f(0.0, 0.60, 0.80);
                glBegin(GL_POLYGON);
                glVertex2i(x2+k*pos, y1);
                glVertex2i(x2+k*pos, y3);
                glVertex2i(x3+k*pos, y3);
                glVertex2i(x3+k*pos, y1);
                glEnd();
            }
            b[j] = temp->info;
            temp = temp->link;
            ar[i].p = b[j]/10;
            ar[i].q = b[j]%10;
            ar[i].p = ar[i].p+48;
            ar[i].q = ar[i].q+48;
            glColor3f(1.0, 1.0, 0.0);
            glRasterPos2f(55.0+k*pos, 215.0);
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ar[i].p);
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ar[i].q);
            drawstring(50.0+k*pos, 185, (char *)"INFO", 1, 1);
            drawstring(85.0+k*pos, 185, (char *)"LINK", 1, 1);
            i++;
            j++;
            glFlush();
        }
    }
}
```

CHAPTER 5

TESTING

5.1 Testing

The primary purpose of testing is to detect software failures so that defects may be discovered and corrected. The scope of software testing often includes examination of codes as well as execution of that code in various environment.

Table 5.1: Result table

TEST CASE	RESULT EXPECTED	RESULT OBTAINED	REMARKS
When enter key pressed	It will direct to the page that contains information	It will direct to the page that contains information	Successful
When mouse right button pressed	Menu will open	Menu will open	Successful
When insert at front clicked	Node inserted at the front	Node inserted at the front	Successful
When insert at rear clicked	Node inserted at the rear	Node inserted at the rear	Successful
When delete at front clicked	Node deleted at the front	Node deleted at the front	Successful
When delete at rear clicked	Node deleted at the rear	Node deleted at the rear	Successful
When exit clicked	Exited from the page	Exited from the page	Successful

5.2 User Interface Testing

Interface testing is done to cover all the functionalities of the system and ensure that this project's graphical user interface meets its specification. We test the interface to perform, such as the movements of the virus and antivirus and also swapping of displays.

5.3 System Testing

After testing the user interface, we can click into the system to test the program. We need to execute a number of times to make sure that the system does not have any bugs during execution. We check the compliance with the specified requirements.

5.4 Performance

The overall performance of the result is efficient and as expected. We need to test in different Linux OS and repeat test the user interface and system testing in these operating systems. The simulation has no delays.

5.5 Operation Testing

The program must be able to execute various operations using graphics. The operations are : This project containing both mouse events and keyboard key events i.e. when you press enter in the keyboard, the new page will display on screen where the page contains some information about the title of the project and message will pop saying, click the right mouse button. When you right click on the message, the menu will be displayed in which insert at front, insert at rear, delete at front, and delete at rear are provided. When you click on Insert at front, the system asks the user to enter the data and entered data will be stored in the information part of the node and the next pointer field connected to itself that will be displayed in the animated view. Similarly, when you click on Insert at rear, the node will be visible at the rear with a entered data and the link field of the last node will always be pointing to the first node when keep on inserting at the front as well as inserting at the rear. During insertion, the array elements are shifted into higher order memory locations. These nodes are linked with line to other elements in the list. When you click on Delete at front, the inserted first node will be deleted at the front side of the list and when you Delete at rear, the inserted last node will be deleted at the end of the list. During deletion, elements are shifted into lower order memory locations. Finally, the exit that is used to exit from the running program.

CHAPTER 6

RESULTS

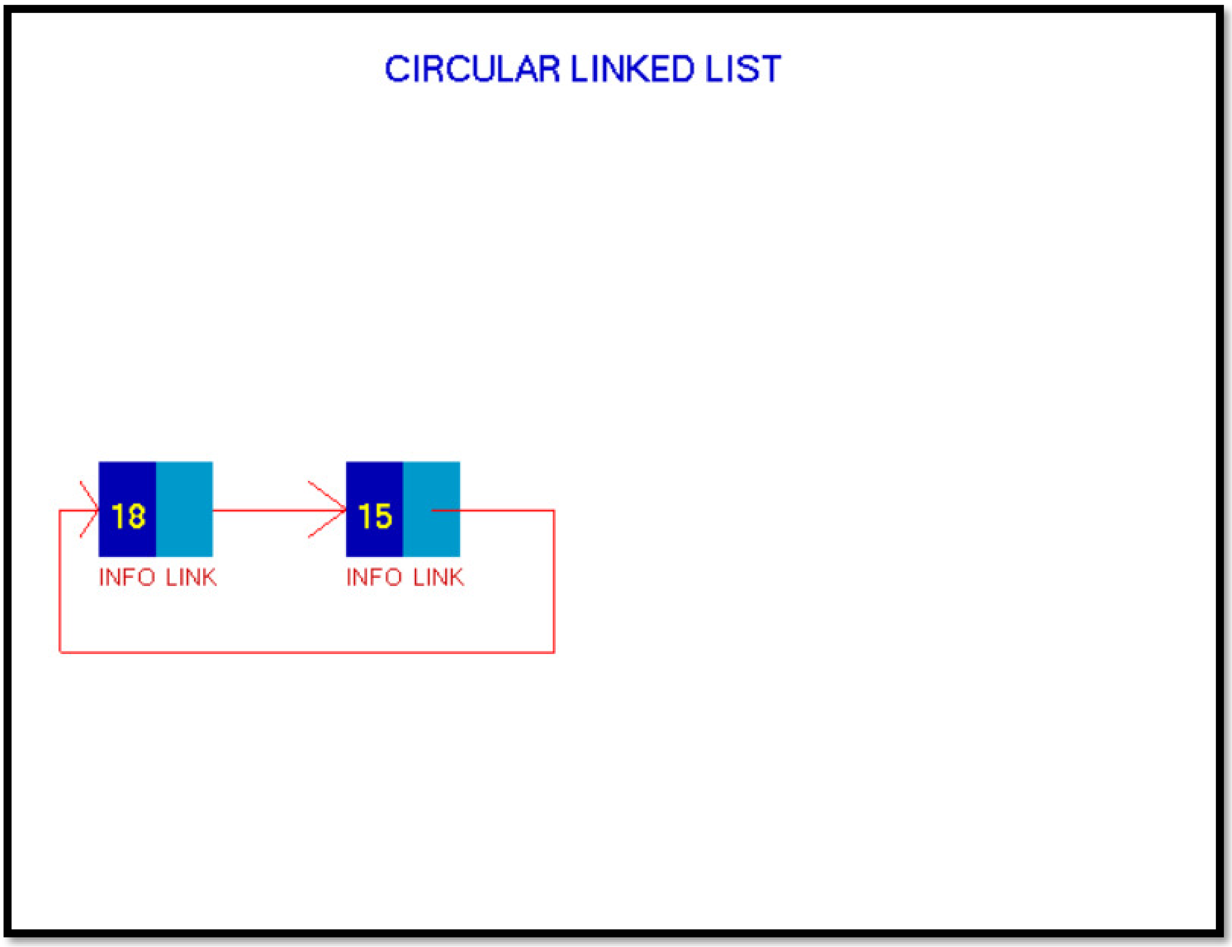

Fig 6.3: Insert at front

The fig 6.4 shows the operation of insertion that inserts the node at the rear with the entered data. When the user enters the data, the pointer of previously inserted data will be disconnected and connected to the newly inserted node.

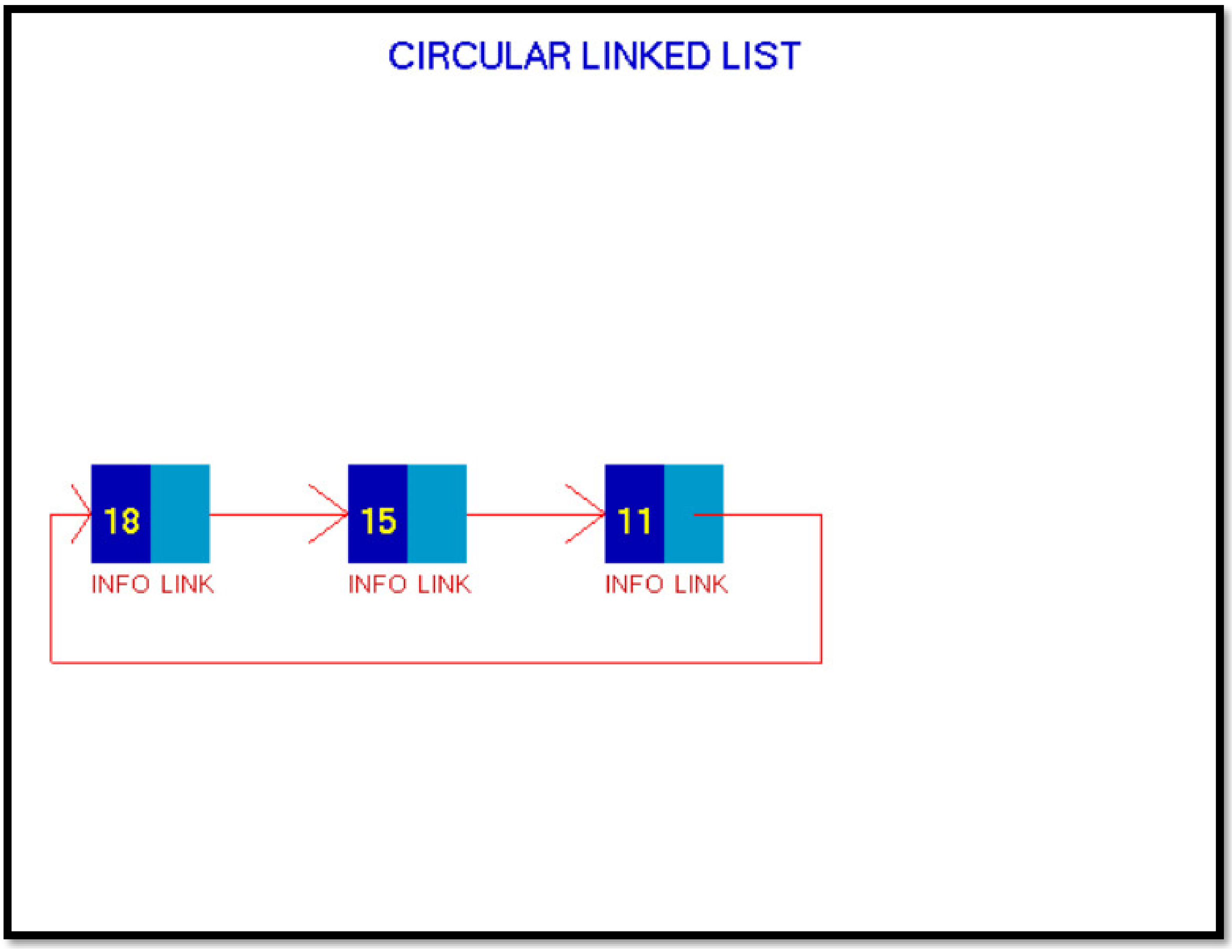

Fig 6.4: Insert at rear

The fig 6.5 shows that the operation of deletion that deletes the node at the front with the entered data.

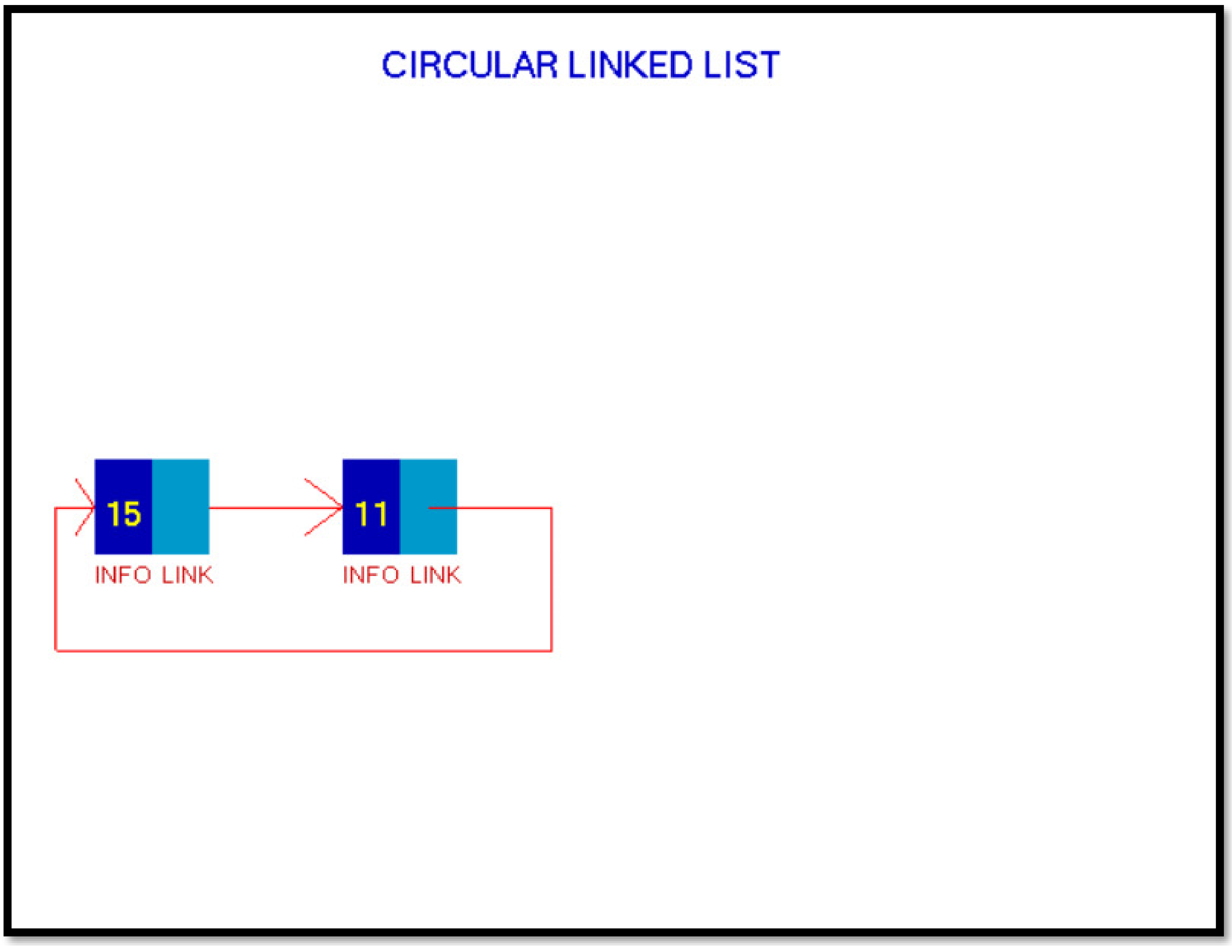

Fig 6.5: Delete at front

The fig 6.6 shows that the operation of deletion that deletes the node at the rear with the entered data.

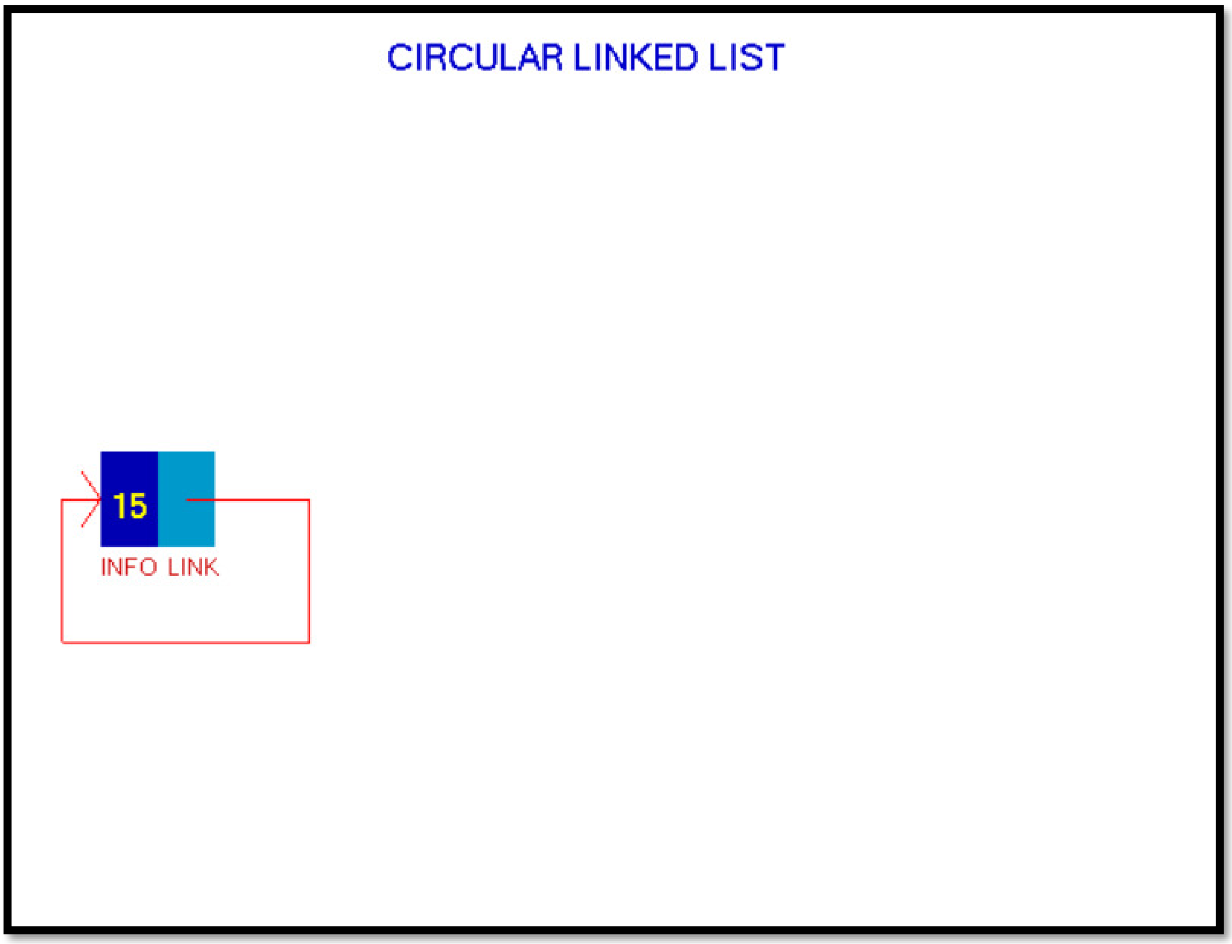

Fig 6.6: Delete at rear

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

In this project we have implemented the circular linked list using computer graphics. This project is quite simple and easy to understand. This project gives the complete concept of circular linked list with some basic operations. This mainly focuses on how the operations will be performed and anyone can understand by looking at the animation. The functionalities like drawing a simple line, creating a polygon, and filling them however if implemented on large scale with sufficient resources, it has the potential to become a standard stand-alone GUI based application for windows operating system.

It also helped us to understand various inbuilt functions of OpenGL and allowed to implement my own. The keyboard and mouse interaction has helped me to implement this project in more easy way and OpenGL methodology was employed, I obtain a good expressive in OpenGL software development. We conclude the project ‘Circular Linked List’ successfully truest of our senses and to best of my ability.

7.2 Future Enhancement

- We can also provide messages while displaying the node inserted and deleted and also include explanation regarding insertion at front, insert at rear, delete at front, and delete at rear for better understanding.
- We can also include more operations of linked list like insert at the specified and delete at the specified and make them more attractive to the user or for those who wants clear picture of view.
- We can add more graphics to the scene using OpenGL functions and making it fully animated mode.

REFERENCES

[1] Introduction to Computer Graphics: https://www.javatpoint.com

[2] Introduction to OpenGL: https://www.ntu.edu.sg

[3] About Circular Linked List: https://www.tutorialspoint.com/index.htm

[4] OpenGL functions: https://docs.microsoft.com/en-us/

[5] About circular linked list: www.javaatpoint.com

[6] About code blocks: https://www.codeblocks.org

[7] About OpenGL: https://www.openglprojects.in