Chapter 1

INTRODUCTION

1.1 Introduction to Project

In computer science, a double-ended queue (abbreviated to dequeue) is an abstract data type that generalizes a queue, for which elements can be added to or removed from either the front (head) or back (tail). It is also often called a head-tail linked list, though properly this refers to a specific data structure implementation of a de-queue. Our project takes uses input and the element where ever required by user i.e. backend Or frontend and also lets us delete from frontend or backend.

1.2 Objective of Project

- Adds an item at the front of Dequeue.
- Adds an item at the rear of Dequeue.
- Deletes an item from front of Dequeue
- Deletes an item from rear of Dequeue.

1.3 Scope of the Project

One example where a dequeue can be used is the A-Steal job scheduling algorithm.[5] This algorithm implements task scheduling for several processors. A separate dequeue with threads to be executed is maintained for each processor. To execute the next thread, the processor gets the first element from the dequeue (using the "remove first element" dequeue operation). If the current thread forks, it is put back to the front of the dequeue ("insert element at front") and a new thread is executed. When one of the processors finishes execution of its own threads (i.e. itsdequeue is empty), it can "steal" a thread from another processor: it gets the last element from the dequeue of another processor ("remove last element") and executes it. The steal-job scheduling algorithm is used by Intel's Threading Building Blocks (TBB) library for parallel programming.

Chapter 2

INTODUCTION TO TECHNOLOGY USED

2.1 Front end : Atom editor v1.27.1

Figure: 2.1- Atom Editor

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained.[8] Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in CoffeeScript and Less. It can also be used as an integrated development environment (IDE). Atom was released from beta, as version 1.0, on 25 June 2015. Its developers call it a "hackable text editor for the 21st Century".

2.2 Back end : GNU Compiler

The GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU toolchain and the standard compiler for most Unix-like operating systems. The Free Software Foundation (FSF) distributes GCC under the GNU General Public License (GNU

GPL). GCC has played an important role in the growth of free software, as both a tool and an example.

Figure:2.2- Gcc compiler

2.3 OpenGL & Glut

Figure:2.3- OpenGL

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3Dvector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. Silicon Graphics Inc., (SGI) started developing OpenGL in 1991 and released it in January 1992; applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006 OpenGL has been managed by the non-profit technology consortium Khronos Group.

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Software Requirement

• OS	:	Windows 7 or above Mac High Sierra or above Linux
• Front end	:	Atom v1.27.1
• Back end	:	gcc complier
• Coding language	:	c++

3.2 Hardware Requirement

• Processor	:	Intel i5 1.5GHz or above.
• Ram	:	8GB DDR3 or above.
• Memory	:	128GB SSD or above.
• Monitor	:	LED-backlit glossy widescreen display.

Chapter 4

FUNCTIONS USED

4.1 Inbuilt Functions

- main(intargc,char **argv){} - main function of the project.
- glutMainLoop(); - Enters GLUT event processing loop.
- strcpy(); - Copies string from one variable to another.
- glPushMatrix(); - glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix.
- glVertex2f(); - Draw a line between two points.
- & many more

4.2 User defined Functions

- myinit() - Initialize the values requried.
- square() - Draws a square where we can place element.
- drawOutline() - Draws a outline to a square.
- drawString() - Draws string and other information to the output screen.
- drawArrow() - Draws arrow where ever requried.
- backEnqueue() - Adds element at the back end of the queue.
- frontEnqueue() - Adds element at front end of the queue.
- & many more

Chapter 5

ALGORITHM

5.1 Algorithm for Insertion at rear end

```
Step -1: [Check for overflow]
          if(rear==MAX)
         Print("Queue is Overflow”);
         return;
Step-2: [Insert element]
         else
         rear=rear+1;
         q[rear]=no;
         [Set rear and front pointer]
         if rear=0
                  rear=1;
         if front=0
                  front=1;
Step-3: return
```

5.2 Implementation of Insertion at rear end

```
voidadd_item_rear()
{
         intnum;
         printf("\n Enter Item to insert : ");
         scanf("%d",&num);
```

```
        if(rear==MAX)
        {
                printf("\n Queue is Overflow");
                return;
        }
        Else
                {
                rear++;
                q[rear]=num;
                if(rear==0)
                rear=1;
                if(front==0)
                front=1;
        }
}
```

5.3 Algorithm for Insertion at font end

```
Step-1 : [Check for the front position]
        if(front<=1)
                Print ("Cannot add item at front end");
        return;
Step-2  : [Insert at front]
        else
                front=front-1;
                q[front]=no;
Step-3  : Return
```

5.4 Implementation of Insertion at font end

```
voidadd_item_front()
{
        intnum;
        printf("\n Enter item to insert:");
        scanf("%d",&num);
        if(front<=1){
        printf("\n Cannot add item at front end");
        return;
                }
        else{
                front—;
                q[front]=num;
        }
}
```

5.5 Algorithm for Deletion from front end

```
Step-1 [ Check for front pointer]
if front=0
        print(" Queue is Underflow”);
        return;
Step-2 [Perform deletion]
else
        no=q[front];
        print(“Deleted element is”,no);
        [Set front and rear pointer]
if front=rear
        front=0;
        rear=0;
else
```

```
        front=front+1;
    Step-3 : Return
```

5.6 Implementation of Deletion at font end

```
Void delete_item_front(){
        intnum;
        if(front==0)
        {
            printf("\n Queue is Underflow\n");
            return;
        }
        else{
            num=q[front];
            printf("\n Deleted item is %d\n",num);
            if(front==rear)
            {
                front=0;
                rear=0;
            }
            else{
                front++;
            }
        }
    }
```

5.7 Algorithm for Deletion from rear end

```
Step-1 : [Check for the rear pointer]
    if rear=0
        print("Cannot delete value at rear end");
```

```
            return;
Step-2: [ perform deletion]
        else
        no=q[rear];
        if(front = rear)
            front=0;
            rear=0;
        else
            rear=rear-1;
            print("Deleted element is",no);
Step-3 : Return
```

5.8 Implementation of Deletion from rear end

```
voiddelete_item_rear()
{
        intnum;
        if(rear==0)
        {
                printf("\n Cannot delete item at rear end\n");
                return;
        }
        else{
                num=q[rear];
                if(front==rear){
                        front=0;
                        rear=0;
                }
                else
                {
```

```
                                rear--;
                                printf("\n Deleted item is %d\n",num);
                        }
                }
        }
```

Chapter 6

SOURCE CODE

6.1 main.cpp

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <GLUT/glut.h>
#include <ctype.h>
#include <time.h>
#include "myheader.h"


//1st function
// initialize the values requried
voidmyinit() {
    // background
    glClearColor(BACKGROUND_R, BACKGROUND_G, BACKGROUND_B,
BACKGROUND_A);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0,SCREEN_X,0.0,SCREEN_Y);



// creates color of cube and outline color
    int i;
    for(i = 0; i < 15; i++) {
            queue[i].r = 0;
            queue[i].g = 0.25;
            queue[i].b = 0.75;
            queue[i].rl = 1;
}

    float step = DIST/MAX;
    queue[0].x1 = OFFSET_X;
    queue[0].x2 = queue[0].x1+step;

    for(i = 1; i <= 15; i++) {
            queue[i].x1 = queue[i-1].x1+step;
            queue[i].x2 = queue[i].x1+step;
    }

}
```

```
//2nd function
//function to draw a square
void square(int x1, int y1, int x2, int y2) {
      glBegin(GL_POLYGON);
                  glVertex2f(x1, y1);
                  glVertex2f(x1, y2);
                  glVertex2f(x2, y2);
                  glVertex2f(x2, y1);
      glEnd();
}

//3rd function
//function to draw the outline the square
voiddrawOutline(int x1, int y1, int x2, int y2) {
      int temp;
      if(x1 < x2) {
                  temp = x1;
                  x1 = x2;
                  x2 = temp;
      }
      if(y1 < y2) {
                  temp = y1;
                  y1 = y2;
                  y2 = temp;
      }
      glBegin(GL_LINES);
                  glVertex2f(x1, y1);
                  glVertex2f(x1, y2);
                  glVertex2f(x2, y2);
                  glVertex2f(x2, y1);
```

```
    glEnd();
}
//4th function
//function to draw a string to the output screen
voiddrawString(const char *str, double x=0, double y=0, double size=5.0) {
    glPushMatrix();
    glTranslatef(x,y,0);
    glScalef(size,size,4.0);
    glColor3f(1, 0, 0);
    intitemCt = 0;
    intlen = strlen(str);
}
    glPopMatrix();
}
//13th function
//main funtion where all the action takes place
int  main(intargc,char **argv) {
    char number[1000];
    int i, j, len;
    char c;
    strcpy(enter_str, "Enter Element to Queue: ");
    start_of_num = strlen("Enter Element to Queue: ");
    jump:
    printf("\n\n\n");
    printf("------------------------------------\n");
    printf("Simulation of  dequeueue in OpenGL\n");
    printf("------------------------------------\n\n");
    printf("Enter the number of elements in the dequeueue\n(Not greater than 15
and not lesser than 3):\n");
    scanf("%d", &MAX);
```

```
        while(MAX > 15 || MAX < 3) {
                printf("ERROR: Invalid value! \nEnter again: ");
                scanf("%d", &MAX);
        }
        printf("\n\nVALUE ACCEPTED! Program Running...\n");

        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(SCREEN_X,SCREEN_Y);
        glutInitWindowPosition(10,10);
        glutKeyboardFunc(mykey);
        myinit();
        glutMainLoop();
        return 0;
    }
```

6.2 myHeader.h

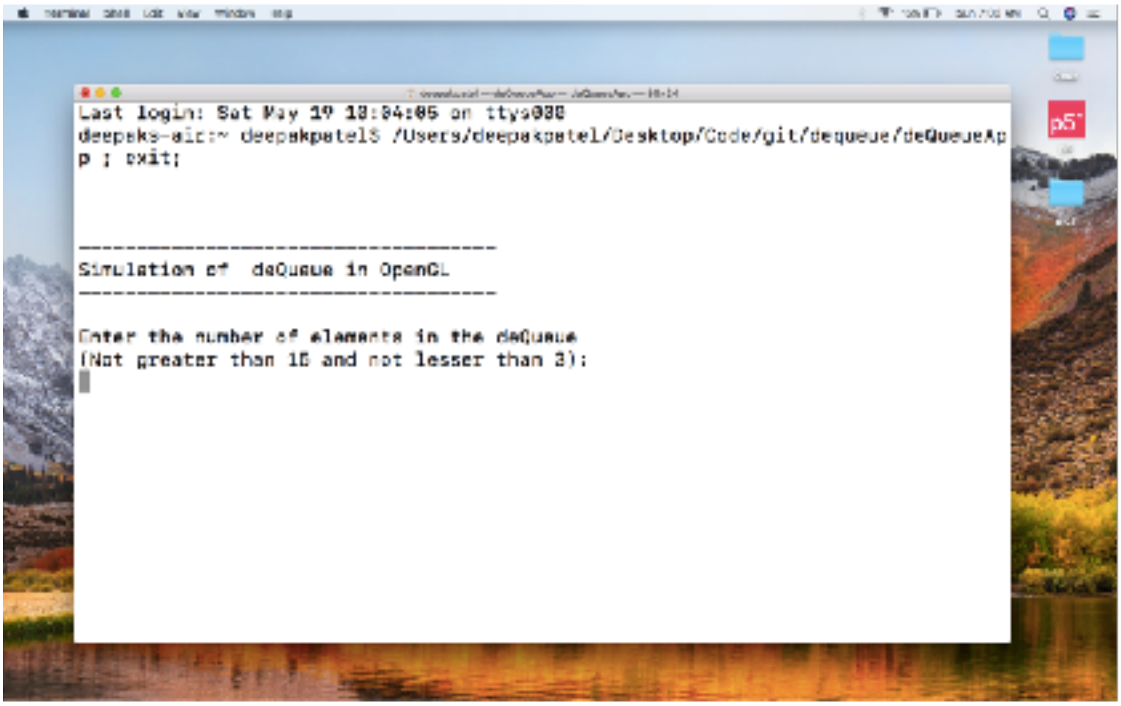
```
#ifndef QUEUE_SIMULA_H
#define QUEUE_SIMULA_H
#define SCREEN_X 1000
#define BACKGROUND_R 1.0
#define BACKGROUND_G 1.0
#define BACKGROUND_B 1.0
#define BACKGROUND_A 1.0
#define SCREEN_Y 500
#define OFFSET_X 50
#define differenc 180.000000
#define fnot 20
#define ARROW_LENGTH 50
#define ENQUEUE 19
#define DEQUEUEUE1 38
#define DEQUEUEUE2 39
```

```
        #define NO_OP 45
        #define OPERATION_POSITION_X SCREEN_X/4+SCREEN_X/20
        #define OPERATION_POSITION_Y SCREEN_Y/5
        #define ENTER_POSITION_Y SCREEN_Y-60
        #define ENTER_POSITION_X OFFSET_X
        #define EMPTY 0
        #define FULL 1
        staticint MAX;              // MAXIMUM NUMBER OF ELEMENTS IN
    THE QUEUE.          TAKEN AS INPUT
    int DIST = SCREEN_X - 2*OFFSET_X;
    int f = 0;              // FRONT OF QUEUE
    int b = 0;              // REAR OF QUEUE
    int YD = SCREEN_Y/10;           // just a random height. HEIGHT OF EACH
ELEMENT (BOX) IN THE QUEUE
    double FONT_ADJUST = 9; //  DENOMINATOR  OF  FONT_RATIO.  LARGER
    THE   VALUE, LARGER THE FONT
    double FONT_RATIO = YD/FONT_ADJUST;  // SETS THE SIZE OF THE FONT
    intenqORdq = NO_OP;                 // indicates what the last operation was.
    // end of all Constants
    intclr = 0;
    intisback = 1;
    char enter_str[10000];
    char blinking[2] = {'_', 0};
    char displayString[10003];
    char displayString1[10003];
    int start_of_num;
    intcnt_of_chars = 0;
    int message = EMPTY;
    // Each element of the queue is encapsulated into a structure.
    structelem {
        float r, g, b; // filling colors
```

```
          floatrl, gl, bl; // for outlines
          float x1, x2;    // start and end x positions
          charnum[11];
          } ;
structelem queue[16];                  // elements of the queue
#endif
```

Chapter 7

SCREENSHOTS



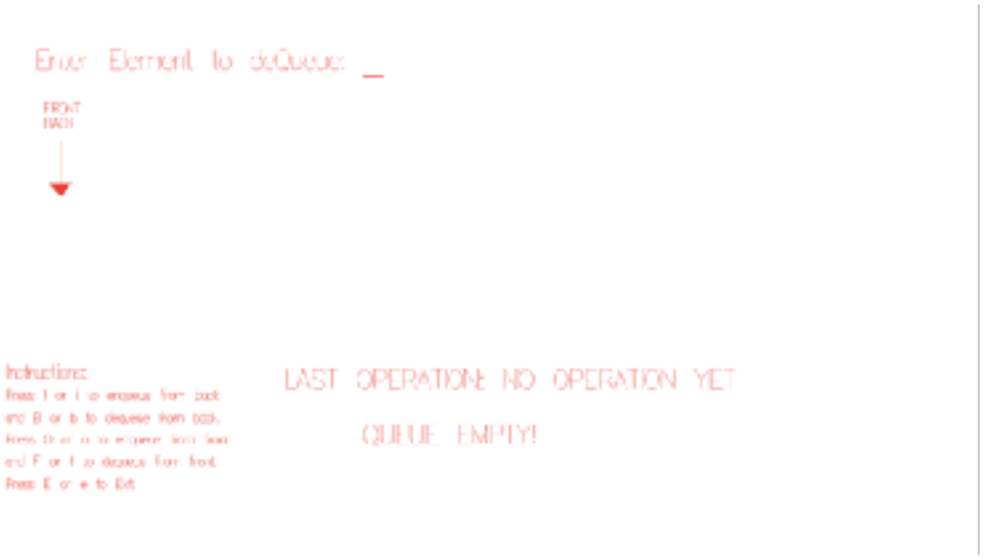
```
Last login: Sat May 19 18:04:05 on ttys000
deepaks-air:~ deepakpatel$ /Users/deepakpatel/Desktop/Code/git/dequeue/deQueueAp
p ; exit;



------------------------------------
Simulation of  deQueue in OpenGL
------------------------------------

Enter the number of elements in the deQueue
(Not greater than 15 and not lesser than 3):
```

Figure: 7.1- User input screen

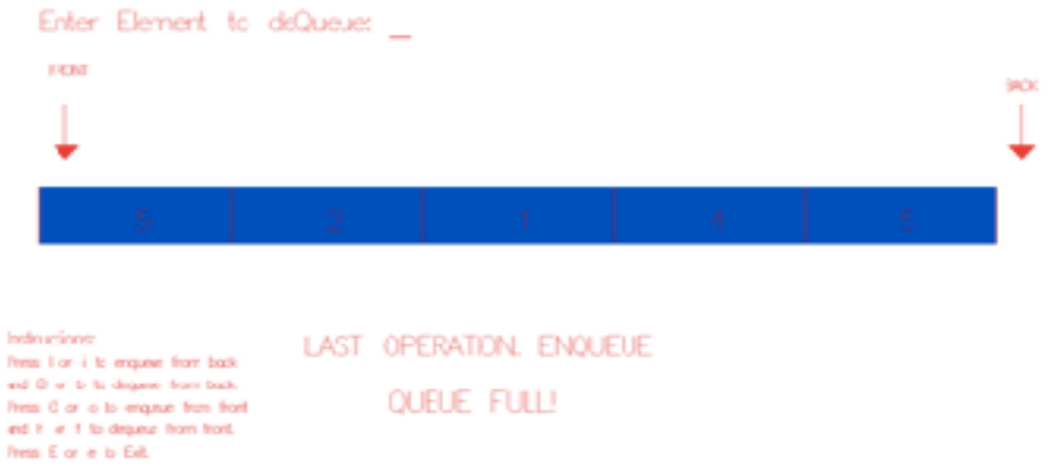


```
Enter Element to deQueue: _

FRONT
BACK


Instructions:                          LAST OPERATION: NO OPERATION YET
Press I or i to enqueue from back
and B or b to dequeue from back.           QUEUE EMPTY!
Press D or d to enqueue from front
and F or f to dequeue from front.
Press E or e to Exit
```

Figure: 7.2- Output screen

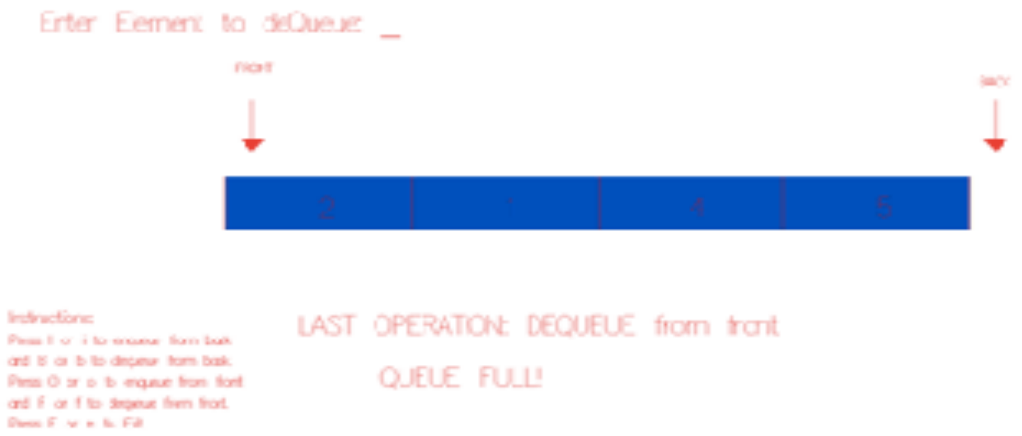


```
Enter Element to deQueue: _
```

```
Instructions:
Press I or i to enqueue from back
and B or b to dequeue from back.
Press O or o to enqueue from front
and F or f to dequeue from front.
Press E or e to Exit.
```

```
LAST OPERATION: ENQUEUE
QUEUE FULL!
```

Figure:- 7.3- Queue full condition



```
Enter Element to deQueue _
```

```
Instructions:
Press I or i to enqueue from back
and B or b to dequeue from back.
Press O or o to enqueue from front
and F or f to dequeue from front.
Press E or e to Exit.
```

```
LAST OPERATON: DEQUEUE from front
QUEUE FULL!
```

Figure: 7.4- dequeueue from front

CONCLUSION

The project is well suited for 2d and 3d objects as well as for carrying out basic graphical functionality. However if implemented on large scale insufficient resources as the potential to become a standard stand alone. GUI based application for Macintosh operating system. Out of many features available the project demonstrates some popular and commonly used features of openGL such as scalene,reshapingetc these graphical function will also work in tandem with various rules involved in double ended queue. Since this project works on dynamic values it can be used for real time computation.OpenGL complexity and the project demonstration the scope of OpenGL platform as a premiere developing launch pad. Hence it has indeed been usefull in developing many algorithms. It serves as important stepping stone for venturing into other fields of computer graphic design and application.

In future version of double ended queue project, addition of some more inbuilt function 2d and 3d models of varioys options to set properties according to usersrequiremnt is a feasible idea making a user interface of this program simpler and more user friendly will certainly help beginners in using this program more easily. The added functionality of giving information about an object on keyboard or mouse clicks.perhaps the most challenging task will be to implement the functionality of layers which are used extensively by proffessional’s in graphics industry.

BIBLIOGRAPHY

Text Book :

- Donald Hearn & Pauline Baker: Computer Graphics-OpenGL Version 3rd Edition, Pearson Education ,2011
- Edward Angel :Interactive Computer graphics – A Top Down approach with OpenGL , 5th edition. Pearson Education, 2011.

Web Links :

www.google.com

https://en.wikipedia.org/wiki/Double-ended_queue/

https://www.geeksforgeeks.org/dequeue-set-1-introduction-applications/

http://scanftree.com/Data_Structure/duble-ended-queue-dequeue/