VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics & Visualization Mini Project Report

on

# "BUCKET SORT"

Submitted in Partial fulfillment of the Requirements for VI Semester of the Degree of

Bachelor of Engineering

In

Computer Science & Engineering

By

S MEGHANA PESHWA
(1CR16CS139)

SOPHIA MARIA AUGUSTINE
(1CR16CS164)

Under the Guidance of

Mr. Kartheek G.C.R

Asst. Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

# CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## CERTIFICATE

This is to certify that the Computer Graphics & Visualization project work entitled **"Bucket Sort"** has been carried out by **S Meghana Peshwa (1CR16CS139)** and **Sophia Maria Augustine (1CR16CS164)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2018-2019**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. This CG project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

| | |
|---|---|
| **-----------------------** | **----------------------** |
| **Signature of Guide** | **Signature of HOD** |
| **Mr. Kartheek G.C.R.**<br>**Assistant Professor**<br>**Dept. of CSE, CMRIT** | **Dr. Jhansi Rani P**<br>**Professor & Head**<br>**Dept. of CSE, CMRIT** |

<u>External Viva</u>

| Name of the examiners | Signature with date |
|---|---|
| 1. | |
| 2. | |

# ABSTRACT

OpenGL provides a set of commands to render a three dimensional scene. That means you provide them in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpeGL applications without licensing.

OpenGL is a hardware and system dependent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation, because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

This project includes the concepts of transformation, motion in objects. We have tried to create a sorting method called “BUCKET SORT”.

This project aims at sorting the given set of numbers using Bucket Sort. It is a sorting algorithm that works by distributing the elements of an array into a number of buckets. Each bucket is then sorted individually, either by using a different sorting algorithm, or by recursively applying the bucket sorting algorithm. It is done using the array data structure.

The user can enter a number of integers and get them sorted either in ascending or descending order as per the user’s requirement. The process of sorting of numbers can be visually seen by the user.

# ACKNOWLEDGEMENT

Success is the result of hard work, determination, proper guidance and help. With such determination and guidance, we have been able to complete this project on time and in an effective manner. We take this opportunity to thank, all those who extended a helping hand, whenever we needed one.

We offer our heartfelt gratitude to our teachers **Prof. Kartheek** and **Prof. Shivaraj**, for their guidance and help throughout this project. They were always available to us with their great knowledge, wisdom and logic.

We are also grateful to our parents, who provided us with the essential and necessary resources to complete this project. We are ever grateful to our friends for their suggestions and criticism which proved very useful for the completion of our project.

Last but not the least, we thank the almighty God for his abundant blessings and guidance on us throughout this project.

**S. Meghana Peshwa (1CR16CS139)**
**Sophia Maria Augustine (1CR16CS164)**

# CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 COMPUTER GRAPHICS

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. Computers have become a powerful medium for the rapid and economical production of pictures.

Interactive graphics is the most important means of producing pictures since the invention of photography and television. We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.

A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

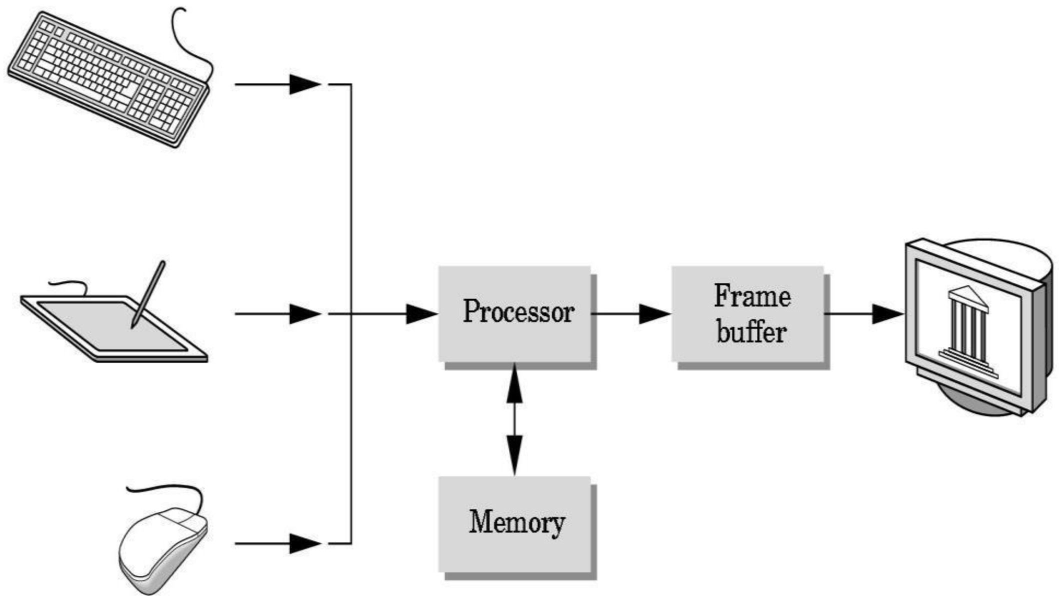


**Fig 1.1:** Graphic System

## 1.2 Areas of Application of Computer Graphics

- User interfaces and Process control
- Cartography
- Office automation and Desktop publishing
- Plotting of graphs and charts
- Computer aided Drafting and designs
- Simulation and Animation

## 1.3 Introduction to OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

It fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

### 1.3.1 The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl,glu,glut.

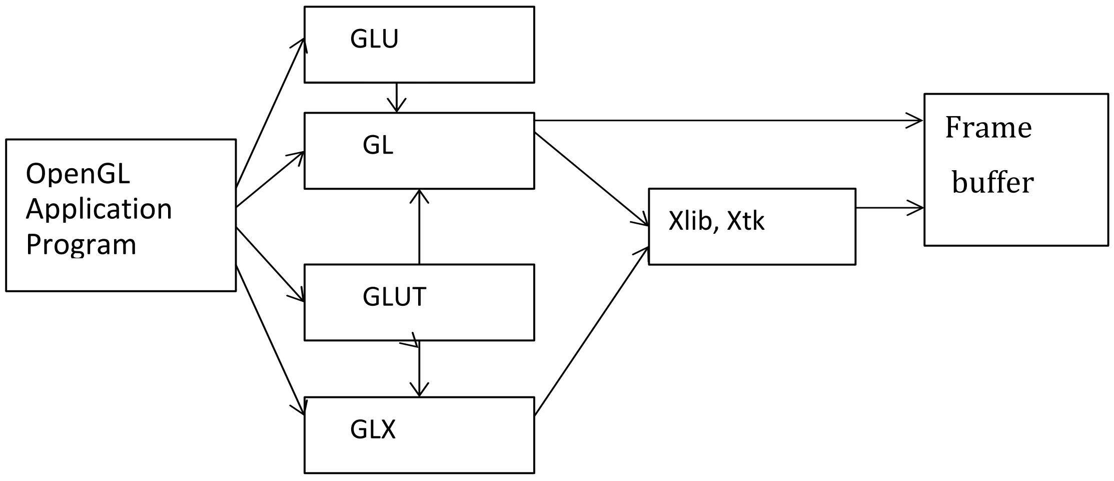


**Fig 1.2:** Library organization of OpenGL

### 1.3.2 Graphics Functions

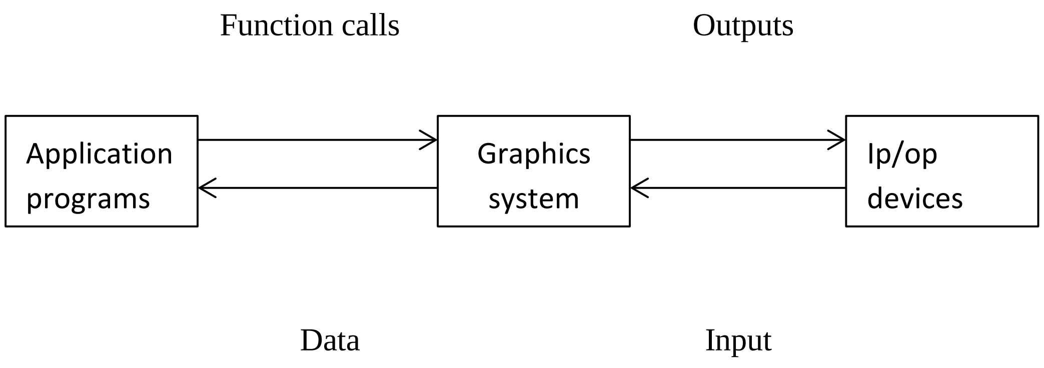


**Fig 1.3:** Graphics system as a black box.

Our basic model of a graphics package is a black box, a term that engineers use to denote a system whose properties are described only by its inputs and outputs. We describe an API through the functions in its library.

# CHAPTER 2

# REQUIREMENTS SPECIFICATION

## 2.1 PURPOSE OF THE REQUIREMENTS DOCUMENT

The software requirement specification is the official statement of what is required for development of particular project. It includes both user requirements and system requirements. This requirement document is utilized by variety of users starting from project manager who gives project to the engineer responsible for development of project.

It should give details of how to maintain, test, verify and what all the actions to be carried out through life cycle of project.

### 2.1.1 Scope of the project

The scope is to use the basic primitives defined in openGL library creating complex objects. We make use of different concepts such as glVertex(), glColor(), gluOrtho2D(), glFlush(), glEnable() and so on.

### 2.1.2 Definition

The project **BUCKET SORT** is created to demonstrate OpenGL's concepts. It encompasses some of the skills learnt in our OpenGL classes such as glVertex(), glColor(), gluOrtho2D() and glEnable().

## 2.2 SPECIFIC REQUIREMENTS

### 2.2.1 User Requirements:

- Easy to understand and should be simple.
- The built-in functions should be utilized to maximum extent.
- OpenGL library facilities should be used.

### 2.2.2 Software Requirements:

- Platform used: UBUNTU
- Technology used: OpenGL Libraries such has OpenGL Utility library, OpenGL Utility toolkit
- Language: C

### 2.2.3 Hardware Requirements:

- Processor-Intel or AMD(Advanced Micro Devices)
- RAM-512MB(minimum)
- Hard Disk-1MB(minimum)
- Mouse
- Keyboard
- Monitor

# CHAPTER 3

# DESIGN

## 3.1 Flowchart

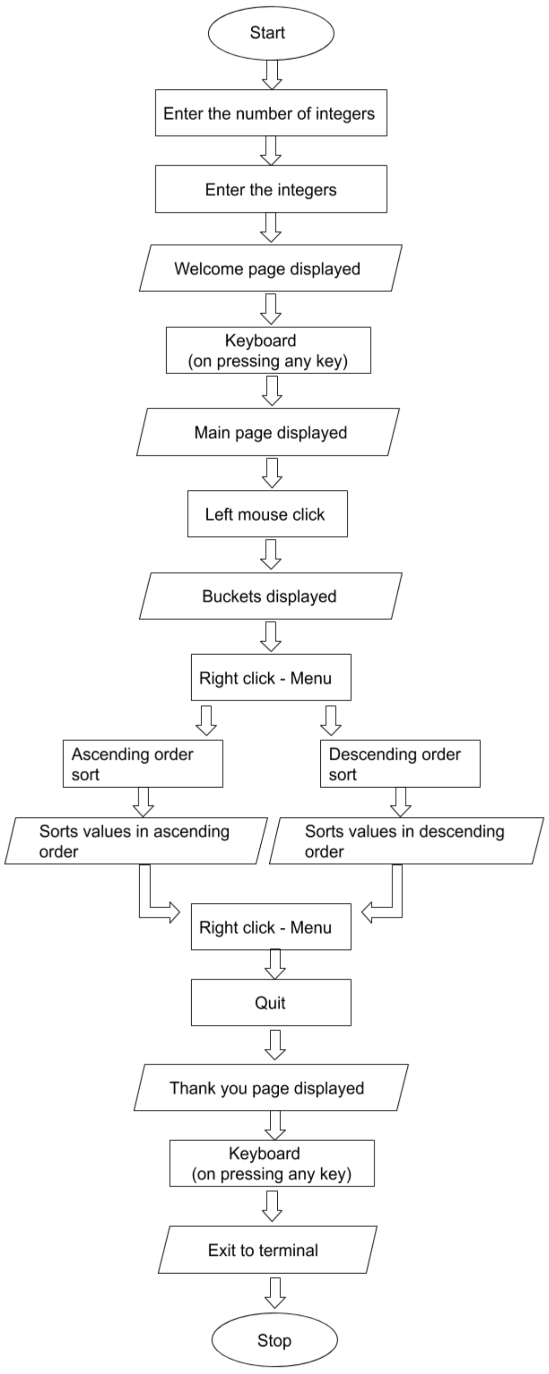


**Fig 3.1:** Flowchart

# CHAPTER 4

# IMPLEMENTATION

## 4.1 Code in C Language

### 4.1.1 Display Function

In this function, we display the heading, input array, the buckets and the sorted array. The welcome() function and the thankyou() function is also called here and is displayed based on their respective flag values.

CODE -

```
void display()

{
   glClearColor(1,0.90,0.90,1);
         glClear(GL_COLOR_BUFFER_BIT);
         glColor3f(0.65,0,0.4);
   if(flag2==1)
      welcome();//display welcome page
         else if(flag2==0){
      redisplay();
      if(flag==1){
         translate_num();
         int j,p=0,b[n];
         if(sort_type==0){
         for(int i=0;i<150;i++)
         {
            j=0;
            if(bucket[i][0]!=9999){//check for non empty bucket
               while(bucket[i][j]!=9999){//loop till last value stored in that bucket
                  b[p]=bucket[i][j];//store sorted values in another array
                  p++;
```

```
                    j++;
                }
            }
        }
    }
        else if(sort_type==1){
        for(int i=149;i>=0;i--)
        {
            j=0;
            if(bucket[i][0]!=9999){//check for non empty bucket
                while(bucket[i][j]!=9999){//loop till last value stored in that bucket
                    b[p]=bucket[i][j];//store sorted values in another array
                    p++;
                    j++;
                }
            }
        }
        }
        delay(1300);
        char s1[30] = "The sorted array is  :  ";
        if(cnt==0){
            cnt++;
            printf(s1);
            for(int i=0;i<n;i++)
                printf("%d  ",b[i]);
        }
        display_ipop(b,s1,140,500);
    }
}
if(flag3==1)
    thankyou();
glFlush();
```

```
}
```

### 4.1.2 Function to Display Buckets

This function is used to display the buckets, the range of values each bucket can hold and the values in each bucket.

CODE -

```
void display_bucket()
{
        int j=0;
        int p=0,x,y;
        double d;
unsigned char s[30];
        char s1[30],s2[10]=" - ",s3[30];
        for(int i=0;i<150;i++)
        {
                j=0;
                if(bucket[i][0]!=9999){
        x = i*10;
        y = x+9;
                        while(bucket[i][j]!=9999){
                                display_int(bucket[i][j],px[i][j],py[i][j]);
                                j++;
                        }
        d = (X2[p]-(X2[p]-X1[p])/2);
        sprintf(s3,"%d",x);
        sprintf(s1,"%d",y);
        strcat(s3,s2);
        strcat(s3,s1);
        //display range of each bucket
        display_string(s3,d-
(screen_x/3000)*glutBitmapLength(GLUT_BITMAP_HELVETICA_18,(unsigned
char*)s3),Y1-450,GLUT_BITMAP_HELVETICA_18);
```

```
        //draw buckets
                        glLineWidth(5);
                        glBegin(GL_LINE_LOOP);
                        glVertex2f(X1[p],Y1+300);
                        glVertex2f(X1[p],Y1-300);
                        glVertex2f(X2[p],Y1-300);
                        glVertex2f(X2[p],Y1+300);
                        glEnd();
                        p++;
                }
        }
}
```

### 4.1.3 Function to sort individual buckets

In this function we sort all the buckets one by one. The sorting algorithm used for this is bubble sort. The sorting can be visualized by the movement of the integers.

CODE -

```
 void translate_num(){
   //this function moves the numbers to be sorted in a semi circle fashion
        double angle1=M_PI,angle2=0,rx=90,ry=100,px1,py1,px2,py2;
        int i,j,p,q=0;
        for(p=0;p<150;p++)
        {
                q=0;
                if(bucket[p][0]!=9999){
                        while(bucket[p][q]!=9999){
                                q++;
                        }
                        //bubble sort
        for(i=0;i<q-1;i++)
        {
           for(j=0;j<q-i-1;j++)
        {

          angle1=M_PI;
          angle2=0;
          if(sort_type==0){
            if(bucket[p][j]>bucket[p][j+1]){ //for ascending order sort
              px1=px[p][j]+90;
              py1=py[p][j];
              px2=px[p][j+1]-90;
              py2=py[p][j+1];
              while((angle2>=-M_PI) && (angle1>=0)){
                px[p][j]=px1+cos(angle1)*rx;
                py[p][j]=py1+sin(angle1)*ry;
                angle1=angle1-(0.5*M_PI/180);
                px[p][j+1]=px2+cos(angle2)*rx;
                py[p][j+1]=py2+sin(angle2)*ry;
                angle2=angle2-(0.5*M_PI/180);
                redisplay();//re display entire screen with updated positions of
numbers
              }
swap(&bucket[p][j],&bucket[p][j+1]);//swap the numbers
            swapf(&px[p][j],&px[p][j+1]);//swap the positions of the numbers
            }
          }

          else if(sort_type==1){
            if(bucket[p][j]<bucket[p][j+1]){ //for descending order sort
              px1=px[p][j]+90;
              py1=py[p][j];
              px2=px[p][j+1]-90;
              py2=py[p][j+1];
              while((angle2>=-M_PI) && (angle1>=0)){
```

```
                        px[p][j]=px1+cos(angle1)*rx;
                        py[p][j]=py1+sin(angle1)*ry;
                        angle1=angle1-(0.5*M_PI/180);
                        px[p][j+1]=px2+cos(angle2)*rx;
                        py[p][j+1]=py2+sin(angle2)*ry;
                        angle2=angle2-(0.5*M_PI/180);
                        redisplay();//re display entire screen with updated positions of
numbers
                     }
                  swap(&bucket[p][j],&bucket[p][j+1]);//swap the numbers
                  swapf(&px[p][j],&px[p][j+1]);//swap the positions of the numbers
                  }
               }
            }
         }
         delay(1500);
      }
   }
}
```

### 4.1.4 Function to display the Welcome Page

The welcome page is displayed using this function.

CODE -

```
void welcome()
{
   GLfloat y = 2570;
   char buff[200];
   char *c;
   char head1[35] = "COMPUTER GRAPHICS MINI PROJECT";
   char head2[30] = "TOPIC : BUCKET SORT";
   char end[70] = "PRESS ANY KEY TO VISUALIZE BUCKET SORT";
```

```
    display_string(head1,screen_x/2-
(screen_x/3000)*glutBitmapLength(GLUT_BITMAP_TIMES_ROMAN_24,(unsigned
char*)head1),2850,GLUT_BITMAP_TIMES_ROMAN_24);
    display_string(head2,screen_x/2-
(screen_x/3000)*glutBitmapLength(GLUT_BITMAP_TIMES_ROMAN_24,(unsigned
char*)head2),2700,GLUT_BITMAP_TIMES_ROMAN_24);
    FILE *fp;
    if ((fp = fopen("welcome.txt","r")) == NULL){
       printf("Error! opening file");
       // Program exits if the file pointer returns NULL.
       exit(1);
   }
   if(fp){
        while (fgets(buff,150,fp)!=NULL){
            glRasterPos2f(100,y);
            for (c=buff; *c!='\0'; c++)
            {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,*c);
            }
            y -= 93;
        }
        fclose(fp);
    }
    display_string(end,screen_x/2-
(screen_x/3000)*glutBitmapLength(GLUT_BITMAP_TIMES_ROMAN_24,(unsigned
char*)end),70,GLUT_BITMAP_TIMES_ROMAN_24);
}
```

### 4.1.5 Function to display Thank You page

This function is used to display the thank you page. It is displayed when the respective flag is set to 1 in the display() function.

CODE -

```
void thankyou()
{
   glViewport(0,0,screen_x/2 - 50,1000);
   glClearColor(0.72,0.92,0.97,1);
         glClear(GL_COLOR_BUFFER_BIT);
         glColor3f(0.22,0.35,0.50);
   display_string("THANK YOU!",1350,2200,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("Done by  :",1420,1950,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("S Meghana
Peshwa",1320,1750,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("1CR16CS139",1380,1550,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("&",1490,1400,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("Sophia Maria
Augustine",1280,1250,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("1CR16CS164",1380,1050,GLUT_BITMAP_TIMES_ROMAN_24);
   display_string("PRESS ANY KEY TO
EXIT",1240,400,GLUT_BITMAP_TIMES_ROMAN_24);
}
```

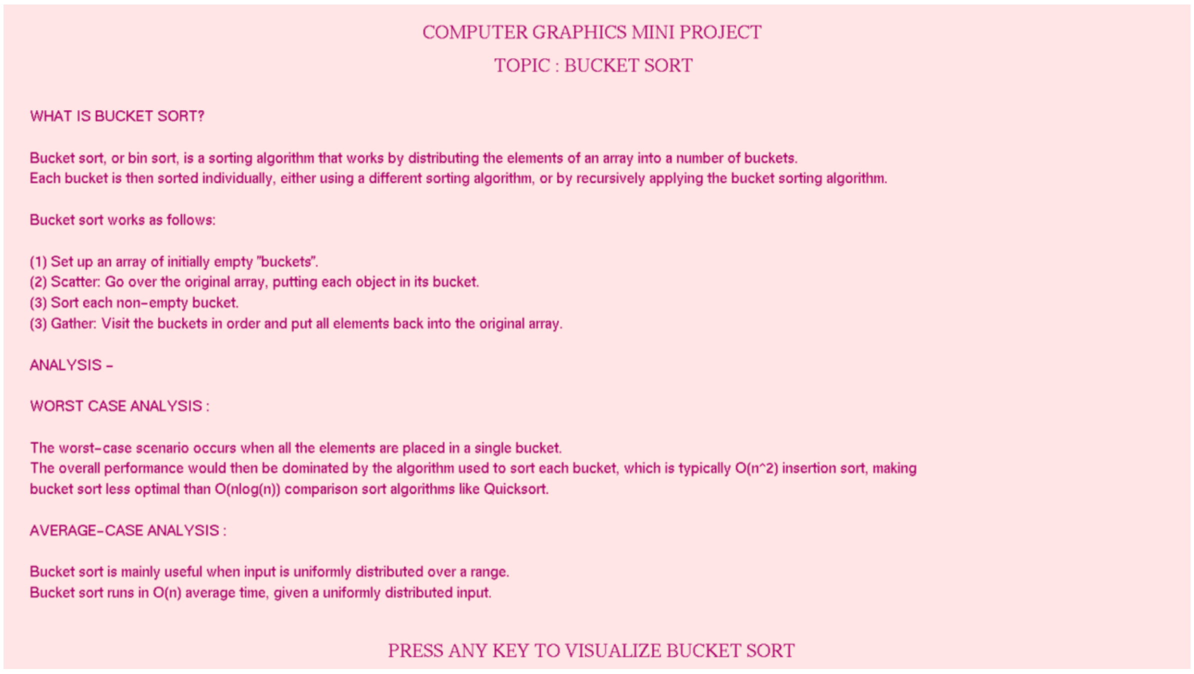
# CHAPTER 5

# DISCUSSIONS AND SCREENSHOTS

## 5.1 Discussion

- After executing the code, the user enters the number of integers and the input values.
- After pressing enter the welcome page is displayed.
- On pressing any key the next page with the provided input values is displayed.
- On clicking left mouse button, the input array values get placed in their respective buckets.
- On right click, menu is displayed and choosing Ascending order sort sorts the array in ascending order.
- Choosing Descending order sort sorted the array in descending order.
- On choosing Quit, the thank you page is displayed.
- Pressing any key will exit to the terminal.

## 5.2 SCREEN SHOTS



COMPUTER GRAPHICS MINI PROJECT

TOPIC : BUCKET SORT

WHAT IS BUCKET SORT?

Bucket sort, or bin sort, is a sorting algorithm that works by distributing the elements of an array into a number of buckets.
Each bucket is then sorted individually, either using a different sorting algorithm, or by recursively applying the bucket sorting algorithm.

Bucket sort works as follows:

(1) Set up an array of initially empty "buckets".
(2) Scatter: Go over the original array, putting each object in its bucket.
(3) Sort each non-empty bucket.
(3) Gather: Visit the buckets in order and put all elements back into the original array.

ANALYSIS -

WORST CASE ANALYSIS :

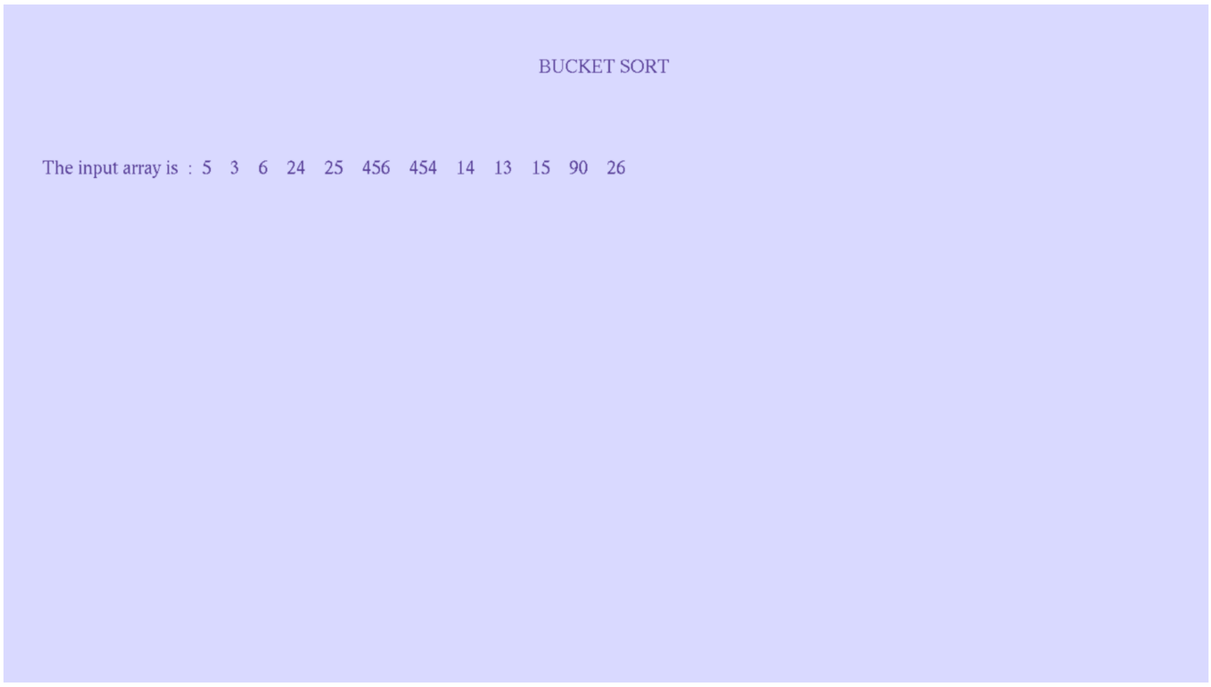
The worst-case scenario occurs when all the elements are placed in a single bucket.
The overall performance would then be dominated by the algorithm used to sort each bucket, which is typically O(n^2) insertion sort, making bucket sort less optimal than O(nlog(n)) comparison sort algorithms like Quicksort.

AVERAGE-CASE ANALYSIS :

Bucket sort is mainly useful when input is uniformly distributed over a range.
Bucket sort runs in O(n) average time, given a uniformly distributed input.

PRESS ANY KEY TO VISUALIZE BUCKET SORT

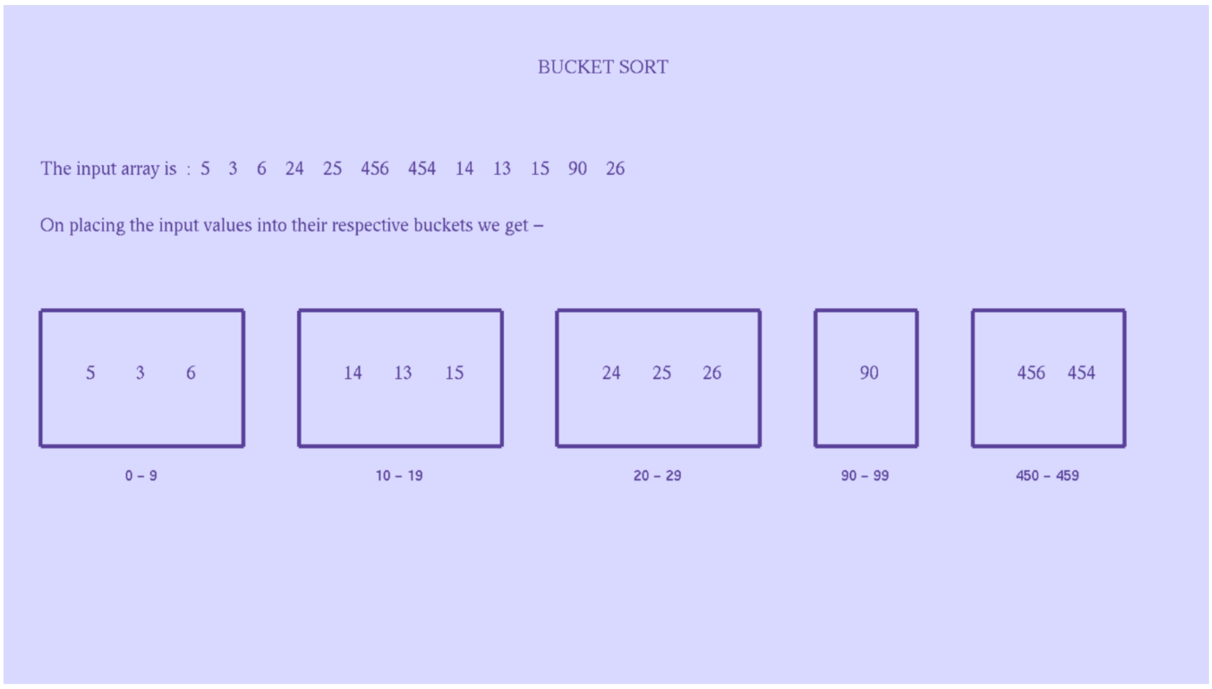
**Fig 5.1:** Welcome Screen



BUCKET SORT

The input array is : 5 3 6 24 25 456 454 14 13 15 90 26

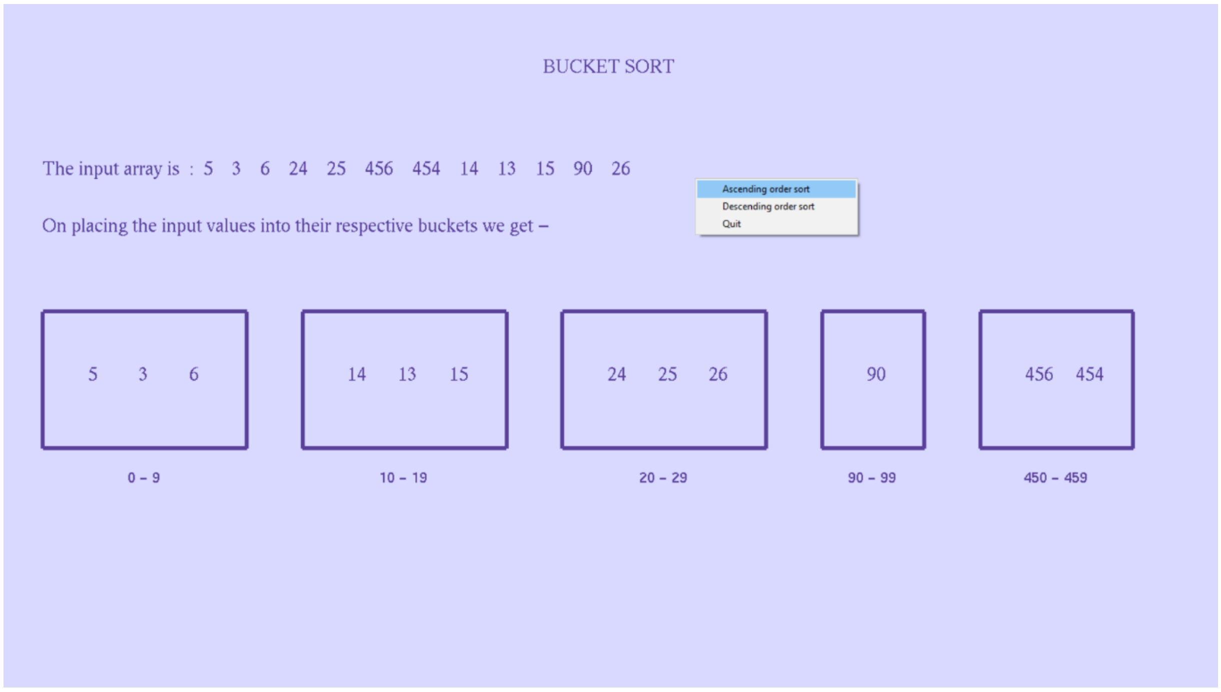
**Fig 5.2:** Input Array


BUCKET SORT

The input array is : 5 3 6 24 25 456 454 14 13 15 90 26

On placing the input values into their respective buckets we get –

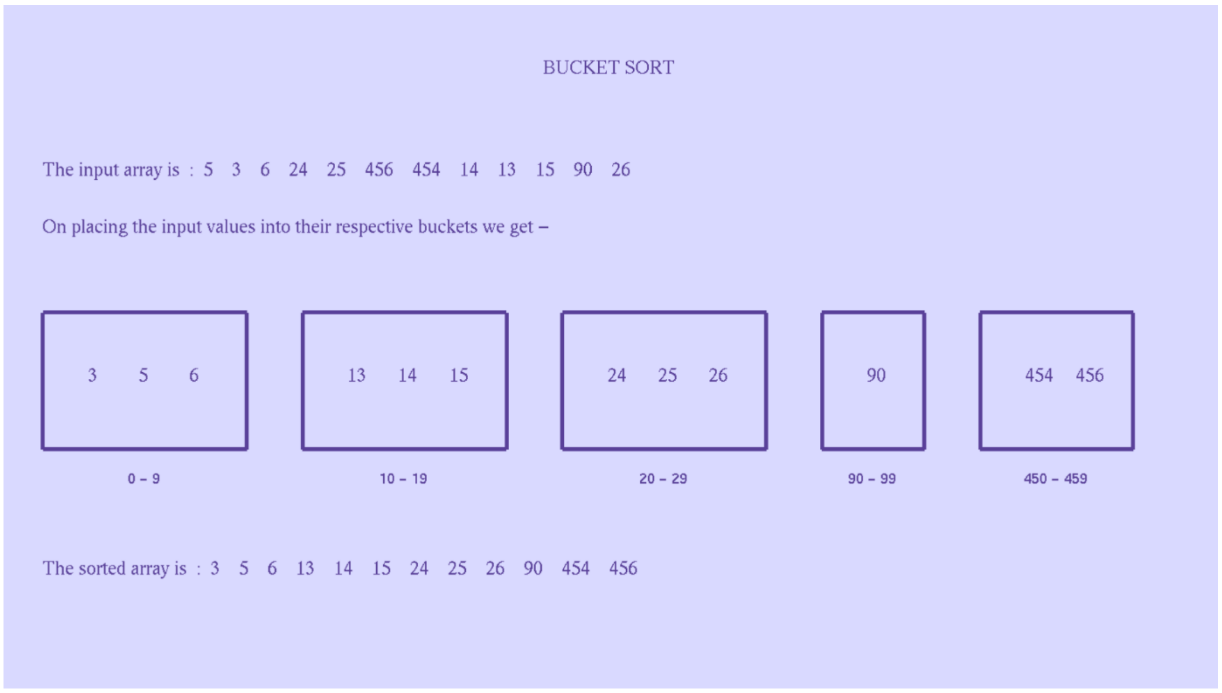
**Fig 5.3:** Splitting into Buckets


BUCKET SORT

The input array is : 5 3 6 24 25 456 454 14 13 15 90 26

On placing the input values into their respective buckets we get –

**Fig 5.4:** When Ascending Sort is selected

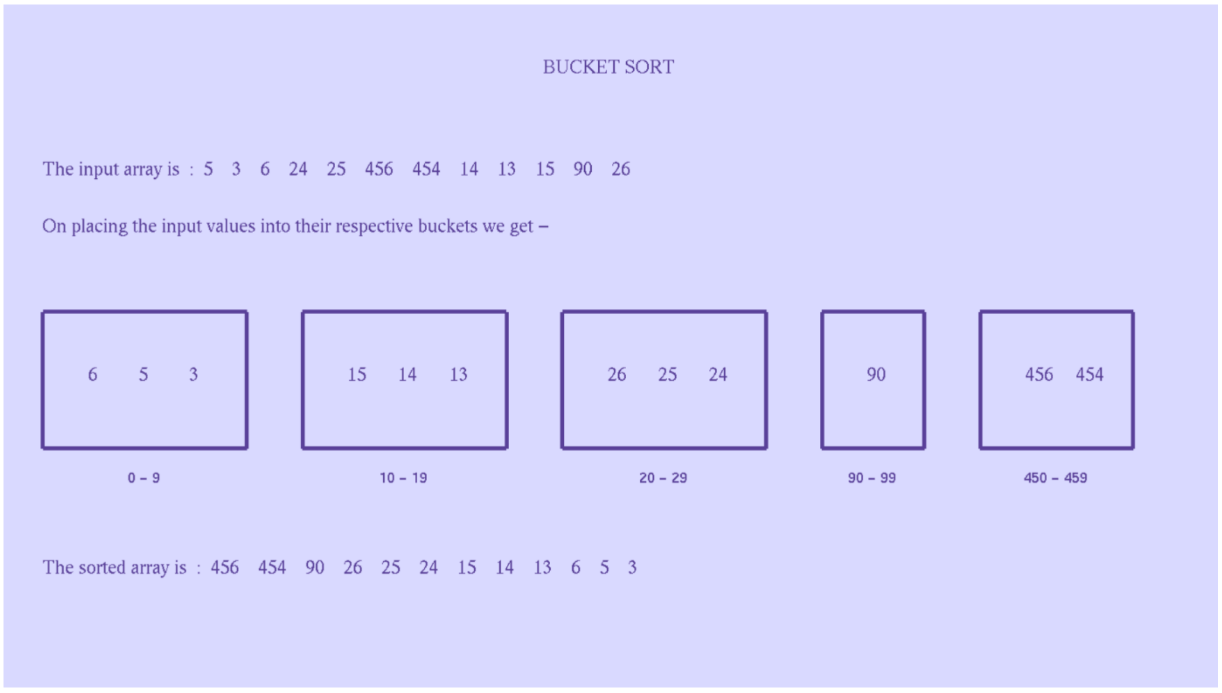

BUCKET SORT

The input array is : 5 3 6 24 25 456 454 14 13 15 90 26

On placing the input values into their respective buckets we get –

The sorted array is : 3 5 6 13 14 15 24 25 26 90 454 456

**Fig 5.5:** Sorted in Ascending Order

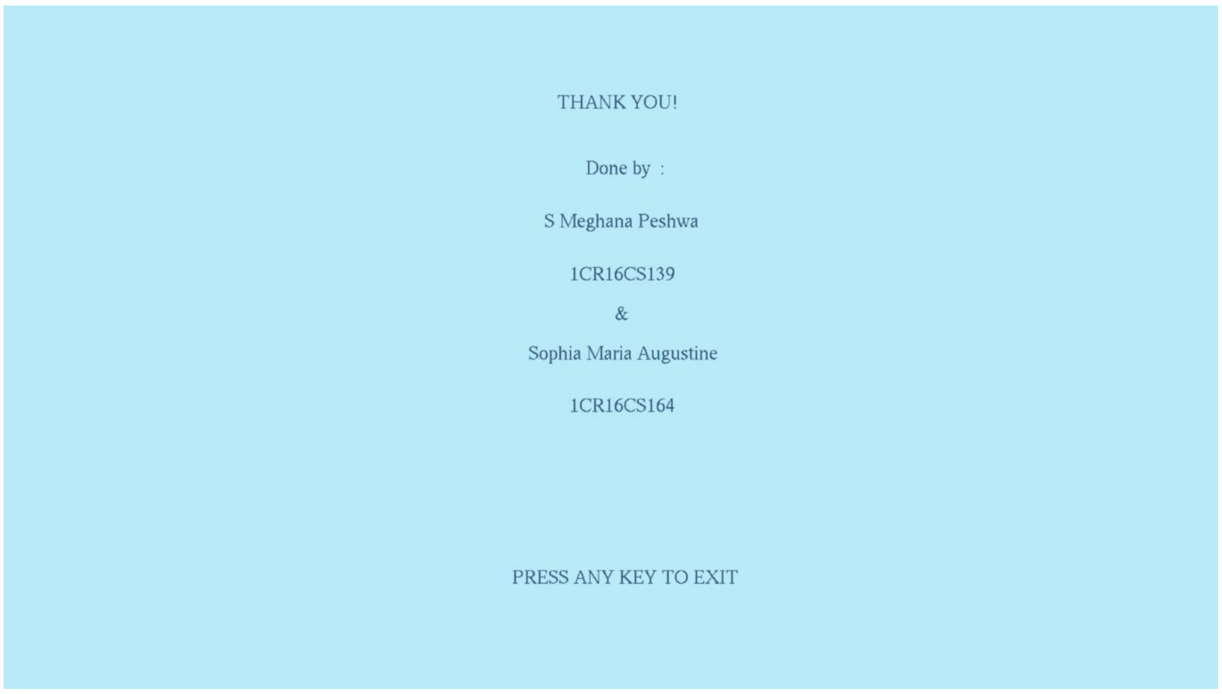

BUCKET SORT

The input array is : 5 3 6 24 25 456 454 14 13 15 90 26

On placing the input values into their respective buckets we get –

The sorted array is : 456 454 90 26 25 24 15 14 13 6 5 3

**Fig 5.6:** Sorted in Descending Order



THANK YOU!

Done by :

S Meghana Peshwa

1CR16CS139

&

Sophia Maria Augustine

1CR16CS164

PRESS ANY KEY TO EXIT

**Fig 5.7:** Thank You Screen

# CHAPTER 6

# CONCLUSION AND FUTURE SCOPE

By implementing this project we got to know how to use some of the built-in functions effectively and how to interact efficiently and easily. We also got a good exposure of how sorting, animation and simulations are developed.

The future scope of this project is that, this application can be changed to sort negative values as well. Efficiency of sorting can be improved. The code can be improved so as to sort larger number of integers and the buckets can be displayed in a better form.

One of the major accomplishments in the specification of OpenGL was the isolation of window system dependencies from OpenGL's rendering model. The result is that OpenGL is window system independent. Window system operations such as the creation of a rendering window and the handling of window system events are left to the native window system to define. Necessary interactions between OpenGL and the window system such as creating and binding an OpenGL context to a window are described separately from the OpenGL specification in a window system dependent specification.

The GLUT application-programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT API (like the OpenGL API) is stateful. Most initial GLUT state is defined and the initial state is reasonable for simple programs. The GLUT routines also take relatively few parameters. No pointers are returned. The only pointers passed into GLUT are pointers to character strings (all strings passed to GLUT are copied, not referenced) and opaque font handles.

# REFERENCES

[1] Bucket sort – https://en.wikipedia.org/wiki/Bucket_sort

[2] https://www.geeksforgeeks.org/bucket-sort-2/

[3] https://gamedev.stackexchange.com/questions/9607/moving-an-object-in-a-circular-path

[4] https://www.hackerearth.com/practice/algorithms/sorting/bucket-sort/tutorial/

[5] http://www.opengl.org

[6] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd Edition, Pearson Education, 2011. (4th Edition)

[7] Edward Angel: Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008. (Chapter 3: 3-1 to 3.11: Input & Interaction)