Name of the Institute ..... RV COLLEGE OF ENGINEERING.

Laboratory Certificate

This is to certify that Smt./Sri. Shreyas S Kasetty

has satisfactorily completed the course of Experiments in

Practical..... Computer Graphics ..... Prescribed by

the..... Computer Science Department .....

University..... V.T.U .....

in the Laboratory of this College in the year 2020 - 2021

Signature of the Teacher in charge of the batch

Date :.....

Head of the Department

Name of the Candidate .....

.....Reg No. .....

Examination Centre .....

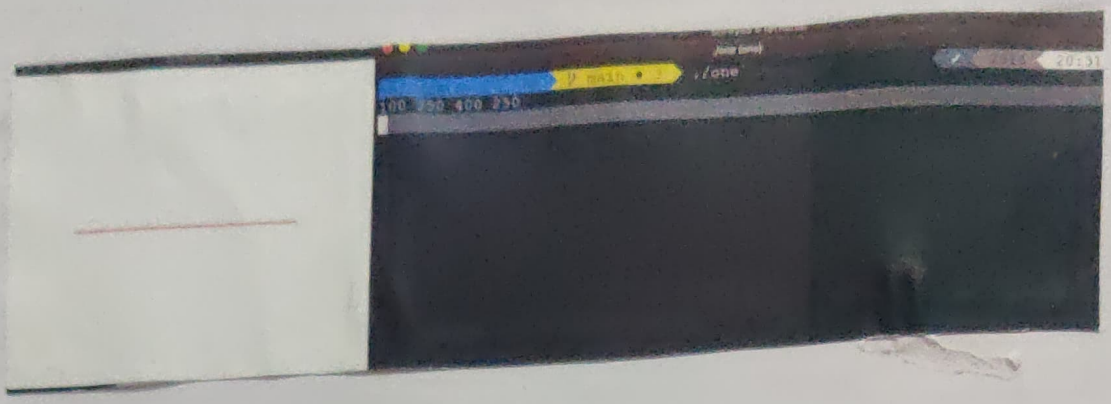
Date of Practical Examination.....

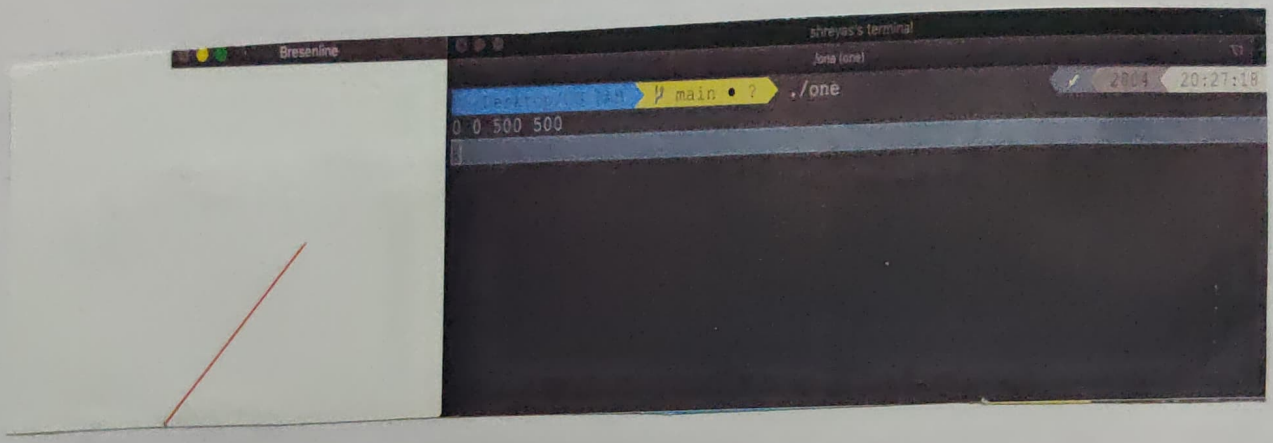
INDEX

Name: Shreyas S Kasetty Class: VII C Year: 2020-21

INDEX

Name..... Class..... Year.....


main • ?
./one
100 100 400 250


Bresenline
shreyas's terminal
Jona (one)
main • ?
./one
2004
20:27:18
0 0 500 500

Name of the Experiment : Bresenham line Date :

Experiment No. : 1 Experiment Result :

Page No. : 1

1. Write a program to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and slopes less than one! User must able to draw as many lines and specify input through keyboard.

```
#include<GLUT/glut.h>
#include<stdio.h>
void display(int x, int y)
{ glBegin(GL_POINTS);
  glVertex2i(x,y);
  glEnd();
}
int x1, y1, x2, y2

void draw_line()
{
  glClear(GL_COLOR_BUFFER_BIT);
 int dx, dy, i, e;
 int incx, incy, inc1, inc2;
int x, y;
 dx = x2 - x1;
 dy = y2 - y1;
 if(dx<0)
   dx = -dx;
 if(dy<0)
   dy = -dy;
 incx=1
```

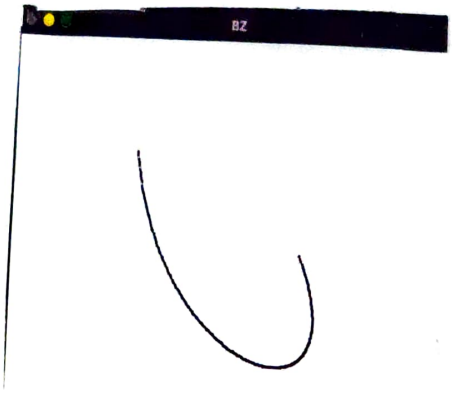
```
if (x2<x1)
    incx = -1;
incy=1;
 if (y2<y1)
    incy=-1;

x = x1
y=y1;
if (dx>dy)
{
   display (x, y);
  e= 2*dy - dx;
 incl= 2*(dy-dx);
 inc2 = 2*dy;
for(i=0; i<dx; i++)
{
   if (e>=0)
   {
      y+=incy;
      e += incl;
   }else
      e += inc2;
  x += incx;
 display (x,y);
}
}else
{  display(x,y);
   e = 2*dy - dx;
   incl = 2* (dy-dx);
   inc2 = 2*dx;
```

```
for(i=0; i<dy; i++)
{  if(e>=0)
   { x += incx;
     e += inc1;
   }
   else
     e += inc2;
   y += incy;
   display(x,y);
 }}
glFlush();
}
void myinit()
{ glClearColor(1,1,1,1);
  glColor3f(1.0,0.0,0.0);
  glPointSize(3.0);
  gluOrtho2D(10,500,10,500);
}
int main(int argc, char *argv)
{ printf("Enter the starting and ending coordinates");
  scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(300, 300);
  glutInitWindowPosition(5,5);
  glutCreateWindow("Bresenham Line");
  glutDisplayFunc(draw_line);
  myinit();
  glutMainLoop();
  return 0;
}
```

```
#include <iostream>
#include <GL/glut.h>
#include <time.h>
using namespace std;
int x1, x2, yc1, y2;
int flag = 0;
void draw_pixel(int x, int y)
{
        glColor3f(1, 0, 0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
        glFlush();

}
void draw_line()
{
        int dx, dy, i, e;
        int incx, incy, inc1, inc2;
        int x, y;
        dx = x2 - x1;
        dy = y2 - yc1;
        if (dx < 0)dx = -dx;
        if (dy < 0)dy = -dy;
        incx = 1;
        if (x2 < x1)
                incx = -1;
        incy = 1;
        if (y2 < yc1)
                incy = -1;
        x = x1;
        y = yc1;
        if (dx > dy)
        {
                draw_pixel(x, y);
                e = 2 * dy - dx;
                inc1 = 2 * (dy - dx);
                inc2 = 2 * dy;
                for (i = 0; i < dx; i++)
                {
                        if (e > 0)
                        {
                                y += incy;
                                e += inc1;
                        }
                        else
                                e += inc2;
                        x += incx;
                        draw_pixel(x, y);
                }
        }
        else
        {

                draw_pixel(x, y);
                e = 2 * dx - dy;
```

```
                inc1 = 2 * (dx - dy);
                inc2 = 2 * dx;
                for (i = 0; i < dy; i++)
                {
                        if (e > 0)
                        {
                                x += incx;
                                e += inc1;

                        }
                        else

                                e += inc2;
                        y += incy;

                        draw_pixel(x, y);
                }
        }
        glFlush();
}
void myinit()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(1, 1, 1, 1);
        gluOrtho2D(-250, 250, -250, 250);
}
void MyMouse(int button, int state, int x, int y)
{
        switch (button)
        {
        case GLUT_LEFT_BUTTON:
                if (state == GLUT_DOWN)
                {
                        if (flag == 0)
                        {
                                printf("Defining x1,y1");
                                x1 = x - 250;
                                yc1 = 250 - y;
                                flag++;
                                cout << x1 << " " << yc1 << " \n";
                        }
                        else
                        {
                                printf("Defining x2,y2");
                                x2 = x - 250;
                                y2 = 250 - y;
                                flag = 0;
                                cout << x2 << " " << y2 << " \n";
                                draw_line();

                        }

                }
                break;
        }
}
void display()
```

```
{}
int main(int ac, char* av[])
{
        /*
        //FOR KEYBOARD
        cout<<"X1\n";
        cin>>x1;
        cout<<"Y1\n";
        cin>>yc1;
        cout<<"X2\n";
        cin>>x2;
        cout<<"Y2\n";
        cin>>y2;
        //END KEYBOARD
        */
        glutInit(&ac, av);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 200);
        glutCreateWindow("LINE");
        myinit();
        glutMouseFunc(MyMouse); //INCLUDE TO USE MOUSE, REMOVE WHILE
USING KEYBOARD
        //draw_line(); //INCLUDE TO USE KEYBOARD, REMOVE WHILE USING
MOUSE
        glutDisplayFunc(display);
        glutMainLoop();
}
```


BZ

Name of the Experiment : ..... Date : | | | | | | | | |

Experiment No. : 12 ..... Experiment Result : .....

12. Write a program to construct Bezier Curve. Control points are supplied through keyboard/ mouse.

```
#include <iostream>
#include <math.h>
#include <gl/glut.h>
using namespace std;
float t, q, r, x[4], yc[4];
int flag=0;
void myInit(){
    glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);
    gluOrtho2D(0,500,0,500);
}
void drawPixel(float x, float y){
    glBegin(GL_POINTS);
    glVertex2f(x,y);
    glEnd();
}
void display().
{   glClear(GL_COLOR_BUFFER_BIT);
    int i;
    double t;
    glColor3f(0,0,0);
    glBegin(GL_POINTS);
    for(t=0; t<1; t=t+0.005){
```

Teachers sign .....

```
double xt= pow(1-t,3)*x1[0]+3*t*pow(1-t,2)*x1[1]
 + 3*pow(t,2)*(1-t)*x1[2]+pow(t,3)*x1[3];
double yt= pow(1-t,3)*yc[0]+3*t*pow(1-t,2)*yc[1]
+ 3* pow(t,2)*(1-t)*yc[2]+pow(t,3)* yc[3];
  glvertex2f(xt, yt);
}
{
 glColor3f(1,1,0);
 for(i=0; i<4; i++){
       glVertex2f(x1[i], yc[i]);
       glEnd();
       glFlush();
    }
}
void mymouse(int btn, int state, int x, int y)
{  if(btn == GLUT_LEFT_BUTTON && state = GLUT_DOWN
    && flag<4)
   {     x1[flag]=x;
         yc[flag]=500-y;
      cout<<"x: "<<x<<" y "<<500-y;
      glPointSize(3);
      glColor3f(1,1,0);
      glBegin(GL_POINTS);
      glVertex2i(x,500-y);
      glEnd();
       glFlush();
       flag++;
   }
```

Teachers sign .....

```
if (flag2==1 && btn==GLUT_LEFT_BUTTON)
{   glColor3f(0,0,1);
    display();
    flag=0;
  }
}
int main(int argc, char* argv[]){
    glutInit(&argc, argv);
cout<<"Enter x coordinates";
cin>>x1[0]>>x1[1]>>x1[2]>>x1[3]
cout<<"Enter y coordinates";
cin>>y[0]>>y[1]>>y[2]>>y[3];
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("BZ");
glutDisplayFunc(display);
glutMouseFunc(mymouse);
myInit();
glutMainLoop();
}
```

```
#include<iostream>
#include<math.h>
#include<gl/glut.h>

using namespace std;
float f, g, r, x1[4], yc[4];
int flag = 0;
void myInit() {

        glClearColor(1, 1, 1, 1);
        glColor3f(1, 1, 1);
        glPointSize(5);
        gluOrtho2D(0, 500, 0, 500);
}

void drawPixel(float x, float y) {
        glBegin(GL_POINTS);
        glVertex2f(x, y);
        glEnd();
}

void display() {

        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double t;
        glColor3f(0, 0, 0);
        glBegin(GL_POINTS);
        for (t = 0; t < 1; t = t + 0.005) {
                double xt = pow(1 - t, 3) * x1[0] + 3 * t * pow(1 - t,
 2) * x1[1] + 3 * pow(t, 2) * (1 - t) * x1[2] + pow(t, 3) * x1[3];
                double yt = pow(1 - t, 3) * yc[0] + 3 * t * pow(1 - t,
 2) * yc[1] + 3 * pow(t, 2) * (1 - t) * yc[2] + pow(t, 3) * yc[3];
                glVertex2f(xt, yt);

        }
        glColor3f(1, 1, 0);
        for (i = 0; i < 4; i++) {
                glVertex2f(x1[i], yc[i]);
                glEnd();
                glFlush();
        }


}
void mymouse(int btn, int state, int x, int y)
{
        if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN && flag < 4)
        {
                x1[flag] = x;
                yc[flag] = 500 - y;
                cout << " X: " << x << " Y" << 500 - y;
                glPointSize(3);
                glColor3f(1, 1, 0);
                glBegin(GL_POINTS);
                glVertex2i(x, 500 - y);
```

```
            glEnd();
            glFlush();
            flag++;

        }
        if (flag >= 4 && btn == GLUT_LEFT_BUTTON)
        {
            glColor3f(0, 0, 1);
            display();
            flag = 0;


        }

}
int main(int argc, char* argv[]) {
        glutInit(&argc, argv);

        /*
        //USE KEYBOARD
        cout << "Enter the x co-ordinates";
        cin >> x1[0] >> x1[1] >> x1[2] >> x1[3];
        cout << "Enter y co-ordinates";
        cin >> yc[0] >> yc[1] >> yc[2] >> yc[3];
        //END KEYBOARD
        */
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("BZ");
        glutDisplayFunc(display);
        glutMouseFunc(mymouse); //INCLUDE FOR MOUSE, REMOVE FOR
KEYBOARD
        myInit();
        glutMainLoop();
}
```