CS352 Assignment 1
openGL init

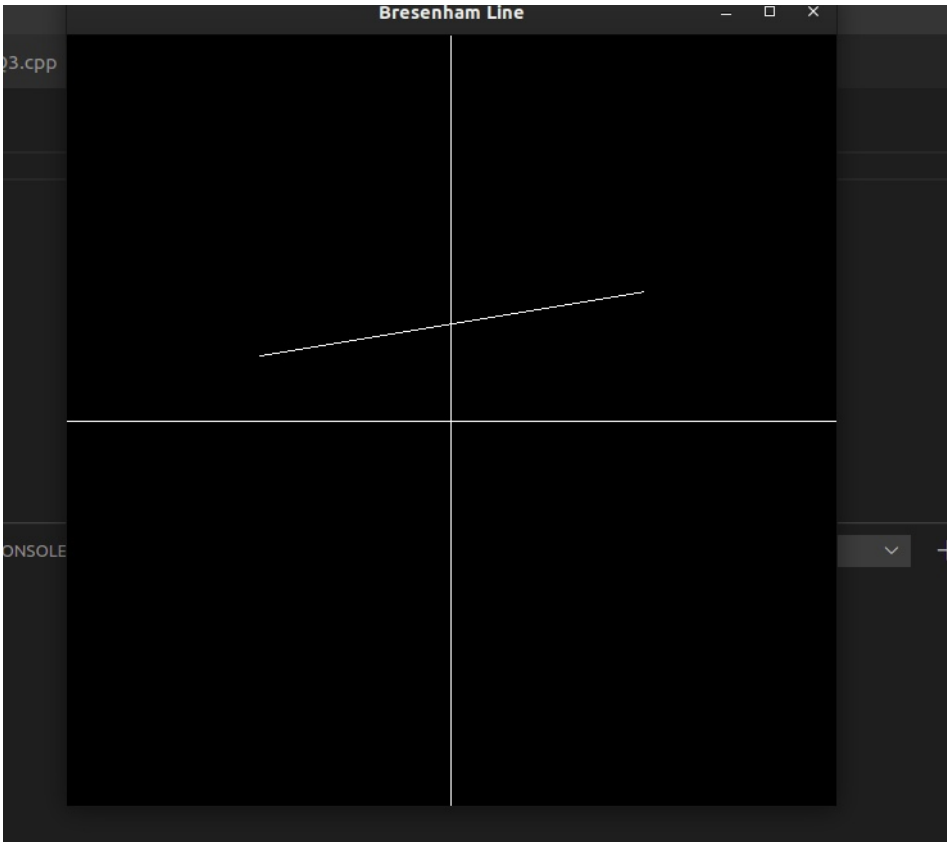
Ayush Agrawal
180001011
01-04-2021

Q1. Bresenham Line - Code

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
// #include <utility>
using namespace std;

const double WIDTH = 600;
const double HEIGHT = 600;

vector<pair<int,int>> points;

#define inRange(x1,x,x2) ((x>min(x1,x2))&&(x<max(x1,x2)))

vector<pair<int,int>> makeBresenham(int x1, int y1, int x2, int y2){
   vector<pair<int,int>> points;
   points.push_back({x1,y1});

         int dx = x2-x1;
         int dy = y2-y1;

   bool mInv = 0;

   if(abs(dy) > abs(dx)){
      mInv = 1;
      swap(x1, y1);
      swap(x2, y2);
      swap(dx, dy);
   }

   int stepX = (dx > 0);
   int stepY = (dy > 0);

   if(dx < 0) {
      stepX = -1;
      dx = -dx;
   }
   if(dy < 0) {
      stepY = -1;
      dy = -dy;
   }

         int x = x1;
```

```
        int y = y1;

        int p = 2*dy-dx;

        while( x != x2 ){
                if(p >= 0){
                        y += stepY;
                        p -= 2*dx;
                }

      // cout<<x<<" "<<y<<"\n";
      if(mInv){
         points.push_back({y,x});
      } else {
         points.push_back({x,y});
      }
      p += 2*dy;
                x += stepX;
         }

   return points;
}

void initBresenham(){
   cout<<"Enter the coordinates of the line - x1 y1 x2 y2\n";
   cout<<"For the sake of viewing, keep the points in the range -600,600\n";

   int x1 = -400, y1 = 50;
   int x2 = 400, y2 = 200;

   cin>>x1>>y1>>x2>>y2;

   points = makeBresenham(x1, y1, x2, y2);
}


void drawBresenham(){
   glClear(GL_COLOR_BUFFER_BIT);

   glBegin(GL_LINES);
      glVertex2f(-1, 0);
      glVertex2f(1, 0);
   glEnd();

   glBegin(GL_LINES);
      glVertex2f(0, -1);
      glVertex2f(0, 1);
   glEnd();

   glBegin(GL_POINTS);
      for(auto &point: points)
         glVertex2f((double)point.first/WIDTH, (double)point.second/HEIGHT);
```

```
    glEnd();


    glFlush();
}

int main(int argc, char** argv){
    initBresenham();
    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Bresenham Line");

    // Register display callback
    glutDisplayFunc(drawBresenham);

    // Glut main Loop
    glutMainLoop();
}
```

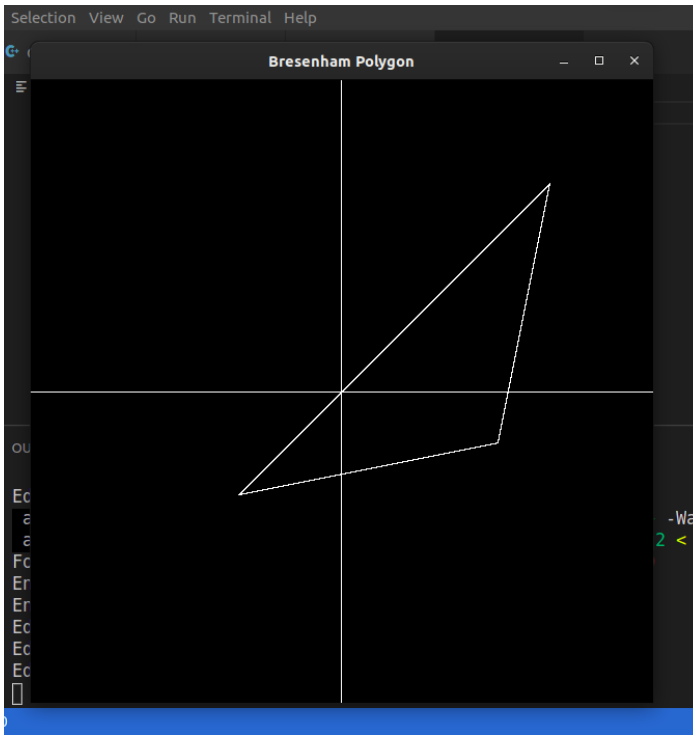


Q2. Polygon with Bresenham - Code

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
// #include <utility>
using namespace std;

const double WIDTH = 600;
const double HEIGHT = 600;

vector<vector<pair<int,int>>> edges;

vector<pair<int,int>> makeBresenham(int x1, int y1, int x2, int y2){
   vector<pair<int,int>> points;
   points.push_back({x1,y1});

         int dx = x2-x1;
         int dy = y2-y1;

   bool mInv = 0;

   if(abs(dy) > abs(dx)){
      mInv = 1;
      swap(x1, y1);
      swap(x2, y2);
      swap(dx, dy);
   }

   int stepX = (dx > 0);
   int stepY = (dy > 0);

   if(dx < 0) {
      stepX = -1;
      dx = -dx;
   }
   if(dy < 0) {
      stepY = -1;
      dy = -dy;
   }

         int x = x1;
         int y = y1;

         int p = 2*dy-dx;

         while( x != x2 ){
                  if(p >= 0){
                           y += stepY;
                           p -= 2*dx;
                  }

      // cout<<x<<" "<<y<<"\n";
```

```
        if(mInv){
            points.push_back({y,x});
        } else {
            points.push_back({x,y});
        }
        p += 2*dy;
                    x += stepX;
            }

    return points;
}

void initBresenham(){
    cout<<"For the sake of viewing, keep the points in the range -600,600\n";

    cout<<"Enter the Number of vertices\n";
    int n;cin>>n;
    if(n < 3) {
        cout<<"Atleast 3 vertices\n";
    }
    edges.resize(n);
    vector<pair<int,int>> points = vector<pair<int,int>>(n);


    cout<<"Enter Each vertice one by one\n";

    for(int i=0;i<n;i++){
        cin>>points[i].first>>points[i].second;
    }
    for(int i=0;i<n;i++){
        int x1 = points[i].first;
        int y1 = points[i].second;

        int x2 = points[(i+1)%n].first;
        int y2 = points[(i+1)%n].second;

        cout<<"Edge from: "<<x1<<","<<y1<<" to "<<x2<<","<<y2<<"\n";
        edges[i] = makeBresenham(x1,y1,x2,y2);
    }



}


void drawBresenham(){
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
        glVertex2f(-1, 0);
        glVertex2f(1, 0);
    glEnd();
```

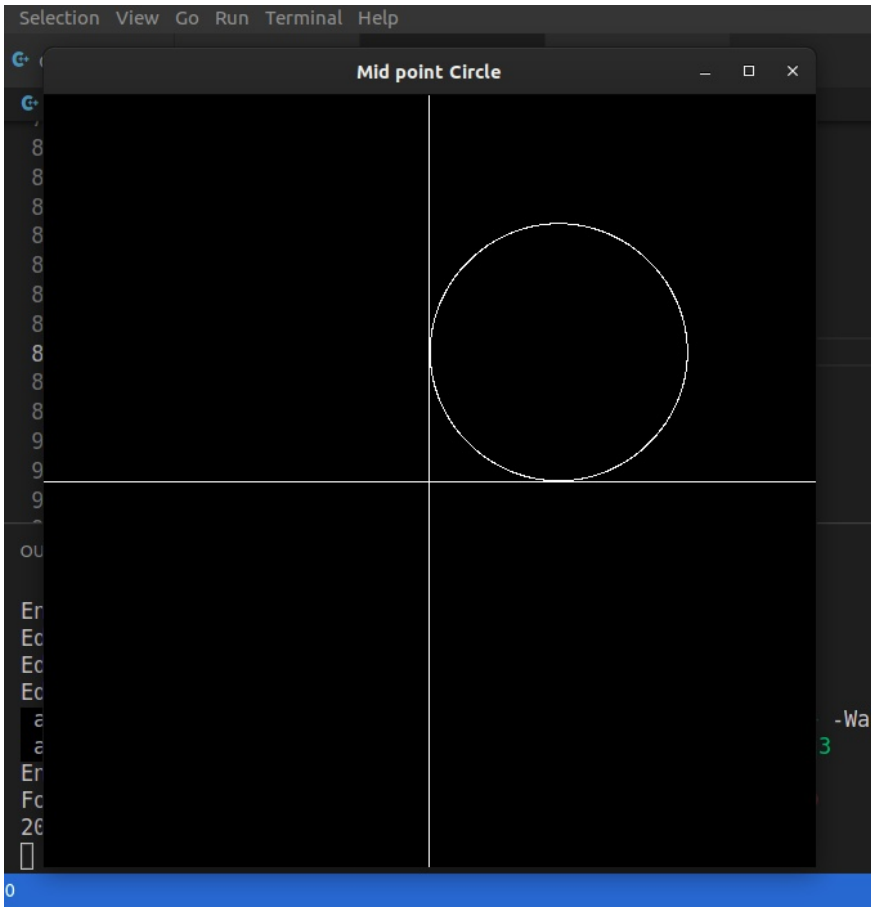
```
    glBegin(GL_LINES);
        glVertex2f(0, -1);
        glVertex2f(0, 1);
    glEnd();

    glBegin(GL_POINTS);
        for(auto &edge: edges){
            for(auto &point: edge)
                glVertex2f((double)point.first/WIDTH, (double)point.second/HEIGHT);
        }
    glEnd();


    glFlush();
}

int main(int argc, char** argv){
    initBresenham();

    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Bresenham Polygon");

    // Register display callback
    glutDisplayFunc(drawBresenham);

    // Glut main Loop
    glutMainLoop();
}
```



Q3. Circle using mid-point algo - Code.

```
#include <iostream>
#include <GL/glut.h>
#include <vector>
// #include <utility>
using namespace std;

const double WIDTH = 600;
const double HEIGHT = 600;

vector<pair<int,int>> points;

#define inRange(x1,x,x2) ((x>min(x1,x2))&&(x<max(x1,x2)))

void push_octant(int x, int y, int xc, int yc){
   points.push_back({xc+x,xc+y});
   points.push_back({xc+y,xc+x});
   points.push_back({xc+y,xc-x});
   points.push_back({xc+x,xc-y});
   points.push_back({xc-x,xc-y});
   points.push_back({xc-y,xc-x});
   points.push_back({xc-y,xc+x});
   points.push_back({xc-x,xc+y});
}

void makeCircle(int xc, int yc, int r){
   int i=0;
   int x = 0, y = r;
   int P = 5-4*r;
   while(x<y) {
      x++;i++;
      if(P<=0) {
         P = P+8*x+12;
      } else {
         y=y-1;
         P = P+8*(x-y)+20;
      }
      push_octant(x,y, xc, yc);
   }
}

void initCircle(){
   cout<<"Enter the dim of circle - x_center y_center radius\n";
   cout<<"For the sake of viewing, keep the points in the range -600,600\n";

   int x0 = 200, y0 = 200;
   int r = 200;

   cin>>x0>>y0>>r;
```

```
    makeCircle(x0,y0,r);
}


void drawCircle(){
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
        glVertex2f(-1, 0);
        glVertex2f(1, 0);
    glEnd();

    glBegin(GL_LINES);
        glVertex2f(0, -1);
        glVertex2f(0, 1);
    glEnd();

    glBegin(GL_POINTS);
        for(auto &point: points)
            glVertex2f((double)point.first/WIDTH, (double)point.second/HEIGHT);
    glEnd();


    glFlush();
}

int main(int argc, char** argv){
    initCircle();
    // GLute init and create window
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(WIDTH,HEIGHT);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Mid point Circle");

    // Register display callback
    glutDisplayFunc(drawCircle);

    // Glut main Loop
    glutMainLoop();
}
```


Selection View Go Run Terminal Help
Mid point Circle