COMPUTER GRAPHICS

LAB PRACTICALS RECORD

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR – 144011, PUNJAB (INDIA)

Submitted To:

Ms.

Asst. Professor
Department of CSE

Submitted By:

Nikhil Bansal

13103011
6th Semester

INDEX

PROGRAM 1

DDA LINE ALGO

Description:

DDA line algorithm is a basic algorithm used to draw line. It requires more computations than other algorithms.

Program:

```
// dda algorithm for line drawing.
// Provide the line coordinates at commandline.

#include<graphics.h>
#include<stdio.h>

int main(int argc,char *argv[])
{
  if(argc<5){
    printf("Enter coordinates of end points of line on commandine\n");
    return -1;
  }

  //coordinates output file
  FILE *coordinates=fopen("coordinates", "w");

  //commandline input
  int i,j,x1,x2,y1,y2;
  float currx,curry;
  x1=atoi(argv[1]);
  y1=atoi(argv[2]);
  x2=atoi(argv[3]);
  y2=atoi(argv[4]);

  //if coordinates are not in increasing order of x then make them
  if(x1>x2){
    int temp=x1;
    x1=x2;
    x2=temp;
    temp=y1;
    y1=y2;
    y2=temp;
  }

  //graphics initialise
  int gd = DETECT,gm;
```

```
initgraph(&gd,&gm,NULL);

//draw line using line function to check the correctness of the algo
line(x1,y1,x2,y2);

//find the slope
float m=((float)y2-y1)/(x2-x1);

if(m<=1&&m>=-1){
  putpixel(x1,y1,RED);
  currx=x1;
  curry=y1;
  while(currx!=x2){
    currx+=1;
    curry=curry+m;
    fprintf(coordinates, "%d %d\n",(int)currx,(int)curry);
    putpixel((int)currx,(int)curry,RED);
  }
}

if(m>1||m<-1){
  if(m>1){
    putpixel(x1,y1,RED);
    currx=x1;
    curry=y1;
    while(curry!=y2){
      curry+=1;
      currx=currx+1/m;
      fprintf(coordinates,"%d %d\n",(int)currx,(int)curry);
      putpixel((int)currx,(int)curry,RED);
    }
  }
  else{
    putpixel(x2,y2,RED);
    currx=x2;
    curry=y2;
    while(curry!=y1){
      curry+=1;
      currx=currx+1/m;
      fprintf(coordinates, "%d %d\n",(int)currx,(int)curry);
      putpixel((int)currx,(int)curry,RED);
    }
  }
}

delay(5000);
closegraph();
```

```
  return 0;
}
```

PROGRAM 2

BRESNHAN'S LINE ALGO

Description:

It is another line drawing algorithm. It is much more efficent than DDA algorithm. It also draws smooth line.

Program:

```
//Bresnham's Line algorithm

#include<graphics.h>
#include<stdio.h>

// absolute i.e mod of x
int abs(int x){
   if(x<0)
      return -x;
   else
      return x;
}

int main(int argc,char *argv[]){
   // command line arguments check
   if(argc<5){
      printf("Enter coordinates of end points of line on commandine\n");
      return -1;
   }

   // coordinates output file
   FILE *coordinates=fopen("coordinates", "w");

   // commandline input
   int x1,x2,y1,y2;
   int p_curr,currx,curry;
   x1=atoi(argv[1]);
   y1=atoi(argv[2]);
   x2=atoi(argv[3]);
   y2=atoi(argv[4]);

   //if coordinates are not in increasing order of x then make them
   if(x1>x2){
      int temp=x1;
      x1=x2;
      x2=temp;
      temp=y1;
```

```
  y1=y2;
  y2=temp;
}

// Initialise graphics
int gd = DETECT,gm;
initgraph(&gd,&gm,NULL);

//slope
float m=((float)y2-y1)/(x2-x1);
int dx=abs(x2-x1);
       int dy=abs(y2-y1);
putpixel(x1, y1, RED);

// algorithm
       if(m<=1 && m>=-1){
                currx=x1;
                curry=y1;
                p_curr=2*dy-dx;
       putpixel(x1,y1,RED);
  if(m>=0){
       for(currx=x1+1;currx<=x2;currx++){
                if(p_curr>=0){
                         curry++;
                         p_curr=p_curr+2*dy-2*dx;
                }
                else{
                         p_curr=p_curr+2*dy;
                }
      fprintf(coordinates,"%d %d\n",currx,curry);
                putpixel(currx,curry,RED);
       }
  }
  else{
    for(currx=x1+1;currx<=x2;currx++){
      if(p_curr>=0){
        curry--;
        p_curr=p_curr+2*dy-2*dx;
      }
      else{
        p_curr=p_curr+2*dy;
      }
      fprintf(coordinates,"%d %d\n",currx,curry);
      putpixel(currx,curry,RED);
    }
  }
       }
  // delay to able to view graphics
          delay(5000);
          return 0;
}
```

PROGRAM 3

TRIGNOMETRIC CIRCLE

Description:

It is used to draw circle with given center and radius. In this algo, we basically find the coordinates by using the trignometry formulas. We find the x and y coordinates by:

x = r * cos(angle)
y = r * sin(angle)

Program:

```
// Trignometric Algo for drawing the circle

#include<graphics.h>
#include<math.h>
#include<stdio.h>

// draw the circle with given integer center and radius
int trignometricCircle(int x,int y,int radius){
  float curr_x,curr_y;
  int angle;
  FILE *coordinates=fopen("coordinates", "w");

  // algo
  for(angle=0;angle<360;angle++){
    curr_x=x+cos((float)angle/180*3.14)*radius;
    curr_y=y+sin((float)angle/180*3.14)*radius;
    putpixel((int)curr_x,(int)curr_y,RED);
    fprintf(coordinates, "%d %d\n", (int)curr_x, (int)curr_y);
  }
  return 0;
}

int main(int argc,char *argv[]){

  //command-line parameters check
  if(argc<3){
    printf("Enter 3 arguments on commandine\n");
    return 0;
  }

  //graphics initialisation
  int gd = DETECT,gm;
  initgraph(&gd,&gm,NULL);
  //get the center and radius
  int x,y,radius;
  x=atoi(argv[1]);
  y=atoi(argv[2]);
  radius=atoi(argv[3]);

  //Circle drawn from inbuilt library to check performance of ours
  putpixel(x,y,YELLOW);
  circle(x,y,radius);

  //Draw the circle using Trignometric algo
  trignometricCircle(x,y,radius);

  //delay so as to view the screen
  delay(5000);
        return 0;
}
```

PROGRAM 4

MID POINT CIRCLE

Description:

It is used to draw circle with given center and radius. It is more efficient than trignometric circle algorithm. It doesnot use any trignometric functions thus it more efficient.

Program:

```
// Circle using mid point algorithm with float radius and center
#include<stdio.h>
#include<graphics.h>
#include<math.h>

int midPointCircle(float x,float y,float radius){
        //coordinates output file
        FILE *coordinates=fopen("coordinates", "w");

        float pinit,pcurr;
        int curr_x,curr_y;

        // calculate the initial decision parameter
        if(floor(radius)-radius==0)
                pinit=1-radius;
        else
                pinit=5.00/4-radius;

        // initialisations
        curr_x=0;
        curr_y=floor(radius);
        pcurr=pinit;

        // operate while loop until x<y
        while(curr_x<=curr_y){
                // output points
                putpixel((int)(curr_x+x),(int)(curr_y+y),RED);
                putpixel((int)(-curr_x+x),(int)(curr_y+y),RED);
                putpixel((int)(curr_x+x),(int)(-curr_y+y),RED);
                putpixel((int)(-curr_x+x),(int)(-curr_y+y),RED);
                putpixel((int)(curr_y+y),(int)(curr_x+x),RED);
                putpixel((int)(-curr_y+y),(int)(curr_x+x),RED);
                putpixel((int)(curr_y+y),(int)(-curr_x+x),RED);
                putpixel((int)(-curr_y+y),(int)(-curr_x+x),RED);
                fprintf(coordinates,"%d %d\n",(int)(curr_x+x),(int)(curr_y+y));
                fprintf(coordinates,"%d %d\n",(int)(-curr_x+x),(int)(curr_y+y));
                fprintf(coordinates,"%d %d\n",(int)(curr_x+x),(int)(-curr_y+y));
                fprintf(coordinates,"%d %d\n",(int)(-curr_x+x),(int)(-curr_y+y));
                fprintf(coordinates,"%d %d\n",(int)(curr_y+y),(int)(curr_x+x));
                fprintf(coordinates,"%d %d\n",(int)(-curr_y+y),(int)(curr_x+x));
                fprintf(coordinates,"%d %d\n",(int)(curr_y+y),(int)(-curr_x+x));
                fprintf(coordinates,"%d %d\n",(int)(-curr_y+y),(int)(-curr_x+x));

                // algo
                if(pcurr<0){
                        curr_x+=1;
                        pcurr=pcurr+2*curr_x+1;
                }
                else{
                        curr_x+=1;
                        curr_y-=1;
                        pcurr=pcurr+2*curr_x+1-2*curr_y;
                }
        }

        // close the output file
        fclose(coordinates);
        return 0;
}

int main(int argc,char *argv[]){

  //command-line parameters check
  if(argc<3){
    printf("Enter 3 arguments on commandine\n");
    return 0;
  }

        //get the center and radius
  float x,y,radius;
  x=atoi(argv[1]);
  y=atoi(argv[2]);
  radius=atof(argv[3]);

  // check if x and y are greater than radius else pixel out of range will be there
  if(x<radius||y<radius){
                printf("Circle cannot be displayed\nAs x and y are less than radius so there will be pixel out of
range.\n");
                return 0;
  }

  //graphics initialisation
  int gd = DETECT,gm;
```

```
    initgraph(&gd,&gm,NULL);

    //Circle drawn from inbuilt library to check performance of ours
    putpixel(x,y,YELLOW);
    circle((int)x,(int)y,(int)radius);

    //Draw the circle using Trignometric algo
    midPointCircle(x,y,radius);

    //delay so as to view the screen
    delay(5000);
            return 0;
} //delay so as to view the screen
   delay(5000);
            return 0;
}
```

PROGRAM 5

TRIGNOMETRIC ELLIPSE

Description:

It is used to draw ellipse. It is less efficient than trignometric ellipse algorithm. In this algo, we basically find the coordinates by using the trignometry formulas. We find the x and y coordinates by:

x = a * cos(angle)
y = b * sin(angle)

Program:

```
// Trignometric Algo for drawing the ellipse

#include<graphics.h>
#include<math.h>
#include<stdio.h>

// draw the circle with given integer center and axes
int trignometricEllipse(int x,int y,int a,int b){
  float curr_x,curr_y;
  int angle;
  FILE *coordinates=fopen("coordinates", "w");

  // algo
  for(angle=0;angle<360;angle++){
    curr_x=x+a*cos((float)angle/180*3.14);
    curr_y=y+b*sin((float)angle/180*3.14);
    putpixel((int)curr_x,(int)curr_y,RED);
    fprintf(coordinates, "%d %d\n", (int)curr_x, (int)curr_y);
  }

  fclose(coordinates);
  return 0;
}

int main(int argc,char *argv[]){

  //command-line parameters check
  if(argc<4){
    printf("Enter 4 arguments on commandine\n");
    return 0;
  }

  //get the center and radius
  int x,y,a,b;
```

```
   x=atoi(argv[1]);
   y=atoi(argv[2]);
   a=atoi(argv[3]);
   b=atoi(argv[4]);

   // check for pixel out of range
   if(x<a||y<b){
      printf("Enter center of ellipse such that center points are less than a and b.\nElse therer will be pixel out of
range.\n");
      return 0;
   }

   //graphics initialisation
   int gd = DETECT,gm;
   initgraph(&gd,&gm,NULL);

   //Ellipse drawn from inbuilt library to check performance of ours
   putpixel(x,y,YELLOW);
   ellipse(x,y,0,360,a,b);

   //Draw the ellipse using Trignometric algo
   trignometricEllipse(x,y,a,b);

   //delay so as to view the screen
   delay(5000);
             return 0;
}
```