Interactive Computer Graphics (Assignment –II)

Submitted in partial fulfilment of the requirements for the degree of

Master of Technology in Information Technology

by

Vijayananda D Mohire

(Enrolment No.921DMTE0113)

Information Technology Department

Karnataka State Open University

Manasagangotri, Mysore – 570006

Karnataka, India

(2009)

Interactive Computer Graphics

CERTIFICATE

This is to certify that the Assignment-II entitled (Interactive Computer Graphics, subject code: MT11) submitted by Vijayananda D Mohire having Roll Number 921DMTE0113 for the partial fulfilment of the requirements of Master of Technology in Information Technology degree of Karnataka State Open University, Mysore, embodies the bonafide work done by him under my supervision.

Place: _______________ **Signature of the Internal Supervisor**

Name

Date: _______________ **Designation**

For Evaluation

Question Number	**Maximum Marks**	**Marks awarded**	**Comments, if any**
1	5		
2	5		
TOTAL	10		

Evaluator's Name and Signature　　　　　Date

Preface

This document has been prepared specially for the assignments of M.Tech – IT I Semester. This is mainly intended for evaluation of assignment of the academic M.Tech - IT, I semester. I have made a sincere attempt to gather and study the best answers to the assignment questions and have attempted the responses to the questions. I am confident that the evaluator's will find this submission informative and evaluate based on the provide content.

For clarity and ease of use there is a Table of contents and Evaluators section to make easier navigation and recording of the marks. A list of references has been provided in the last page – Bibliography that provides the source of information both internal and external. Evaluator's are welcome to provide the necessary comments against each response, suitable space has been provided at the end of each response.

I am grateful to the Infysys academy, Koramangala, Bangalore in making this a big success. Many thanks for the timely help and attention in making this possible within specified timeframe. Special thanks to Mr. Vivek and Mr. Prakash for their timely help and guidance.

Candidate's Name and Signature Date

Table of Contents

Table of Figures

INTERACTIVE COMPUTER GRAPHICS

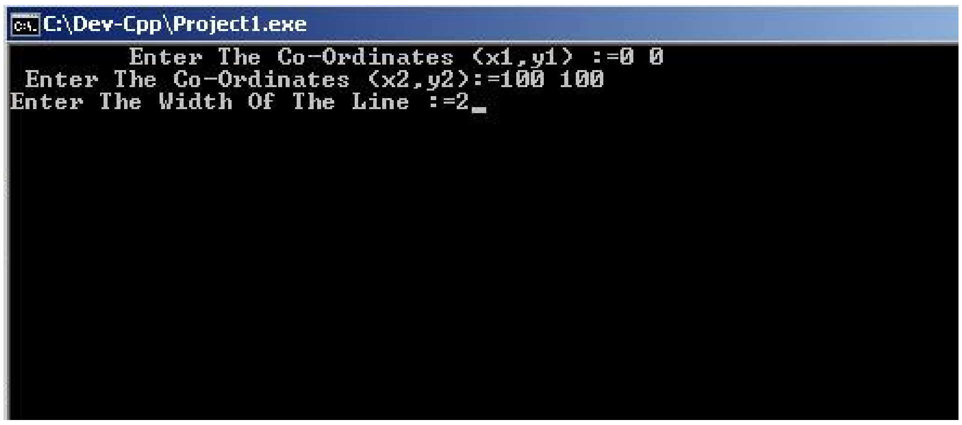
RESPONSE TO ASSIGNMENT - II

Question 1 Write any two program in C/C++ for different line drawing algorithms? Also explain me O/P obtained?

Answer 1(a)

DDA Algorithm for Line drawing

Input:



C:\Dev-Cpp\Project1.exe

```
        Enter The Co-Ordinates (x1,y1) :=0 0
 Enter The Co-Ordinates (x2,y2):=100 100
Enter The Width Of The Line :=2_
```

Figure 1 DDA Input for Line (Colin, 1991)

Output:

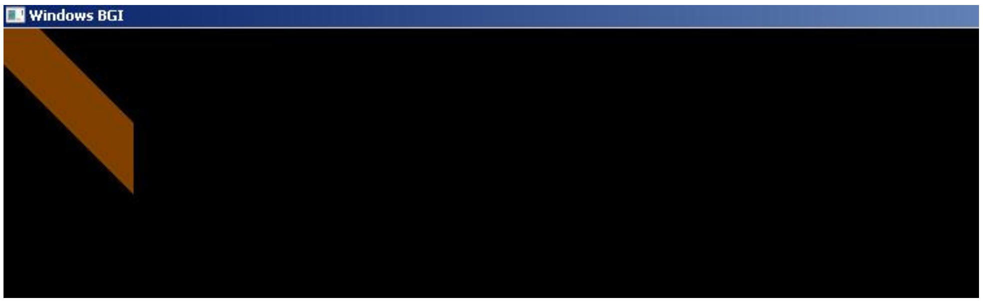
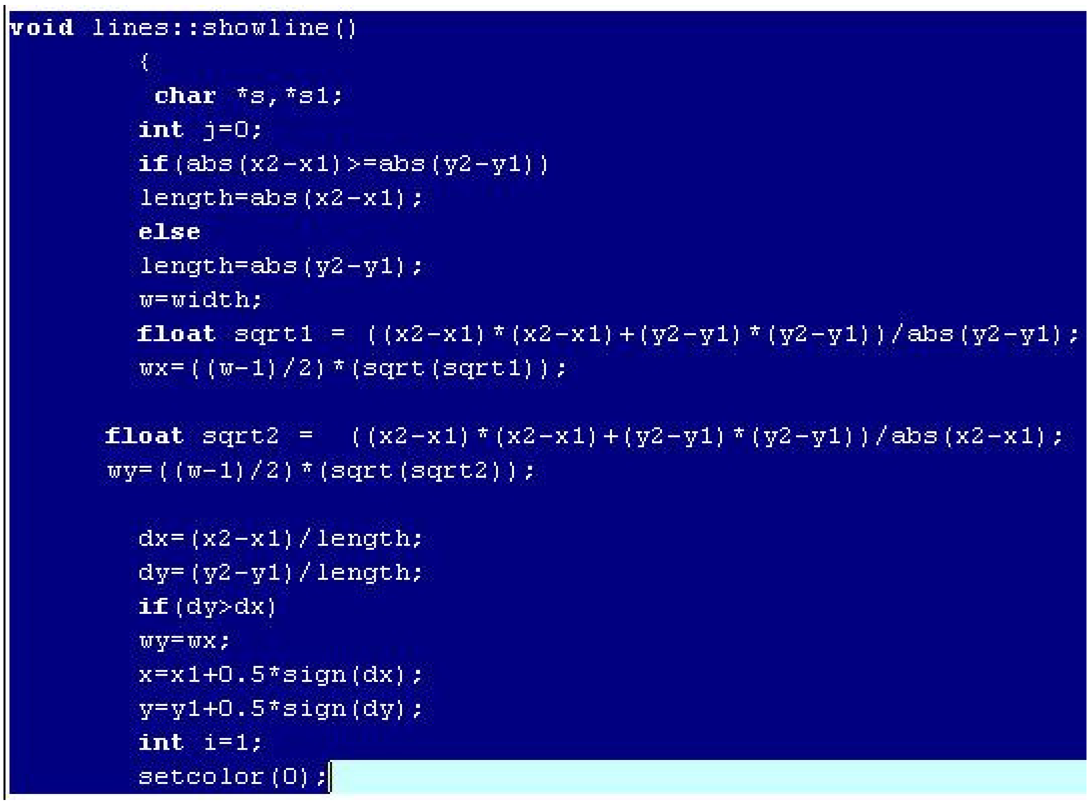


Figure 2 DDA Output for Line (Colin, 1991)

Explanation:

Below provides the method used to demonstrate the DDA Algorithm. More details can be obtained from the C++ code provided in Appendix**.**



```
void lines::showline()
        {
         char *s,*s1;
        int j=0;
        if(abs(x2-x1)>=abs(y2-y1))
        length=abs(x2-x1);
        else
        length=abs(y2-y1);
        w=width;
        float sqrt1 = ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(y2-y1);
        wx=((w-1)/2)*(sqrt(sqrt1));

      float sqrt2 =  ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(x2-x1);
      wy=((w-1)/2)*(sqrt(sqrt2));

        dx=(x2-x1)/length;
        dy=(y2-y1)/length;
        if(dy>dx)
        wy=wx;
        x=x1+0.5*sign(dx);
        y=y1+0.5*sign(dy);
        int i=1;
        setcolor(0);
```

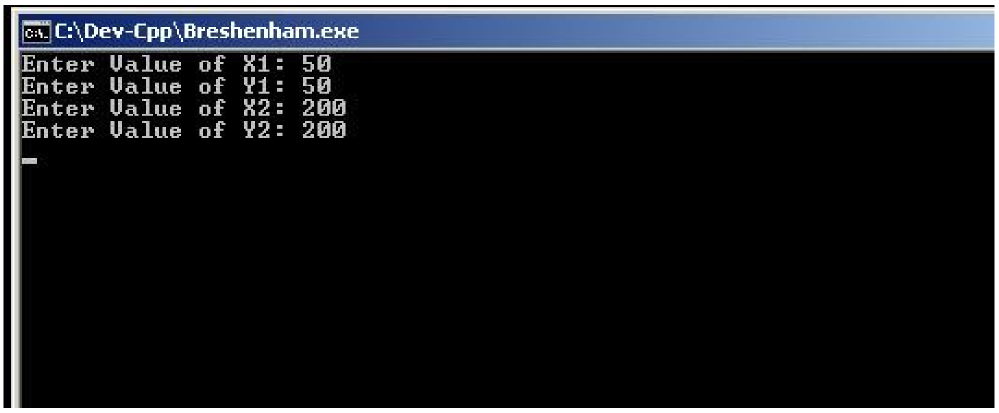
Figure 3 DDA Algorithm (Anonymous, C++ > Computer Graphics sample source codes, 2009)

From above code it is illustrated that the DDA provides fair amount of LINE drawing.

Answer 1(b)

Bresenham Algorithm for Line drawing:

Input:



```
C:\Dev-Cpp\Breshenham.exe
Enter Value of X1: 50
Enter Value of Y1: 50
Enter Value of X2: 200
Enter Value of Y2: 200
_
```

Figure 4 Bresenham Input for Line (Colin, 1991)

Output:

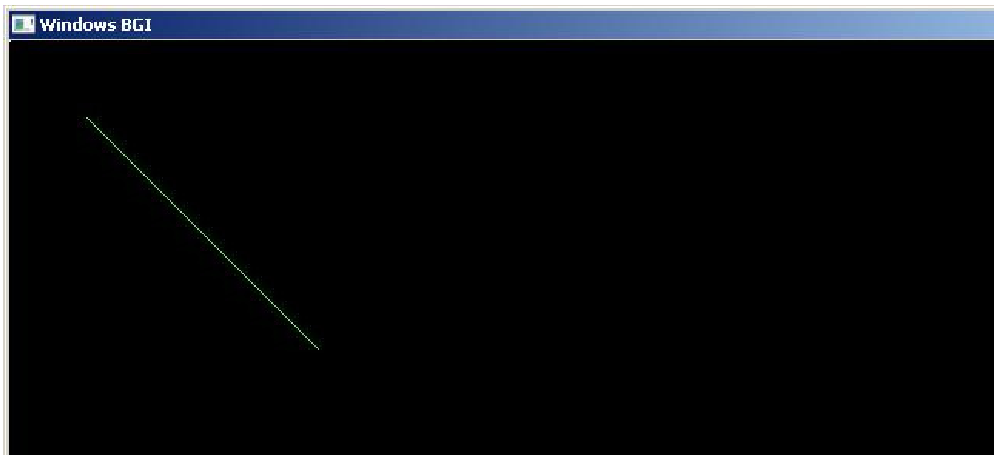
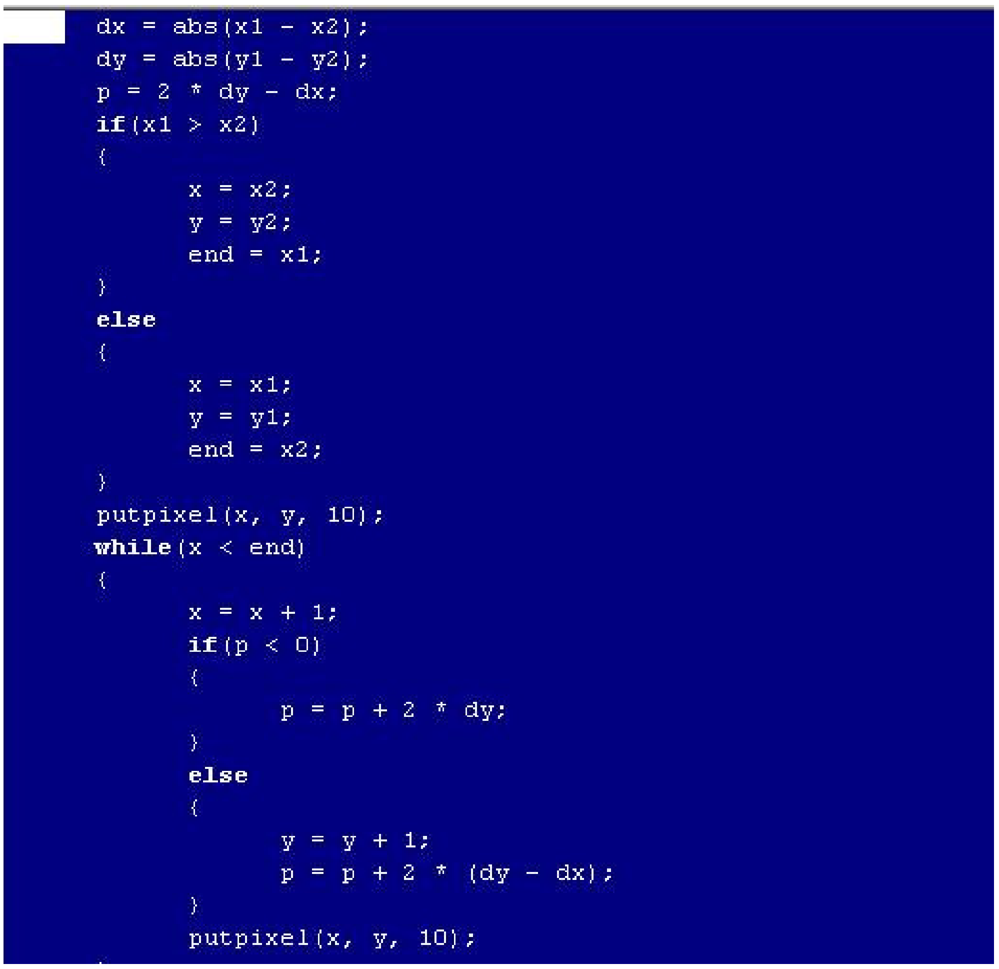


Figure 5 Bresenham Output for Line (Colin, 1991)

Explanation:

Below provides the method used to demonstrate the Bresenham Algorithm. More details can be obtained from the C++ code provided in Appendix.



```
dx = abs(x1 - x2);
dy = abs(y1 - y2);
p = 2 * dy - dx;
if(x1 > x2)
{
        x = x2;
        y = y2;
        end = x1;
}
else
{
        x = x1;
        y = y1;
        end = x2;
}
putpixel(x, y, 10);
while(x < end)
{
        x = x + 1;
        if(p < 0)
        {
                p = p + 2 * dy;
        }
        else
        {
                y = y + 1;
                p = p + 2 * (dy - dx);
        }
        putpixel(x, y, 10);
```

Figure 6 Bresenham Algorithm (Anonymous, C > Games and Graphics sample

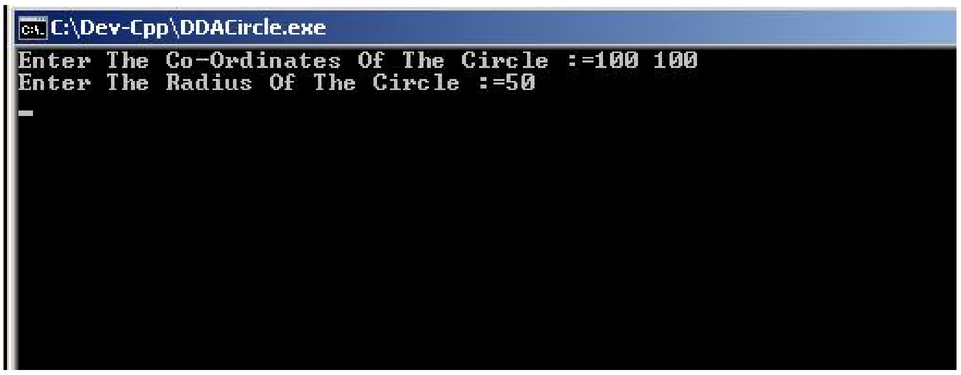
source codes, 2009)

Evaluator’s Comments if any:

Question 2 Write a program in C/C++ for circle generating algorithm?

Answer 2

Input:



```
C:\Dev-Cpp\DDACircle.exe
Enter The Co-Ordinates Of The Circle :=100 100
Enter The Radius Of The Circle :=50
```

Figure 7 DDA Circle input (Colin, 1991)

Output:

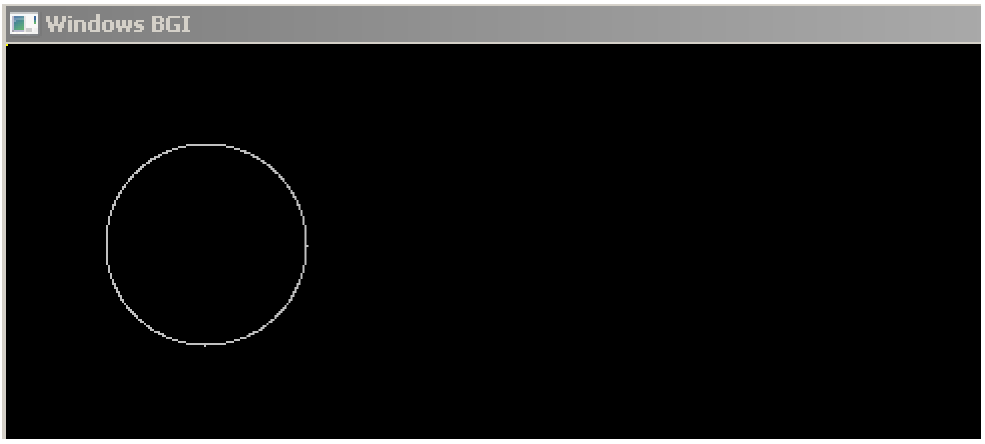
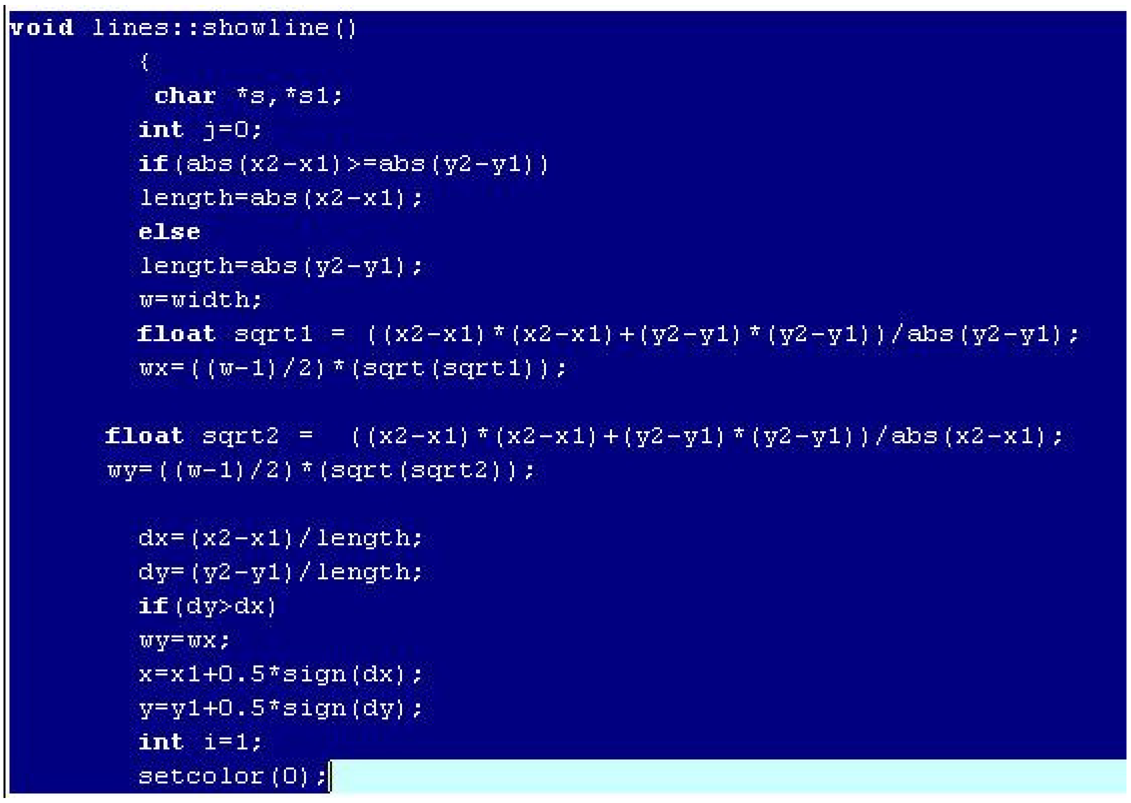


Figure 8 DDA Circle output (Colin, 1991)

Explanation:



```
void lines::showline()
        {
         char *s,*s1;
        int j=0;
        if(abs(x2-x1)>=abs(y2-y1))
        length=abs(x2-x1);
        else
        length=abs(y2-y1);
        w=width;
        float sqrt1 = ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(y2-y1);
        wx=((w-1)/2)*(sqrt(sqrt1));

      float sqrt2 =  ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(x2-x1);
      wy=((w-1)/2)*(sqrt(sqrt2));

        dx=(x2-x1)/length;
        dy=(y2-y1)/length;
        if(dy>dx)
        wy=wx;
        x=x1+0.5*sign(dx);
        y=y1+0.5*sign(dy);
        int i=1;
        setcolor(0);
```

Figure 9 DDA Algorithm for Circle (Anonymous, C++ > Computer Graphics sample source codes, 2009)

Evaluator’s Comments if any:

Appendix Complete Code

DDA Line Algorithm

```
#include "winbgim.h"
#include <iostream>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
#include<stdio.h>
using namespace std;

class lines
      {
      private:
      int length,x1,y1,x2,y2,x,y,dx,dy,wx,wy,w,width;
      public:
      lines();      //Constructor
      void showline();
      int sign(int);
      };


int lines::sign(int xx)
            {
            if(xx<0)
            return -1;
            if(xx==0)
            return 0;
            if(xx>0)
            return 1;
            return 0;
            }
lines::lines()
            {
            x=0;y=0;
```

```
        cout<<"     Enter The Co-Ordinates (x1,y1) :=";

      //      cout<<" Enter The Co-Ordinates (x1,y1):=";
              cin >> x1 >> y1;

              cout << " Enter The Co-Ordinates (x2,y2):=";
              cin >> x2 >> y2;

              cout<<"Enter The Width Of The Line :=";
              cin >> width;

              }
void lines::showline()
              {
           char *s,*s1;
              int j=0;
              if(abs(x2-x1)>=abs(y2-y1))
              length=abs(x2-x1);
              else
              length=abs(y2-y1);
              w=width;
              float sqrt1 = ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(y2-y1);
              wx=((w-1)/2)*(sqrt(sqrt1));

          float sqrt2 =  ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))/abs(x2-x1);
        wy=((w-1)/2)*(sqrt(sqrt2));

              dx=(x2-x1)/length;
              dy=(y2-y1)/length;
              if(dy>dx)
              wy=wx;
              x=x1+0.5*sign(dx);
              y=y1+0.5*sign(dy);
              int i=1;
              setcolor(0);

              while(i<=length)
                    {
                    for(j=0;j<wy;j++)
              putpixel((x),(y+j),6);
                    for(j=0;j<wy;j++)

              putpixel((x),(y-j),6);
                  putpixel((x),(y),6);
                    x+=dx;
                    y+=dy;
                    i++;
                    }
              setcolor(15);
              outtextxy(800,10,"The Points Are:=");
        sprintf(s,"A(%d,%d)",x1,y1);
              outtextxy(800,20,s);
              sprintf(s,"B(%d,%d)",x2,y2);
              outtextxy(800,30,s);

              getch();
              }

int main()
{
  lines a;

  initwindow(800,600);

   a.showline();

   closegraph();
return 0;
}
```

Bresenham Line Algorithm

```
#include<stdio.h>
#include<conio.h>
#include "winbgim.h"
int main()
{
      int gd = DETECT, gm;
      int dx, dy, p, end;
      float x1, x2, y1, y2, x, y;
    //  initgraph(&gd, &gm, "c:\tc\bgi");
    initwindow(800,600);
      printf("Enter Value of X1: ");
      scanf("%f", &x1);
      printf("Enter Value of Y1: ");
      scanf("%f", &y1);
      printf("Enter Value of X2: ");
      scanf("%f", &x2);
      printf("Enter Value of Y2: ");
      scanf("%f", &y2);
      dx = abs(x1 - x2);
      dy = abs(y1 - y2);
      p = 2 * dy - dx;
      if(x1 > x2)
      {
            x = x2;
            y = y2;
            end = x1;
      }
      else
      {
            x = x1;
            y = y1;
            end = x2;
      }
      putpixel(x, y, 10);
      while(x < end)
      {
            x = x + 1;
            if(p < 0)
            {
                  p = p + 2 * dy;
            }
            else
            {
                  y = y + 1;
                  p = p + 2 * (dy - dx);
            }
            putpixel(x, y, 10);
      }
      getch();
      closegraph();
      return 0;
}
```

DDA Circle Algorithm

```
//Program to implement DDA Circle Drawing Algorithm

#include<iostream.h>
#include "winbgim.h"
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<stdlib.h>
#include<stdio.h>


class myCircle
      {
      private:
      float x,y,r,d,x1,y1;
      public:
      myCircle();      //Constructor
```

```
          void showCircle();
          int sign(int);
          };

int myCircle::sign(int xx)
                {
                if(xx<0)
                return -1;
                if(xx==0)
                return 0;
                if(xx>0)
                return 1;
                return 0;
                }
myCircle::myCircle()
                {
                x=0;y=0;
                cout << "Enter The Co-Ordinates Of The Circle :=";
                cin >> x1 >> y1;
                cout << "Enter The Radius Of The Circle :=";
                cin >> r;
                }

void myCircle::showCircle()
                {
                char *s;
                int s1,s2,ic;
                x=x1;y=y1;
                float i=0;
                while(i<=360)
                      {
                      x=x1+r*cos(i);
                      y=y1+r*sin(i);
                      putpixel((x),(y),7);
                      i+=0.5;
                      }
                getch();
                }

int main()
```

```
      {
      int i,j,xx=220,xxx=430;

      myCircle a;
      initwindow(800,600);
    a.showCircle();
      closegraph();
      return 0;
}
```

Bibliography

Anonymous. (2009). *C > Games and Graphics sample source codes*. Retrieved October 2009, from Happy Codings: http://www.c.happycodings.com/Games_and_Graphics/code18.html

Anonymous. (2009). *C++ > Computer Graphics sample source codes*. Retrieved October 2009, from Happy Codings: http://www.cplusplus.happycodings.com/Computer_Graphics/code18.html

Anonymous. (2009). *C++ > Computer Graphics sample source codes*. Retrieved October 2009, from Happy Codings: http://www.cplusplus.happycodings.com/Computer_Graphics/code17.html

Colin, M. ,. (1991). Dev-C++. USA.