Computer Graphics.

Lab Manual

Computer Graphics Lab.

1. Syllabus from the university

1. Implementation of line generation using slope's method, DDA and Bresenhem's algorithms.
2. Implementation of circle generation using Mid-point method and Bresenhem's algorithm.
3. Implementation of ellipse generation using Mid-point method.
4. Implementation of polygon filling using Flood-fill, Boundary-fill and Scan-line algorithms.
5. Implementation of 2D transformation: Translation, Scaling, Rotation, Mirror Reflection and Shearing (write a menu driven program).
6. Implementation of Line Clipping using Cohen-Sutherland algorithm and Bisection Method.
7. Implementation of Polygon Clipping using Sutherland-Hodgeman algorithm.
8. Implementation of 3D geometric transformations: Translation, Scaling and rotation.
9. Implementation of Curve generation using Interpolation methods.
10. Implementation of Curve generation using B-spline and Bezier curves.
11. Implementation of any one of Back face removal algorithms such as Depth-Buffer algorithm, Painter's algorithm, Warnock's algorithm, Scan-line algorithm)

2. Rational behind the coverage

a) **User Interface** -: Most application that runs on personal computer and workstations have user interfaces that rely on desktop window system to manage multiple simultaneous activities, and point and click facilities to allow user to select

a) **Interactive plotting in business science and technology:-** Computer graphics provide facilities to create 2D and 3D graphs of mathematical, Physical and Economical functions;histograms bar and pie charts; task scheduling charts; inventory and production charts ; and the like.

b) **Office Automation and Electronics Publishing** -: Office automation and electronic publishing can produce both traditional and printed (hardcopy) documents and electronic (softcopy) documents that contain text, tables, graphs and other form of drawn or scanned graphics.

c) **Computer Aided Design** -: In computer Aided Design (CAD), interactive graphics is used to design Components and systems of mechanical, electrical , electromechanical, and electronic devices including structures such as buildings , automobile bodies , aero plane and ship hulls , very large scale integration chips, optical system, and telephone and computer networks.

Menu items and other things.

a) **Simulation and animation for scientific visualization and entertainment -**: Computer produced animated movies and displays of the time- varying behavior of real and simulated objects and become increasingly popular for scientific and engineering visualization.

b) **Art and Commerce -:** Computer graphics is used to produce pictures that express a message and attract attention. Slide production for commercial, Scientific or educational presentation is another coast effective use of graphics.

c) **Process Control -:** whereas flight simulators or arcade games let users interact with a simulation of a real or artificial world, many other application enable people to interact with some aspect of the real world itself.

3. Hardware and Software requirement

a) **Software requirement :** Turbo C / C++

4. List of Experiments

1. Study of basic graphics functions defined in “graphics.h “.
2. Write a program to draw a Hut or other geometrical figures.
3. Write a program to draw a line using Bresenhem’s Algo.
4. Write a program to draw a line using DDA algorithm.
5. Write a program to draw a line using Mid-Point algorithm.
6. Write a program to draw a circle using mid-point algorithm.
7. Write a program to draw an Ellipse using Mid-Point algorithm.
8. Write a program to rotate a Circle around any arbitrary point or around the boundary of another circle.
9. Write a menu driven program to rotate, scale and translate a line point, square, triangle about the origin.
10. Write a program to perform line clipping.
11. Write a program to implement reflection of a point, line.
12. Write a program to perform shearing on a line.
13. Write a program to implement polygon filling.
14. Write a program to implement transformations in three dimensions.

BASIC GRAPHICS FUNCTION

1) INITGRAPH

- Initializes the graphics system.

Declaration

- Void far initgraph(int far *graphdriver)

Remarks

- To start the graphic system, you must first call initgraph.
- Initgraph initializes the graphic system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode.
- Initgraph also resets all graphics settings (color, palette, current position, viewport, etc) to their defaults then resets graph.

2) GETPIXEL, PUTPIXEL

- Getpixel gets the color of a specified pixel.
- Putpixel places a pixel at a specified point.

Decleration

- Unsigned far getpixel(int x, int y)
- Void far putpixel(int x, int y, int color)

Remarks

- Getpixel gets the color of the pixel located at (x,y);
- Putpixel plots a point in the color defined at (x, y).

Return value

- Getpixel returns the color of the given pixel.
- Putpixel does not return.

3) CLOSE GRAPH

- Shuts down the graphic system.

Decleration

- Void far closegraph(void);

Remarks

- Close graph deallocates all memory allocated by the graphic system.
- It then restores the screen to the mode it was in before you called initgraph.

Return value

- None.

4) ARC, CIRCLE, PIESLICE

- arc draws a circular arc.
- Circle draws a circle
- Pieslice draws and fills a circular pieslice

Declaration

- Void far arc(int x, int y, int stangle, int endangle, int radius);
- Void far circle(int x, int y, int radius);
- Void far pieslice(int x, int y, int stangle, int endangle, int radius);

Remarks

- Arc draws a circular arc in the current drawing color

- Circle draws a circle in the current drawing color
- Pieslice draws a pieslice in the current drawing color, then fills it using the current fill pattern and fill color.

5) ELLIPSE, FILL ELIPSE, SECTOR

- Ellipse draws an elliptical arc.
- Fill ellipse draws and fills ellipse.
- Sector draws and fills an elliptical pie slice.

Declaration

- Void far ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)
- Void far fill ellipse(int x, int y, int xradius, int yradius)
- Void farsectoe(int x, int y, int stangle, int endangle, int xradius, int yradius)

Remarks

- Ellipse draws an elliptical arc in the current drawing color.
- Fill ellipse draws an elliptical arc in the current drawing color and than fills it with fill color and fill pattern.
- Sector draws an elliptical pie slice in the current drawing color and than fills it using the pattern and color defined by setfill style or setfill pattern.

6) FLOODFILL

- Flood-fills a bounded region.

Decleration

- Void far floodfill(int x, int y, int border)

Remarks

- Floodfills an enclosed area on bitmap device.
- The area bounded by the color border is flooded with the current fill pattern and fill color.

- (x,y) is a “seed point”
    - If the seed is within an enclosed area, the inside will be filled.
    - If the seed is outside the enclosed area, the exterior will be filled.
- Use fillpoly instead of floodfill wherever possible so you can maintain code compatibility with future versions.
- Floodfill doesnot work with the IBM-8514 driver.

Return value

- If an error occurs while flooding a region, graph result returns „1‟.

7) GETCOLOR, SETCOLOR

- Getcolor returns the current drawing color.
- Setcolor returns the current drawing color.

Decleration

- Int far getcolor(void);
- Void far setcolor(int color)

Remarks

- Getcolor returns the current drawing color.
- Setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.
- To set a drawing color with setcolor , you can pass either the color number or the equivalent color name.

8) LINE,LINEREL,LINETO

- Line draws a line between two specified pints.
- Onerel draws a line relative distance from current position(CP).
- Linrto draws a line from the current position (CP) to(x,y).

Declaration

- Void far lineto(int x, int y)

Remarks

- Line draws a line from (x1, y1) to (x2, y2) using the current color, line style and thickness. It does not update the current position (CP).
- Linerel draws a line from the CP to a point that is relative distance (dx, dy) from the CP, then advances the CP by (dx, dy).
- Lineto draws a line from the CP to (x, y), then moves the CP to (x,y).

Return value

- None

9) RECTANGLE

- Draws a rectangle in graphics mode.

Decleration

- Void far rectangle(int left, int top, int right, int bottom)

Remarks

- It draws a rectangle in the current line style, thickness and drawing color.
- (left, top) is the upper left corner of the rectangle, and (right, bottom) is its lower right corner.

Return value

- None.

BRESENHEM'S ALGORITHM FOR LINE DRAWING.

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,p,dx,dy and also declare gdriver=DETECT,gmode.
3. Initialize the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer; that is, plot the first point put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1,and obtain the initial value of decision parameter p as:
    a. p=(2dy-dx).
7. Starting from first point (x,y) perform the following test:
8. Repeat step 9 while(x<=x2).
9. If p<0,next point is (x+1,y) and p=(p+2dy).
10. Otherwise, the next point to plot is (x+1,y+1) and p=(p+2dy-2dx).
11. Place pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.

WAP TO DRAW A LINE USING MID POINT ALGORITHM OR BRESENHAM'S ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int x,y,x1,y1,x2,y2,p,dx,dy;
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\tc\\BGI:");
        printf("\nEnter the x-coordinate of the first point ::");
        scanf("%d",&x1);
        printf("\nEnter the y-coordinate of the first point ::");
        scanf("%d",&y1);
        printf("\nEnter the x-coordinate of the second point ::");
        scanf("%d",&x2);
        printf("\nEnter the y-coordinate of the second point ::");
        scanf("%d",&y2);
        x=x1;
        y=y1;
        dx=x2-x1;
        dy=y2-y1;
        putpixel(x,y,2);
        p=(2dy-dx);
        while(x<=x2)
        {
                if(p<0)
                {
                        x=x+1;
                p=2*x-dx;
        }
        else
        {
                x=x+1;
                y=y+1;
                p=p+2*dy;
        }
        putpixel(x,y,7);
}
getch();
closegraph();
```

ALGORITHM TO DRAWA LINE USING DDA ALGORITHM.

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer;that is,plot the first point.put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs(dx) > abs(dy), do s=abs(dx).
8. Otherwise s= abs(dy).
9. Then xi=dx/s and yi=dy/s.
10. Start from k=0 and continuing till k<s,the points will be
    1. x=x+xi.
    2. y=y+yi.
11. Place pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.

WAP TO DRAW A LINE USING DDA ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int x,y,x1,x2,y1,y2,k,dx,dy,s,xi,yi;
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
        printf("enter first point");
        scanf("%d%d",&x1,&y1);
        printf("enter second point");
        scanf("%d%d",&x2,&y2);
        x=x1;
        y=y1;
        putpixel(x,y,7);
        dx=x2-x1;
        dy=y2-y1;
         if(abs(dx)>abs(dy))
                s=abs(dx);
        else
                s=abs(dy);
        xi=dx/s;
        yi=dy/s;
        x=x1;
        y=y1;
        putpixel(x,y,7);
        for(k=0;k<s;k++)
        {
                x=x+xi;
                y=y+yi;
                putpixel(x,y,7);
        }
        getch();
        closegraph();
}
```

BRESENHAM'S ALGORITHM TO DRAW A CIRCLE.

1. Start.
2. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the radius of the circle r.
5. Load x-0,y=r,initial decision parameter p=1-r.so the first point is (0,r).
6. Repeat Step 7 while (x<y) and increment x-value simultaneously.
7. If (p>0),do p=p+2*(x-y)+1.
8. Otherwise p=p+2*x+1 and y is decremented simultaneously.
9. Then calculate the value of the function circlepoints() with p.arameters (x,y).
10. Place pixels using putpixel at points (x+300,y+300) in specified colour in circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
11. Close Graph.
12. Stop.

WAP TO DRAW A CIRCLE USING BRESENHAM'S ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void circlepoints(int,int);
void main()
{
        int x,y,p,r;
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
        clrscr();
        printf("enter the radius");
        scanf("%d",&r);
        x=0;y=r;p=1-r;
        while(x<y)
        {
                x++;
                if(p>0)
                {
                        p=p+2*(x-y)+1;
                        y--;
                }
                else
                        p=p+2*x+1
```

```
  circlepoints(x,y);
          }
          getch();
          closegrap
          h();
}

void circlepoints(int x,int y)
              {
          putpixel(x+300,y+300,8);
          putpixel(x+300,-y+300,8);
          putpixel(-x+300,y+300,8);
          putpixel(-x+300,-y+300,8);
          putpixel(y+300,x+300,8);
          putpixel(y+300,-x+300,8);
          putpixel(-y+300,x+300,8);
          putpixel(-y+300,-x+300,8);
}
```