

PROGRAMMING LABORATORY LAB MANUAL

Shree Ramchandra College of Engineering

Lonikand, PUNE (MH)



Department of Computer Engineering

LAB MANUAL

S.E. COMPUTER (SEMESTER-IV)

PROGRAMMING LABORATORY

Subject Code: 210251

Teaching Scheme
Lectures: 3 Hrs/week
Practical: 2Hrs/Week

Examination Scheme
Theory: 50 Marks
Online: 50 Marks
Oral: 50 Marks
Term Work: 50 Marks

PROGRAMMING LABORATORY LAB MANUAL

INDEX

Sr. no.	TITLE	Page No.
1	Writing a C/C++ Program to emulate CPU Architecture (Central Bus) Develop register, ALU level GUI to display results.	
2	Writing a C++ class for displaying pixel or point on the screen	
3	Write a C++ class for a Line drawing method using overloading DDA and Bresenham's Algorithms, inheriting the pixel or point	
4	Write a C++ class for a circle drawing inheriting line class	
5	Write a program in C/C++ to draw a circle of desired radius.	
6	Write a program using C/ C++ to draw a line with line styles (Thick, Thin, Dotted)	
7	Write a program using Java/python to draw a line with line styles (Thick, Thin, Dotted)	
8	Write a C/C++ program to draw a convex polygons (Square, Rectangle, Triangle)	
9	Write a C/C++ program to draw a Convex polygon with programmable edges	
10	Write a program in Java to draw a concave polygon.	
11	Write a C/C++ program to fill polygon using scan line algorithm	
12	Draw a line using OpenGL	

Lonikand, Pune

PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 01

Title: - Writing a C/C++ Program to emulate CPU Architecture (Central Bus) Develop register, ALU level GUI to display results.

Pre-requisite: Basic Graphics Primitives and functions.

Aim: Basic Graphics Primitives and functions are studied to emulate a CPU Architecture using registers, ALU, Buses, I/O devices.

Hardware Used: P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: TC++

Theory:-

Emulation refers to the ability of a computer program in an electronic device to emulate (imitate) another program or device. The standard definition for emulation is “try to be equal or better than someone or something”. An emulator is thus someone or something who or which emulates someone or something else. Emulation in computers is the same, to emulate the behavior of a hardware device in software or with a different hardware, or to emulate the behavior of a piece of software either with another hardware or software. That is still a too general definition, because you can emulate from an OS to a sound card. And the techniques used are absolutely different. Emulating a hardware device with another hardware device just has to care about to output the same values than the original hardware for the same input values. That is a task for electronic or VLSI design. But if you are emulating an OS over another OS that is a software problem.

Basic Graphics Function: -

INITGRAPH: - Initializes the graphics system
void far initgraph (int far *graphdriver, int far *graphmode, char far *pathtodriver);

To start the graphics system, you must first call initgraph. initgraph initializes the graphics 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 graphresult to 0.

***graphdriver:** - Integer that specifies the graphics driver to be used. You can give graphdriver a value using a constant of the graphics drivers enumeration type.

***graphmode:** - Integer that specifies the initial graphics mode (unless *graphdriver = DETECT). If *graphdriver = DETECT, initgraph sets *graphmode to the highest resolution available for the detected driver. You can give *graphmode a value using a constant of the graphics_modes enumeration type.

***pathtodriver:** - Specifies the directory path where initgraph looks for graphics drivers (*.BGI) first. ■ If they're not there, initgraph looks in the current directory. ■ If pathtodriver is null, the driver

PROGRAMMING LABORATORY LAB MANUAL

files must be in the current directory. This is also the path `settextstyle` searches for the stroked character font files (*.CHR).

GETPIXEL, PUTPIXEL :-

```
int getpixel(int x, int y);
```

```
void putpixel(int x, int y, int color);
```

`getpixel` returns the color of pixel present at point(x, y).

`putpixel` plots a pixel at a point(x, y) of specified color.

CLOSE GRAPH :-

Void `far closegraph(void)`:- 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`.

Following are inbuilt functions of graphics library used for assignment:-

rectangle():- It draws a rectangle in the current line style, thickness, and drawing color.
`void rectangle(int left, int top, int right, int bottom);`

ellipse():- It draws an elliptical arc in the current drawing color.
`void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);`

sector():- It draws and fills an elliptical pie
`void sector(int x, int y, int stangle, int endangle, int xradius, int yradius);`

Line ():- `Line ()` is used to draw a line.

```
Line(x1,y1,x2,y2);
```

Circle(): is used to draw the circle.

```
Syntax: circle(x1,y1,radius);
```

drawpoly ():- draws the outline of a polygon, draws a polygon using the current line style and color.

```
Drawpoly(numpoints,*polypoints);
```

Numpoints:- specifies number of points

*polypoints :- points to a sequence of (numpoints*2) integers.

Outtext: -Display a string in the viewport

```
Void far outtext(char far textstring);
```

```
Example: - outtext(display text in graphics);
```

Putpixel: - plots a pixel at specified point.



PROGRAMMING LABORATORY LAB MANUAL

Syntax: void far putpixel (int x, inty,intpixelcolor);

Example:putpixel(100,100,15);

Closegraph: shuts down the graphics mode

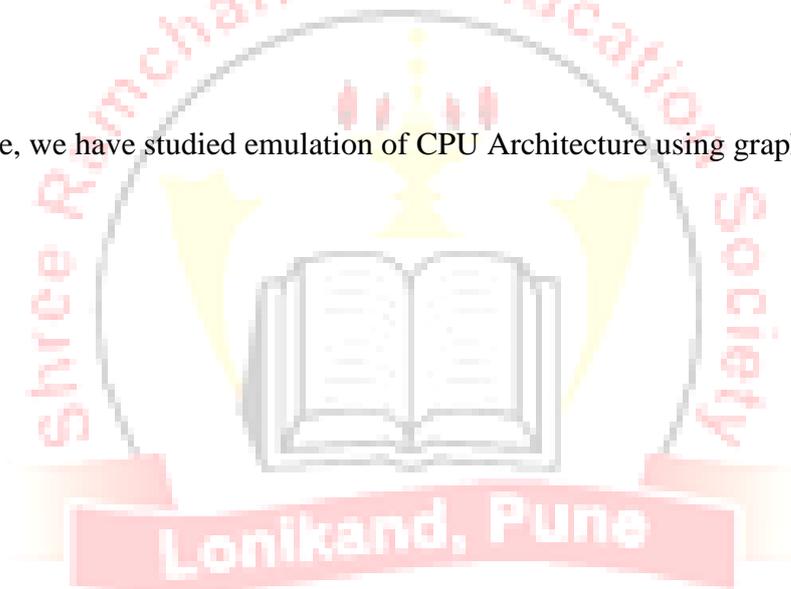
Syntax: -void far close graph(void);

Example: - closegraph();

Application in Real world (Min. Two) with explanation:

1. Education and Training Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.
2. Visualization For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

Conclusion: -Hence, we have studied emulation of CPU Architecture using graphics functions.





PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 02

Title:- Writing a C++ class for displaying pixel or point on the screen.

Pre-requisite: Basic Graphics Primitives and functions

Aim: - Pixel is drawn as per the coordinates and color.

Hardware Used:- P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: - TC++

Theory:

A pixel (short for picture element, using the common abbreviation "pix" for "picture") is one of the many tiny dots that make up the representation of a picture in a computer's memory. Each such information element is not really a dot, nor a square, but an abstract sample. With care, pixels in an image can be reproduced at any size without the appearance of visible dots or squares; but in many contexts, they are reproduced as dots or squares and can be visibly distinct when not fine enough. The intensity of each pixel is variable; in color systems, each pixel has typically three or four dimensions of variability such as red, green and blue, or cyan, magenta, yellow and black.

Fig. Pixel Resolution

Another popular convention is to cite resolution as the total number of pixels in the image, typically given as number of mega pixels, which can be calculated by multiplying pixel columns by pixel rows and dividing by one million. Other conventions include describing pixels per length unit or pixels per area unit, such as pixels per inch or per square inch. None of these pixel resolutions are true resolutions, but they are widely referred to as such; they serve as upper bounds on image resolution.

Below is an illustration of how the same image might appear at different pixel resolutions, if the pixels were poorly rendered as sharp squares (normally, a smooth image reconstruction from pixels would be preferred, but for illustration of pixels, the sharp squares make the point better).

1) Loading computer graphics

When we start with the turbo c editor. We first includes the graphics library to convert the text editor into graphics editor. To include the graphics library the steps are

- i) Select the option menu from the menu bar
- ii) Select linker option
- iii) Select libraries from linker option
- iv) Select graphics library from the libraries window
- v) Click on Ok.

2) Initialized the graphics Mode:-

Before writing the computer graphics program, first we initialize the graphics driver and graphics mode, for that purpose we use detectgraph() function and initgraph() function.



PROGRAMMING LABORATORY LAB MANUAL

Detectgraph():- determines the graphics driver and chooses mode by checking hardware.

Syntax:-detectgraph(&gdriver, &gmode);

Initgraph():- It initializes graphics system. To start the graphic system , you must first callinitgraph(), initgraph initializes the graphic system by loading a graphics driver from disk, then putting into the graphics mode.

Syntax:-initgraph(&gdriver,&gmode,"C:\\tc\\bgi");

“Bgi” is a file that stores the graphics driver and for each bgi driver there are different graphics mode. In short, it provides graphics interface.

Inbuilt functions :-

Following are inbuilt functions of graphics library:-

1). **Line ()**:- Line () is used to draw a line.

Line(x1,y1,x2,y2);

2). **Circle()**: is used to draw the circle.

Syntax: circle(x1,y1,radius);

3). **drawpoly()** :- draws the outline of a polygon, draws a polygon using the current line style andcolor.

Drawpoly(numpoints,*polypoints);

Numpoints:- specifies number of points

*polypoints :- points to a sequence of (numpoints*2) integers.

4). **Outtext**: -Display a string in the viewport
Void far outtext(char far textstring);

Example: - outtext(display text in graphics);

5). **Putpixel**: - plots a pixel at specified point.

Syntax: void far putpixel (int x, inty,intpixelcolor);

Example:putpixel(100,100,15);

6). **Closegraph**: shuts down the graphics mode
Syntax: -void far close graph(void); Example: - closegraph();

Conclusion:- Hence, we have studied displaying pixel or point on the screen.

Department Of Computer Engineering | S.R.C.O.E.



PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 03

Title: - Write a C++ class for a Line drawing method using overloading DDA and Bresenham's Algorithms, inheriting the pixel or point.

Pre-requisite: Basic Graphics Primitives and functions.

Aim: - Line is generated as per the coordinates.

Hardware Used: P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: TC++

Theory:

1) Digital Differential Analyzer (DDA)

The slope of a straight line is given as

$$M = (y_2 - y_1) / (x_2 - x_1) \text{ i.e. } \Delta y / \Delta x$$

This equation can be used to obtain a rasterized straight line. For any given x interval Δx along a line, we can compute the corresponding y interval Δy as,

$$\Delta y = \frac{(y_2 - y_1) \Delta x}{(x_2 - x_1)}$$

Similarly, we can obtain the x interval Δx corresponding to a specified Δy as,

$$\Delta x = \frac{(x_2 - x_1) \Delta y}{(y_2 - y_1)}$$

Once the intervals are known the values for next x and next y on the straight line can be obtained as follows

$$X_{i+1} = x_i + \Delta x$$

$$= x_i + \frac{(x_2 - x_1) \Delta y}{(y_2 - y_1)}$$

And

$$y_{i+1} = y_i + \Delta y$$

$$= y_i + \frac{(y_2 - y_1) \Delta x}{(x_2 - x_1)}$$

These equations represent a recursion relation for successive values of x and y along the required line. Such a way of rasterizing a line is called a digital differential analyzer (DDA).

For simple DDA either Δx or Δy , whichever is larger, is chosen as one raster unit, i.e.

If $|\Delta x| \geq |\Delta y|$



PROGRAMMING LABORATORY LAB MANUAL

Then $\Delta x=1$

Else $\Delta y=1$

With this simplification, if $\Delta x=1$ then

We have,

$$y_{i+1} = y_i + \frac{(y_2-y_1)}{(x_2-x_1)}$$

$$x_{i+1} = x_i + 1$$

If $\Delta y=1$ then

We have,

$$y_{i+1} = y_i + 1$$

$$x_{i+1} = x_i + \frac{(x_2-x_1)}{(y_2-y_1)}$$

Algorithm:

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.

(If equal then plot that point and exit)

2. $\Delta x = |x_2-x_1|$ and $\Delta y = |y_2-y_1|$

3. if $(\Delta x \geq \Delta y)$ then

length= Δx

else

length= Δy

end if

4. $\Delta x = (x_2-x_1) / \text{length}$

$\Delta y = (y_2-y_1) / \text{length}$

5. $x = x_1 + 0.5 * \text{Sign}(\Delta x)$

$y = y_1 + 0.5 * \text{Sign}(\Delta y)$

(Sign function makes the algorithm work in all quadrants. It returns $-1, 0, 1$ depending on whether its argument is $<0, =0, >0$ respectively. The factor 0.5 makes it possible to round the values in the integer function rather than truncating them.)

6. $I=1$ (Begins the loop, in this loop points are plotted)

While ($I \leq \text{length}$)

{

Plot (Integer (x), Integer

(y)) $X=x+\Delta x$

$Y=y+\Delta y$

$I=I+1$

}

7. Stop.

2) Bresenham's line Algorithm

The basic principle of Bresenham's line algorithm is to select the optimum raster locations to



PROGRAMMING LABORATORY LAB MANUAL

represent a straight line. To get this the algorithm always increments either x or y by one unit depending on the slope of line. The increment in the other variable is determined by examining the distance between the actual line location and the nearest pixel. This distance is called decision variable or the error.

In mathematical terms decision variable or the error is defined as

$$E = DB - DA \text{ or } DA - DB$$

If $e > 0$, then it implies that $DB > DA$, i.e. the pixel above the line is closer to the true line.

If $DB < DA$, then we can say that the pixel below the line is closer to the true line.

The error term is initially set as

$$e = 2\Delta y - \Delta x$$

Where $\Delta y = y_2 - y_1$,

$$\Delta x = x_2 - x_1$$

Then according to value of e following actions are taken.

While ($e \geq 0$)

```
{  
y=y+1  
e=e-2*Δx  
}
```

```
x=x+1
```

```
e=e+2*Δy
```

Algorithm:

1. Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal.

(If equal then plot that point and exit)

2. $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$

3. [Initialize starting point]

```
x=x1
```

```
y=y1
```

4. $e = 2 * \Delta y - \Delta x$

[Initialize value of decision variable or error to compensate for nonzero intercepts]

4. $I = 1$ [Initialize counter]

5. Plot (x, y)

6. While ($e \geq 0$)

```
{ y=y+1  
  e=e-2*Δx
```

```
}
```

```
x=x+1
```

```
e=e+2*Δy
```

7. $I = I + 1$

8. If ($I \leq \Delta x$) then go to step 6.

9. Stop.

Application in Real world (Min. Two) with explanation:

1. Image Processing: Image processing is to apply techniques to modify or interpret existing pictures. It is widely used in medical applications.

2. Graphical User Interface: Multiple window, icons, menus allow a computer setup to be utilized more efficiently.



PROGRAMMING LABORATORY LAB MANUAL

Conclusion: - Thus we have implemented Line Drawing Algorithms (DDA & Bresenham).

Assignment no: 04

Title:- Write a C++ class for a circle drawing inheriting line class.

Pre-requisite:- Basic Graphics Primitives and functions.

Aim:- Circle is generated as per the coordinates and radius.

Hardware Used:- P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used:- TC++

Theory :

1. DDA Circle Drawing Algorithm

The equation of circle, with origin as the center of the circle is given as $x^2 + y^2 = r^2$

The DDA Algorithm can be used to draw the circle by defining circle as a differential equation. It is given as below

$2x dx + 2y dy = 0$ (where r is constant)

$x dx + y dy = 0$

$y dy = -x dx$

$\frac{dy}{dx} = \frac{-x}{y}$

From above equation, we can construct the circle by using incremental x value, $\Delta x = \epsilon y$ and incremental y value, $\Delta y = -\epsilon x$, where ϵ is calculated from the radius of the circle as given below $2n-1 \leq r < 2n$ (where r is radius of circle)

$\epsilon = 2 - n$

Applying these incremental steps we have, $x_{n+1} = x_n + \epsilon y_n$

$y_{n+1} = y_n - \epsilon x_n$

The points plotted using above equations give the spiral instead of the circle. For getting circle we have to replace x_n by x_{n+1} in the equation of y_{n+1} .

Therefore we have

$x_{n+1} = x_n + \epsilon y_n$

$y_{n+1} = y_n - \epsilon x_{n+1}$

Algorithm

1. Read the radius (r), of the circle and calculate value of ϵ

2. Start_x = 0

Start_y = r

3. $x_1 = \text{start_x}$

$y_1 = \text{start_y}$

4. do

{

$x_2 = x_1 + \epsilon y_1$

$y_2 = y_1 - \epsilon x_2$

Plot ($\text{int}(x_2)$, $\text{int}(y_2)$)



PROGRAMMING LABORATORY LAB MANUAL

```
X1=x2
Y1=y2
  (Reinitialize the current point)
} while ( y1-start_y) <ε or ( start_x - x1) >ε
```

(Check if the current point is the starting point or not. If current point is not starting point repeat step 4)

5. Stop.

2) Bresenham's Circle Drawing Algorithm:

This algorithm considers the eight-way symmetry of the circle to generate it. It plots 1/8th part of the circle, i.e. from 90° to 45° . As the circle is drawn from 90° to 45° , the x moves in positive direction and y moves in the negative direction.

To achieve the best approximation to the true circle we have to select those pixels in the raster that fall the least distance from the true circle.

Here the equation for d_i at starting point, i.e. at $x=0$ and $y=r$ can be simplified as,

```
di=3-2r (where r is radius)
For di<0, xi+1 = xi + 1 and
For di>=0, xi+1 = xi +1 and yi+1 = yi -1
```

Similarly, the equations for d_{i+1} for both the cases are given as

```
For di<0, di+1 = di + 4xi+6 and
For di>=0, di+1 = di + 4(xi-yi) +10.
```

Reflecting about x & y-axis can draw the remaining part of the circle.

Algorithm:

```
1. Read the radius(r) of the circle.
2. d=3-2r (Initialize the decision variable)
3. x=0, y=r ( Initialize starting point)
4. do
{
  plot(x,y)
  plot(y,x)
  plot(y,-x)
  plot(x,-y)
  plot(-x,-y)
  plot(-y,-x)
  plot(-y,x)
  plot(-x,y)

  if(d<0) then
  {
    d=d+4x+6
  }
  else
  {
```



PROGRAMMING LABORATORY LAB MANUAL

```
d=d+4(x-
y)+10 y=y-1
}
x=x+1
}while(x<y)
5. Stop.
```

Application in Real world (Min. Two) with explanation:

1 .Image Processing

Image processing is to apply techniques to modify or interpret existing pictures. It is widely used in medical applications.

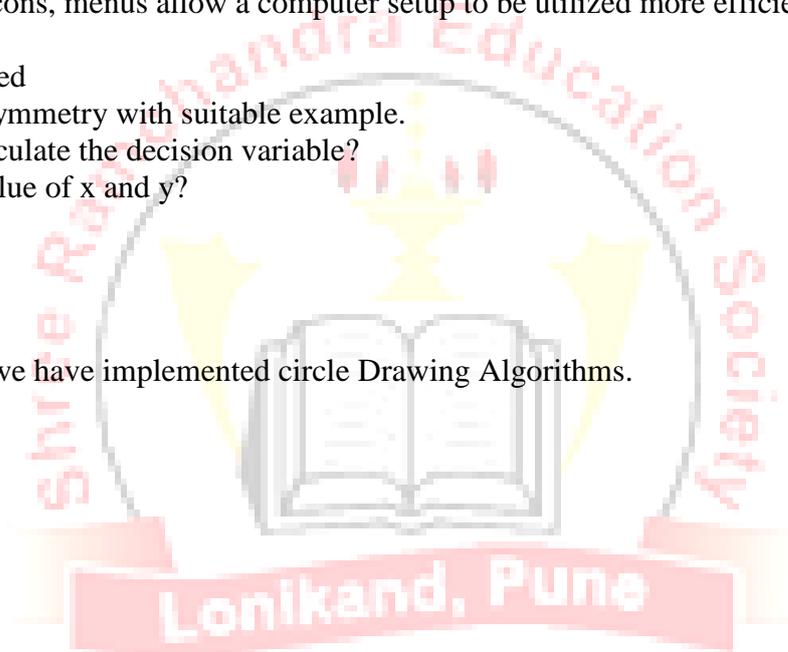
2. Graphical User Interface

multiple window, icons, menus allow a computer setup to be utilized more efficiently.

Questions to be asked

1. Explain 8 way symmetry with suitable example.
2. How can you calculate the decision variable?
3. What is initial value of x and y?

Conclusion: Thus we have implemented circle Drawing Algorithms.





PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 05

Title: -Write a program in C/C++ to draw a circle of desired radius.

Pre-requisite: Basic Graphics Primitives and functions.

Aim: Circle is drawn as per the coordinates and radius.

Hardware Used:-P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used:-TC++

Theory:

Midpoint Circle Drawing Algorithm

1. Accept radius 'r' and center of circle (XC, yc) from user and plot 1st point on circumference of circle.

$$(X_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter

$$p_0 = 1 - r$$

3. If we are using octant symmetry property to plot the pixels, then until ($x < y$) we have to perform following steps.

if $p_k < 0$

 modify p_k as $p_k = p_k + 2x + 1$ and

 then $x = x + 1$

otherwise

 modify p_k as $p_k = p_k + 2(x - y) + 1$

$x = x + 1$

$y = y - 1$

4. Determine the symmetry property points in other octants also

5. Derive all formulas by considering center point as origin. Move each calculated pixel position (x,y) on to the circular path centered on (xc, yc) and plot the coordinate values as

$$x = x + x_c \text{ and } y = y + y_c$$

Application in Real world (Min. Two) with explanation:

1. Presentation Graphics

To produce illustrations which summarize various kinds of data. Except 2D, 3D graphics are good tools for reporting more complex data.

2. Computer Art

Painting packages are available. With cordless, pressure-sensitive stylus, artists can produce electronic paintings which simulate different brush strokes, brush widths, and colors.

Photorealistic techniques, morphing and animations are very useful in commercial art. For films, 24 frames per second are required. For video monitor, 30 frames per second are required.

Questions to be asked

1. Differentiate between 4way & 8 way connected pixel. Which one is better?



PROGRAMMING LABORATORY LAB MANUAL

2. How can you calculate the decision variable?
3. What is initial value of x and y?

Conclusion: - Thus we have implemented Circle Drawing Algorithms .

Assignment no: 06

Title: Write a program using C/C++ to draw a line with line styles (Thick, Thin, Dotted).

Pre-requisite: Basic Graphics Primitives and functions.

Aim: Line is generated as per the choice.

Hardware Used:-P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: - TC++

Theory:

Basic attributes of a straight line segment are its type, its width, and its color. In some graphics packages, lines can also be displayed using selected pen or brush options. In the following sections, we consider how line drawing routines can be modified to accommodate various attribute specifications.

Line Type

Possible selections for the line-type attribute include solid lines, dashed lines, and dotted lines. We modify a line drawing algorithm to generate such lines by setting the length and spacing of displayed solid sections along the line path. A dashed line could be displayed by generating an inter dash spacing that is equal to the length of the solid sections. Both the length of the dashes and the inter dash spacing are often specified as user options. A dotted line can be displayed by generating very short dashes with the spacing equal to or greater than the dash size. Similar methods are used to produce other line-type variations. Line Attributes

To set line type attributes in a PHIGS application program, a user invokes the function

`setLinetype(lt)`

where parameter `lt` is assigned a positive integer value of 1,2,3, or 4 to generate lines that are, respectively, solid, dashed, dotted, or dash-dotted.

Line Width

Implementation of line- width options depends on the capabilities of the output device. A heavy line on a monitor could be displayed as adjacent parallel lines, while a pen plotter might require pen changes. As with other PHIGS attributes, a line-width command is used to set the current line-width value in the attribute list. This value is then used by line-drawing algorithms to control the thickness of lines generated with subsequent output primitive commands

We set the line-width attribute with the command:

`SetLinewidthScaleFactor(lw)`

Line-width parameter `lw` is assigned a positive number to indicate the relative width of the line to be displayed. A value of 1 specifies a standard width line. On a pen plotter, for instance, a user could set `lw`

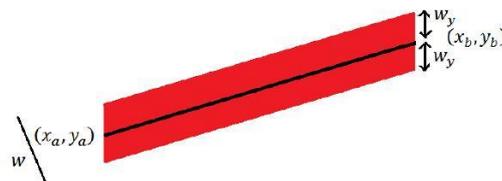


PROGRAMMING LABORATORY LAB MANUAL

to a value of 0.5 to plot a line whose width is half that of the standard line. Values greater than 1 produce lines thicker than the standard.

Thick Line Segments

We can draw lines with thickness greater than one pixel. To produce a thick line segment, we can draw two lines in parallel along the thick-line edges. As we step along the line finding successive edge pixels, we must turn ON all the pixels which lie between the boundaries.



To draw a line between (x_a, y_a) and (x_b, y_b) with thickness w , we would have a top boundary between the points $(x_a, y_a + w_y)$ and $(x_b, y_b + w_y)$ and a lower boundary between $(x_a, y_a + w_y)$ and $(x_b, y_b - w_y)$ where w_y is given by,

$$w_y = \frac{(w-1) [(x_b - x_a)^2 + (y_b - y_a)^2]^{1/2}}{2 |x_b - x_a|}$$

Application in Real world (Min. Two) with explanation: 1. Entertainment

Motion pictures, Music videos, and TV shows, Computer games Education and Training Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.

2. Visualization

For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

Questions to be asked

1. Explain the logic of Thick, thin and dotted lines.
2. Explain different types of slope. How that will affect in drawing a line?

Conclusion:

Thus, we have successfully studied to implement a program in C/C++ to draw a line with line style (Thick, Thin, Dotted).



PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 07

Title:-Write a python program to draw a simple polygon (Square, Rectangle, Triangle)

Aim:-To draw python program to draw a simple polygon.

Hardware Used:-P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: - python

Theory:-

Introduction

Python is a powerful and widely used programming language. Python is often used as a scripting language (i.e., a programming language that is used to control software applications). Javascript embedded in a webpage can be used to control how a web browser such as Firefox displays web content, so javascript is a good example of a scripting language. Python can be used as a scripting language for various applications (such as Sage [S]), and is ranked in the top 5-10 worldwide in terms of popularity. Python is fun to use. In fact, the origin of the name comes from the television comedy series Monty Python's Flying Circus.

Python has seen extensive use in the information security industry, and has been used in a number of commercial software products, including 3D animation packages such as Maya and Blender, and 2D imaging programs like GIMP and Inkscape.

What is Python?

The idea resonates in both mathematics and in computer programming. Statements must be constructed from carefully defined terms with a clear and unambiguous meaning, or things can go wrong. Python is a computer programming language designed for readability and functionality. One of Python's design goals is that the meaning of the code is easily understood because of the very clear syntax of the language. The Python programming language has a specific syntax (form) and semantics (meaning) which enables it to express computations and data manipulations which can be performed by a computer.

Python's implementation was started in 1989 by Guido van Rossum at CWI (a national research institute in the Netherlands) as a successor to the ABC programming language (an obscure language made more popular by the fact that it motivated Python!). Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, Benevolent Dictator for Life (BDFL). Python is an interpreted language, i.e., a programming language whose programs are not directly executed by the host cpu but rather executed (or interpreted) by a program known as an interpreter. The source code of a Python program is translated or (partially) compiled to a "bytecode" form of a Python "process virtual machine" language. This is in distinction to C



PROGRAMMING LABORATORY LAB MANUAL

code which is compiled to cpu-machine code before runtime. Python is a “dynamically typed” programming language. A programming language is said to be dynamically typed, when the majority of its type checking is performed at run-time as opposed to at compile-time. Dynamically typed languages include JavaScript, Lisp, Lua, Objective-C, Python, Ruby, and Tcl.

The data which a Python program deals with must be described precisely. This description is referred to as the data type. In the case of Python, the fact that Python is dynamically typed basically means that the interpreter or compiler will go out for you what type a variable is at run-time, so you don't have to declare variable types yourself. The fact that Python is “strongly typed” means that it will actually raise a run-time type error when you have violated a Python grammar/syntax rule as to how types can be used together in a statement. Of course, just because Python is dynamically and strongly typed does not mean you can neglect “type discipline”, that is carelessly mixing types in your statements, hoping Python to give out things. Here is an example showing how Python can give out the type from the command at run-time.

Python

```
>>> a = 2012
```

```
>>> type(a)
```

```
<type 'int'>
```

```
>>> b = 2.011
```

The Python compiler can also “coerce” types as needed.

In this example below, the interpreter coerces at runtime the integer a into b so that it can compute a+b:

```
>>> c = a+b
```

```
>>> c
```

```
>>> type(c)
```

```
<type 'float'>
```

However, if you try to do something illegal, it will raise a type error.

```
Python >>> 3+"3"
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Also, Python is an object-oriented language. Object-oriented programming (OOP) uses “objects” - data structures consisting of data fields and methods - to design computer programs. For example, a matrix could be the “object” you want to write programs to deal with. You could be a class of matrices and, for example, a method for that class might be addition.



PROGRAMMING LABORATORY LAB MANUAL

Python has several ways to read in files which are filled with legal Python commands. One is the import command. This is really designed for Python "modules" which have been placed in specific places in the Python directory structure. Another is to "execute" the commands in the say myfile.py, using the Python command: python myfile.py. To have Python read in a file of data, or to write data to a file, you can use the open command, which has both read and write methods.

Introduction to Graphics

The computer uses a different coordinate system. Understanding why it is different requires a bit of computer history. During the early 80's, most computer systems were text-based and did not support graphics.

When positioning text on the screen, programmers started at the top calling it line 1. The screen continued down for 24 lines and across for 40 characters. Even with plain text, it was possible to make rudimentary graphics by using just characters on the keyboard. Characters were still positioned starting with line 1 at the top. The character set was expanded to include boxes and other primitive drawing shapes. Characters could be drawn in different colors. the graphics got more advanced. Search the web for "ASCII art" and many more examples can be found. Once computers moved to being able to control individual pixels for graphics, the text-based coordinate system stuck.

Colors

Colors are defined in a list of three colors: red, green, and blue. The numbers range from 0 to 255. Zero means there is none of the color, and 255 tells the monitor to display as much of the color as possible. The colors combine, so if all three colors are specified, the color on the monitor appears white. Lists in Python are surrounded by square brackets. Individual numbers are separated by commas. Below is an example that creates variables and sets them equal to lists of three numbers. These lists will be used later to specify colors.

Defining colors

```
1 black = [ 0, 0, 0]
```

```
2 white = [255,255,255]
```

```
3 blue = [ 0, 0,255]
```

```
4 green = [ 0,255, 0]
```

```
5 red = [255, 0, 0]
```

Using the interactive shell in IDLE, try defining these variables and printing them out. If the five colors above aren't the colors you are looking for, you can define your own.

Open a window

The programs did not open any windows like most modern programs do in Windows or Macs. The code to open a window is not complex. Below is the required code, which creates a window sized to 400 x 400 pixels:

```
# Set the height and width of the  
screen size =[400 ,400]  
screen = pygame . display .set_mode( size )
```

Opening and setting the window size



PROGRAMMING LABORATORY LAB MANUAL

Set the height and width of the screen size =[400 ,400]
`screen=pygame.display.set_mode(size)`

To set the title of the window in its title bar and the title shown when it is minimized, use the following line of code:

Setting the window title
`pygame.display.set_caption("Welcome to NBSSOE, Ambegaon")`

Drawing

The following code clears whatever might be in the window with a white background. Remember that the variable `white` was defined earlier as a list of 3 RGB values.

Clear the screen and set the screen background `screen.fill(white)`

This code shows how to draw a line on the screen.

It will draw on the screen a green line from (0,0) to (100,100) that is 5 pixels wide.

Drawing a single line

`pygame.draw.line(screen ,green ,[0,0],[100,100],5)`

Drawing a rectangle

A rectangle with the origin at (20,20), a width of 250 and a height of 100.

When specifying a rectangle the computer needs a list of these four numbers in the order of (x,y,width,height). The next line of code draws this rectangle. The first two numbers in the list define the upper left corner at (20,20). The next two numbers specify first the width of 250 pixels, and then the height of 100 pixels. The 2 at the end specifies a line width of 2 pixels.

`pygame.draw.rect(screen ,black ,[20,20,250,100],2)`



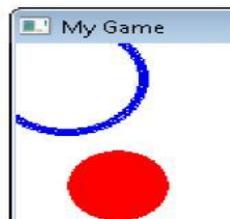
Drawing arcs

Draw an arc as part of an ellipse. Use radians to determine an angle to draw. `pygame.draw.arc(screen ,black ,[20,220,250,200], 0, pi/2, 2)`

`pygame.draw.arc(screen ,green ,[20,220,250,200], pi/2, pi, 2)`

`pygame.draw.arc(screen ,blue , [20,220,250,200], pi ,3*pi/2, 2)`

`pygame.draw.arc(screen ,red , [20,220,250,200],3*pi/2, 2*pi, 2)`



Drawing polygon



PROGRAMMING LABORATORY LAB MANUAL

The next line of code draws a polygon.

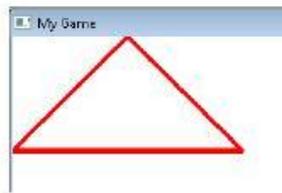
The triangle shape is defined with three points at (100,100) (0,200) and (200,200). It is possible to list as many points as desired. Note how the points are listed. Each point is a list of two numbers, and the points themselves are nested in another list that holds all the points.

Drawing a triangle

This draws a triangle using the polygon command

```
pygame.draw.polygon(screen ,black ,[[100,100],[0,200] ,[200,200]] ,5)
```

B.1. DRAW



Ending the program

Right now, clicking the “close” button of a window while running this Pygame program in IDLE will cause the program to crash. This is a hassle because it requires a lot of clicking to close a crashed program.

By calling the command below, the program will exit as desired. This command closes and cleans up the resources used by creating the window.

Proper shutdown of a Pygame program

Be IDLE friendly. If you forget this line , the program will hang and not exit properly.

```
pygame.quit ()
```

Conclusion:

Thus we have successfully studied how to implement a python program to draw a simple polygon (Square, Rectangle, Triangle)



PROGRAMMING LABORATORY LAB MANUAL

Assignment no: 08

Title:- Write a C/C++ program to draw a convex polygons (Square, Rectangle, Triangle).

Pre-requisite:- Basic Graphics Primitives and functions.

Aim:- Convex polygons is generated as per the choice

Hardware Used:- P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: - TC++

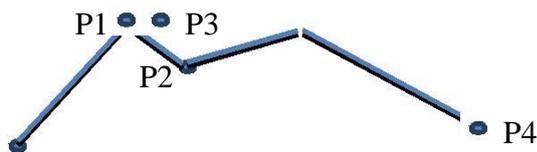
Theory:

A convex polygon is a simple polygon whose interior is a convex set. In a convex polygon, all interior angles are less than 180 degrees. The following properties of a simple polygon are all equivalent to convexity:

Every internal angle is less than or equal to 180 degrees. Every line segment between two vertices remains inside or on the boundary of the polygon. The polygon is entirely contained in a closed half-plane defined by each of its edges. For each edge, the vertices not contained in the edge are on the same side of the line that the edge defines. The angle at each vertex contains all other vertices in its interior (except the three vertices defining the angle).

Convex Polygons: In a convex polygon, any line segment joining any two inside points lies inside the polygon.

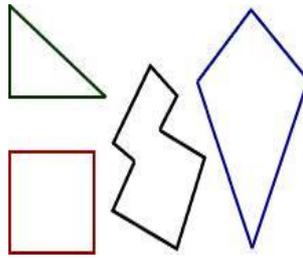
A polyline is a chain of connected line segment. It is specified by the vertices $P_0, P_1, P_2 \dots$ and so on. The first vertices is called the initial or starting point and the last vertex is called the final or terminal point. When starting point and terminal point is same then it is called polygon.



The line segments which make up the polygon boundary are called sides or edges. The endpoints of polygon are called the polygon vertices. Simplest polygon is a triangle



PROGRAMMING LABORATORY LAB MANUAL

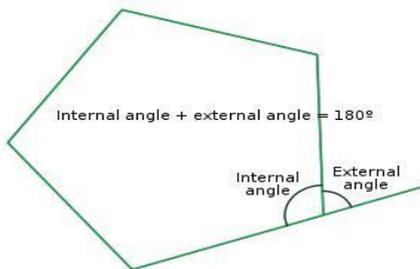


We can divide polygon into three classes

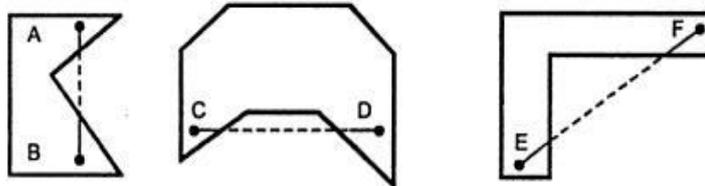
1. Convex
2. Concave
3. Complex

A convex polygon is a simple polygon whose interior is a convex set. Following are properties of a simple polygon

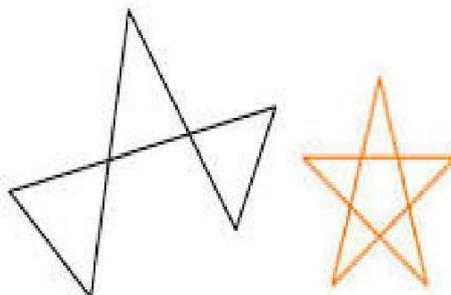
- Every internal angle is less than 180 degrees.
- Every line segment between two vertices remains inside or on the boundary of the polygon.



A concave polygon will always have an interior angle greater than 180 degrees. It is possible to cut a concave polygon into a set of convex polygons



- A complex polygon is neither convex nor concave.





PROGRAMMING LABORATORY LAB MANUAL

Application in Real world (Min. Two) with explanation:

1. Education and Training Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.
2. Visualization For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

Questions to be asked

1. What is polygon? Explain their types?
2. What is mean by programmable edges?

Conclusion: Thus we've understood the concept of drawing convex polygon.

Assignment no: 09

Title: - Write a C/C++ program to draw a convex polygons (Square, Rectangle, Triangle) using programmable edges.

Pre-requisite:- Basic Graphics Primitives and functions.

Aim:-Convex polygons is generated as per the choice

Hardware Used:-P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used:-TC++

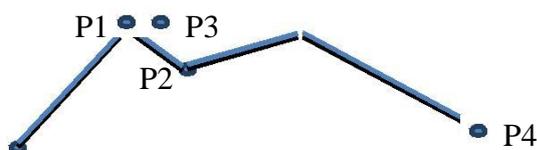
Theory: -

A convex polygon is a simple polygon whose interior is a convex set. In a convex polygon, all interior angles are less than 180 degrees. The following properties of a simple polygon are all equivalent to convexity:

Every internal angle is less than or equal to 180 degrees. Every line segment between two vertices remains inside or on the boundary of the polygon. The polygon is entirely contained in a closed half-plane defined by each of its edges. For each edge, the vertices not contained in the edge are on the same side of the line that the edge defines. The angle at each vertex contains all other vertices in its interior (except the three vertices defining the angle).

Convex Polygons: In a convex polygon, any line segment joining any two inside points lies inside the polygon.

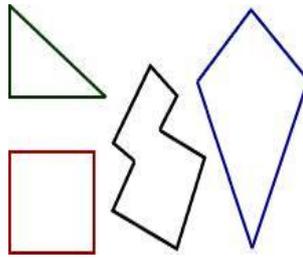
A polyline is a chain of connected line segment. It is specified by the vertices $P_0, P_1, P_2 \dots$ and so on. The first vertices is called the initial or starting point and the last vertex is called the final or terminal point. When starting point and terminal point is same then it is called polygon.



The line segments which make up the polygon boundary are called sides or edges. The endpoints of polygon are called the polygon vertices. Simplest polygon is a triangle



PROGRAMMING LABORATORY LAB MANUAL

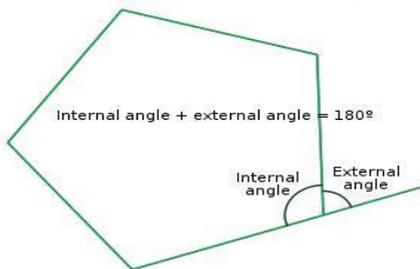


We can divide polygon into three classes

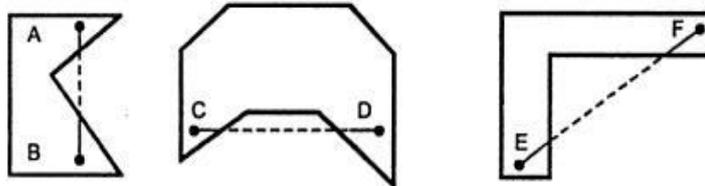
1. Convex
2. Concave
3. Complex

A convex polygon is a simple polygon whose interior is a convex set. Following are properties of a simple polygon

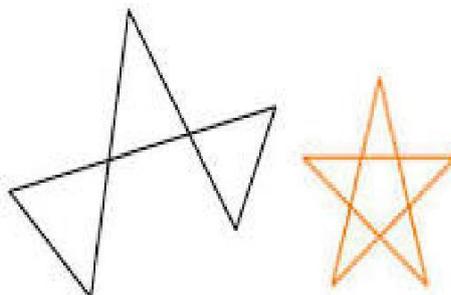
- Every internal angle is less than 180 degrees.
- Every line segment between two vertices remains inside or on the boundary of the polygon.



A concave polygon will always have an interior angle greater than 180 degrees. It is possible to cut a concave polygon into a set of convex polygons



- A complex polygon is neither convex nor concave.





PROGRAMMING LABORATORY LAB MANUAL

Conclusion: Thus we've understood the concept of drawing convex polygon.

FAQ:

1. What is polygon and Polyline?
2. What are the different types of polygon?
3. Which are the various approaches used to represent polygon?
4. Explain the condition for convex polygon.
5. Explain the condition for concave polygon.

Conclusion: Thus we've understood the concept of drawing convex polygons (Square, Rectangle, Triangle) using programmable edges.

Assignment no: 10

Title: Write a program in Java to draw a concave polygon.

Pre-requisite: Basic Graphics Primitives and functions.

Aim: Convex polygons is generated as per the choice

Hardware Used:-P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used:-Java

Theory:

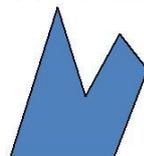
Polygon :It is a closed polyline.

Types of Polygon:

1. **Convex Polygon:** A convex polygon is a polygon such that for any two points inside the polygon, all points on the line segment connecting them are also inside the Polygon.



2. **Concave Polygon :**A concave polygon is a polygon such that for any two points inside the polygon, if some points on the line segment connecting them are not inside the Polygon.



Graphics in java :To do custom graphics in a JAVA application, write a new class that extends the [JPanel](#) class. In that class, override the definition of the paintComponent() method.

A Custom Graphics Template:



PROGRAMMING LABORATORY LAB MANUAL

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import java.swing.event.*;

public class ClassName extends JPanel {

    public void paintComponent(Graphics g) {        super.paintComponent(g);

        }

    }

}
```

Java Swing:

It is java framework. Java Swing is a lightweight Java graphical user interface (GUI) widget toolkit that includes a rich set of widgets. It includes several packages for developing rich desktop applications in Java. Swing includes built-in controls such as trees, image buttons, tabbed panes, sliders, toolbars, color choosers, tables, and text areas to display HTTP or rich text format (RTF).

Scanner class simplifies console input. The Scanner class is a class in java.util, which allows the user to read values of various type.

1. **import java.awt.*** : AWT stands for **Abstract Window Toolkit**. The Abstract Window Toolkit provides many classes for programmers to use. It is the connection between the application and the GUI. It contains classes that programmers can use to make graphical components, e.g., buttons, labels, frames
2. **import java.util.*** : The java.util package contains classes that deal with collections, events, date and time, internationalization and various helpful utilities.
3. **import java.swing.*** : Swing is built on top of AWT, and provides a new set of more sophisticated graphical interface components.
4. **import java.swing.event.*** : This package defines classes and interfaces used for event handling in the AWT.

Here **Jpanel** is the **canvas** on which drawing is done.

paintComponent() method is pre-defined. If we want to draw something on the panel, then you need to override it.

A **Graphics object g** controls the visual appearance of a Swing component.

Graphics object is obtained as a parameter to the paintComponent() method. Once you have the Graphics object, you can send it messages to change the color or font that it uses, or to draw a variety of geometric figures.



PROGRAMMING LABORATORY LAB MANUAL

super.paintComponent(g) invokes the paintComponent method from the superclass of JPanel (theJComponent class) to erase whatever is currently drawn on the panel. **This is useful for animation.**

JFrame:

A Frame is a top-level window with a title and a border. A frame, implemented as an instance of the JFrameclass, is a window that has decorations such as a border, a title, and supports button components that close or magnify the window. Applications with a GUI usually include at least one frame.

Conclusion:- Thus we've drawn Simple Polygon in Java.

Assignment no: 11

Title: - Write a C/C++ program to fill polygon using scan line algorithm.

Pre-requisite: Basic Graphics Primitives and functions.

Aim: - Scan-fill method fills the polygon.

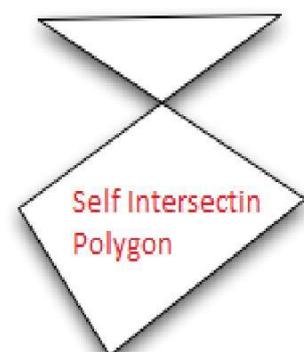
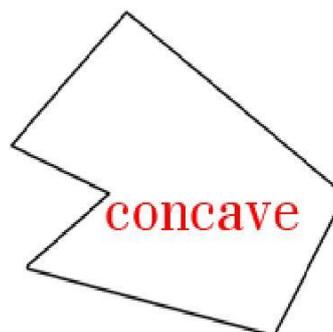
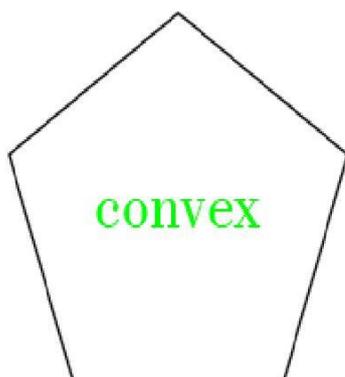
Hardware Used:- P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used:- TC++

Theory: -

Different types of Polygons

- Simple Convex
- Simple Concave
- Non-Simple: Self-intersecting
- With holes





PROGRAMMING LABORATORY LAB MANUAL

Polygon Filling Approaches:

1. Scan Line Fill approaches

Determine overlap intervals for scan lines that cross the area. This approach is mostly used in general graphics packages

E.g. Scanline fill algorithm

2. Seed fill approaches

Start from interior point and paint outward until boundary conditions are encountered. This approach is used in applications having complex boundaries and interactive painting systems E.g. Boundary Fill & Flood Fill algorithm

Non Recursive Flood Fill algorithm with implementation of inside test

Seed(x, y) is the seed pixel

Push is a function for placing a pixel on the stack

Pop is a function for removing a pixel from the stack Pixel(x, y) =seed(x, y)

Initialize

Stack Push

Pixel(x, y)

While (stack not empty)

Get a Pixel from the stack pop

Pixel(x, y)

If (pixel(x, y) != new value)

Then Pixel(x, y) =new value

End if

Examine the surrounding pixels to see if they should be placed onto the

stack if(pixel(x+1,y)!=new value and inside pixel(x+1,y)==true) then

Pixel(x+1,y)=new value

End if

if(pixel(x,y+1)!=new value and inside

pixel(x,y+1)==true) then Pixel(x,y+1)==new value

End if

if(pixel(x-1,y)!=new value and inside pixel(x-

1,y)==true) Then pixel(x-1,y)=new value

End if

If(pixel(x,y-1)!=new value and inside pixel(x,y-1)==true)

then pixel(x,y-1)==new value

End if

End while

Inside/Outside Test: insidepixel(a,b)

Odd/Even Test

- Draw the line from point P(a,b) to a point outside the polygon
- Count the number of edges crossing along that line
- If this count is odd then P is interior return true



PROGRAMMING LABORATORY LAB MANUAL

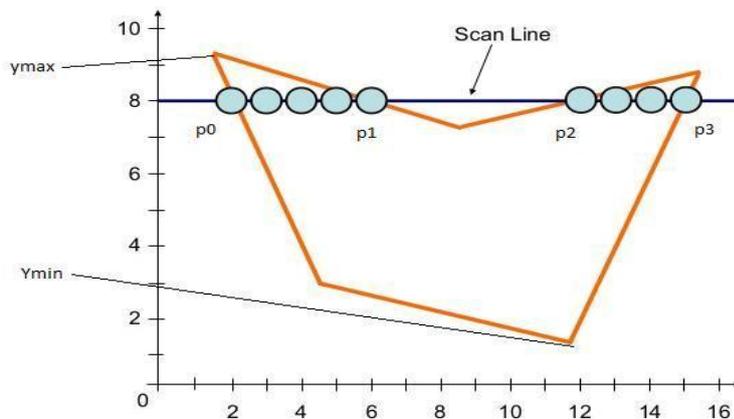
- Else P is exterior return false

Scan Line Polygon Fill Algorithm

Basic algorithm:

For $y=y_{min}$ to y_{max}

- Intersect scanline y with each edge
- Sort intersections by increasing $x[p_0, p_1, p_2, p_3]$
- Fill Pairwise($p_1 \rightarrow p_1, p_2 \rightarrow p_3, \dots$)



Global Edge Table:

- Contains information about all edges of the polygon
- Entries are
 - Theymax value for the edge
 - The x value at y_{min} of that edge
 - $1/\text{slope}$ ($1/m$) for that edge

Active Edge Table:

- Contains information about all edges intersected by the current scan line
- Entries are
 - Theymax value for that edge
 - The x value for the intersection of the polygon edge with this scan line
 - The x increment value ($1/m$)

Scan Line Polygon Fill Algorithm

Set the Global Edge Table to include all edges of the polygon.

Set Y to be the smallest y coordinate

Initialize the Active Edge Table(AET) to be empty

Repeat until the AET and ET are empty:

Add edges from the ET to the AET in which $y_{min}=Y$

Remove edges from the AET in which

$y_{max}=Y$ sort AET on x

Fill pixels between pairs of intersections in the AET



PROGRAMMING LABORATORY LAB MANUAL

For each edge in the AET, replace x with $x+1/m$
Set $Y=Y+1$ to move to the next scan line.

Conclusion-

Thus we've understood different polygon scan fill algorithms and implemented them in laboratory

Assignment no: 12

Title: - Draw a line using OpenGL

Pre-requisite:- Basic Graphics Primitives and functions.

Aim: - Bouncing ball animation in Maya

Hardware Used: P-IV Machine, RAM-1GB HDD: 10GB, Graphics card

Software Used: Maya

Theory:-

Comprehensive 3D animation software

Maya® 3D animation, modeling, simulation, rendering and compositing software offers a comprehensive creative feature set for 3D computer animation, modeling, simulation and rendering on a highly extensible production platform. Maya provides high-end character and effects toolsets along with increased productivity for modeling, texturing and shaded creation tasks.

Toolsets for character creation and digital animation

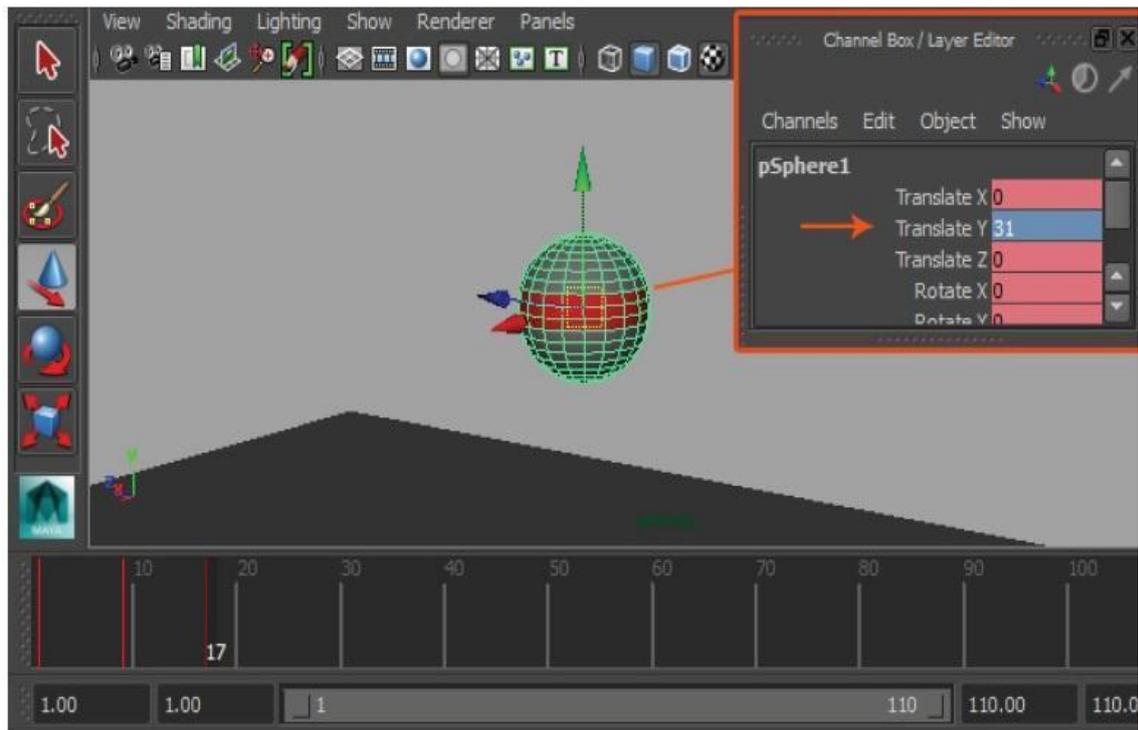
View Maya® demo videos showing features that enable you to tackle challenging character creation and digital animation productions. Get powerful integrated 3D animation, modeling, simulation, effects and rendering tools on a robust and extensible CG pipeline core.

Bouncing ball animation in Maya 2015

Screenshots-



PROGRAMMING LABORATORY LAB MANUAL



Lonikand, Pune



PROGRAMMING LABORATORY LAB MANUAL

