Assignment 1

Q1. What are the different approaches to fill a polygon

Ans. The approaches to filling a polygon are as follows:-

1. Seed fill algorithms:-

i) Boundary fill

- Boundary fill algorithm starts with some interior pixel of a polygon, called seed pixel and keeps filling neighbouring pixels in outward directions until the boundary color is encountered.
- Neighbouring pixels are approached using 4-connectivity or 8-connectivity.
- Algorithm:-

```
void BoundaryFill4(int x, int y, colour newcolor, colour edgecolor)
{
    int current;
    current = ReadPixel (x,y);
    if (current != edgecolor && current != newcolor )
    {
        BoundaryFill4(x+1, y, newcolor, edgecolor);
        BoundaryFill4(x-1, y, newcolor, edgecolor);
        BoundaryFill4 (x, y+1, newcolor, edgecolor);
        BoundaryFill4 (x, y-1, newcolor, edgecolor);
    }
}
```

ii) Flood Fill

- Many realistic objects have a different colored boundaries. Boundary fill algorithm cannot fill the polygon with multi-colored boundary.
- Flood fill algorithm can handle this issue.
- The algorithm starts with the interior pixel, fill color and old color. If the color of the current pixel is the same as old color then it fills the pixel with fill color and examines its neighbours

recursively.

- Algorithm:

```
void FloodFill4 (int x, int y, color newcolor, color old Color)
{
    if (Read Pixel (x,y) == old Color)
    {
        Flood Fill 4 (x+1, y, newcolor, old Color);
        Flood Fill 4 (x-1, y, newcolor, old Color);
        Flood Fill 4 (x, y+1, newcolor, old Color);
        Flood Fill 4 (x, y-1, newcolor, old Color);
    }
}
```

2. Scan line Fill

- ~~Scan~~ Scan line fill algorithm computes the visible span of each scan line within polygon ~~var~~ boundaries.
- It works with convex, concave, self intersecting and polygon with holes.
- Approach:-

→ Find intersections of scan line with polygon edges

→ Sort all intersections on their x-co-ordinate.

→ Fill the span using a pair of sorted intersection points.

- Algorithm:

```
→ For y = ymin to ymax
i)   intersect scanline y with each edge
ii)  sort intersections in increasing x [p0, p1, p2, p3]
iii) fill pairwise (p0 → p1, p2 → p3, ......)
```

Q2. what are the advantages and drawbacks of scanline polygon fill algorithm?

Ans. Advantages:-

- It takes the advantage of edge coherence, which results in to faster execution.
- It only draws visible pixels.
- It requires less memory
- This is a common choice in software routine.

→ Drawbacks:

- It is more complex
- It requires all polygon should be sent to render before drawing.

Dp.

Assignment - 2

Q1. What is the limitation of Cohen Sutherland Line Clipping Algorithm?

Ans. Limitations:-

i) • Clipping window region can be rectangular in shape only and no ~~outer~~ other polygonal shaped window is allowed.

• Edges of rectangular shaped clipping window has to be parallel to the x-axis and y-axis.

• If end points of line segment lies in the extreme limits i.e, one at RHS and other at LHS, one at the top and other at the bottom (~~dig~~ diagonally) then, even if the line doesn't pass through the clipping region it will have logical intersection of 0000 implying that line segment will be clipped but in fact it is not so.

ii) Repeated clipping is expensive.

iii) It can be improved using more regions (eg: Nichol ~~Lee~~ Lee Nichol Approach).

Q2. What are the advantages of Cohen Sutherland Line clipping?

Ans. • Advantages:-

i) It is easy to ~~implement and~~ understand.

ii) Simple to implement.

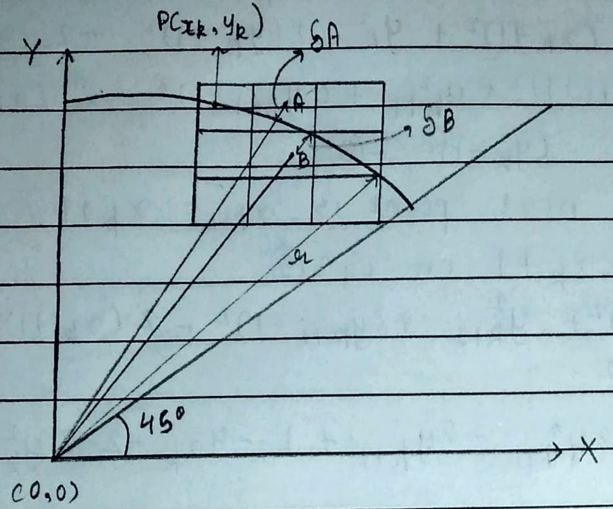
iii) Best suitable for the lines fully inside or outside.

iv) It can easily be extended for 3D line clipping.

Assignment -3

Q1. Explain the derivation of decision parameters in Bresenham's circle drawing algorithm.

Ans.



δA and $\delta B \rightarrow$ distances of A and B from true circle

- $P(x_k, y_k)$ is a point on circle.
- Co-ordinates of Pixel A $\rightarrow$ (x_k+1, y_k-1)
- Co-ordinates of pixel B $\rightarrow$ (x_k+1, y_k-1)

- Distance of A and B from origin:-

$dA = \sqrt{(x_k+1-0)^2+(y_k-0)^2}$

$dA = \sqrt{(x_k+1)^2+y_k^2}$ — ①

$dB = \sqrt{(x_k+1-0)^2+(y_k-1-0)^2}$

$dB = \sqrt{(x_k+1)^2+(y_k-1)^2}$ — ②

- $\delta A = dA - r$

  $\delta B = dB - r$

- To avoid square root, we will define δA and δB as follows:

$\delta A = dA^2 - r^2$ — ③

$\delta B = dB^2 - r^2$ — ④

putting eqn ① and ② in eqn ③ and ④ respectively

$\delta A = (x_k+1)^2 + y_k^2 - r^2$ — ⑤

$\delta B = (x_k+1)^2 + (y_k-1)^2 - r^2$ — ⑥

- Decision parameter d_k can be defined as:-

~~d_k~~ $d_k = \delta A + \delta B$ — putting eqn ⑤ and ⑥

$d_k = (x_k+1)^2 + y_k^2 - r^2 + (x_k+1)^2 + (y_k-1)^2 - r^2$

$d_k = 2(x_k+1)^2 + y_k^2 + (y_k-1)^2 - 2r^2$ — ⑦

replacing k by k+1, next decision parameter,

$d_{k+1} = 2(x_{k+1}+1)^2 + y_{k+1}^2 + (y_{k+1}-1)^2 - 2r^2$

Now,

$$d_{k+1} - d_k = 2(x_{k+1}+1)^2 + y_{k+1}^2 + (y_{k+1}-1)^2 - 2r^2$$
$$- [2(x_k+1)^2 + y_k^2 + (y_k-1)^2 - 2r^2]$$
$$= 2(x_{k+1}+1)^2 + y_{k+1}^2 + (y_{k+1}-1)^2 - 2(x_k+1)^2$$
$$- y_k^2 - (y_k-1)^2 \quad — ⑧$$

- For both A and B, next point is $x_{k+1} = x_k+1$

$\therefore$ put $x_{k+1} = x_k + 1$ in eqn ⑧

$$d_{k+1} - d_k = 2(x_k+1+1)^2 + y_{k+1}^2 + (y_{k+1}-1)^2 - 2(x_k+1)^2 - y_k^2$$
$$-(y_k-1)^2$$
$$= 8x_k + 8 + 2y_{k+1}^2 - 2y_{k+1} + 1 - 4x_k - 2 - 2y_k^2$$
$$+ 2y_k - 1$$
$$d_{k+1} - d_k = 4x_k + 2(y_{k+1}^2 - y_k^2) - 2(y_{k+1} - y_k) + 6 \quad — ⑨$$

Now, $d_k = \delta_A + \delta_B$

→ Case 1: Point A is selected
- if $\delta_A < \delta_B$
- $d_k < 0$
- $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$ } put in eqn

$$d_{k+1} = d_k + 4x_k + 6 \quad — (I)$$

→ Case 2: Point B is selected
- if $\delta_A \geq \delta_B$
- $d_k \geq 0$
- $y_{k+1} = y_k - 1$

$x_{k+1} = x_k + 1$ → put in eqn ⑨

$$d_{k+1} = d_k + 4(x_k - y_k) + 10 \quad — (II)$$

→ To find initial decision parameter d_k :-
- At first pixel in first octant, $x = 0$; $y = r$

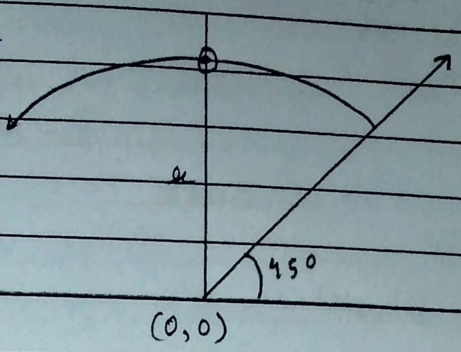
at initial point,

$x_k = 0$; $y_k = r$

Put these values in eqn ⑦,

$d_k = 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2$

$d_k = 3 - 2r$ — $\boxed{III}$



Q2. Explain the concept of encapsulation with example.

Ans.
- In general, encapsulation is a process of wrapping similar code in one place.
- In C++, we can bundle data members and functions that operate together inside a single class.
- For eg:

```
class Rectangle
{
    public:
        int length;
        int breadth;

        int getArea ()
        {
            return length * breadth;
        }
};
```

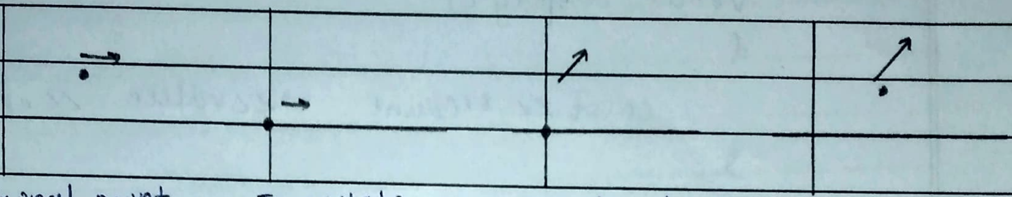
- In the above program, the function getArea() calculates the area of a rectangle.
- To calculate the area, it needs length and breadth.
- Hence, the data members (length and breadth) and the function getArea() are kept together in the Rectangle class.

Assignment-4

Q1. How to rotate any 2-D object an arbitrary point? Explain in brief.

Ans. A point rotation can be performed by a sequence of the following steps:-

1. Translate the pivot point to the origin.
2. Rotate the object.
3. Translate ~~it~~ back the pivot point to its original place.



Original point | Translate to origin | Rotate about Origin | Translate back to original point

- If M is final transformation matrix then,

$$M = T^{-1}.R.T = \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Q2. Explain the concept of operator overloading with example.

Ans. • Operator overloading is a way of providing new implementation of existing operators to work with user defined data types.

- Operator overloading is basically function overloading, where different operator functions have the same symbol but different operands.

eg:

```
class Count
{
```

```
private:
    int value;
public:
    count (): value (5) { }
    void operator ++ (){
        value =value+1;
    }
    void display ()
    {
        cout << "Count: "<<value <<endl;
    }
};
int main()
{   Count   count1;
    ++count 1;
    count1.display();
    return 0;
}
//O/P:
Count: 6
```

Assignment - 5

Q1. What is the ~~most~~ importance of curves and fractals in computer graphics?

Ans. Importance of fractals and curves

- Fractals and curves generating software creates mathematical beauty through ~~[illegible]~~ visualization.
- Modern computers may take seconds or minutes to complete a single high resolution fractal image.
- Images are generated for both simulation (modeling) and random fractals for art.
- Fractal generation used for modeling is part of realism in computer graphics.
- Fractal generation software can be used to mimic natural landscapes with fractal landscapes and scenery generation programs.
- Fractal imagery can be used to introduce irregularity to an otherwise sterile computer generated environment.

Q2. What are the applications of curves and fractals?

Ans. Applications:-

1. Astronomy:-
   - For analysing galaxies, rings of Saturn, etc.
2. Biology / Chemistry:-
   - For depicting bacteria cultures, chemical reactions, human anatomy, molecules, plants.
3. Image compression and resolutions:-
   - Single fractals allow us to convey seemingly random patterns with little data, working with 3D model creation and image resolution.
4. Art:-
   - The simplistic to the complext range of rules that govern fractal creation are down right

allowing for artists.

5. others:-

- For depicting clouds, coastline and borderlines, data compression, diffusion, economy, fractal ~~etc~~ art, fractal music, landscapes special effect etc.

Assignment - 6

Q1. What are the advantages of open GL over other ~~with~~ API's ?

Ans. Advantages of open GL over other API's:

1. Speed
- Raw speed is the main advantage for using open GL.
- The engine powering this speeding is the Graphics Processing Unit (GPU).

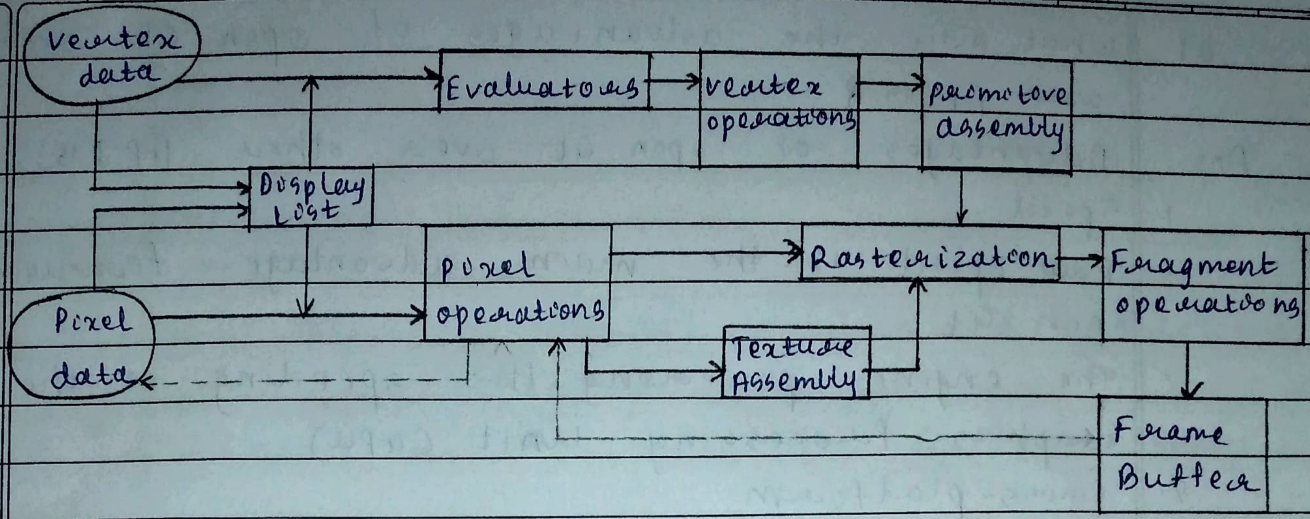
2. Cross-platform
- Open-GL is designed to be ~~plato~~ platform agnostic.
- If you write an openGL implementation for the iPhone, it can be ported to Android with almost no changes and ~~identical~~ you'd get identical results.

3. Web GL:-
- Sometimes a website or ~~we~~ web app makes more business sense than a mobile app ~~(more about~~, basically ruling out high performance graphics.
- Now, we can add hardware-accelerated graphics to websites too, further improving open GL's platform-independence.

4. Shrinking barrier to entry:-
- Open GL can be difficult to get into because of its complexity.
- But every day, new frameworks are being developed that make it increasingly easier, particularly with Web-GL.

Q2. Explain rendering pipeline with reference to OpenGL.

Ans.



1. Geometric data:
- Follows the path that includes evaluators and per-vertex operations, while pixel data (pixels, images and bit maps) are treated different for part of the process.

2. Display Lists:
- All data, whether it describes geometry or pixels, can be saved in a display ~~device~~ list for current or later use.

3. Evaluators:
- It converts curves and surfaces that are described by parametric equations into vertex data.

4. Per-vertex operations:
- Spatial co-ordinates are projected from a position in the 3D world to a position on your screen by Vertex shader.

5. Primitive Assembly:
- After three vertices have been processed by the vertex shader, they are taken to the Primitive Assembly stage.
- This is where a primitive is constructed by connecting the vertices ~~of~~ in a specified order.

6. Rasterization:

- In this stage, pixels are tested to see if they are inside the primitive's perimeter. If they are not, they are discarded.
- If they are within the primitive, they are taken to the next stage.

7. Pixel operations:

- Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components.
- Next, the data is scaled, biased, mapped and processed.

8. Texture Assembly:

- Applying texture images onto geometric objects to make them look more ~~reto~~ realistic.

9. Fragment operations:

- When a fragment leaves the rasterization stage, it is taken to Fragment shader.
- The job of fragment shader is to determine the final color for each fragment.
- Before the pixels on the fragment are sent to the framebuffer, fragments are submitted to several tests like Pixel ownership Test, Scissor Test, Alpha Test, Depth Test.

10. Frame buffer:

- At the end of the pipeline, the pixels are saved in Frame buffer.

Q3. Write OpenGL Advantages.

Ans. Advantages of OpenGL:-

- Easy to begin with
- Learning curve scales from totally simple to most

complex realtime 3D graphics.

- Cross platform (Windows, LINUX, mac, even some handheld devices)
- Stable interface
- Extensible, new hardware features exposed quickly.
- Great discussion and help forum.

Assignment -7

Q1. What is polymorphism?

Ans. • Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

• Polymorphism ~~to~~ uses inherited methods to perform different tasks.

• This allows us to perform a single action in different ways.

• eg:

```
class Animal
{
    public:
        void animalSound()
        {
            cout << "The animal makes a sound\n";
        }
};
class Pig: public Animal
{
    public:
        void animalSound()
        {
            cout<< "The pigs says: wee wee \n";
        }
};
class Dog: public Animal
{
    public:
        void animal Sound()
        {
            cout << "The dog says: bow wow \n";
        }
};
```

```
int main ()
{
    Animal myAnimal;
    Pig myPig;
    Dog myDog;
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
    return 0;
}
//O/P:
The animal makes a sound
The pig says: wee wee
The dog says: bow wow
```