4/5/22 Computer Graphics - Assignment - 01 PAGE NO:

B 1. What is computer graphics? Explain the various applications of the computer graphics

→ Computer graphics deals with generating images with the aid of computers. It refers to several different things: the representation and manipulation of image data by a computer.

→ various applications of the computer graphics are:

1. Computer Art: Using computer graphics, we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes & specifying object motion. Cartoon drawing, paintings, logo design can also be done.

2. Computer Aided Drawing: Designing of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specifications

3. Presentation Graphics: For the preparation of reports or summarising the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer Graphics.

4. Entertainment: Computer Graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures, music video, tv shows, cartoon animation films.

5. Education: Computer generated models are extremely useful for teaching huge number of concepts and fundamentals

in an easy to understand and learn manner.

6. Training: Specialized system for training like simulators can be used for training the candidates in a way that can be grasped in a short span of time with better understanding.

7. Visualization: Today the need of visualize things have increased drastically, the need of visualization can be seen in many advance technologies etc;

8. Image processing: Processing of existing images into refined ones for better interpretation is one of the many applications of computer Graphics.

9. Machine Drawing: CG is very frequently used for designing, modifying and creation of various parts of machine and the whole machine itself.

10. Graphical User Interface: The use of pics, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant.

B2. What is the difference b/w the Raster Scan and Random Scan display

→

Base of Difference	Raster Scan System	Random Scan System.
1. Electron Beam.	The electron beam is swept across the screen, one row at a time form top to bottom	The electron beam is directed only to the parts of screen where a picture is to be drawn.

2.	Resolution	Its resolution is poor because raster system in contract produces zig-zag lines that are plotted as discrete point sets	Its resolution is good because this system produces smooth lines drawings because CRT beam directly follows the line path.
3.	picture Definition	picture definition is stored as set of Integrity values for all screen points called pixels in buffer areas	picture definition is stored as a set of line drawing instruction in a display file.
4.	Realistic Display	The capability of the system to store Integrity value for pixel makes it well suited for the realistic display of screens contain shadow & color pattern.	Here systems are designed for line drawing and can't display realistic shaded scenes.
5.	Draw an Image	Screen points/ pixels are used to draw an Image	Mathematical functions are used to draw an image.

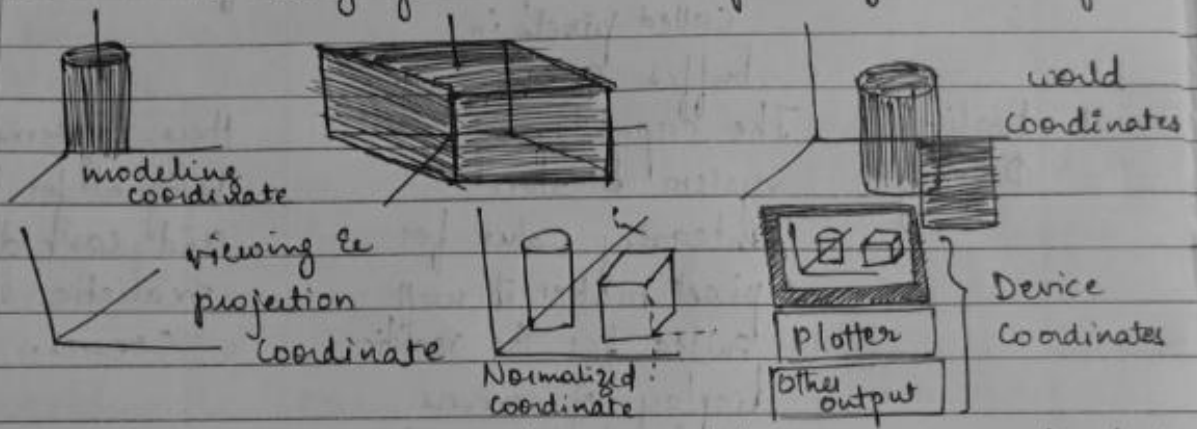
Q3. Explain the diagram of the different Cartesian reference frames are used in the process of Constructing & displaying a Scene.

→ Frame-buffer local line & the corresponding screen positions are referred in cartesian coordinates.

* In an application program we use the commands with in a graphics software package to set coordinate position for

displayed objects relative to the origin of the coordinates

→ Viewing pipeline : World coordinate position are first converted to viewing coordinates corresponding to the view we want of a scene, based on the position & orientation of a hypothetical camera. Then object location are transformed to a 2 dimensional projection of the scene, which corresponds to what we will see on the output device. the scene is then stored in normalized coordinates where each coordinate value is in the range form -1 to 1 (or) in the range form 0 to 1 depending on the system →



→ The transformation sequence from modeling coordinates to device coordinates for a three-dimensional scene. objects shapes can be individually defined in modelling coordinate reference system. Then the shapes are positioned within the world coordinate scenes. Next, world coordinate specifications are transformed through the viewing pipeline to viewing & projection coordinates. At the final step representation of the scene to the output device for display.

→ Normalized coordinates: are also referred to as normalized device coordinates, since using them representation makes a graphics package independent of the coordinate range for any specific output device we also need to identify

visible surface and eliminate picture parts outside the bounds for the view we want to show on display devices.

3D objects: an initial modeling coordinate position (x_{m1}, y_{m1}, z_{m1}) in the illustrate is transformed to world coordinates, then viewing & projection coordinates then to left handed normalized coordinates & finally to device coordinate position. (x_{dc}, y_{dc}) with the sequence.

$(x_{m1}, y_{m1}, m_{m1}) \rightarrow (x_{w1}, y_{w1}, z_{w1}) \rightarrow (x_{v1}, y_{v1}, z_{v1}) \rightarrow (x_{p1}, y_{p1}, z_{p1}) \rightarrow (x_{n1}, y_{n1}, z_{n1}) \rightarrow (x_{d1}, y_{d1})$

→ Device coordinates (x_{d1}, y_{d1}) are integers within the range $(0,0)$ to (x_{max}, y_{max}) for a particular o/p device

In addition to the 2-D positions $(x_{d1} - y_{d1})$ on viewing surface.

Q4 With neccessary steps. Explain the Bresenham's line drawing algorithm Consider the line from (5,5) to (13,9). Use the Bresenham's algorithm to rasterize the line

→ (5,5) B(13,9) wkt, $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \rightarrow (1)$

(x_1, y_1) (x_2, y_2)

Using Eqn (1) we get

$m = \frac{9-5}{13-5} = \frac{4}{8} = \frac{\Delta y}{\Delta x} \rightarrow (2) \Rightarrow m = y_2 = 0.5 \Rightarrow \boxed{m = 0.5}$ (3)

$\therefore \Delta y = 4$ and $2\Delta y = 8$ } → (4)

$\Delta x = 8$ $2\Delta x = 16$ }

while $m = 0.5$ Case $m < 1$

$P_K = 2\Delta y - \Delta x \rightarrow (5)$

$P_{K+1} = P_K + 2\Delta y - 2\Delta x (y_{K-1} - y_K) \rightarrow (6)$

if $(P_K = 0)$

{ $x_{K+1} = x_{K+1}$

$y_{K+1} = y_{K+1}$ } condition to closer x_{K+1} & y_{K+1}

else {

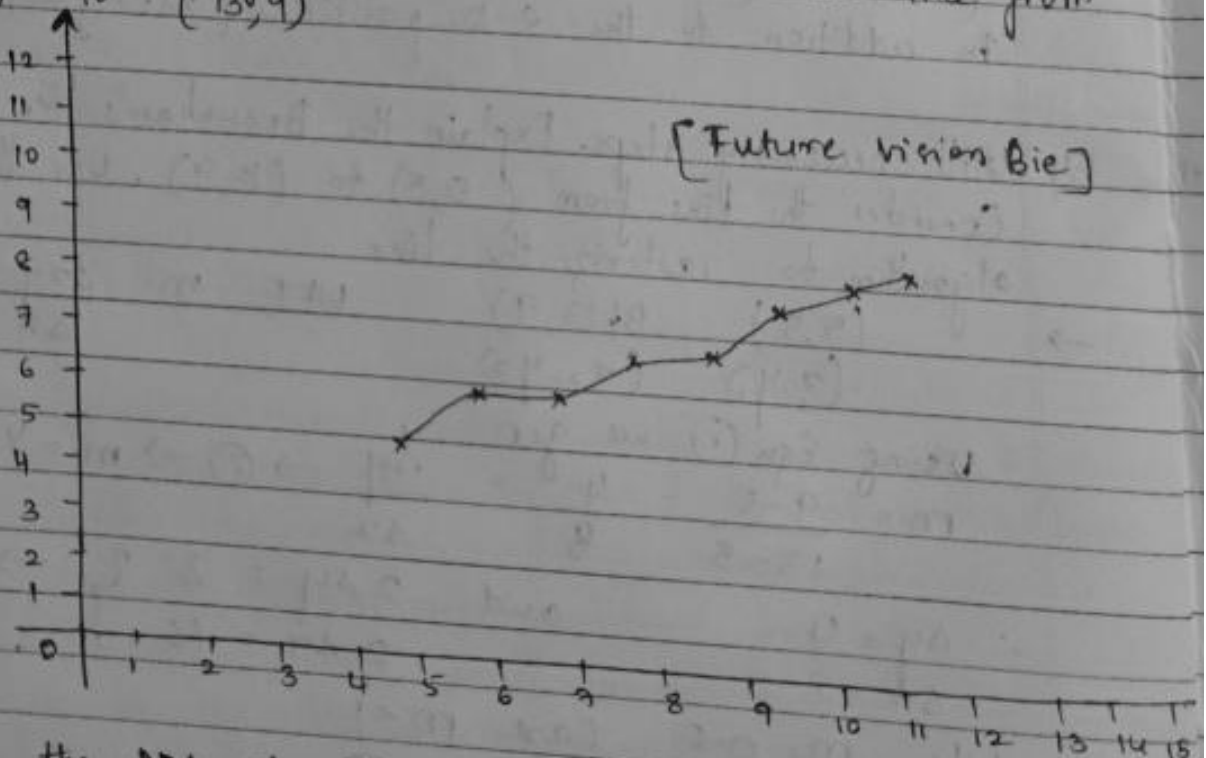
$x_{K+1} = x_{K+1}$

$y_{K+1} = y_K$

PAGE NO.

x_k	y_k	P_k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
-	-	-	5	5	(5,5)
5	5	8-8=0	6	6	(6,6)
6	6	0+8-16(6-5)=-8	7	6	(7,6)
7	6	-8+8+6(6-6)=0	8	7	(8,7)
8	7	0+8-10(7-6)=-8	9	7	(9,7)
9	7	-8+8-16(7-7)=0	10	8	(10,8)
10	8	0+8-16(8-7)=-8	11	8	(11,8)
11	8	0+8-16(8-8)=0	12	9	(12,9)
12	9	0+8-16(9-8)=-8	13	9	(13,9)

Now we have got the value to draw the line from (5,5) to (13,9)



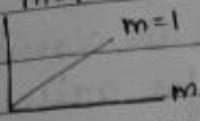
Q5. Explain the DDA algorithm with procedure

→ DDA - Digital Differential ~~Algorithm~~. Analyzer

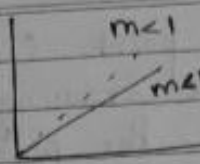
→ points to remember:

* The screen is filled with pixels
* We are in a disclose which pixel to class
* there are 3 cases.

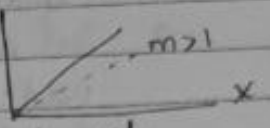
Case 1: m=1



Case 2: m<1



Case 3: m>1



Case 1: m=1 WKT both x and y incremented by 1 unit

$x_{k+1} = x_k + 1 \rightarrow (1)$

$y_{k+1} = y_k + 1 \rightarrow (2)$

Case 2: m<1 WKT x increments by 1 unit

$x_{k+1} = x_k + 1 \rightarrow (3)$

we are in the dilema about y_{k+1}

So ∴ $y = mx + 1 \rightarrow (4)$

$m = \frac{y_2 - y_1}{x_2 - x_1} \rightarrow (5)$

Current pixel = (x_k, y_k) 1 unit in eqn (5)

Next pixel = (x_{k+1}, y_{k+1}).

$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \rightarrow (6)$

$y_{k+1} = ?$ using (6) eqn we get

$m = \frac{y_k + y_{k+1}}{1}$

$\therefore \boxed{y_{k+1} = m + y_k} \rightarrow (7)$

Case 3: m>1

WKT y always increments by 1

$y_{k+1} = y_k + 1 \rightarrow (8)$

$x_k = ?$ (in dilema)

Using Eqn (6) with (8) we get

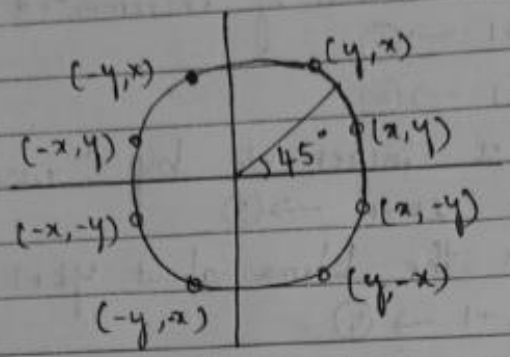
$m = \frac{1}{x_{k+1} - x_k}$

$\boxed{x_{k+1} = x_k + 1/m} \rightarrow (9)$

PAGE NO

Q6. Explain the mid point Circle Algorithm

→ The mid point circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle.



For any given pixel (x,y), the next pixel to be plotted is either (x, y+1) or (x-1, y+1). This can be decided by following the steps below.

1. find the midpoint p of the 2 possible pixels i,e (x-0.5, y+1)
2. If p lies inside or the circle perimeter we plot the pixel (x, y+1), otherwise if it's outside we plot the pixel (x-1, y+1)

→ Boundary Condition:- Whether the midpoint lies inside or outside the circle can be decided by using the formula:
Given a circle centred at (0,0) and radius r and a point p(x,y)

$F(P) = x^2 + y^2 - r^2$

if $F(P) < 0$, the point is inside the circle
$F(P) = 0$, the point is on the perimeter
$F(P) > 0$, the point is outside the circle.

→ In our program we denote F(P) with p. The value of p is calculated at the midpoint of the 2 contending pixels i,e (x-0.5, y+1). Each pixel is described with a subscript k.

$p_k = (x_k - 0.5)^2 + (y_k + 1)^2 - r^2$.

Now,

$x_{k+1} = x_k$ or x_{k-1}, $y_{k+1} = y_{k+1}$

$\therefore p_{k+1} = (x_{k+1} - 0.5)^2 + (y_{k+1}+1)^2 - r^2$

$= (x_{k+1} - 0.5)^2 + [(y_{k+1}) + 1]^2 - r^2$

$= (x_{k+1} - 0.5)^2 + (y_{k+1})^2 + 2(y_{k+1}) + 1 - r^2$

$= (x_{k+1} - 0.5)^2 + [-(x_k - 0.5)^2] + (x_k - 0.5)^2] + (y_{k+1})^2 - r^2 + 2(y_{k+1}) + 1$

$= p_k + (x_{k+1} - 0.5)^2 - (x_k - 0.5)^2 + 2(y_{k+1}) + 1$

$= p_k + (x^2_{k+1} - x^2_k) - (x_{k+1} - x_k) + 2(y_{k+1}) + 1$

$= p_k + 2(y_{k+1}) + 1$, when $p_k <= 0$ i.e the midpoint is inside the circle. $(x_{k+1} = x_k)$

$p_k + 2(y_{k+1}) - 2(x_k - 1) + 1$, when $p_k > 0$ i.e the midpoint is outside the circle $(x_{k+1} = x_{k-1})$

→ The first point to be plotted is $(r, 0)$ on the 'x-axis. The initial value of P is calculated as follows:-

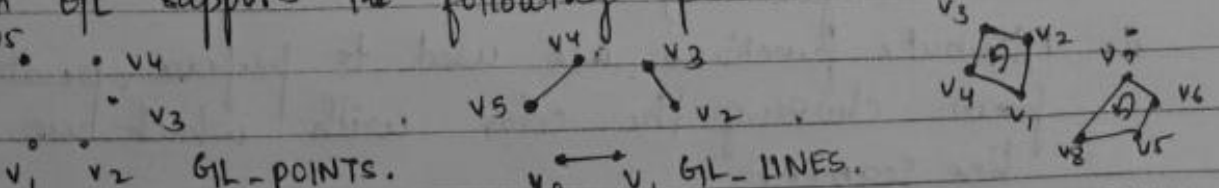
$p_1 = (r - 0.5)^2 + (0+1)^2 - r^2$

$= 1.25 - r$

$= 1 - r$ (when rounded off).

Q7. Explain the different open GL primitive. Explain each primitive with an Eg.

→ In open GL, an object is made up of geometric primitives such as triangle, quad, line segment and point. A primitive is made up of one or more vertices. open GL supports the following primitives.



	Value	Meaning
1.	GL_POINTS	individual points
8.	GL_LINES	pairs of vertices interpreted as individual line segments

3. GL_POLYGON - boundary of a simple, convex polygon
4. GL_TRIANGLES - triples of vertices interpreted as triangles
5. GL_QUADS - quadruples of vertices interpreted as 4-sided polygons
6. GL_LINE_STRIP - series of connected line segments
7. GL_TRIANGLE_STRIP - linked strip of triangles
8. GL_TRIANGLE_FAN - linked fan of triangles
9. GL_QUAD_STRIP - linked strip of Quadrilaterals

Q8. list the major groups of API functions in open GL. With an Eg. Explain any 4 of them.

→ A software API consisting of around several hundred functions that allow you to talk to your graphics hardware. It is the cross-platform and the most commonly used in professional graphics applications

1. primitive functions
2. Attribute functions
3. viewing functions
4. Input functions
5. Transformation functions
6. Control functions
7. Query functions.

1. Primitive functions define the low level objects | atomic entries that our system can display.

Eg:
```
glBegin (GL_POINTS);
    glVertex2f (100.0, 200.0);
glEnd();
```

2. Attribute functions are used to perform operations ranging from choosing the color with which we display a line segment

Eg:
```
glColor3f(1.0, 0.0, 0.0);
glLineWidth (1.0);
```

3. Transformation functions: allows to carry out transformations of objects, such as rotation, translation & scaling.

```
gl Translatef ( tx, ty, tz);
gl Rotatef ( theta, vx, vy, vz);
gl Scalef ( sx, sy, sz);
```

4. Input function: Allows us to deal with the diverse forms of i/p that characterize modern graphics systems

```
void glutKeyboardFunc (void (*func)
(unsigned char key, int x, int y));
```