USN | | | | | | | | | | |

Internal Assessment Test 1 – May 2022

Sub:	COMPUTER GRAPHICS AND VISUALIZATION					Sub Code:	18CS62	Branch:	CSE
Date:	05/05/2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	6th A,B,C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1	Derive matrices for basic 2D transformations with necessary diagrams.	[10]	CO3	L2
2	Explain the following OpenGL primitives with syntax, examples and neat diagrams for the same. a) GL_TRIANGLES b) GL_POLYGON c) GL_TRIANGLE_FAN d) GL_TRIANGLE_STRIP e) GL_QUAD_STRIP	[10]	CO2	L2
3	Consider a triangle with vertices (2,2) (6,2) and (4,4). Apply rotation around the origin and rotation around fixed point (2,2) on the triangle and calculate the coordinates of rotated triangles. Take angle of rotation as 90 degree for both cases.	[10]	CO3	L3
4	Explain Midpoint circle drawing algorithm in detail. Derive corresponding equations	[10]	CO2	L2
5	Illustrate the Bresenham's line drawing algorithm for digitizing the line with end points (20,10) and (30,18).	[10]	CO2	L3
6	What are the polygon classifications? How to identify a convex polygon? Illustrate the methods used to split a concave polygon.	[10]	CO1	L2

Scheme of Evaluation

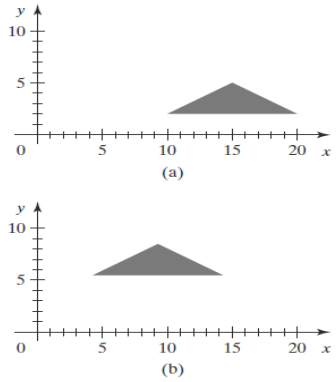
1	Derive matrices for basic 2D transformations with necessary diagrams. Translation derivation with diagram – 3 Rotation derivation with diagram – 4 Scaling derivation with diagram – 3	[10]	CO3	L2
2	Explain the following OpenGL primitives with syntax, examples and neat diagrams for the same. a) GL_TRIANGLES – 2 marks b) GL_POLYGON – 2 marks c) GL_TRIANGLE_FAN – 2 marks d) GL_TRIANGLE_STRIP – 2 marks e) GL_QUAD_STRIP – 2 marks	[10]	CO2	L2
3	Consider a triangle with vertices (2,2) (6,2) and (4,4). Apply rotation around the origin and rotation around fixed point (2,2) on the triangle and calculate the coordinates of rotated triangles. Take angle of rotation as 90 degree for both cases. Rotation around the origin with matrix calculations – 5 marks Rotation around the (2,2) with matrix calculations – 5 marks	[10]	CO3	L3
4	Explain Midpoint circle drawing algorithm in detail. Derive corresponding equations Diagram and Symmetric approach – 3 marks Pk derivation for both cases – 5 marks P0 derivation – 2 makrs	[10]	CO2	L2
5	Illustrate the Bresenham's line drawing algorithm for digitizing the line with end points (20,10) and (30,18). P0 calculation 2 marks. Digitizing table 8 marks.	[10]	CO2	L3
6	What are the polygon classifications? How to identify a convex polygon? Illustrate the methods used to split a concave polygon. Polygon classifications – 2 marks Identifying convex polygon 2 marks Vector method for splitting with diagram and example – 3 marks Rotation method for splitting with diagram and example – 3 marks	[10]	CO1	L2

Solutions

1 Derive matrices for basic 2D transformations with necessary diagrams.

Translation:

- A translation on a single coordinate point can be performed by adding offsets to its coordinates so as to generate a new coordinate position. This relocates the point to a new special position along a straight line.
- Similarly, a translation is applied to an object that is defined with multiple coordinate positions, such as a quadrilateral, by relocating all the coordinate positions by the same displacement along parallel paths. Then the complete object is displayed at the new location.
- To translate a two-dimensional position, we add translation distances tx and ty to the original coordinates x,y to obtain the new coordinate position x1,y1 as shown in the below figure.

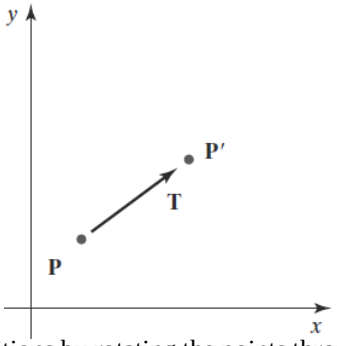


$x' = x + tx$

$y' = y + ty$

- The translation distance pair tx,ty is called a translation vector or shift vector.
- We can express the above equations as a single matrix equation by using the following column vectors:

$P = x\ y \qquad P' = x'\ y' \qquad T = tx\ ty$

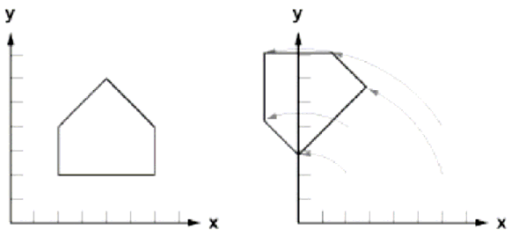


Using the above matrices, the 2D translation equation in the matrix form can be given as,

$$P' = P + T$$

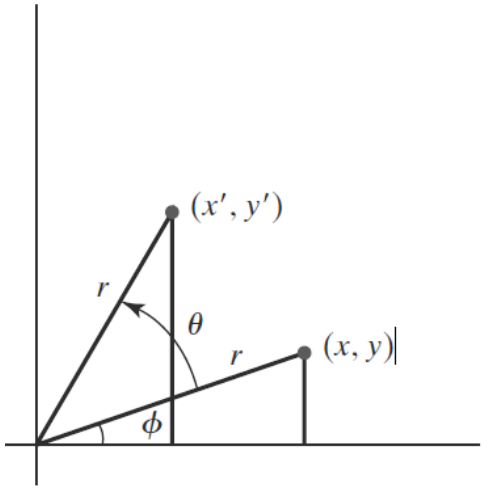
Rotation:

- A rotation transformation of an object can be obtained by specifying a rotation axis and a rotation angle. All points of the object are then transformed to new positions by rotating the points through the specified angle about the rotation axis.
- A two-dimensional rotation of an object is obtained by repositioning the object along a circular path in the xy plane. In this case, we are rotating the object about a rotation axis that is perpendicular to the xy plane (parallel to the coordinate z axis).



- Consider a point position P in the co-ordinate system where P=(x,y) as shown in the below figure.
- In this figure, r is the constant distance of the point from the origin; angle ϕ is the original angular position of the point from the horizontal (x-axis). The (x, y) coordinates can be represented as,

$$x = r\cos\phi \quad y = r\sin\phi \qquad \} \text{ Eq 1}$$



- If we want to rotate the point P by rotation angle θ, Using standard trigonometric identities, we can express the transformed coordinates in terms of angles θ and ϕ as,

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$

$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

Substituting Eq1 in the above equations we get,

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

- We can express the above equations as a single matrix equation by using the following column vectors:

$P = x\ y \qquad P' = x'\ y'$

Using the above matrices, the 2D translation equation in the matrix form can be given as,

$$P' = R.P$$

Where, $R = \cos\theta \quad \sin\theta \quad -\sin\theta \quad \cos\theta$

Scaling:

- Scaling is applied on an object to alter its size. A simple two dimensional scaling operation is performed by multiplying object positions (x, y) by scaling factors sx and sy to produce the transformed coordinates x1,y1 :

$$x' = x.sx$$

$$y' = y.sy$$

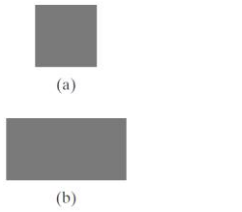


FIGURE 6
Turning a square (a) into a rectangle (b) with scaling factors $s_x = 2$ and $s_y = 1$.

- Scaling factor sx scales an object in the x direction, while sy scales in the y direction.

$$P' = S.P$$

i.e $\quad x'\ y' = sx\ 0\ \ 0\ sy\ \ x\ y$

Where 'S' is the scaling matrix in 2D.

- Any positive values can be assigned to the scaling factors sx and sy. Values less than 1 reduce the size of objects; values greater than 1 produce enlargements. Specifying a value of 1 for both sx and sy leaves the size of objects unchanged.
- Objects transformed using the above equations are both scaled and repositioned. Scaling factors with absolute values less than 1 move objects closer to the coordinate origin, while absolute values greater than 1 move coordinate positions farther from the origin, as shown in the below figure where both the line length and the distance from the origin are reduced by a factor of 0.5.

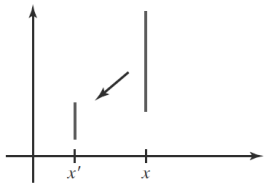


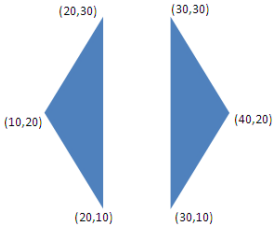
FIGURE 7
A line scaled with Equation 12 using $s_x = s_y = 0.5$ is reduced in size and moved closer to the coordinate origin.

2 Explain the following OpenGL primitives with syntax, examples and neat diagrams for the same.

a) GL_TRIANGLES

OpenGL primitive constant GL_TRIANGLES is used to obtain triangles. Every set of three successive points are considered for one triangle.

```
glBegin(GL_TRIANGLES);
      glVertex2i(30,10);
      glVertex2i(40,20);
      glVertex2i(30,30);
      glVertex2i(20,30);
      glVertex2i(10,20);
      glVertex2i(20,10);
glEnd();
```



In this example, the first three coordinate points define the vertices for one triangle, the next three points define the next triangle, and so forth. For each triangle fill area, we specify the vertex positions in a counterclockwise order.

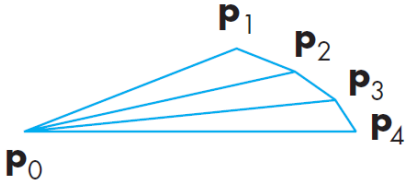
b) GL_POLYGON

OpenGL primitive constant GL_POLYGON is used to obtain polygon of N vertices. All points are joined successively for creating one polygon.

```
glBegin(GL_TRIANGLES);
      glVertex2i(30,10);
      glVertex2i(40,20);
      glVertex2i(30,30);
      glVertex2i(20,30);
      glVertex2i(10,20);
      glVertex2i(20,10);
glEnd();
```

c) GL_TRIANGLE_FAN

A triangle fan is based on one fixed point. The next two points determine the first triangle, and subsequent triangles are formed from one new point, the previous point, and the first (fixed) point.



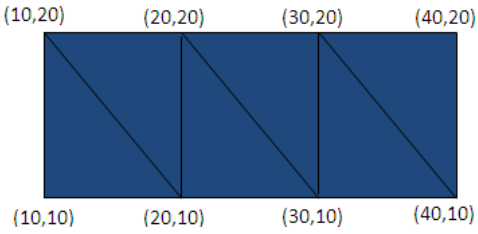

```
glBegin(GL_TRIANGLES);
      glVertex2fv(P0);
      glVertex2fv(P1);
      glVertex2fv(P2);
```

```
        glVertex2fv(P3);
        glVertex2fv(P4);
    glEnd();
```

d) GL_TRIANGLE_STRIP

In the triangle strip, after the first triangle, each additional vertex is combined with the previous two vertices to define a new triangle.

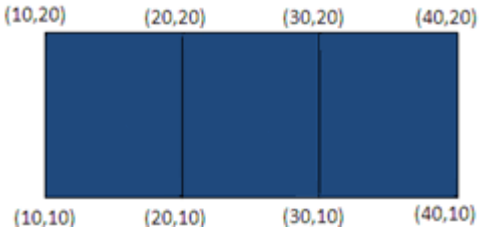
```
    glBegin(GL_TRIANGLES);
        glVertex2i(10,10);
        glVertex2i(10,20);
        glVertex2i(20,10);
        glVertex2i(20,20);
        glVertex2i(30,10);
        glVertex2i(30,20);
        glVertex2i(40,10);
        glVertex2i(40,20);
    glEnd();
```



d) GL_QUAD_STRIP

In the Quad strip, after the first quadrilateral, each additional pair of new vertices are combined with the previous two vertices to define a new quadrilateral.

```
    glBegin(GL_QUAD_STRIP);
        glVertex2i(10,10);
        glVertex2i(10,20);
        glVertex2i(20,10);
        glVertex2i(20,20);
        glVertex2i(30,10);
        glVertex2i(30,20);
        glVertex2i(40,10);
        glVertex2i(40,20);
    glEnd();
```



3

Consider a triangle with vertices (2,2) (6,2) and (4,4). Apply rotation around the origin and rotation around fixed point (2,2) on the triangle and calculate the coordinates of rotated triangles. Take angle of rotation as 90 degree for both cases.

$$R(\theta,0,0) = \begin{matrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$R(\theta,2,2) = \begin{matrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{matrix} = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{matrix}$$

Let $A = \begin{matrix} 2 \\ 2 \\ 1 \end{matrix}$ $B = \begin{matrix} 6 \\ 2 \\ 1 \end{matrix}$ $C = \begin{matrix} 4 \\ 4 \\ 1 \end{matrix}$

Around origin:

$$A' = R(\theta,0,0) * A = \begin{matrix} -2 \\ 2 \\ 1 \end{matrix}$$

$$B' = R(\theta,0,0) * B = \begin{matrix} -2 \\ 6 \\ 1 \end{matrix}$$

$$C' = R(\theta,0,0) * C = \begin{matrix} -4 \\ 4 \\ 1 \end{matrix}$$

Around (2,2):

$$A' = R(\theta,2,2) * A = \begin{matrix} 2 \\ 2 \\ 1 \end{matrix}$$

$$A' = R(\theta,2,2) * B = \begin{matrix} 2 \\ 6 \\ 1 \end{matrix}$$

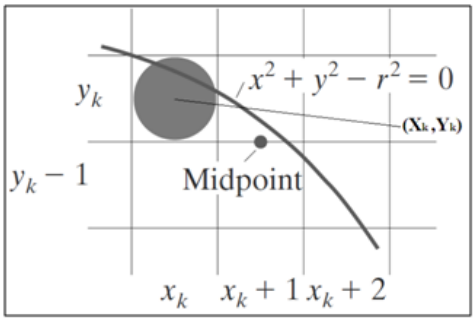
$$A' = R(\theta,2,2) * C = \begin{matrix} 0 \\ 4 \\ 1 \end{matrix}$$

4 Explain Midpoint circle drawing algorithm in detail. Derive corresponding equations

Consider the equation of the circle at some arbitrary point (x, y). The spatial relationship between an arbitrary (x, y) and a circle of radius r centered at the origin can be given as:

$$f(x,y) = x^2 + y^2 - r^2 \begin{cases} < 0 & (x,y) \textit{is inside the circle} \\ = 0 & (x,y) \textit{is on the cirlce} \\ > 0 & (x,y) \textit{is outside the circle} \end{cases}$$

The above tests are performed for the mid positions between pixels near the circle path at each sampling step. Consider the below figure which shows the midpoint between the two candidate pixels at sampling position $x_k + 1$.



- Assume that we have just plotted the pixel at (x_k, y_k) as shown in the above picture. Next we have to decide if the pixel at position (x_{k+1}, y_k) or the one at position (x_{k+1}, y_{k-1}) is closer to the circle.
- To decide on this we make use of the mid-point between these pixels, which is $(x_{k+1}, y_{k-1/2})$ and apply the circle function on the midpoint.

$$p_k = f(x_{k+1}, y_k - 1/2)$$

$$p_k = (x_{k+1})^2 + (y_k - 1/2)^2 + r^2$$

- If $p_k < 0$, this midpoint is inside the circle and hence the pixel at (x_{k+1}, y_k) is closer to the circle boundary and we select it as next pixel to draw.
- Otherwise, the mid-point position is outside or on the circle boundary, and we select the pixel (x_{k+1}, y_{k-1}) as this pixel is closer to the circle boundary.
- Instead of calculating next mid-point and then calculating its mid-point, we calculate the successive decision parameters using incremental calculations.
- We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $(x_{k+1} + 1) = ((x_k + 1) + 1) = x_k + 2$:

$$p_{k+1} = f(x_{k+1} + 1, y_{k+1} - 1/2)$$

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

- Subtracting p_{k+1} and p_k we get,

$$p_{k+1} - p_k = 2(x_k + 1) + ({y_{k+1}}^2 - {y_k}^2) - (y_{k+1} - y_k) + 1$$

- If $p_k < 0$ then $y_{k+1} = y_k$, Hence

$$p_{k+1} = p_k + 2(x_k + 1) + 1$$

$$p_{k+1} = p_k + 2x_k + 3$$

- If $p_k \geq 0$ then $y_{k+1} = y_k - 1$, Hence

$$p_{k+1} = p_k + 2(x_k + 1) - 2y_k + 3$$

$$p_{k+1} = p_k + 2x_k - 2y_k + 5$$

The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$:

$$p_0 = f\left(1, r - \frac{1}{2}\right)$$

$$p_0 = 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$p_0 = \frac{5}{4} - r = 1 - r$$

5

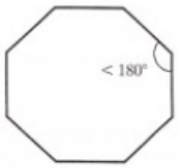
Illustrate the Bresenham's line drawing algorithm for digitizing the line with end points (20,10) and (30,18).
P0 = 2*dy – dx = 6

k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	–2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)

k	p_k	(x_{k+1}, y_{k+1})
5	6	(26, 15)
6	2	(27, 16)
7	–2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

6

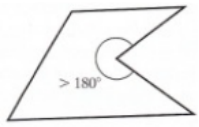
What are the polygon classifications? How to identify a convex polygon? Illustrate the methods used to split a concave polygon.



Polygons are classified into two types: 1) Con

- A polygon is called a convex polygon if all the interior angles of the polygon are less than or equal to 180 degree.
- A polygon is called a concave polygon if at least one of the interior angle of the polygon is greater than or equal to 180 degree.

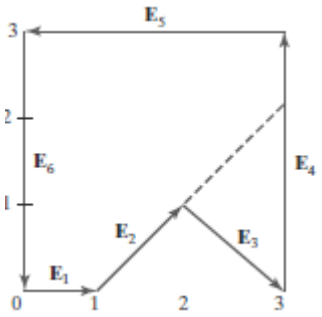
Identifying a Concave polygon:



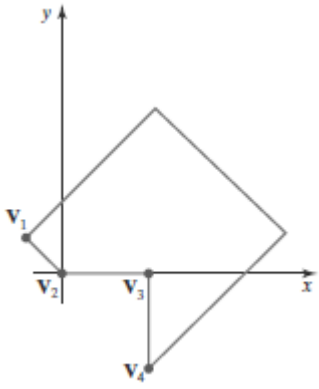
- If we select any two points in the interior of the polygon, for a convex the line segment joining these two points will also be in the interior of the polygon.
- Another way to identify a concave polygon is to extend all edges of the polygon and for some edge if some vertices are on one side of the extension line and some vertices are on the other side, the polygon is concave.
- If we setup a vector for each of edges of the polygon, then we can use cross product of adjacent edges can be used to test concavity. All such vector cross products will produce same sign (+ve or –ve) for a convex polygon. Therefore, if some cross products have +ve and some have –ve signs then such polygon is concave polygon.

Splitting Concave Polygons:

a) Vector method: we first need to form the edge vectors. Given two consecutive vertex positions, Vk and Vk+1, we define the edge vector between them as: $\mathbf{E}_k = \mathbf{V}_{k+1} - \mathbf{V}_k$



Next we calculate the cross-products of successive edge vectors in order around the polygon perimeter. If the z component of some cross-products is positive while other cross-products have a negative z component, the polygon is concave. Otherwise, the polygon is convex. This assumes that no series of three successive vertices are collinear, in which case the cross-product of the two edge vectors for these vertices would be zero.
The following example illustrates this method for splitting a concave polygon.
The figure shows a concave polygon with six edges. Edge vectors for this polygon can be expressed as:
$\mathbf{E}_1 = (1, 0, 0)$ $\mathbf{E}_2 = (1, 1, 0)$
$\mathbf{E}_3 = (1, -1, 0)$ $\mathbf{E}_4 = (0, 2, 0)$
$\mathbf{E}_5 = (-3, 0, 0)$ $\mathbf{E}_6 = (0, -2, 0)$
The crossproduct $\mathbf{E}_j \times \mathbf{E}_k$ for two successive edge vectors is a vector perpendicular to the xy plane with z component equal to $E_{jx}E_{ky} - E_{kx}E_{jy}$:
$\mathbf{E}_1 \times \mathbf{E}_2 = (0, 0, 1)$ $\mathbf{E}_2 \times \mathbf{E}_3 = (0, 0, -2)$
$\mathbf{E}_3 \times \mathbf{E}_4 = (0, 0, 2)$ $\mathbf{E}_4 \times \mathbf{E}_5 = (0, 0, 6)$
$\mathbf{E}_5 \times \mathbf{E}_6 = (0, 0, 6)$ $\mathbf{E}_6 \times \mathbf{E}_1 = (0, 0, 2)$
Since the cross-product $\mathbf{E}_2 \times \mathbf{E}_3$ has a negative z component, we split the polygon along the line of vector $\mathbf{E}_2$.



b) Rotational method:
In this method we proceed counterclockwise around the polygon edges, we shift the position of the polygon so that each vertex Vk in turn is at the coordinate origin. Then, we rotate the

	polygon about the origin in a clockwise direction so that the next vertex Vk+1 is on the x axis. If the following vertex, Vk+2, is below the x axis, the polygon is concave. We then split the polygon along the x axis to form two new polygons, and we repeat the concave test for each of the two new polygons. These steps are repeated until we have tested all vertices in the polygon list. In the above figure after moving V2 to the coordinate origin and rotating V3 onto the x axis, we find that V4 is below the x axis. So we split the polygon along the line of V2-V3, which is the x axis.