

USN

--	--	--	--	--	--	--	--	--	--



Internal Assessment Test 3 – July 2022

Sub:	Computer Graphics and Visualization				Sub Code:	18CS62	Branch:	CSE
Date:	8/7/2022	Duration:	90 mins	Max Marks:	50	Sem/Sec:	6 A,B,C	OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1	Discuss the transformation between color spaces in computer Graphics. Explain the Specular Reflection and the Phong Model.	[10]	CO4	L2
2	Explain the general classification of Visible surface detection and elaborate the Z-buffer algorithm	[10]	CO4	L2
3	What are the different classes of logical input? List and explain the input modes with suitable block diagram.	[10]	CO5	L1
4	Explain the definition and execution of Display List. Write a suitable code snippet to define a display list to create a red square.	[10]	CO5	L3
5	Demonstrate how an event driven input can be performed for a keyboard and mouse device.	[10]	CO5	L2
6	Explain Bezier spline curves with their properties.	[10]	CO5	L2

CI

CCI

HOD/CSE

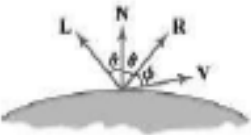

CO PO Mapping

Course Outcomes		Modules	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Understand basics concepts and applications of Computer Graphics	1,2	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CO2	Design and implement algorithms for 2D graphics primitives and attributes.	2,3,5	2	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-
CO3	Illustrate Geometric transformations on both 2D and 3D objects.	2,3,4	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CO4	Understand concepts of clipping and visible surface detection in 2D and 3D viewing, and Illumination Models.	3,4	2	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-
CO5	Design and implement interactive OpenGL graphics programs.	1,2,3,4,5	3	3	3	3	2	-	-	-	-	-	-	2	2	-	3	-

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				

Solution

Sub:	Computer Graphics and Visualization				Sub Code:	18CS62	Branch:	CSE	
Date:	8/7/2022	Duration:	90 mins	Max Marks:	50	Sem/Sec:	6 A,B,C		
Answer any FIVE FULL Questions								MARKS	CO
1	<p>Discuss the transformation between color spaces in computer Graphics. Explain the Specular Reflection and the Phong Model.</p> <p><u>Specular Reflection and the Phong Model</u></p> <ul style="list-style-type: none"> ✓ The bright spot, or specular reflection, that we can see on a shiny surface is the result of total, or near total, reflection of the incident light in a concentrated region around the specular-reflection angle. ✓ The below figure shows the specular reflection direction for a position on an illuminated surface  <ol style="list-style-type: none"> 1. N represents: unit normal surface vector The specular reflection angle equals the angle of the incident light, with the two angles measured on opposite sides of the unit normal surface vector N 2. R represents the unit vector in the direction of ideal specular reflection, 3. L is the unit vector directed toward the point light source, and 4. V is the unit vector pointing to the viewer from the selected surface position. <ul style="list-style-type: none"> ✓ Angle ϕ is the viewing angle relative to the specular-reflection direction R ✓ An empirical model for calculating the specular reflection range, developed by Phong Bui Tuong and called the Phong specular-reflection model or simply the Phong model, sets the intensity of specular reflection proportional to $\cos^m \phi$ ✓ Angle ϕ can be assigned values in the range 0° to 90°, so that $\cos \phi$ varies from 0 to 1.0. ✓ The value assigned to the specular-reflection exponent m is determined by the type of surface that we want to display. ✓ A very shiny surface is modeled with a large value for m (say, 100 or more), and smaller values (down to 1) are used for duller surfaces. ✓ For a perfect reflector, m is infinite. For a rough surface, such as chalk or cinderblock, m is assigned a value near 1. 						[10]	CO4	
2	<p>Explain the general classification of Visible surface detection and elaborate the Z-buffer algorithm</p> <ul style="list-style-type: none"> • We can broadly classify visible-surface detection algorithms according to whether they deal with the object definitions or with their projected images. <ul style="list-style-type: none"> • Object-space methods: compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible. • Image-space methods: visibility is decided point by point at each pixel position on the projection plane. • Although there are major differences in the basic approaches taken by the various visible-surface detection algorithms, most use sorting and coherence methods to improve performance. • Sorting is used to facilitate depth comparisons by ordering the individual surfaces in a scene according to their distance from the view plane. • Coherence methods are used to take advantage of regularities in a scene 						[10]	CO4	

Depth - Buffer Algorithm:

1. Initialize the depth buffer and frame buffer so that for all buffer positions (x, y) ,

$$\text{depthBuff}(x, y) = 1.0, \text{frameBuff}(x, y) = \text{backgndColor}$$

2. Process each polygon in a scene, one at a time, as follows:

- For each projected (x, y) pixel position of a polygon, calculate the depth z (if not already known).

- If $z < \text{depthBuff}(x, y)$, compute the surface color at that position and set

$$\text{depthBuff}(x, y) = z, \text{frameBuff}(x, y) = \text{surfColor}(x, y)$$

After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding color values for those surfaces.

- ❖ Given the depth values for the vertex positions of any polygon in a scene, we can calculate the depth at any other point on the plane containing the polygon.

- ❖ At surface position (x, y) , the depth is calculated from the plane equation as

$$z = \frac{-Ax - By - D}{C}$$

- ❖ If the depth of position (x, y) has been determined to be z , then the depth z' of the next position $(x + 1, y)$ along the scan line is obtained as

$$z' = \frac{-A(x + 1) - By - D}{C}$$

$$z' = z - \frac{A}{C}$$

- ❖ The ratio $-A/C$ is constant for each surface, so succeeding depth values across a scan line are obtained from preceding values with a single addition.

- ❖ We can implement the depth-buffer algorithm by starting at a top vertex of the polygon.

- ❖ Then, we could recursively calculate the x -coordinate values down a left edge of the polygon.

- ❖ The x value for the beginning position on each scan line can be calculated from the beginning (edge) x value of the previous scan line as

$$x' = x - \frac{1}{m}$$

where m is the slope of the edge (Figure below).

- ❖ Depth values down this edge are obtained recursively as

$$z' = z + \frac{A/m + B}{C}$$

- ❖ If we are processing down a vertical edge, the slope is infinite and the recursive calculations reduce to

$$z' = z + \frac{B}{C}$$

- ❖ One slight complication with this approach is that while pixel positions are at integer (x, y) coordinates, the actual point of intersection of a scan line with the edge of a polygon may not be.

- ❖ As a result, it may be necessary to adjust the intersection point by rounding its fractional part up or down, as is done in scan-line polygon fill algorithms.

- ❖ An alternative approach is to use a midpoint method or Bresenham-type algorithm for determining the starting x values along edges for each scan line.

What are the different classes of logical input? List and explain the input modes with suitable block diagram.

- | | | |
|------------|---|--------------|
| 1. String | } | 4. Choice |
| 2. Locator | | 5. Valuator. |
| 3. Pick | | 6. Stroke. |

1. String:

- * Provides ASCII values of input to user program.
- * Implemented by means of physical keyboard.

2. Locators:

- * Provides a position in world coordinates to the user program.
- * Implemented by means of pointing devices like mouse or trackball.

3. Pick:

- * Returns the identifiers of an object on the display to user program.
- * Implemented by pointing devices, ~~but~~ but has a separate software interface to user program.

4) Choice:

- * Allows user to select one of a discrete number of options.
- * OpenGL uses widgets.
- * Widgets - Menus, scroll bars and graphical buttons.

5) Valuator:

- * Provides analog input to user program
- * Boxes or Dials to provide value.

6) Stroke:

- * Returns an array of locations.
- * Example: Pressing a mouse button in sequence and stored in an array. and on release - transfers the data.

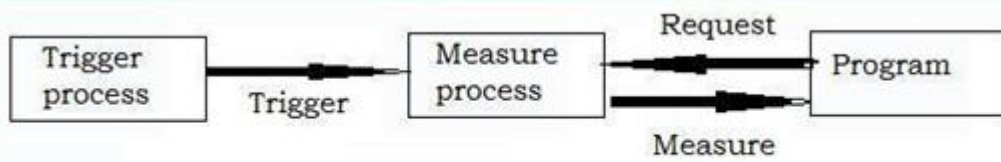


Figure: Request Mode

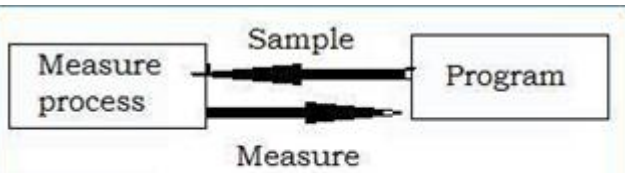


Figure: Sample Mode

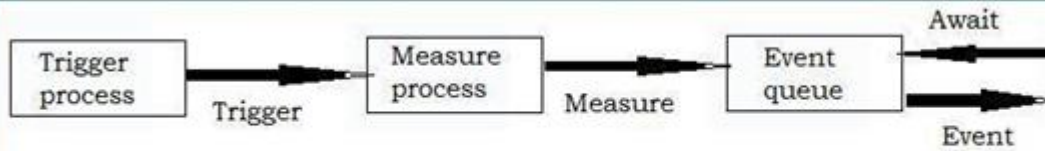


Figure: Event-mode model

4 Explain the definition and execution of Display List. Write a suitable code snippet to define a display list to create a red square.

Definition and Execution of Display List.

Definition:

- * If some objects are to be displayed many times, we can use display list.
- * It is a group of OpenGL commands that is been stored for later execution.

Why use Display list?

- * improves performance
- * same definition can be used multiple times.

Creating a Display List:

```
glNewList(i);
:
glEndList(i);
```

Two flags are defined in NewList.

- ① GL_COMPILE - send list to server but not to display its content
- ② GL_COMPILE_AND_EXECUTE - immediate display of content

Execution of Display List:

```
glCallList(i);
```

Naming and Creating a Display List:

- * display list is identified by an integer index.
- * To avoid overwriting of existing display list

[10]

CO5

`glGenList()` - generates one or more unused indices.

Example:

```
void init ()
```

```
{
```

```
glNewList(SQUARE, GL_COMPILE);
```

```
glColor3f(1.0, 0.0, 0.0);
```

```
glBegin(GL_POLYGON);
```

```
glVertex2f(10, 10);
```

```
glVertex2f(20, 10);
```

```
glVertex2f(20, 20);
```

```
glVertex2f(10, 20);
```

```
glEnd();
```

```
glEndList();
```

```
}
```

```
void display ()
```

```
{ glClear(GL_COLOR_BUFFER_BIT);
```

```
glCallList(SQUARE);
```

```
glFlush();
```

```
}
```

5 Demonstrate how an event driven input can be performed for a keyboard and mouse device.

Keyboard Events:

* Keyboard returns ASCII values to the user program.

* Keyboard event is generated when mouse is in the window and key is pressed.

* Programming keyboard event involves 2 steps:

① The keyboard callback function must be defined as follows.

```
void mykey(unsigned char key, int x, int y)
{
    if (key == 'q')
        exit(0);
}
```

② The keyboard callback function can be registered in the main function as follows:

```
glutKeyboardFunc(mykey);
```

[10]

CO5

Mouse Event:

* There are two types of events associated with a mouse.

① Move event - generated when a mouse is moved with one of the buttons being pressed.

~~* Active mouse event~~

* If mouse is moved without a button being pressed, the event is called Passive mouse event

② Mouse Event: - Generated when mouse button is either pressed or released.

Programming a mouse involves two steps:

① The mouse callback function must be defined.

```
void mymouse (int button, int state, int x, int y)
```

button: GLUT_LEFT_BUTTON

GLUT_RIGHT_BUTTON.

GLUT_MIDDLE_BUTTON.

(x, y) - position of mouse arrow.

state: GLUT_DOWN

GLUT_UP.

② Register the mouse callback function in the main function.

```
glutMouseFunc(mymouse);
```

Example:

```
void square(x, y)
```

```
void mymouse (int button, int state, int x, int y)
```

```
{  
  if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
```

```
    square(x, y);
```

```
  if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)  
    exit(0);
```

```
}
```

```
int main (int argc, char *argv)
```

```
{
```

```
  glutInit (&argc, argv);
```

```
  glutInitDisplayMode (GLUT_RGB | GLUT_SINGLE);
```

```
  glutCreateMenu ("Example");
```

```
  glutDisplayFunc (display);
```

```
  glutMouseFunc (mymouse);
```

```
  glutMainLoop();
```

```
}
```


Bezier Spline Curves:

- * Developed by French engineer Pierre Bezier.
- * used it for Renault automobile.
- * These curves are used for surface design.
- * Easy to implement, used in CAD systems, graphics packages, painting packages.
- * Bezier curves can be filled by control points they are generally limited to four.
- * degree of Bezier polynomial depends upon control points.

Bezier Curve Equations:

Consider, $(n+1)$ control points specified as, $P_k(x_k, y_k, z_k)$

k vary from 0 to n .

- * The points are blended to produce a position vector $P(u)$, which explains the path of a Bezier polynomial function between P_0 and P_n .

$$P(u) = \sum_{k=0}^n p_k \text{BEZ}_{k,n}(u) \quad 0 \leq u \leq 1$$

Bezier Blend function, $\text{BEZ}_{k,n}(u)$ - Bernstein Polynomial.

$$\text{BEZ}_{k,n}(u) = C(n,k) u^k (1-u)^{n-k}$$

$$C(n, k) = \frac{n!}{k!(n-k)!}$$

3 Geo parametric equations

$$x(u) = \sum_{k=0}^n x_k BEZ_{k,n}(u)$$

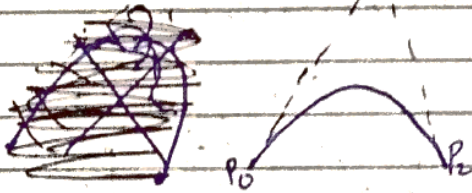
$$y(u) = \sum_{k=0}^n y_k BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^n z_k BEZ_{k,n}(u)$$

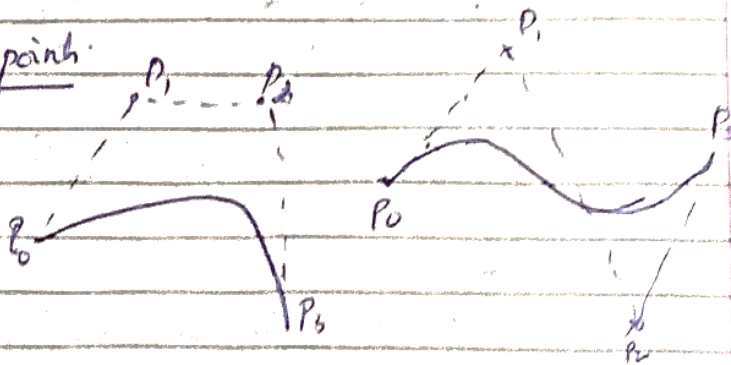
The degree of Bezier curve polynomial is n - no. of control points.

Example:

3 control points - Parabola



4 control points



Recursive Calculation:

$$C(n, k) = \frac{n-k+1}{k} C(n, k-1)$$

for $n \geq k$.

Recursive Relationship

$$BEZ_{k,n}(u) = (1-u) BEZ_{k,n-1}(u) +$$

$$u \cdot BEZ_{k-1,n-1}(u), \quad n \geq k \geq 1.$$

with $BEZ_{k,k} = u^k$.

$$BEZ_{0,k} = (1-u)^k$$

Q. 2) Properties of Bezier Curves -X-

-X-

Property 1

① Curves connects first and last control points.

$$P(0) = P_0$$

$$P(1) = P_n$$

First derivatives:

$$P'(0) = -nP_0 + nP_1$$

$$P'(1) = -nP_{n-1} + nP_n$$

Slope beginning of curve is along the line joining the 1st two control points.
Slope end of the curve is along the line joining last two control points.