

Master of Computer Application (MCA)

Computer Graphics Lab (DMCACO109P24)

Self-Learning Material (SEM 1)



Jaipur National University Centre for Distance and Online Education

**Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
&
NAAC A+ Accredited**

TABLE OF CONTENTS

Question No.	Topic	Page No.
1	Basic Shapes Drawing	1
2	3D Cube Rotation	1
3	Bezier Curve Implementation	1
4	Color Palette Selector	2
5	Flood Fill Algorithm	2
6	Fractal Tree Generator	3
7	Solar System Simulation	3
8	Graphical User Interface Design	3
9	Ray Casting in 2D Grids	4
10	Image Filters and Effects	4
11	Particle System Simulation	5
12	Terrain Generation Using Perlin Noise	5
13	Morphological Transformations on Shapes	5
14	Text Rendering and Styling	6
15	Vector Graphics Editor	6
16	Shadow Mapping in 3D Scenes	6
17	Cloth Simulation	7
18	Cellular Automata for Pattern Generation	7
19	Real-time Video Processing	8
20	Interactive 3D Model Viewer	8

Question 1: Basic Shapes Drawing

Program Statement:

Develop a program to draw basic geometric shapes like lines, rectangles, and circles using a graphics library of your choice.

Solution Hints:

- Start by setting up the graphics environment and choosing functions from the library to set pixel colors.
- Implement Bresenham's algorithm for line drawing.
- Use the midpoint circle algorithm for drawing circles.
- Create user input handlers to specify parameters like size and position for each shape.

Question 2: 3D Cube Rotation

Program Statement:

Write a program that renders a 3D cube and allows the user to rotate it along the X, Y, and Z axes using keyboard inputs.

Solution Hints:

- Use a 3D graphics API like OpenGL or Three.js.
- Define vertices for the cube and use matrix transformations for rotation.
- Implement event listeners for keyboard inputs to adjust rotation angles.
- Apply perspective projection to render the cube in 3D space.

Question 3: Bezier Curve Implementation

Program Statement:

Implement a program to draw a Bezier curve based on user-defined control points.

Solution Hints:

- Start with the cubic Bezier curve formula.
- Allow users to place control points using mouse clicks.
- Dynamically update and render the curve as control points are adjusted.
- Use the de Casteljau's algorithm for curve generation.

Question 4: Color Palette Selector

ProgramStatement:

Create a color palette selector that allows users to mix RGB colors and apply these colors to draw on a canvas.

Solution Hints:

- Implement sliders for Red, Green, and Blue values, updating a display color dynamically.
- Use an on-screen canvas for users to draw with the selected color.
- Handle mouse events to capture drawing actions.
- Store and retrieve different color selections.

Question 5: Flood Fill Algorithm

ProgramStatement:

Program a paint application tool using the flood fill algorithm to fill enclosed areas with a selected color.

Solution Hints:

- Allow users to draw simple shapes as boundaries.
- Implement the flood fill algorithm with stack (non-recursive) to avoid stack overflow issues.
- Provide a GUI with color selection and fill options.
- Ensure boundary and fill color detection to prevent leaks.

Question 6: Fractal Tree Generator

Program Statement:

Write a program that generates a fractal tree with adjustable parameters for branch length, depth, and angles.

Solution Hints:

- Use recursive functions to draw each branch of the tree.
- Provide sliders or input fields to adjust the angle between branches, depth of recursion, and branch length.
- Redraw the tree dynamically as parameters are adjusted to show immediate feedback.

Question 7: Solar System Simulation

Program Statement:

Develop a simple simulation of the solar system with the sun, earth, and moon, demonstrating their rotations and orbits.

Solution Hints:

- Represent celestial bodies with spheres in 3D space.
- Use orbital mechanics basics to calculate and animate the motion of the bodies.
- Provide controls to adjust the speed of rotation and orbit.

Question 8: Graphical User Interface Design

Program Statement:

Create a simple graphical user interface (GUI) with buttons, sliders, and checkboxes that interact with each other.

Solution Hints:

- Use a GUI toolkit like Tkinter, PyQt, or Swing.
- Implement event handling for user interactions with the GUI components.
- Design a layout that dynamically adjusts based on window size.

Question 9: Ray Casting in 2D Grids

Program Statement:

Implement a 2D ray casting algorithm to simulate a light source casting rays within a bounded area with obstacles.

Solution Hints:

- Define a grid with blocks as obstacles.
- Use ray casting mathematics to calculate intersections and reflections off the obstacles.
- Visualize rays extending from the light source until they hit an obstacle or grid boundary.

Question 10: Image Filters and Effects

Program Statement:

Develop a program that applies various filters and effects to images, such as grayscale, sepia, and blur.

Solution Hints:

- Use a graphics library that supports image manipulation.
- Implement pixel-wise operations to alter image data.
- Provide real-time previews of the filters and an option to save the modified image.

Question 11: Particle System Simulation

Program Statement:

Create a particle system where particles are generated at a point and move outward in random directions, simulating smoke or fire.

Solution Hints:

- Generate particles with initial random velocities.
- Update particle positions and reduce life over time until they disappear.
- Use blending modes for particle rendering to create realistic effects.

Question 12: Terrain Generation Using Perlin Noise

Program Statement:

Implement a program to generate a 2D terrain map using Perlin noise, allowing users to adjust the roughness and scaling.

Solution Hints:

- Use a noise function to generate height values at each point in a grid.
- Allow dynamic adjustment of noise parameters to see changes in the terrain.
- Color-code terrain features such as mountains, plains, and water based on height.

Question 13: Morphological Transformations on Shapes

Program Statement:

Program morphological transformations such as erosion and dilation on binary shapes drawn on a canvas.

Solution Hints:

- Allow users to draw arbitrary shapes on a canvas.
- Implement erosion and dilation using structuring elements.
- Update the canvas in real-time to show the results of the transformations.

Question 14: Text Rendering and Styling

Program Statement:

Create a tool that allows users to input text, select a font, and apply styles like bold, italic, and underline.

Solution Hints:

- Use a text rendering library to display text.
- Provide options for font selection and style toggles.
- Render the styled text on a canvas and allow for position adjustments.

Question 15: Vector Graphics Editor

Program Statement:

Develop a simple vector graphics editor that allows users to create and manipulate shapes like polygons, ellipses, and paths.

Solution Hints:

- Implement shape drawing tools and selection mechanisms.
- Allow for transformations such as translation, rotation, and scaling.
- Provide a layering system for complex compositions.

Question 16: Shadow Mapping in 3D Scenes

Program Statement:

Write a program that implements shadow mapping technique to add shadows to objects in a 3D scene.

Solution Hints:

- Use a 3D graphics API to render scenes.
- Implement shadow maps by rendering from the light's perspective.
- Apply depth comparisons to determine shadowed regions.

Question 17: Cloth Simulation

Program Statement:

Develop a simulation of a hanging cloth interacting with environmental forces like wind and gravity.

Solution Hints:

- Model the cloth as a grid of interconnected particles.
- Use physics-based algorithms to simulate forces acting on each particle.
- Render the cloth dynamically, updating the simulation based on user inputs for wind strength and direction.

Question 18: Cellular Automata for Pattern Generation

Program Statement:

Implement a cellular automaton like Conway's Game of Life to generate evolving patterns on a grid.

Solution Hints:

- Set up a grid where each cell can be alive or dead.
- Apply rules for cell survival, birth, and death based on neighbor states.
- Provide start, stop, and reset controls for the simulation.

Question 19: Real-time Video Processing

Program Statement:

Create a program that captures video from a webcam and applies real-time effects like edge detection or color inversion.

Solution Hints:

- Capture video frames using a multimedia library.
- Implement shaders or filter algorithms to modify the frames.
- Display the processed video stream in real-time.

Question 20: Interactive 3D Model Viewer

Program Statement:

Develop an interactive viewer for 3D models where users can load models, view them from different angles, and apply basic lighting.

Solution Hints:

- Support common 3D model formats for loading.
- Implement arcball rotation for user interaction.
- Use basic lighting models to enhance the visual appearance of the models.