**Lab Taks-6**

Submission Guidelines-

* Rename the file with your serial number only
* Must submit within the time discussed in class
* Must include resources for all the section in the table

|  |
| --- |
| **Question-**  Develop an animation that will change the background color of the window after 20ms. Use at least two different colors. |
| **Graph-** |
| **Code-**  **#include <Windows.h>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <cstdlib>**  **#include <cstdio>**  **void display()**  **{**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glFlush();**  **}**  **void updateColor(int a)**  **{**  **float red = ((double)rand() / (RAND\_MAX));**  **float green = ((double)rand() / (RAND\_MAX));**  **float blue = ((double)rand() / (RAND\_MAX));**  **glClearColor(red, green, blue, 1.0f);**  **glutPostRedisplay();**  **glutTimerFunc(20, updateColor, 0);**  **}**  **int main(int argc, char \*\*argv)**  **{**  **glutInit(&argc, argv);**  **glutInitWindowSize(500,500);**  **glutInitWindowPosition(50, 50);**  **glutCreateWindow("Background Color Change");**  **glutDisplayFunc(display);**  **glutTimerFunc(20, updateColor, 0);**  **// Initialize OpenGL state with white background**  **glClearColor(1.0f, 1.0f, 1.0f, 1.0f);**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glFlush();**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-**  **1st color**    **2nd color** |

|  |
| --- |
| **Question-**  Develop an animation that will call four objects separately, each after 20 ms. |
| **Graph-** |
| **Code-**  #include <GL/glut.h>  #include <cmath>  int currentObject = 0;  void drawTriangle() {  glColor3f(1.0f, 0.0f, 0.0f);  glBegin(GL\_TRIANGLES);  glVertex2f(-3.0f, -3.0f);  glVertex2f(3.0f, -3.0f);  glVertex2f(0.0f, 6.0f);  glEnd();  }  void drawRectangle() {  glColor3f(0.0f, 1.0f, 0.0f);  glBegin(GL\_QUADS);  glVertex2f(-4.0f, -2.0f);  glVertex2f(4.0f, -2.0f);  glVertex2f(4.0f, 4.0f);  glVertex2f(-4.0f, 4.0f);  glEnd();  }  void drawCircle() {  glColor3f(0.0f, 0.0f, 1.0f);  glBegin(GL\_TRIANGLE\_FAN);  glVertex2f(0.0f, 0.0f);  for (int i = 0; i <= 360; i += 10) {  float angle = i \* 3.14159265358979323846 / 180.0;  float x = 4.0f \* cos(angle);  float y = 4.0f \* sin(angle);  glVertex2f(x, y);  }  glEnd();  }  void drawPentagon() {  glColor3f(1.0f, 1.0f, 0.0f);  glBegin(GL\_POLYGON);  for (int i = 0; i < 5; ++i) {  float angle = i \* 2 \* 3.14159265358979323846 / 5;  float x = 5.0f \* cos(angle);  float y = 5.0f \* sin(angle);  glVertex2f(x, y);  }  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  switch (currentObject) {  case 0:  drawTriangle();  break;  case 1:  drawRectangle();  break;  case 2:  drawCircle();  break;  case 3:  drawPentagon();  break;  }  glutSwapBuffers();  }  void update(int value) {  currentObject = (currentObject + 1) % 4;  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  // Main function  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(400, 400);  glutCreateWindow("Call Four Objects");  glutDisplayFunc(display);  gluOrtho2D(-10, 10, -10, 10);  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  glutTimerFunc(1000, update, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop a code that will have four different objects (keep it simple). The objects will move to the left, right, up and down in a loop. |
| **Graph-** |
| **Code-** **#include <iostream>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <math.h>**  **using namespace std;**  **float squareX = 0.0f;**  **float triangleX = 100.0f;**  **float circleY = 0.0f;**  **float lineY = 100.0f;**  **void drawSquare(float x, float y) {**  **glBegin(GL\_QUADS);**  **glColor3f(1.0f, 0.0f, 0.0f);**  **glVertex2f(x - 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y + 10.0f);**  **glVertex2f(x - 10.0f, y + 10.0f);**  **glEnd();**  **}**  **void drawTriangle(float x, float y) {**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0.0f, 1.0f, 0.0f);**  **glVertex2f(x, y + 15.0f);**  **glVertex2f(x - 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y - 10.0f);**  **glEnd();**  **}**  **void drawCircle(float x, float y) {**  **glBegin(GL\_POLYGON);**  **glColor3f(0.0f, 0.0f, 1.0f);**  **float radius = 10.0f;**  **for (int i = 0; i <= 360; i += 10) {**  **float theta = i \* 3.14159f / 180.0f;**  **glVertex2f(x + cos(theta) \* radius, y + sin(theta) \* radius);**  **}**  **glEnd();**  **}**  **void drawLine(float x, float y) {**  **glBegin(GL\_LINES);**  **glColor3f(1.0f, 1.0f, 0.0f);**  **glVertex2f(x - 10.0f, y);**  **glVertex2f(x + 10.0f, y);**  **glEnd();**  **}**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **drawSquare(squareX, 50.0f);**  **drawTriangle(triangleX, -50.0f);**  **drawCircle(50.0f, circleY);**  **drawLine(-50.0f, lineY);**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **squareX += 2.0f;**  **if (squareX > 100.0f) squareX = -100.0f;**  **triangleX -= 2.0f;**  **if (triangleX < -100.0f) triangleX = 100.0f;**  **circleY += 2.0f;**  **if (circleY > 100.0f) circleY = -100.0f;**  **lineY -= 2.0f;**  **if (lineY < -100.0f) lineY = 100.0f;**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **void initialize() {**  **glClearColor(0.0f, 0.0f, 0.0f, 1.0f);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluOrtho2D(-100, 100, -100, 100);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 600);**  **glutCreateWindow("Moving Objects with Fixed Vertices");**  **initialize();**  **glutDisplayFunc(drawScene);**  **glutTimerFunc(20, update, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop a code that will have four different objects (keep it simple). Four different keys will be dedicated each objects. The objects will move to the left, right, up and down in a loop as the keys are pressed individually. |
| **Graph-** |
| **Code-**  **#include <iostream>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <cmath>**  **float squareY = 0.0f;**  **float triangleX = 0.0f;**  **float circleY = 0.0f;**  **float lineX = 0.0f;**  **const float boundary = 100.0f;**  **void drawSquare(float x, float y) {**  **glBegin(GL\_QUADS);**  **glColor3f(1.0f, 0.0f, 0.0f);**  **glVertex2f(x - 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y + 10.0f);**  **glVertex2f(x - 10.0f, y + 10.0f);**  **glEnd();**  **}**  **void drawTriangle(float x, float y) {**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0.0f, 1.0f, 0.0f);**  **glVertex2f(x, y + 15.0f);**  **glVertex2f(x - 10.0f, y - 10.0f);**  **glVertex2f(x + 10.0f, y - 10.0f);**  **glEnd();**  **}**  **void drawCircle(float x, float y) {**  **glBegin(GL\_POLYGON);**  **glColor3f(0.0f, 0.0f, 1.0f);**  **float radius = 10.0f;**  **for (int i = 0; i <= 360; i += 10) {**  **float theta = i \* 3.14159f / 180.0f;**  **glVertex2f(x + cos(theta) \* radius, y + sin(theta) \* radius);**  **}**  **glEnd();**  **}**  **void drawLine(float x, float y) {**  **glBegin(GL\_LINES);**  **glColor3f(1.0f, 1.0f, 0.0f);**  **glVertex2f(x - 10.0f, y);**  **glVertex2f(x + 10.0f, y);**  **glEnd();**  **}**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **drawSquare(50.0f, squareY);**  **drawTriangle(triangleX, -50.0f);**  **drawCircle(0.0f, circleY);**  **drawLine(lineX, 50.0f);**  **glutSwapBuffers();**  **}**  **void handleKeypress(unsigned char key, int x, int y) {**  **switch (key) {**  **case 'w':**  **case 'W':**  **squareY += 5.0f;**  **if (squareY > boundary) squareY = -boundary;**  **break;**  **case 's':**  **case 'S':**  **squareY -= 5.0f;**  **if (squareY < -boundary) squareY = boundary;**  **break;**  **case 'a':**  **case 'A':**  **triangleX -= 5.0f;**  **if (triangleX < -boundary) triangleX = boundary;**  **break;**  **case 'd':**  **case 'D':**  **triangleX += 5.0f;**  **if (triangleX > boundary) triangleX = -boundary;**  **break;**  **case 'i':**  **case 'I':**  **circleY += 5.0f;**  **if (circleY > boundary) circleY = -boundary;**  **break;**  **case 'k':**  **case 'K':**  **circleY -= 5.0f;**  **if (circleY < -boundary) circleY = boundary;**  **break;**  **case 'j':**  **case 'J':**  **lineX -= 5.0f;**  **if (lineX < -boundary) lineX = boundary;**  **break;**  **case 'l':**  **case 'L':**  **lineX += 5.0f;**  **if (lineX > boundary) lineX = -boundary;**  **break;**  **}**  **glutPostRedisplay();**  **}**  **void initialize() {**  **glClearColor(0.0f, 0.0f, 0.0f, 1.0f);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluOrtho2D(-100, 100, -100, 100);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 600);**  **glutCreateWindow("Interactive Moving Objects");**  **initialize();**  **glutDisplayFunc(drawScene);**  **glutKeyboardFunc(handleKeypress);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop a scenario where it will rain and gradually create flood |
| **Graph-** |
| **Code-**  **#include <iostream>**  **#include <GL/gl.h>**  **#include <GL/glut.h>**  **#include <cstdlib>**  **#include <ctime>**  **#define MAX\_RAINDROPS 100**  **float raindropX[MAX\_RAINDROPS];**  **float raindropY[MAX\_RAINDROPS];**  **float raindropSpeed[MAX\_RAINDROPS];**  **float waterLevel = -100.0f;**  **float rainIntensity = 0.1f;**  **float floodSpeed = 0.02f;**  **int raindropCount = 50;**  **const float boundaryTop = 100.0f;**  **const float boundaryBottom = -100.0f;**  **const float boundaryLeft = -100.0f;**  **const float boundaryRight = 100.0f;**  **void createRaindrops(int count) {**  **for (int i = 0; i < count; i++) {**  **raindropX[i] = static\_cast<float>(rand() % 200 - 100);**  **raindropY[i] = boundaryTop;**  **raindropSpeed[i] = rainIntensity + static\_cast<float>(rand() % 10) / 100.0f;**  **}**  **}**  **void drawRaindrops() {**  **glColor3f(0.5f, 0.5f, 1.0f);**  **glBegin(GL\_LINES);**  **for (int i = 0; i < raindropCount; i++) {**  **glVertex2f(raindropX[i], raindropY[i]);**  **glVertex2f(raindropX[i], raindropY[i] - 5.0f);**  **}**  **glEnd();**  **}**  **void updateRaindrops() {**  **for (int i = 0; i < raindropCount; i++) {**  **raindropY[i] -= raindropSpeed[i];**  **if (raindropY[i] < boundaryBottom) {**  **raindropY[i] = boundaryTop;**  **raindropX[i] = static\_cast<float>(rand() % 200 - 100);**  **}**  **}**  **}**  **void drawFloodWater() {**  **glColor3f(0.0f, 0.0f, 1.0f);**  **glBegin(GL\_QUADS);**  **glVertex2f(boundaryLeft, waterLevel);**  **glVertex2f(boundaryRight, waterLevel);**  **glVertex2f(boundaryRight, boundaryBottom);**  **glVertex2f(boundaryLeft, boundaryBottom);**  **glEnd();**  **}**  **void updateWaterLevel() {**  **if (waterLevel < boundaryTop) {**  **waterLevel += floodSpeed;**  **}**  **}**  **void drawScene() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **drawFloodWater();**  **drawRaindrops();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **updateRaindrops();**  **updateWaterLevel();**  **rainIntensity += 0.001f;**  **if (raindropCount < MAX\_RAINDROPS) {**  **raindropCount++;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **void initialize() {**  **glClearColor(0,255,255,0);**  **glMatrixMode(GL\_PROJECTION);**  **glLoadIdentity();**  **gluOrtho2D(boundaryLeft, boundaryRight, boundaryBottom, boundaryTop);**  **}**  **int main(int argc, char\*\* argv) {**  **srand(time(0));**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(800, 600);**  **glutCreateWindow("Rain and Flood Simulation");**  **initialize();**  **createRaindrops(raindropCount);**  **glutDisplayFunc(drawScene);**  **glutTimerFunc(20, update, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |