**Lab Taks-6**

|  |
| --- |
| **Question-**  Develop an animation that will change the background color of the window after 20ms. Use at least two different colors. |
| **Graph-** |
| **Code-**  **#include <GL/glut.h>**  **int colorIndex = 0;**  **void setBackgroundColor() {**  **if (colorIndex == 0)**  **glClearColor(0.0f, 0.0f, 1.0f, 1.0f);**  **else**  **glClearColor(1.0f, 0.0f, 0.0f, 1.0f);**  **}**  **void display() {**  **setBackgroundColor();**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glutSwapBuffers();**  **}**  **void timer(int value) {**  **colorIndex = (colorIndex + 1) % 2;**  **glutPostRedisplay();**  **glutTimerFunc(20, timer, 0);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(500, 500);**  **glutCreateWindow("Two-Color Background Animation");**  **glMatrixMode(GL\_PROJECTION);**  **gluOrtho2D(0, 500, 0, 500);**  **glutDisplayFunc(display);**  **glutTimerFunc(20, timer, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop an animation that will call four objects separately, each after 20 ms. |
| **Graph-** |
| **Code-**  #include <GL/glut.h>  #include <cmath>  int currentStep = 0;  void drawSquare() {  glColor3f(1, 0, 0);  glBegin(GL\_QUADS);  glVertex2f(-0.4f, -0.1f);  glVertex2f(-0.2f, -0.1f);  glVertex2f(-0.2f, 0.1f);  glVertex2f(-0.4f, 0.1f);  glEnd();  }  void drawCircle() {  glColor3f(0, 1, 0);  float cx = 0.0f, cy = 0.0f, r = 0.1f;  glBegin(GL\_POLYGON);  for (int i = 0; i < 100; ++i) {  float theta = 2.0f \* 3.1416f \* float(i) / 100;  float x = r \* cosf(theta);  float y = r \* sinf(theta);  glVertex2f(x + cx, y + cy);  }  glEnd();  }  void drawTriangle() {  glColor3f(0, 0, 1);  glBegin(GL\_TRIANGLES);  glVertex2f(0.2f, -0.1f);  glVertex2f(0.4f, -0.1f);  glVertex2f(0.3f, 0.1f);  glEnd();  }  void drawRectangle() {  glColor3f(1, 1, 0);  glBegin(GL\_QUADS);  glVertex2f(-0.1f, 0.2f);  glVertex2f(0.1f, 0.2f);  glVertex2f(0.1f, 0.35f);  glVertex2f(-0.1f, 0.35f);  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  if (currentStep >= 1) drawSquare();  if (currentStep >= 2) drawCircle();  if (currentStep >= 3) drawTriangle();  if (currentStep >= 4) drawRectangle();  glutSwapBuffers();  void timer(int) {  currentStep++;  if (currentStep > 4) currentStep = 0;  glutPostRedisplay();  glutTimerFunc(20, timer, 0);  }  void init() {  glClearColor(0.1f, 0.1f, 0.1f, 1.0f);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-1, 1, -1, 1);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);  glutInitWindowSize(600, 600);  glutCreateWindow("Sequential Object Animation");  init();  glutDisplayFunc(display);  glutTimerFunc(0, timer, 0)  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A computer screen shot of a black and white screen  Description automatically generatedA screenshot of a computer  Description automatically generated** |

|  |
| --- |
| **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 <GL/glut.h>**  **#include <stdbool.h>**  **bool showObj1 = true;**  **bool showObj2 = true;**  **bool showObj3 = true;**  **bool showObj4 = true;**  **float x1 = 0.0f;**  **float x2 = 11.0f;**  **float y3 = 11.0f;**  **float y4 = 0.0f;**  **void drawObject1() {**  **glPointSize(10);**  **glBegin(GL\_POINTS);**  **glColor3f(1.0, 0.0, 0.0);**  **glVertex2f(x1, 2.0f);**  **glEnd();**  **}**  **void drawObject2() {**  **glLineWidth(3);**  **glBegin(GL\_LINES);**  **glColor3f(0.0, 1.0, 0.0);**  **glVertex2f(x2, 4.0f);**  **glVertex2f(x2 - 1.0f, 4.0f);**  **glEnd();**  **}**  **void drawObject3() {**  **glBegin(GL\_TRIANGLES);**  **glColor3f(1.0, 1.0, 1.0);**  **glVertex2f(7.0f, y3);**  **glVertex2f(6.5f, y3 + 1.0f);**  **glVertex2f(7.5f, y3 + 1.0f);**  **glEnd();**  **}**  **void drawObject4() {**  **glBegin(GL\_QUADS);**  **glColor3f(1.0, 1.0, 0.0);**  **glVertex2f(3.0f, y4);**  **glVertex2f(4.0f, y4);**  **glVertex2f(4.0f, y4 + 1.0f);**  **glVertex2f(3.0f, y4 + 1.0f);**  **glEnd();**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **if (showObj1) drawObject1();**  **if (showObj2) drawObject2();**  **if (showObj3) drawObject3();**  **if (showObj4) drawObject4();**  **glutSwapBuffers();**  **}**  **void update(int) {**    **x1 += 0.05f;**  **if (x1 > 11.0f) x1 = 0.0f;**    **x2 -= 0.05f;**  **if (x2 < 0.0f) x2 = 11.0f;**  **y3 -= 0.05f;**  **if (y3 < 0.0f) y3 = 11.0f;**    **y4 += 0.05f;**  **if (y4 > 11.0f) y4 = 0.0f;**  **glutPostRedisplay();**  **glutTimerFunc(30, update, 0);**  **}**  **void init() {**  **glClearColor(0.2f, 0.2f, 0.2f, 1.0f);**  **glMatrixMode(GL\_PROJECTION);**  **gluOrtho2D(0, 11, 0, 11);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(500, 500);**  **glutCreateWindow("4 Object Animation scaled 1 to 10");**  **init();**  **glutDisplayFunc(display);**  **glutTimerFunc(0, 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 <GL/glut.h>**  **#include <stdbool.h>**  **float x1 = 5.0f, y1 = 5.0f;**  **float x2 = 5.0f, y2 = 6.5f;**  **float x3 = 5.0f, y3 = 4.0f;**  **float x4 = 5.0f, y4 = 3.0f;**  **bool moveObj1 = false;**  **bool moveObj2 = false;**  **bool moveObj3 = false;**  **bool moveObj4 = false;**  **const float speed = 0.05f;**  **const float minCoord = 1.0f;**  **const float maxCoord = 10.0f;**  **void drawObject1() {**  **glPointSize(10);**  **glBegin(GL\_POINTS);**  **glColor3f(1, 0, 0);**  **glVertex2f(x1, y1);**  **glEnd();**  **}**  **void drawObject2() {**  **glLineWidth(3);**  **glBegin(GL\_LINES);**  **glColor3f(0, 1, 0);**  **glVertex2f(x2 - 0.3f, y2);**  **glVertex2f(x2 + 0.3f, y2);**  **glEnd();**  **}**  **void drawObject3() {**  **glBegin(GL\_TRIANGLES);**  **glColor3f(0, 0, 1);**  **glVertex2f(x3, y3 + 0.5f);**  **glVertex2f(x3 - 0.3f, y3 - 0.3f);**  **glVertex2f(x3 + 0.3f, y3 - 0.3f);**  **glEnd();**  **}**  **void drawObject4() {**  **glBegin(GL\_QUADS);**  **glColor3f(1, 1, 0);**  **glVertex2f(x4 - 0.3f, y4 - 0.3f);**  **glVertex2f(x4 + 0.3f, y4 - 0.3f);**  **glVertex2f(x4 + 0.3f, y4 + 0.3f);**  **glVertex2f(x4 - 0.3f, y4 + 0.3f);**  **glEnd();**  **}**  **void display() {**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **drawObject1();**  **drawObject2();**  **drawObject3();**  **drawObject4();**  **glutSwapBuffers();**  **}**  **void update(int value) {**  **if (moveObj1) {**  **x1 -= speed;**  **if (x1 < minCoord) x1 = maxCoord;**  **}**  **if (moveObj2) {**  **x2 += speed;**  **if (x2 > maxCoord) x2 = minCoord;**  **}**  **if (moveObj3) {**  **y3 += speed;**  **if (y3 > maxCoord) y3 = minCoord;**  **}**  **if (moveObj4) {**  **y4 -= speed;**  **if (y4 < minCoord) y4 = maxCoord;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(30, update, 0);**  **}**  **void keyboard(unsigned char key, int x, int y) {**  **switch (key) {**  **case '1': moveObj1 = !moveObj1; break;**  **case '2': moveObj2 = !moveObj2; break;**  **case '3': moveObj3 = !moveObj3; break;**  **case '4': moveObj4 = !moveObj4; break;**  **case 27: exit(0); break;**  **}**  **}**  **void init() {**  **glClearColor(0.2f, 0.2f, 0.2f, 1.0f);**  **glMatrixMode(GL\_PROJECTION);**  **gluOrtho2D(minCoord, maxCoord, minCoord, maxCoord);**  **}**  **int main(int argc, char\*\* argv) {**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(500, 500);**  **glutCreateWindow("4 Objects Controlled with Keys 1-4");**  **init();**  **glutDisplayFunc(display);**  **glutKeyboardFunc(keyboard);**  **glutTimerFunc(30, update, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop a scenario where it will rain and gradually create flood |
| **Graph-** |
| **Code-**  **Code-**  #include <GL/glut.h>  #include <vector>  #include <cstdlib>  #include <ctime>  const int numDrops = 100;  float rainX[numDrops];  float rainY[numDrops];  float waterLevel = -1.0f;  float waterRiseSpeed = 0.001f;  void initRain() {  for (int i = 0; i < numDrops; ++i) {  rainX[i] = (rand() % 200 - 100) / 100.0f;  rainY[i] = (rand() % 200) / 100.0f;  }  }  void drawRain() {  glColor3f(0.5f, 0.7f, 1.0f);  glBegin(GL\_LINES);  for (int i = 0; i < numDrops; ++i) {  glVertex2f(rainX[i], rainY[i]);  glVertex2f(rainX[i], rainY[i] - 0.05f);  }  glEnd();  }  void updateRain() {  for (int i = 0; i < numDrops; ++i) {  rainY[i] -= 0.02f;  if (rainY[i] < -1.0f) {  rainY[i] = 1.0f;  rainX[i] = (rand() % 200 - 100) / 100.0f;  }  }  }  void drawWater() {  glColor3f(0.0f, 0.3f, 0.8f); // Dark blue  glBegin(GL\_QUADS);  glVertex2f(-1.0f, -1.0f);  glVertex2f(1.0f, -1.0f);  glVertex2f(1.0f, waterLevel);  glVertex2f(-1.0f, waterLevel);  glEnd();  }  void updateWater() {  if (waterLevel < 1.0f)  waterLevel += waterRiseSpeed;  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  drawRain();  drawWater();  glutSwapBuffers();  }  void timer(int) {  updateRain();  updateWater();  glutPostRedisplay();  glutTimerFunc(40, timer, 0);  }  void init() {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-1, 1, -1, 1);  srand(time(0));  initRain();  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);  glutInitWindowSize(800, 600);  glutCreateWindow("Rain and Flood Simulation");  init();  glutDisplayFunc(display);  glutTimerFunc(0, timer, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A screenshot of a computer  Description automatically generatedA screenshot of a computer  Description automatically generated** |