**Lab Taks-6**

Submission Guidelines-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* 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. |
| **Code-**  #include <GL/glut.h>  #include <cstdlib>  int currentColor = 0; // 0: First color, 1: Second color  // RGB values for two different background colors  GLfloat backgroundColors[2][3] = {  {0.2f, 0.2f, 0.2f}, // First color: Dark Gray  {0.8f, 0.8f, 0.8f} // Second color: Light Gray  };  void drawScene() {  glClearColor(backgroundColors[currentColor][0], backgroundColors[currentColor][1], backgroundColors[currentColor][2], 1.0);  glClear(GL\_COLOR\_BUFFER\_BIT);  glutSwapBuffers();  }  void update(int value) {  // Toggle between the two colors  currentColor = (currentColor + 1) % 2;  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(20, update, 0); // Call update again after 20 milliseconds  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Background Color Animation");  glutDisplayFunc(drawScene);  glutTimerFunc(0, update, 0);  gluOrtho2D(-1, 1, -1, 1);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop an animation that will call four objects separately, each after 20 ms. |
| **Code-**  #include <iostream>  #include <GL/glut.h>  float squareSize = 0.2;  float objectSpeed = 0.06;  struct Square {  float x;  float y;  bool isVisible;  };  Square square1 = {-0.5, 0.0, false};  Square square2 = {0.5, 0.0, false};  Square square3 = {0.0, 0.5, false};  Square square4 = {0.0, -0.5, false};  void drawSquare(float x, float y) {  glBegin(GL\_QUADS);  glVertex2f(x - squareSize, y - squareSize);  glVertex2f(x + squareSize, y - squareSize);  glVertex2f(x + squareSize, y + squareSize);  glVertex2f(x - squareSize, y + squareSize);  glEnd();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  if (square1.isVisible) {  glPushMatrix();  glTranslatef(square1.x, square1.y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  }  if (square2.isVisible) {  glPushMatrix();  glTranslatef(square2.x, square2.y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  }  if (square3.isVisible) {  glPushMatrix();  glTranslatef(square3.x, square3.y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  }  if (square4.isVisible) {  glPushMatrix();  glTranslatef(square4.x, square4.y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  }  glutSwapBuffers();  }  void update(int value) {  square1.isVisible = true;  glutPostRedisplay();  glutTimerFunc(80, update, 1);  glutTimerFunc(320, [](int value) {  square2.isVisible = true;  glutPostRedisplay();  }, 2);  glutTimerFunc(500, [](int value) {  square3.isVisible = true;  glutPostRedisplay();  }, 3);  glutTimerFunc(800, [](int value) {  square4.isVisible = true;  glutPostRedisplay();  }, 4);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Object Animation");  glutDisplayFunc(drawScene);  glutTimerFunc(0, update, 0); // Add a timer  gluOrtho2D(-1, 1, -1, 1);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  A red squares on a black background  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. |
| **Code-**  #include <iostream>  #include <GL/glut.h>  float squareSize = 0.2;  float objectSpeed = 0.01;  float square1X = -0.5;  float square1Y = 0.0;  float square2X = 0.5;  float square2Y = 0.0;  float square3X = 0.0;  float square3Y = 0.5;  float square4X = 0.0;  float square4Y = -0.5;  void drawSquare(float x, float y) {  glBegin(GL\_QUADS);  glVertex2f(x - squareSize, y - squareSize);  glVertex2f(x + squareSize, y - squareSize);  glVertex2f(x + squareSize, y + squareSize);  glVertex2f(x - squareSize, y + squareSize);  glEnd();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  // Move square 1 to the left  glPushMatrix();  glTranslatef(square1X, square1Y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  // Move square 2 to the right  glPushMatrix();  glTranslatef(square2X, square2Y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  // Move square 3 upwards  glPushMatrix();  glTranslatef(square3X, square3Y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  // Move square 4 downwards  glPushMatrix();  glTranslatef(square4X, square4Y, 0.0);  drawSquare(0.0, 0.0);  glPopMatrix();  glutSwapBuffers();  }  void update(int value) {  // Update positions for each square independently  square1X -= objectSpeed;  if (square1X < -1.0)  square1X = 1.0;  square2X += objectSpeed;  if (square2X > 1.0)  square2X = -1.0;  square3Y += objectSpeed;  if (square3Y > 1.0)  square3Y = -1.0;  square4Y -= objectSpeed;  if (square4Y < -1.0)  square4Y = 1.0;  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(16, update, 0); // Call update again in 16 milliseconds  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Moving Objects");  glutDisplayFunc(drawScene);  glutTimerFunc(0, update, 0); // Add a timer  gluOrtho2D(-1, 1, -1, 1);  glutMainLoop();  return 0;  } |
| A red cross with black background  Description automatically generated**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. |
| **Code-**  #include <iostream>  #include <GL/glut.h>  float squareSize = 0.2;  float objectSpeed = 0.05;  struct Square {  float x;  float y;  };  Square square1 = {-0.5, 0.0};  Square square2 = {0.5, 0.0};  Square square3 = {0.0, 0.5};  Square square4 = {0.0, -0.5};  void drawSquare(float x, float y) {  glBegin(GL\_QUADS);  glVertex2f(x - squareSize, y - squareSize);  glVertex2f(x + squareSize, y - squareSize);  glVertex2f(x + squareSize, y + squareSize);  glVertex2f(x - squareSize, y + squareSize);  glEnd();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity(); // Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  drawSquare(square1.x, square1.y);  drawSquare(square2.x, square2.y);  drawSquare(square3.x, square3.y);  drawSquare(square4.x, square4.y);  glutSwapBuffers();  }  void update(int value) {  glutPostRedisplay(); // Notify GLUT that the display has changed  glutTimerFunc(16, update, 0); // Call update again in 16 milliseconds  }  void handleKeyPress(unsigned char key, int x, int y) {  // Handle key presses for moving objects  switch (key) {  case 'a': // Move square1 to the left  square1.x -= objectSpeed;  if (square1.x < -1.0)  square1.x = 1.0;  break;  case 'd': // Move square2 to the right  square2.x += objectSpeed;  if (square2.x > 1.0)  square2.x = -1.0;  break;  case 'w': // Move square3 upwards  square3.y += objectSpeed;  if (square3.y > 1.0)  square3.y = -1.0;  break;  case 's': // Move square4 downwards  square4.y -= objectSpeed;  if (square4.y < -1.0)  square4.y = 1.0;  break;  }  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Moving Objects with Keys");  glutDisplayFunc(drawScene);  glutKeyboardFunc(handleKeyPress); // Register the keypress function  glutTimerFunc(0, update, 0); // Add a timer  gluOrtho2D(-1, 1, -1, 1);  glutMainLoop();  return 0;  } |
| **A red squares on a black background  Description automatically generatedOutput Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop a code that will have four different objects (keep it simple). Two of the objects will move to the right as the right click is made on the mouse and two of the objects will move to the left as the left key is pressed on the mouse. |
| **Code-**  #include <iostream>  #include <GL/glut.h>  float squareSize = 0.2;  float objectSpeed = 0.05;  struct Square {  float x;  float y;  };  Square square1 = {-0.5, 0.0};  Square square2 = {0.5, 0.0};  Square square3 = {0.0, 0.5};  Square square4 = {0.0, -0.5};  // Flags to indicate whether the right or left mouse button is pressed  bool rightButtonPressed = false;  bool leftButtonPressed = false;  void drawSquare(float x, float y) {  glBegin(GL\_QUADS);  glVertex2f(x - squareSize, y - squareSize);  glVertex2f(x + squareSize, y - squareSize);  glVertex2f(x + squareSize, y + squareSize);  glVertex2f(x - squareSize, y + squareSize);  glEnd();  }  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1, 0, 0);  glLoadIdentity();  glMatrixMode(GL\_MODELVIEW);  drawSquare(square1.x, square1.y);  drawSquare(square2.x, square2.y);  drawSquare(square3.x, square3.y);  drawSquare(square4.x, square4.y);  glutSwapBuffers();  }  void update(int value) {  if (rightButtonPressed) {  square1.x += objectSpeed;  square2.x += objectSpeed;  }  if (leftButtonPressed) {  square3.x -= objectSpeed;  square4.x -= objectSpeed;  }  glutPostRedisplay();  glutTimerFunc(16, update, 0);  }  // Callback function for mouse button presses  void handleMouse(int button, int state, int x, int y) {  if (button == GLUT\_LEFT\_BUTTON) {  if (state == GLUT\_DOWN) {  leftButtonPressed = true;  } else if (state == GLUT\_UP) {  leftButtonPressed = false;  }  } else if (button == GLUT\_RIGHT\_BUTTON) {  if (state == GLUT\_DOWN) {  rightButtonPressed = true;  } else if (state == GLUT\_UP) {  rightButtonPressed = false;  }  }  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Moving Objects with Mouse Clicks");  glutDisplayFunc(drawScene);  glutMouseFunc(handleMouse); // Register the mouse callback function  glutTimerFunc(0, update, 0);  gluOrtho2D(-1, 1, -1, 1);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A group of red squares  Description automatically generated** |