**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/gl.h>  #include <GL/glut.h>  float r = 0;  float g = 0;  float b = 0;  void window(){  glColor3f(0.5, 0.3, 0);  glBegin(GL\_POLYGON);  //upper black portion  glBegin(GL\_POLYGON);  glVertex2f(-95, 60);  glVertex2f(-80, 52);  glVertex2f(70, 52);  glVertex2f(82, 60);  glEnd();  //left window  glBegin(GL\_POLYGON);  glVertex2f(-80, 52);  glVertex2f(-120, 75);  glVertex2f(-125,70);  glVertex2f(-125,-75);  glVertex2f(-120,-80);  glVertex2f(-80,-68);  glVertex2f(-80,-68);  glEnd();  //under black portion  glBegin(GL\_POLYGON);  glVertex2f(-80,-68);  glVertex2f(-80,-73);  glVertex2f(70,-73);  glVertex2f(70,-68);  glEnd();  //right window  glBegin(GL\_POLYGON);  glVertex2f(70,-68);  glVertex2f(110,-75);  glVertex2f(118,-72);  glVertex2f(118,74);  glVertex2f(110,80);  glVertex2f(70,52);  glEnd();  //left window part 2 square like shape  glColor3f(1, 1, 1);  glBegin(GL\_POLYGON);  glVertex2f(-85, 45);  glVertex2f(-115, 60);  glVertex2f(-115, 5);  glVertex2f(-85, 1);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-85, -10);  glVertex2f(-115, -5);  glVertex2f(-115, -70);  glVertex2f(-85, -60);  glEnd();  //Right window 2 square like shape  glBegin(GL\_POLYGON);  glVertex2f(76, 48);  glVertex2f(102, 66);  glVertex2f(101.5, 5.5);  glVertex2f(76, 0);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(76, -10);  glVertex2f(76, -60);  glVertex2f(102, -64);  glVertex2f(102, -6);  glEnd();  //plus sign in the middle of the window  glColor3f(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(-80,0);  glVertex2f(-80,-5);  glVertex2f(70,-5);  glVertex2f(70,0);  glEnd();  glColor3f(0,0,0);  glBegin(GL\_POLYGON);  glVertex2f(-10,52);  glVertex2f(-10,-68);  glVertex2f(-5,-68);  glVertex2f(-5,52);  glEnd();  // under window extension  glColor3f(0.5, 0.3, 0);  glBegin(GL\_POLYGON);  glVertex2f(-80,-73);  glVertex2f(-60,-90);  glVertex2f(50,-90);  glVertex2f(70,-73);  glEnd();  }  void window\_background(){  //window background  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glColor3f(r, g, b);  glBegin(GL\_POLYGON);  glVertex2f(-80,52);  glVertex2f(-80,-68);  glVertex2f(70,-68);  glVertex2f(70,52);  glEnd();  glPopMatrix();  }  void update\_color(int value){  r += 0.01;  g += 0.01;  if (r >= 1){  r = 0;  }  if (g >= 1){  g = 0.5;  }  //b += 0.01;  glutPostRedisplay();  glutTimerFunc(20, update\_color, 0);  }  void display() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer with current clearing color  window\_background();  window();  //BOX1();  //BOX2();  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInitWindowSize(720, 620); // Set the window's initial width & height  glutInit(&argc, argv);  glutCreateWindow("Rotating Wheel"); // Create window with the given title  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  gluOrtho2D(-140,+140,-100,+100);  //glutTimerFunc(20, box1move, 0);  //glutTimerFunc(20, box2move, 0);  glutTimerFunc(20, update\_color, 0);  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Develop an animation that will call four objects separately, each after 20 ms. |
| **Graph-** |
| **Code-**  #include <iostream>  #include <GL/gl.h>  #include <GL/glut.h>  using namespace std;  float \_move = 0.0f;  float \_move1 = -7.0f;  int state = 1;  void box1() {  glBegin(GL\_QUADS);  glVertex2f(-0.8f, 1.0f);  glVertex2f(-0.8f, 0.8f);  glVertex2f(-0.6f, 0.8f);  glVertex2f(-0.6f, 1.0f);  glEnd();  }  void box2() {  glBegin(GL\_QUADS);  glVertex2f(-0.8f, -0.8f);  glVertex2f(-0.8f, -1.0f);  glVertex2f(-0.6f, -1.0f);  glVertex2f(-0.6f, -0.8f);  glEnd();  }  void box3() {  glBegin(GL\_QUADS);  glVertex2f(0.6f, 1.0f);  glVertex2f(0.6f, 0.8f);  glVertex2f(0.8f,0.8f);  glVertex2f(0.8f, 1.0f);  glEnd();  }  void box4() {  glBegin(GL\_QUADS);  glVertex2f(0.6f, -0.8f);  glVertex2f(0.6f, -1.0f);  glVertex2f(0.8f, -1.0f);  glVertex2f(0.8f, -0.8f);  glEnd();  }  void display() {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  switch(state) {  case 1:  box1();  break;  case 2:  box2();  break;  case 3:  box3();  break;  case 4:  box4();  break;  }  glFlush();  }  void call(int value) {  state++;  glutPostRedisplay();  if (state < 5) {  glutTimerFunc(1000, call, 0); // Adjust the delay as needed (here 1000 milliseconds)  }  else{  state = 1;  glutTimerFunc(1000, call, 0);  }  //cout << state << endl;  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitWindowSize(720, 620);  glutCreateWindow("Rotating Wheel");  glutDisplayFunc(display);  glutTimerFunc(1000, call, 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>  using namespace std;  float \_move = 0.0f;  float \_move1 = 0.0f;  float \_move3 = 0.0f;  float \_move4= 0.0f;  int state1 = 1;  int state2 = 2;  int state3 = 3;  int state4 = 4;  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1,0,0);  //glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(0.0f,\_move, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(0.6f, 1.0f);  glVertex2f(0.6f, 0.8f);  glVertex2f(0.8f, 0.8f);  glVertex2f(0.8f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(0.0f,\_move1, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(-0.8f, -0.8f);  glVertex2f(-0.8f, -1.0f);  glVertex2f(-0.6f, -1.0f);  glVertex2f(-0.6f, -0.8f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move3, 0.0f, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(-0.8f, 1.0f);  glVertex2f(-0.8f, 0.8f);  glVertex2f(-0.6f, 0.8f);  glVertex2f(-0.6f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move4, 0.0f, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(0.6f, -0.8f);  glVertex2f(0.6f, -1.0f);  glVertex2f(0.8f, -1.0f);  glVertex2f(0.8f, -0.8f);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  //for object 1  void update(int value) {  switch(state1){  case 1:  if (\_move < 0.0){  \_move += 0.03;  }  else{  state1 = -1;  }  break;  case -1:  if (\_move > -1.8 ){  \_move -= 0.03;  }  else {  state1 = 1;  }  break;  }  /\* \_move -= .02;  if(\_move < -2)  {  \_move = 0.0f;  }\*/  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  //for animation bottom to up  void update1(int value) {  switch(state2){  case 2:  if (\_move1 > 0.0){  \_move1 -= 0.03;  }  else{  state2 = -2;  }  break;  case -2:  if (\_move1 < 1.8 ){  \_move1 += 0.03;  }  else {  state2 = 2;  }  break;  }  /\*\_move1 += .02;  if(\_move1 > 2)  {  \_move1 = 0.0f;  }\*/  glutPostRedisplay();  glutTimerFunc(20, update1, 0);  }  void update3(int value) {  switch(state3){  case 3:  if (\_move3 < 1.6 ){  \_move3 += 0.03;  }  else {  state3 = -3;  }  break;  case -3:  if (\_move3 > -0.2){  \_move3 -= 0.03;  }  else{  state3 = 3;  }  break;  }  glutPostRedisplay();  glutTimerFunc(20, update3, 0);  }  void update4(int value) {  switch(state4){  case 4:  if (\_move4 > -1.6 ){  \_move4 -= 0.03;  }  else {  state4 = -4;  }  break;  case -4:  if (\_move4 < 0.2){  \_move4 += 0.03;  }  else{  state4 = 4;  }  break;  }  glutPostRedisplay();  glutTimerFunc(20, update4, 0);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Transformation");  //gluOrtho2D(-2,2,-2,2);  glutDisplayFunc(drawScene);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 0); //Add a timer  glutTimerFunc(20, update3, 0); //Add a timer  glutTimerFunc(20, update4, 0); //Add a timer  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>  using namespace std;  float \_move = 0.0f;  float \_move1 = 0.0f;  float \_move3 = 0.0f;  float \_move4= 0.0f;  int state1 = 1;  int state2 = 2;  int state3 = 3;  int state4 = 4;  bool a = false;  bool b = false;  bool c = false;  bool d = false;  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1,0,0);  //glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(0.0f,\_move, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(0.6f, 1.0f);  glVertex2f(0.6f, 0.8f);  glVertex2f(0.8f, 0.8f);  glVertex2f(0.8f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(0.0f,\_move1, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(-0.8f, -0.8f);  glVertex2f(-0.8f, -1.0f);  glVertex2f(-0.6f, -1.0f);  glVertex2f(-0.6f, -0.8f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move3, 0.0f, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(-0.8f, 1.0f);  glVertex2f(-0.8f, 0.8f);  glVertex2f(-0.6f, 0.8f);  glVertex2f(-0.6f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move4, 0.0f, 0.0f); // for bottom to up movement  glBegin(GL\_QUADS);  glVertex2f(0.6f, -0.8f);  glVertex2f(0.6f, -1.0f);  glVertex2f(0.8f, -1.0f);  glVertex2f(0.8f, -0.8f);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  //for object 1  void update(int value) {  if (a){  switch(state1){  case 1:  if (\_move < 0.0){  \_move += 0.03;  }  else{  state1 = -1;  }  break;  case -1:  if (\_move > -1.8 ){  \_move -= 0.03;  }  else {  state1 = 1;  }  break;  }  /\* \_move -= .02;  if(\_move < -2)  {  \_move = 0.0f;  }\*/  glutPostRedisplay();  }  glutTimerFunc(20, update, 0);  }  //for animation bottom to up  void update1(int value) {  if (b){  switch(state2){  case 2:  if (\_move1 > 0.0){  \_move1 -= 0.03;  }  else{  state2 = -2;  }  break;  case -2:  if (\_move1 < 1.8 ){  \_move1 += 0.03;  }  else {  state2 = 2;  }  break;  }  /\*\_move1 += .02;  if(\_move1 > 2)  {  \_move1 = 0.0f;  }\*/  glutPostRedisplay();  }  glutTimerFunc(20, update1, 0);  }  void update3(int value) {  if (c){  switch(state3){  case 3:  if (\_move3 < 1.6 ){  \_move3 += 0.03;  }  else {  state3 = -3;  }  break;  case -3:  if (\_move3 > -0.2){  \_move3 -= 0.03;  }  else{  state3 = 3;  }  break;  }  glutPostRedisplay();  }  glutTimerFunc(20, update3, 0);  }  void update4(int value) {  if (d){  switch(state4){  case 4:  if (\_move4 > -1.6 ){  \_move4 -= 0.03;  }  else {  state4 = -4;  }  break;  case -4:  if (\_move4 < 0.2){  \_move4 += 0.03;  }  else{  state4 = 4;  }  break;  }  glutPostRedisplay();  }  glutTimerFunc(20, update4, 0);  }  void handleKeypress(unsigned char key, int x, int y) {  switch (key) {  case 'i':  a = !a;  cout << "I pressed" << endl;  break;  case 'j':  b = !b;  cout <<" j pressed" << endl;  break;  case 'k':  c = !c;  cout << "k pressed" << endl;  break;  case 'l':  d = !d;  cout << "d pressed" << endl;  break;  }  glutPostRedisplay();  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Transformation");  //gluOrtho2D(-2,2,-2,2);  glutDisplayFunc(drawScene);  glutKeyboardFunc(handleKeypress);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 0); //Add a timer  glutTimerFunc(20, update3, 0); //Add a timer  glutTimerFunc(20, update4, 0); //Add a timer  glutMainLoop();  return 0;  } |
| **Output 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. |
| **Graph-** |
| **Code-**  #include <iostream>  #include<GL/gl.h>  #include <GL/glut.h>  using namespace std;  float \_move = 0.0f;  float \_move1 = 0.0f;  float \_move3 = 0.0f;  float \_move4= 0.0f;  int state1 = 1;  int state2 = 2;  int state3 = 3;  int state4 = 4;  bool a = false;  bool b = false;  bool c = false;  bool d = false;  void drawScene() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3d(1,0,0);  //glLoadIdentity(); //Reset the drawing perspective  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(\_move,0.0f, 0.0f); // for bottom to up movement //3  glBegin(GL\_QUADS);  glVertex2f(0.6f, 1.0f);  glVertex2f(0.6f, 0.8f);  glVertex2f(0.8f, 0.8f);  glVertex2f(0.8f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move1,0.0f, 0.0f); // for bottom to up movement // 2  glBegin(GL\_QUADS);  glVertex2f(-0.8f, -0.8f);  glVertex2f(-0.8f, -1.0f);  glVertex2f(-0.6f, -1.0f);  glVertex2f(-0.6f, -0.8f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move1, 0.0f, 0.0f); // for bottom to up movement //1  glBegin(GL\_QUADS);  glVertex2f(-0.8f, 1.0f);  glVertex2f(-0.8f, 0.8f);  glVertex2f(-0.6f, 0.8f);  glVertex2f(-0.6f, 1.0f);  glEnd();  glPopMatrix();  glPushMatrix();  glTranslatef(\_move, 0.0f, 0.0f); // for bottom to up movement //4  glBegin(GL\_QUADS);  glVertex2f(0.6f, -0.8f);  glVertex2f(0.6f, -1.0f);  glVertex2f(0.8f, -1.0f);  glVertex2f(0.8f, -0.8f);  glEnd();  glPopMatrix();  glutSwapBuffers();  }  //for object 1  void update(int value) {  if (a){  switch(state1){  case 1:  if (\_move > -1.6){  \_move -= 0.03;  }  else{  state1 = -1;  }  break;  case -1:  if (\_move < 0.2 ){  \_move += 0.03;  }  else {  state1 = 1;  }  break;  }  /\* \_move -= .02;  if(\_move < -2)  {  \_move = 0.0f;  }\*/  glutPostRedisplay();  }  glutTimerFunc(20, update, 0);  }  //for animation bottom to up  void update1(int value) {  if (b){  switch(state2){  case 2:  if (\_move1 < 1.6){  \_move1 += 0.03;  }  else{  state2 = -2;  }  break;  case -2:  if (\_move1 > -0.2 ){  \_move1 -= 0.03;  }  else {  state2 = 2;  }  break;  }  /\*\_move1 += .02;  if(\_move1 > 2)  {  \_move1 = 0.0f;  }\*/  glutPostRedisplay();  }  glutTimerFunc(20, update1, 0);  }  void handleMouse(int button, int state, int x, int y) {  if (button == GLUT\_LEFT\_BUTTON)  { a = !a;  cout << "left button clicked" << endl;  }  if (button == GLUT\_RIGHT\_BUTTON)  {b = !b;  couit << "right button clicked" << endl;  }  glutPostRedisplay();}  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 800);  glutCreateWindow("Transformation");  //gluOrtho2D(-2,2,-2,2);  glutDisplayFunc(drawScene);  glutMouseFunc(handleMouse);  glutTimerFunc(20, update, 0); //Add a timer  glutTimerFunc(20, update1, 0); //Add a timer  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |