**Lab Taks-5**

Submission Guidelines-

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

|  |
| --- |
| **Question-1**  Create an animation using two box that will move in the opposite direction. |
| **Graph Plot (Picture)-** |
| **Code-**  #include <GL/glut.h>  float box1\_x = -0.9f;  float box2\_x = 0.9f;  float speed = 0.01f;  void drawBox(float x, float y, float width, float height, float r, float g, float b) {  glColor3f(r, g, b);  glBegin(GL\_QUADS);  glVertex2f(x, y);  glVertex2f(x + width, y);  glVertex2f(x + width, y + height);  glVertex2f(x, y + height);  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  drawBox(box1\_x, 0.2f, 0.2f, 0.2f, 1.0f, 0.0f, 0.0f);  drawBox(box2\_x - 0.2f, -0.4f, 0.2f, 0.2f, 0.0f, 0.0f, 1.0f);  glutSwapBuffers();  }  void update(int value) {  box1\_x += speed;  if (box1\_x > 1.0f) box1\_x = -1.0f;  box2\_x -= speed;  if (box2\_x < -1.0f) box2\_x = 1.0f;  glutPostRedisplay();  glutTimerFunc(16, update, 0);  }  void init() {  glClearColor(0.0, 0.0, 0.0, 1.0);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(600, 400);  glutCreateWindow("Two Moving Boxes");  init();  glutDisplayFunc(display);  glutTimerFunc(0, update, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-2**  Design a car which will have rotating wheels. |
| **Graph Plot (Picture)-** |
| **Code:**  #include <GL/glut.h>  #include <math.h>  float angle = 0.0f;  void drawWheel(float cx, float cy, float radius) {  glPushMatrix();  glTranslatef(cx, cy, 0);  glRotatef(angle, 0, 0, 1);  glColor3f(0.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; ++i) {  float theta = i \* 3.14159f / 180.0f;  glVertex2f(radius \* cos(theta), radius \* sin(theta));  }  glEnd();  glColor3f(1.0f, 1.0f, 1.0f);  glLineWidth(2.0f);  glBegin(GL\_LINES);  glVertex2f(0.0f, 0.0f); glVertex2f(radius, 0.0f);  glVertex2f(0.0f, 0.0f); glVertex2f(0.0f, radius);  glEnd();  glPopMatrix();  }  void drawCar() {  float car\_x = -0.3f;  glColor3f(1.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  glVertex2f(car\_x, 0.1f);  glVertex2f(car\_x + 0.6f, 0.1f);  glVertex2f(car\_x + 0.6f, 0.3f);  glVertex2f(car\_x, 0.3f);  glEnd();  glColor3f(0.8f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  glVertex2f(car\_x + 0.1f, 0.3f);  glVertex2f(car\_x + 0.5f, 0.3f);  glVertex2f(car\_x + 0.4f, 0.4f);  glVertex2f(car\_x + 0.2f, 0.4f);  glEnd();  drawWheel(car\_x + 0.15f, 0.1f, 0.05f);  drawWheel(car\_x + 0.45f, 0.1f, 0.05f);  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  drawCar();  glutSwapBuffers();  }  void update(int value) {  angle -= 5.0f;  if (angle <= -360.0f) angle = 0;  glutPostRedisplay();  glutTimerFunc(16, update, 0);  }  void init() {  glClearColor(0.7f, 0.9f, 1.0f, 1.0f);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 600);  glutCreateWindow("Car with Rotating Wheels");  init();  glutDisplayFunc(display);  glutTimerFunc(0, update, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-3**  Now move your car of question-2 from left to right in a loop. |
| **Graph Plot (Picture)-** |
| **Code**:  #include <GL/glut.h>  #include <math.h>  float car\_x = -1.0f;  float angle = 0.0f;  float speed = 0.01f;  void drawWheel(float cx, float cy, float radius) {  glPushMatrix();  glTranslatef(cx, cy, 0);  glRotatef(angle, 0, 0, 1);  glColor3f(0.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; ++i) {  float theta = i \* 3.14159f / 180.0f;  glVertex2f(radius \* cos(theta), radius \* sin(theta));  }  glEnd();  glColor3f(1.0f, 1.0f, 1.0f);  glLineWidth(2.0f);  glBegin(GL\_LINES);  glVertex2f(0.0f, 0.0f); glVertex2f(radius, 0.0f);  glVertex2f(0.0f, 0.0f); glVertex2f(0.0f, radius);  glEnd();  glPopMatrix();  }  void drawCar() {  glColor3f(1.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  glVertex2f(car\_x, 0.1f);  glVertex2f(car\_x + 0.6f, 0.1f);  glVertex2f(car\_x + 0.6f, 0.3f);  glVertex2f(car\_x, 0.3f);  glEnd();  glColor3f(0.8f, 0.0f, 0.0f);  glBegin(GL\_POLYGON);  glVertex2f(car\_x + 0.1f, 0.3f);  glVertex2f(car\_x + 0.5f, 0.3f);  glVertex2f(car\_x + 0.4f, 0.4f);  glVertex2f(car\_x + 0.2f, 0.4f);  glEnd();  drawWheel(car\_x + 0.15f, 0.1f, 0.05f);  drawWheel(car\_x + 0.45f, 0.1f, 0.05f);  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  drawCar();  glutSwapBuffers();  }  void update(int value) {  car\_x += speed;  if (car\_x > 1.0f) car\_x = -0.6f;  angle -= 5.0f;  if (angle <= -360.0f) angle = 0;  glutPostRedisplay();  glutTimerFunc(16, update, 0);  }  void init() {  glClearColor(0.7f, 0.9f, 1.0f, 1.0f);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 600);  glutCreateWindow("Moving Car");  init();  glutDisplayFunc(display);  glutTimerFunc(0, update, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-4**  Design a windmill with rotating blades |
| **Graph Plot (Picture)-** |
| **Code-**  #include <GL/glut.h>  #include <math.h>  float bladeAngle = 0.0f;  void drawTower() {  glColor3f(0.4f, 0.2f, 0.0f);  glBegin(GL\_POLYGON);  glVertex2f(-0.05f, -0.5f);  glVertex2f(0.05f, -0.5f);  glVertex2f(0.05f, 0.2f);  glVertex2f(-0.05f, 0.2f);  glEnd();  }  void drawBlade() {  glBegin(GL\_POLYGON);  glColor3f(0.9f, 0.9f, 0.9f);  glVertex2f(0.0f, 0.0f);  glVertex2f(0.02f, 0.4f);  glVertex2f(-0.02f, 0.4f);  glEnd();  }  void drawBlades() {  glPushMatrix();  glTranslatef(0.0f, 0.2f, 0.0f);  glRotatef(bladeAngle, 0.0f, 0.0f, 1.0f);  for (int i = 0; i < 4; ++i) {  drawBlade();  glRotatef(90.0f, 0.0f, 0.0f, 1.0f);  }  glColor3f(0.3f, 0.3f, 0.3f);  glBegin(GL\_POLYGON);  for (int i = 0; i < 360; ++i) {  float theta = i \* 3.14159f / 180.0f;  glVertex2f(0.03f \* cos(theta), 0.03f \* sin(theta));  }  glEnd();  glPopMatrix();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  drawTower();  drawBlades();  glutSwapBuffers();  }  void update(int value) {  bladeAngle += 2.0f;  if (bladeAngle >= 360.0f) bladeAngle = 0.0f;  glutPostRedisplay();  glutTimerFunc(16, update, 0);  }  void init() {  glClearColor(0.6f, 0.9f, 1.0f, 1.0f);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(800, 600);  glutCreateWindow("Windmill");  init();  glutDisplayFunc(display);  glutTimerFunc(0, update, 0);  glutMainLoop();  return 0;  } |
| **Output Screenshot (Full Screen)-** |