```
#include <iostream>
#include <GL/glut.h>
#include <windows.h>
#include <mmsystem.h>
#include <vector>
#include <ctime>
#include <cstdlib>
#define PI 3.1416



float playerX = -18.0f;
float playerY = -12.0f;


float bulletX = 1.0f;
float bulletY = 1.0f;


bool isBulletActive = false;
bool isPlayerMoving = true;


int score = 0;
int level = 1;
bool levelIncreased = false;



const int numEnemies = 5;  // Adjust the number of enemies as needed
std::vector<float> enemyX(numEnemies, 0.0f);
std::vector<float> enemyY(numEnemies, 0.0f);


void renderBitmapString(float x, float y, float z, void* font, char* string) {
  glColor3f(1.0, 1.0, 1.0);
  char* c;
  glRasterPos3f(x, y, z);
  for (c = string; *c != '\0'; c++) {
    glutBitmapCharacter(font, *c);
  }
}
```

```
void checkCollision() {
   for (int i = 0; i < numEnemies; ++i) {
      if (isBulletActive && bulletX >= enemyX[i] - 1.0f && bulletX <= enemyX[i] + 1.0f && bulletY >=
enemyY[i] - 4.0f && bulletY <= enemyY[i] + 1.0f) {
         std::cout << "Collision detected!" << std::endl;
         score += 1;
         std::cout << "Score: " << score << std::endl;
         isBulletActive = false;


          // Regenerate enemy at initial position
         enemyX[i] = 20.0f;
         enemyY[i] = -12.0f + static_cast<float>(rand() % 300) / 100.0f;  // Randomize Y position
      }
   }


   // Increasing level from 1 to 2
    if (score > 20){
      level = 2;



   }


   if (score > 30){
      level = 3;
   }
}


void drawEnemy() {
   for (int i = 0; i < numEnemies; ++i) {
      if (enemyX[i] > -20.0f) {
         glBegin(GL_POLYGON);
         glColor3f(0.f, 0.0f, 1.0f);
         glVertex2f(enemyX[i] + 1.0f, enemyY[i] - 4.0f);
         glVertex2f(enemyX[i] - 1.0f, enemyY[i] - 4.0f);
         glVertex2f(enemyX[i] - 1.0f, enemyY[i] + 1.0f);
         glVertex2f(enemyX[i] + 1.0f, enemyY[i] + 1.0f);
         glEnd();
      }
```

```
    }
}


void drawBullet() {
    if (isBulletActive) {
        glColor3f(1.0f, 1.0f, 0.0f);
        glBegin(GL_QUADS);
        glVertex2f(bulletX - 0.2f, bulletY - 0.2f);
        glVertex2f(bulletX + 0.2f, bulletY - 0.2f);
        glVertex2f(bulletX + 0.2f, bulletY + 0.2f);
        glVertex2f(bulletX - 0.2f, bulletY + 0.2f);
        glEnd();
    }
}


void drawPlayer() {
    glColor3f(1.0f, 0.5f, 0.0f);
    glBegin(GL_QUADS);
    glVertex2f(playerX - 1.0f, playerY - 5.0f);
    glVertex2f(playerX + 1.0f, playerY - 5.0f);
    glVertex2f(playerX + 1.0f, playerY + 3.0f);
    glVertex2f(playerX - 1.0f, playerY + 3.0f);
    glEnd();
}


void background() {
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(0.5);


    //background
    glBegin(GL_POLYGON);
    glColor3ub(184, 145, 150);
    glVertex2f(-20.0f,20.0f);
    glVertex2f(20.0f,20.0f);
    glVertex2f(20.0f,-20.0f);
    glVertex2f(-20.0f,-20.0f);
    glEnd();
```

```
//door 1
glBegin(GL_POLYGON);
glColor3f(0.1f, 0.1f, 0.1f);
glVertex2f(-15.0f,14.0f);
glVertex2f(-9.0f,14.0f);
glVertex2f(-9.0f,-10.0f);
glVertex2f(-15.0f,-10.0f);
glEnd();


//door 1 glass
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(-11.0f,11.0f);
glVertex2f(-9.0f,11.0f);
glVertex2f(-9.0f,5.0f);
glVertex2f(-11.0f,5.0f);
glEnd();


//door 2
glBegin(GL_POLYGON);
glColor3f(0.1f, 0.1f, 0.1f);
glVertex2f(10.0f,14.0f);
glVertex2f(16.0f,14.0f);
glVertex2f(16.0f,-10.0f);
glVertex2f(10.0f,-10.0f);
glEnd();
//door2 glass
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(14.0f,11.0f);
glVertex2f(16.0f,11.0f);
glVertex2f(16.0f,5.0f);
glVertex2f(14.0f,5.0f);
glEnd();



//fingerprint1
glBegin(GL_POLYGON);
glColor3ub(37, 48, 47);
glVertex2f(-8.0f,5.0f);
```

```
glVertex2f(-7.0f,5.0f);
glVertex2f(-7.0f,2.0f);
glVertex2f(-8.0f,2.0f);
glEnd();


  //fingerprint1  display
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(-7.75f,4.50f);
glVertex2f(-7.25f,4.50f);
glVertex2f(-7.25f,4.25f);
glVertex2f(-7.75f,4.25f);
glEnd();


//finger print1 position



glBegin(GL_POLYGON);
glColor3ub(9, 230, 212);
glVertex2f(-7.62f,3.00f);
glVertex2f(-7.33f,3.00f);
glVertex2f(-7.33f,2.50f);
glVertex2f(-7.62f,2.50f);
glEnd();


  //fingerprint 2
glBegin(GL_POLYGON);
glColor3ub(37, 48, 47);
glVertex2f(18.0f,5.0f);
glVertex2f(19.0f,5.0f);
glVertex2f(19.0f,2.0f);
glVertex2f(18.0f,2.0f);
glEnd();
//finer print 2 display
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(18.25f,4.50f);
glVertex2f(18.75f,4.50f);
glVertex2f(18.75f,4.25f);
```

```
glVertex2f(18.25f,4.25f);
glEnd();
//finger print2 position



glBegin(GL_POLYGON);
glColor3ub(9, 230, 212);
glVertex2f(18.33f,3.00f);
glVertex2f(18.62f,3.00f);
glVertex2f(18.62f,2.50f);
glVertex2f(18.33f,2.50f);
glEnd();
//noticeboard
glBegin(GL_POLYGON);
glColor3ub(4, 89, 81);
glVertex2f(-3.0f,12.0f);
glVertex2f(5.0f,12.0f);
glVertex2f(5.0f,4.0f);
glVertex2f(-3.0f,4.0f);
glEnd();


//dustbin trinagle
glBegin(GL_POLYGON);
glColor3f(0.0f, 1.0f, 0.0f);
glVertex2f(-3.0f,0.0f);
glVertex2f(-2.0f,1.0f);
glVertex2f(-1.0f,1.0f);
glVertex2f(0.0f,0.0f);
glEnd();


//dustbin big rectangle
glBegin(GL_POLYGON);
glColor3ub(33, 156, 61);
glVertex2f(-3.0f,0.0f);
glVertex2f(0.0f,0.0f);
glVertex2f(0.0f,-10.0f);
glVertex2f(-3.0f,-10.0f);
glEnd();
```

```
  //dustbin midlle rectangle


  glBegin(GL_POLYGON);
  glColor3ub(256, 256, 256);
  glVertex2f(-2.0f,-2.0f);
  glVertex2f(-1.0f,-2.0f);
  glVertex2f(-1.0f,-3.0f);
  glVertex2f(-2.0f,-3.0f);
  glEnd();



  //floor
  glBegin(GL_POLYGON);
  glColor3ub(47, 62, 66);
  glVertex2f(-20.0f,-10.0f);
  glVertex2f(20.0f,-10.0f);
  glVertex2f(20.0f,-20.0f);
  glVertex2f(-20.0f,-20.0f);
  glEnd();
}


void gunShotSound() {
  sndPlaySound("gun_shoot_sound.wav", SND_ASYNC);
}



// Level 2 background code
void background2(){


  glClear(GL_COLOR_BUFFER_BIT);
  glLineWidth(0.5);


  glBegin(GL_POLYGON);
  glColor3ub(184, 145, 150);
  glVertex2f(-20.0f,20.0f);
  glVertex2f(20.0f,20.0f);
  glVertex2f(20.0f,-20.0f);
```

```
glVertex2f(-20.0f,-20.0f);
glEnd();
//door 1
glBegin(GL_POLYGON);
glColor3f(0.1f, 0.1f, 0.1f);
glVertex2f(-15.0f,14.0f);
glVertex2f(-9.0f,14.0f);
glVertex2f(-9.0f,-10.0f);
glVertex2f(-15.0f,-10.0f);
glEnd();
//door 1 glass
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(-11.0f,11.0f);
glVertex2f(-9.0f,11.0f);
glVertex2f(-9.0f,5.0f);
glVertex2f(-11.0f,5.0f);
glEnd();


//fingerprint
glBegin(GL_POLYGON);
glColor3ub(37, 48, 47);
glVertex2f(-8.0f,5.0f);
glVertex2f(-7.0f,5.0f);
glVertex2f(-7.0f,2.0f);
glVertex2f(-8.0f,2.0f);
glEnd();


//fingerprint1  display
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(-7.75f,4.50f);
glVertex2f(-7.25f,4.50f);
glVertex2f(-7.25f,4.25f);
glVertex2f(-7.75f,4.25f);
glEnd();


//finger print1 position
glBegin(GL_POLYGON);
glColor3ub(9, 230, 212);
glVertex2f(-7.62f,3.00f);
```

```
glVertex2f(-7.33f,3.00f);
glVertex2f(-7.33f,2.50f);
glVertex2f(-7.62f,2.50f);
glEnd();


//bench big back
 glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(5.0f,-7.0f);
glVertex2f(4.0f,-2.0f);
glVertex2f(18.0f,-2.0f);
glVertex2f(17.0f,-7.0f);
glEnd();


//bench midle rect
glBegin(GL_POLYGON);
glColor3ub(60, 113, 166);
glVertex2f(4.0f,-7.0f);
glVertex2f(18.0f,-7.0f);
glVertex2f(18.0f,-8.0f);
glVertex2f(4.0f,-8.0f);
glEnd();


//rect bench
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(4.00f,-7.00f);
glVertex2f(18.00f,-7.00f);
glVertex2f(18.00f,-8.00f);
glVertex2f(4.00f,-8.00f);
glEnd();



//1legbench
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(4.00f,-8.00f);
glVertex2f(5.00f,-8.00f);
glVertex2f(6.00f,-11.00f);
```

```
glVertex2f(5.00f,-11.00f);
glEnd();


//2leg
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(6.00f,-8.00f);
glVertex2f(6.75f,-8.00f);
glVertex2f(7.25f,-11.00f);
glVertex2f(6.50f,-11.00f);
glEnd();


//3rdleg
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(15.25f,-8.00f);
glVertex2f(16.00f,-8.00f);
glVertex2f(15.5f,-11.00f);
glVertex2f(14.75f,-11.00f);
glEnd();


//4th leg
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(17.0f,-8.00f);
glVertex2f(18.00f,-8.00f);
glVertex2f(17.25f,-11.00f);
glVertex2f(16.25f,-11.00f);
glEnd();


//fire box
glBegin(GL_POLYGON);
glColor3ub(219, 15, 36);
glVertex2f(9.0f,6.00f);
glVertex2f(12.00f,6.00f);
glVertex2f(12.00f,3.00f);
glVertex2f(9.00f,3.00f);
glEnd();
```

```
//fireboxsmall
glBegin(GL_POLYGON);
glColor3ub(47, 62, 66);
glVertex2f(10.0f,5.00f);
glVertex2f(11.00f,5.00f);
glVertex2f(11.00f,4.00f);
glVertex2f(10.00f,4.00f);
glEnd();


glBegin(GL_LINES);
glColor3ub(23, 21, 21);
glVertex2f(4.50f,-4.00f);
glVertex2f(17.50f,-4.00f);
glEnd();


//line
glBegin(GL_LINES);
glColor3ub(23, 21, 21);
glVertex2f(4.75f,-5.50f);
glVertex2f(17.25f,-5.50f);
glEnd();


//noticeboard
glBegin(GL_POLYGON);
glColor3ub(4, 89, 81);
glVertex2f(-3.0f,12.0f);
glVertex2f(5.0f,12.0f);
glVertex2f(5.0f,4.0f);
glVertex2f(-3.0f,4.0f);
glEnd();


//floor
glBegin(GL_POLYGON);
glColor3ub(47, 62, 66);
glVertex2f(-20.0f,-10.0f);
glVertex2f(20.0f,-10.0f);
glVertex2f(20.0f,-20.0f);
glVertex2f(-20.0f,-20.0f);
glEnd();
```

```
}


// Creating level 3 background
void background3() {
  glClear(GL_COLOR_BUFFER_BIT);
  glLineWidth(0.5);



  glBegin(GL_POLYGON);
  glColor3ub(242, 234, 198);
  glVertex2f(-20.0f,20.0f);
  glVertex2f(20.0f,20.0f);
  glVertex2f(20.0f,-20.0f);
  glVertex2f(-20.0f,-20.0f);
  glEnd();
  //door 1
  glBegin(GL_POLYGON);
  glColor3ub(145, 87, 65);
  glVertex2f(-15.0f,14.0f);
  glVertex2f(-9.0f,14.0f);
  glVertex2f(-9.0f,-10.0f);
  glVertex2f(-15.0f,-10.0f);
  glEnd();
  //door 1 glass
  glBegin(GL_POLYGON);
  glColor3ub(158, 216, 236);
  glVertex2f(-11.0f,11.0f);
  glVertex2f(-9.0f,11.0f);
  glVertex2f(-9.0f,5.0f);
  glVertex2f(-11.0f,5.0f);
  glEnd();


  //fingerprint
  glBegin(GL_POLYGON);
  glColor3ub(37, 48, 47);
  glVertex2f(-8.0f,5.0f);
  glVertex2f(-7.0f,5.0f);
  glVertex2f(-7.0f,2.0f);
```

```
glVertex2f(-8.0f,2.0f);
glEnd();


//fingerprint1  display
glBegin(GL_POLYGON);
glColor3ub(205, 209, 206);
glVertex2f(-7.75f,4.50f);
glVertex2f(-7.25f,4.50f);
glVertex2f(-7.25f,4.25f);
glVertex2f(-7.75f,4.25f);
glEnd();


//finger print1 position
glBegin(GL_POLYGON);
glColor3ub(9, 230, 212);
glVertex2f(-7.62f,3.00f);
glVertex2f(-7.33f,3.00f);
glVertex2f(-7.33f,2.50f);
glVertex2f(-7.62f,2.50f);
glEnd();


//bench big back
glBegin(GL_POLYGON);
glColor3ub(96, 133, 158);
glVertex2f(5.0f,-7.0f);
glVertex2f(4.0f,-2.0f);
glVertex2f(18.0f,-2.0f);
glVertex2f(17.0f,-7.0f);


glEnd();
//bench midle rect
glBegin(GL_POLYGON);
glColor3ub(60, 113, 166);
glVertex2f(4.0f,-7.0f);
glVertex2f(18.0f,-7.0f);
glVertex2f(18.0f,-8.0f);
glVertex2f(4.0f,-8.0f);
glEnd();
```

```
 //rect bench
 glBegin(GL_POLYGON);
 glColor3ub(96, 133, 158);
 glVertex2f(4.00f,-7.00f);
 glVertex2f(18.00f,-7.00f);
 glVertex2f(18.00f,-8.00f);
 glVertex2f(4.00f,-8.00f);
 glEnd();



//1legbench
 glBegin(GL_POLYGON);
 glColor3ub(96, 133, 158);
 glVertex2f(4.00f,-8.00f);
 glVertex2f(5.00f,-8.00f);
 glVertex2f(6.00f,-11.00f);
 glVertex2f(5.00f,-11.00f);
 glEnd();
 //2leg
 glBegin(GL_POLYGON);
 glColor3ub(96, 133, 158);
 glVertex2f(6.00f,-8.00f);
 glVertex2f(6.75f,-8.00f);
 glVertex2f(7.25f,-11.00f);
 glVertex2f(6.50f,-11.00f);
 glEnd();
 //3rdleg


 glBegin(GL_POLYGON);
 glColor3ub(96, 133, 158);
 glVertex2f(15.25f,-8.00f);
 glVertex2f(16.00f,-8.00f);
 glVertex2f(15.5f,-11.00f);
 glVertex2f(14.75f,-11.00f);
 glEnd();
 //4th leg
 glBegin(GL_POLYGON);
 glColor3ub(96, 133, 158);
 glVertex2f(17.0f,-8.00f);
 glVertex2f(18.00f,-8.00f);
 glVertex2f(17.25f,-11.00f);
 glVertex2f(16.25f,-11.00f);
```

```
glEnd();
//fire box
glBegin(GL_POLYGON);
glColor3ub(219, 15, 36);
glVertex2f(9.0f,6.00f);
glVertex2f(12.00f,6.00f);
glVertex2f(12.00f,3.00f);
glVertex2f(9.00f,3.00f);
glEnd();


//fireboxsmall
glBegin(GL_POLYGON);
glColor3ub(47, 62, 66);
glVertex2f(10.0f,5.00f);
glVertex2f(11.00f,5.00f);
glVertex2f(11.00f,4.00f);
glVertex2f(10.00f,4.00f);
glEnd();


glBegin(GL_LINES);
glColor3ub(23, 21, 21);
glVertex2f(4.50f,-4.00f);
glVertex2f(17.50f,-4.00f);
glEnd();
//line
glBegin(GL_LINES);
glColor3ub(23, 21, 21);
glVertex2f(4.75f,-5.50f);
glVertex2f(17.25f,-5.50f);
glEnd();


 //noticeboard
glBegin(GL_POLYGON);
glColor3ub(4, 89, 81);
glVertex2f(-3.0f,12.0f);
glVertex2f(5.0f,12.0f);
glVertex2f(5.0f,4.0f);
glVertex2f(-3.0f,4.0f);
glEnd();
```

```
   glColor3ub(47, 62, 66);
   // Draw floor
   glBegin(GL_POLYGON);
   glVertex2f(-20.0f, -10.0f);
   glVertex2f(20.0f, -10.0f);
   glVertex2f(20.0f, -20.0f);
   glVertex2f(-20.0f, -20.0f);
   glEnd();


}



void update(int value) {


   if (isBulletActive) {
      bulletX += 0.5f;
      if (bulletX > 20.0f) {
         isBulletActive = false;
      }
   }


   for (int i = 0; i < numEnemies; ++i) {
      enemyX[i] -= 0.1f;
      if (enemyX[i] < -20.0f) {
         enemyX[i] = 20.0f;
         enemyY[i] = -12.0f + static_cast<float>(rand() % 300) / 100.0f;  // Randomize Y position
      }
   }


   checkCollision();
   glutPostRedisplay();
   glutTimerFunc(5, update, 0);
}
```

```
void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  background();



  if (level == 2){
    background2();
  }
  else if (level == 3){
    background3();
  }


  // Draw player, bullet, and enemies
  drawPlayer();
  drawBullet();
  drawEnemy();


  // Render text and other UI elements
  renderBitmapString(-0.2f, 10.0f, 0.0f, GLUT_BITMAP_HELVETICA_12, "Notice Board");
  glRasterPos2f(-0.2f, 8.0f);
  std::string scoreText = "Score: " + std::to_string(score);
  for (char c : scoreText) {
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, c);
  }


  glRasterPos2f(-0.2f, 6.0f);
  std::string levelText = "Level: " + std::to_string(level);
  for (char c : levelText) {
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, c);
  }


  glutSwapBuffers();
}


void handleKeypress(unsigned char key, int x, int y) {
  switch (key) {
  case 'a':
```

```
        if (playerX < -18) {
            isPlayerMoving = false;
        }
        else {
            playerX -= 0.5f;
        }
        break;
    case 'd':
        playerX += 0.5f;
        break;
    case 's':
        if (!isBulletActive) {
            // Fire bullet
            bulletX = playerX;
            bulletY = playerY;
            isBulletActive = true;
            gunShotSound();
        }
        break;
    }
    glutPostRedisplay();
}


int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitWindowSize(1000, 550);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Course Shooting Game");
    glutDisplayFunc(display);
    glutKeyboardFunc(handleKeypress);
    glutTimerFunc(25, update, 0);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    gluOrtho2D(-20.0, 20.0, -20.0, 20.0);



    // Seed for random number generation
    srand(static_cast<unsigned int>(time(nullptr)));
```

```
    glutMainLoop();
    return 0;
}
```