Computer graphics practical 7

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

int matrix[3][3];
int playerturn;
int result;
bool gameover;

void initgame() {
   playerturn = 1;
   for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
         matrix[i][j] = 0;
      }
   }
}

void keypress(unsigned char key, int x, int y) {
   switch (key) {
   case 'y':
      if (gameover == true) {
         gameover = false;
         initgame();
      }
      break;
   case 'n':
      if (gameover == true)
         exit(0);
      break;
   case 27:
      exit(0);
   }
}

void click(int button, int state, int x, int y) {
   if (!gameover && button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
      if (playerturn == 1) {
         if (matrix[(y - 50) / 100][x / 100] == 0) {
            matrix[(y - 50) / 100][x / 100] = 1;
            playerturn = 2;
         }
      } else {
```

```
            if (matrix[(y - 50) / 100][x / 100] == 0) {
                matrix[(y - 50) / 100][x / 100] = 2;
                playerturn = 1;
            }
        }
    }
}

void drawstring(void* font, const char* s, float x, float y) {
    unsigned int i;
    glRasterPos2f(x, y);
    for (i = 0; i < strlen(s); i++)
        glutBitmapCharacter(font, s[i]);
}

void drawlines() {
    glBegin(GL_LINES);
    glVertex2f(100, 50);
    glVertex2f(100, 340);
    glVertex2f(200, 340);
    glVertex2f(200, 50);
    glVertex2f(0, 150);
    glVertex2f(300, 150);
    glVertex2f(0, 250);
    glVertex2f(300, 250);
    glEnd();
}

void drawxo() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (matrix[i][j] == 1) {
                glBegin(GL_LINES);
                glVertex2f(50 + j * 100 - 25, 100 + i * 100 - 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 + 25);
                glVertex2f(50 + j * 100 - 25, 100 + i * 100 + 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 - 25);
                glEnd();
            } else if (matrix[i][j] == 2) {
                glColor3f(1.0, 1.0, 1.0); // Set the color to white
                glBegin(GL_QUADS);
                glVertex2f(50 + j * 100 - 25, 100 + i * 100 - 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 - 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 + 25);
                glVertex2f(50 + j * 100 - 25, 100 + i * 100 + 25);
                glEnd();

                glColor3f(0.0, 0.0, 0.0); // Set the color to black for the border
                glBegin(GL_LINES);
                glVertex2f(50 + j * 100 - 25, 100 + i * 100 - 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 - 25);
                glVertex2f(50 + j * 100 + 25, 100 + i * 100 - 25);
```

```
            glVertex2f(50 + j * 100 + 25, 100 + i * 100 + 25);
            glVertex2f(50 + j * 100 + 25, 100 + i * 100 + 25);
            glVertex2f(50 + j * 100 - 25, 100 + i * 100 + 25);
            glVertex2f(50 + j * 100 - 25, 100 + i * 100 + 25);
            glVertex2f(50 + j * 100 - 25, 100 + i * 100 - 25);
            glEnd();
         }
      }
   }
}



bool checkifwin() {
   int i, j;
   for (i = 0; i < 3; i++) {
      for (j = 1; j < 3; j++) {
         if (matrix[i][0] != 0 && matrix[i][0] == matrix[i][j]) {
            if (j == 2)
               return true;
         } else
            break;
      }
   }

   for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
         if (matrix[0][i] != 0 && matrix[0][i] == matrix[j][i]) {
            if (j == 2)
               return true;
         } else
            break;
      }
   }

   for (int i = 1; i < 3; i++) {
      if (matrix[0][0] != 0 && matrix[0][0] == matrix[i][i]) {
         if (i == 2)
            return true;
      } else
         break;
   }

   for (int i = 1; i < 3; i++) {
      if (matrix[2][0] != 0 && matrix[2][0] == matrix[2 - i][i]) {
         if (i == 2)
            return true;
      } else
         break;
   }
   return false;
}
```

```
bool checkifdraw() {
   int i, j;
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 3; j++) {
         if (matrix[i][j] == 0)
            return false;
      }
   }
   return true;
}

void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   glClearColor(1, 1, 1, 1);
   glColor3f(0, 0, 0);
   if (playerturn == 1) {
      drawstring(GLUT_BITMAP_HELVETICA_18, "PLAYER 1's turn (X)", 100, 30);
   } else {
      drawstring(GLUT_BITMAP_HELVETICA_18, "player 2's turn (Square)", 100, 30);
   }
   drawlines();
   drawxo();
   if (checkifwin()) {
      if (playerturn == 1) {
         gameover = true;
         result = 2;
      } else {
         gameover = true;
         result = 1;
      }
   } else if (checkifdraw()) {
      gameover = true;
      result = 0;
   }
   if (gameover) {
      drawstring(GLUT_BITMAP_HELVETICA_18, "game over", 100, 160);
      if (result == 0) {
         drawstring(GLUT_BITMAP_HELVETICA_18, "it's a draw", 110, 185);
      }
      if (result == 1) {
         drawstring(GLUT_BITMAP_HELVETICA_18, "PLAYER 1 WIN", 95, 185);
      }
      if (result == 2) {
         drawstring(GLUT_BITMAP_HELVETICA_18, "PLAYER 2 WIN", 95, 185);
      }
      drawstring(GLUT_BITMAP_HELVETICA_18, "DO YOU WANT TO CONTINUE (Y/N)",
40, 210);
   }
   glutSwapBuffers();
}
```

```
void reshape(int x, int y) {
    glViewport(0, 0, x, y);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, x, y, 0, 0, 1);
}

int main(int argc, char** argv) {
    initgame();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(500, 550);
    glutCreateWindow("Tic Tac Toe");
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutKeyboardFunc(keypress);
    glutMouseFunc(click);
    glutIdleFunc(display);
    glutMainLoop();
    return 0;
}
```

Output: