

**Patuakhali Science and Technology University**

**Report on**

**Tic-Tac-Toe Game Using OpenGL in C++**

**Course Title: Computer Graphics and Image Processing Sessional**

**Course Code: CIT-422**

**Submitted to:**

**Md. Mahbubur Rahman**

Assistant Professor

Department of Computer Science and Information Technology

Faculty of Computer Science and Engineering

Patuakhali Science & Technology University, Dumki, Patuakhali, Bangladesh

**Submitted by:**

**Mehedi Hasan Rabbi**

ID: 1802052

Reg No: 08462

Session:2018-19

**Introduction**

This report describes the implementation of a simple **Tic-Tac-Toe game** using **OpenGL** in **C++**. The game is built in **Code::Blocks**, which is a popular IDE for C++ programming. OpenGL, a cross-platform API, is used for rendering the game graphics, while C++ handles the game logic.

**Objective**

The objective of this project is to create a graphical Tic-Tac-Toe game that allows two players to play the game on the same computer. The players alternate turns to place their marks (X or O) on a 3x3 grid. The game ends when a player gets three of their marks in a row, column, or diagonal, or when the grid is filled without a winner, resulting in a tie.

**Tools Used**

* **Code::Blocks IDE**: This project was developed in Code::Blocks, a free C++ IDE.
* **OpenGL**: The OpenGL graphics library was used to render the 2D game board and player moves.
* **GLUT**: The OpenGL Utility Toolkit (GLUT) was used for window management and input handling.

**System Requirements**

1. **Operating System**: Windows, Linux, or macOS
2. **IDE**: Code::Blocks with GNU GCC Compiler
3. **OpenGL**: OpenGL library for rendering graphics
4. **GLUT**: GLUT for input and window management

**Game Design**

**1. Game Structure**

The game consists of a 3x3 grid, where players take turns to place their respective marks. The game keeps track of player turns and checks for a win condition after each move.

**2. Components**

* **Grid**: The game grid is a 3x3 matrix, where each cell can hold an ‘X’, an ‘O’, or remain empty.
* **Player Input**: Players will click on a grid cell to place their mark. The mouse input is captured using OpenGL/GLUT functions.
* **Rendering**: OpenGL is used to draw the grid lines, player marks (X and O), and display the game status.

**3. Win Condition**

A player wins if they succeed in placing three of their marks in a:

* Horizontal row
* Vertical column
* Diagonal

If the grid is completely filled without any player meeting these conditions, the game results in a tie.

**Game Flow**

1. **Initialize the Game Board**: The 3x3 grid is initialized and drawn using OpenGL primitives.
2. **Player Turn**: The game alternates between Player 1 (X) and Player 2 (O). Mouse clicks are used to place marks.
3. **Check for Win or Tie**: After every move, the game checks for a win or tie condition.
4. **End Game**: When a win or tie is detected, a message is displayed, and the game can be reset.

**Implementation Details**

**1. Setting Up OpenGL**

In Code::Blocks, OpenGL and GLUT libraries were configured to handle the rendering and input for the game. The basic functions like glClear(), glBegin(), and glEnd() were used to handle the drawing of the grid and player marks.

|  |
| --- |
| #include <GL/glut.h>  #include <iostream>  #include <cmath>  #include <string>  const int WINDOW\_WIDTH = 600;  const int WINDOW\_HEIGHT = 600;  const int GRID\_SIZE = 3;  int board[GRID\_SIZE][GRID\_SIZE] = {0};  bool isXTurn = true;  std::string gameStatus = "";  bool gameEnded = false;  // Other functions ….. // ……………………  void initializeGL() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-1.0, 1.0, -1.0, 1.0);  glMatrixMode(GL\_MODELVIEW);  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);  glutCreateWindow("Tic-Tac-Toe");  glutDisplayFunc(drawGrid);  glutMouseFunc(handleMouseClick);  // Set up OpenGL state  initializeGL();  // Enter GLUT event loop  glutMainLoop();  return 0;  } |

**2. Drawing the Tic-Tac-Toe Board**

The grid was drawn using OpenGL lines (GL\_LINES) to create the 3x3 matrix. The board is drawn once at the start and updated after every move.

|  |
| --- |
| void drawGrid() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3f(0.0f, 0.0f, 0.0f);  glLineWidth(5.0f);  // Vertical grid lines  for (int i = 1; i < GRID\_SIZE; ++i) {  float x = -1.0f + (2.0f / GRID\_SIZE) \* i;  glBegin(GL\_LINES);  glVertex2f(x, -1.0f);  glVertex2f(x, 1.0f);  glEnd();  }  // Horizontal grid lines  for (int i = 1; i < GRID\_SIZE; ++i) {  float y = -1.0f + (2.0f / GRID\_SIZE) \* i;  glBegin(GL\_LINES);  glVertex2f(-1.0f, y);  glVertex2f(1.0f, y);  glEnd();  }  // Draw the current state of the board  for (int row = 0; row < GRID\_SIZE; ++row) {  for (int col = 0; col < GRID\_SIZE; ++col) {  if (board[row][col] == 1)  drawX(row, col);  else if (board[row][col] == 2)  drawO(row, col);  }  }  // Draw the game status if the game has ended  if (gameEnded) {  glColor3f(0.0f, 0.0f, 0.0f);  glBegin(GL\_QUADS);  glVertex2f(-0.5f, -0.1f);  glVertex2f(0.5f, -0.1f);  glVertex2f(0.5f, 0.1f);  glVertex2f(-0.5f, 0.1f);  glEnd();  glColor3f(1.0f, 1.0f, 1.0f);  glRasterPos2f(-0.4f, 0.0f);  for (char c : gameStatus) {  glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, c);  }  }  glutSwapBuffers();  } |

**3. Handling Mouse Input**

Mouse clicks were captured using glutMouseFunc(). When a player clicks on a cell, the function calculates the corresponding grid position and places the appropriate mark (X or O).

|  |
| --- |
| void handleMouseClick(int button, int state, int x, int y) {  if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN && !gameEnded) {  // Convert mouse coordinates to grid coordinates  int row = (WINDOW\_HEIGHT - y) / (WINDOW\_HEIGHT / GRID\_SIZE);  int col = x / (WINDOW\_WIDTH / GRID\_SIZE);  if (board[row][col] == 0) // If the cell is empty  {  if (isXTurn) {  board[row][col] = 1; // Place X  std::cout << "X placed at (" << row << ", " << col << ")\n";  } else {  board[row][col] = 2; // Place O  std::cout << "O placed at (" << row << ", " << col << ")\n";  }  // Check for win conditions  if (checkWin(isXTurn ? 1 : 2)) {  gameStatus = "Winner is " + std::string(isXTurn ? "X" : "O");  gameEnded = true;  std::cout << gameStatus << "\n";  } else {  // Check for draw  bool draw = true;  for (int i = 0; i < GRID\_SIZE; i++) {  for (int j = 0; j < GRID\_SIZE; j++) {  if (board[i][j] == 0) {  draw = false;  break;  }  }  }  if (draw) {  gameStatus = "Draw";  gameEnded = true;  std::cout << gameStatus << "\n";  }  }  isXTurn = !isXTurn; // Switch turn  }  glutPostRedisplay(); // Redraw the screen  }  } |

**4. Drawing X and O**

The player marks are drawn using OpenGL primitives. An ‘X’ is represented by two diagonal lines, and an ‘O’ is drawn as a circle.

|  |
| --- |
| // Function to draw an X in the specified cell  void drawX(int row, int col) {  float cellWidth = 2.0f / GRID\_SIZE;  float xStart = -1.0f + col \* cellWidth;  float yStart = -1.0f + row \* cellWidth;  glColor3f(1.0f, 0.0f, 0.0f);  glLineWidth(3.0f);  glBegin(GL\_LINES);  glVertex2f(xStart + 0.1f, yStart + 0.1f);  glVertex2f(xStart + cellWidth - 0.1f, yStart + cellWidth - 0.1f);  glVertex2f(xStart + 0.1f, yStart + cellWidth - 0.1f);  glVertex2f(xStart + cellWidth - 0.1f, yStart + 0.1f);  glEnd();  }  // Function to draw an O in the specified cell  void drawO(int row, int col) {  float cellWidth = 2.0f / GRID\_SIZE;  float xCenter = -1.0f + (col + 0.5f) \* cellWidth;  float yCenter = -1.0f + (row + 0.5f) \* cellWidth;  glColor3f(0.0f, 0.0f, 1.0f);  glLineWidth(3.0f);  glBegin(GL\_LINE\_LOOP);  for (int i = 0; i < 360; ++i) {  float theta = i \* 3.14159f / 180.0f;  float x = 0.3f \* cos(theta);  float y = 0.3f \* sin(theta);  glVertex2f(xCenter + x, yCenter + y);  }  glEnd();  } |

**5. Checking Win Conditions**

After each move, the game checks for any winning combinations or a tie.

|  |
| --- |
| bool checkWin(int player) {  // Check rows  for (int i = 0; i < GRID\_SIZE; i++) {  if (board[i][0] == player && board[i][1] == player && board[i][2] == player) {  return true;  }  }  // Check columns  for (int i = 0; i < GRID\_SIZE; i++) {  if (board[0][i] == player && board[1][i] == player && board[2][i] == player) {  return true;  }  }  // Check diagonals  if (board[0][0] == player && board[1][1] == player && board[2][2] == player) {  return true;  }  if (board[0][2] == player && board[1][1] == player && board[2][0] == player) {  return true;  }  return false;  } |

**Conclusion**

The Tic-Tac-Toe game successfully demonstrates how to create a simple 2D game using OpenGL in C++. OpenGL efficiently handles the rendering of the grid and player moves, while C++ manages the game logic. This project offers a foundation for more complex OpenGL applications and showcases the power of using OpenGL for game development.