Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

Department of Electronics and Communication Engineering

III IA EVALUATION REPORT

for

BLENDED LEARNING PROJECT BASED LEARNING

A report submitted by

Name of the Student	**Brijin G**
Register Number	**URK22EC1021**
Subject Name	**Object Oriented Concepts using C++**
Subject Code	**18EC2022**
Date of Report submission	**4/11/2023**

Project Rubrics for Evaluation

First Review: to be awarded for **10 Marks** – PPT should have four slides (Title page, Introduction, Circuit/Block Diagram, and Description of Project).

Second Review: to be awarded for **10 Marks** – PPT should have three slides (Description of Circuit/Concept, Design of Circuit in Tools/Design of Flow graph, and Partial Results)

Third Review: to be awarded for **10 Marks** – PPT should have two slides (Tool based simulation/Hardware based simulation/Concept based interaction/ Case Study)

Fourth Review: to be awarded for **10 Marks** –PPT should have two slides (Output Results & Discussion)

[NOTE: The instructions are subjected to change if required to combine either of two reviews and to be evaluated for 20 Marks.]

Total marks: _____/ 40 Marks

Signature of Faculty with date:

Implementation of Tic-Tac-Toe in C++

PROJECT REPORT

Submitted by

Brijin(URK22EC1021)

Project/Skill based Assessment for the course

Object Oriented concepts using C++

Handled by

Dr. J. SAMSON IMMANUEL, M.E. Ph.D

Report submitted in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF

ENGINEERING AND TECHNOLOGY

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Deemed to be university)

Karunya Nagar, Coimbatore - 641 114. INDIA

November 2023

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

BONAFIDE CERTIFICATE

This is to certify that the mini project report entitled, "**Implementation of Tic-Tac-Toe in C++**" is a bonafide record of work of the candidate who carried out the project work under my supervision during the academic year 2022-2023.

Brijin(URK22EC1021)

Submitted for the review examination held on …….

(Internal Examiner)

CONTENTS

1. Introduction

Tic-Tac-Toe, a classic and timeless game, has captivated players of all ages for generations. It is a simple yet engaging two-player board game that challenges your strategic thinking and tactical skills. In this C++ implementation of Tic-Tac-Toe, we will explore the thrill of creating a program that allows two players to compete in this iconic contest. With the power of C++ programming, we can craft a virtual battlefield where Xs and Os clash in a battle of wits, aiming to secure a row, column, or diagonal of their symbols. Get ready to delve into the world of C++ and embark on a journey to create your very own Tic-Tac-Toe game that will provide endless hours of fun and entertainment.

2. Description

Tic-Tac-Toe in C++ is an iconic, grid-based game that challenges two players to strategically outwit each other. The game board consists of a 3x3 grid, with each cell capable of holding an 'X' or an 'O'. Two players take turns to place their respective symbols on the grid, aiming to create a winning combination of three symbols either horizontally, vertically, or diagonally. The program ensures input validation, ensuring that players cannot make illegal moves. It provides a user-friendly interface, displaying the current state of the board after each move and prompting the next player for their turn. The game continues until one of the

players achieves a winning combination or if no more valid moves are left, resulting in a draw. This C++ implementation of Tic-Tac-Toe offers a comprehensive introduction to programming concepts, including data structures, conditional statements, loops, and user input handling. It's a great way to learn and practice C++ while enjoying a classic game.

3. Concepts Involved

GTK Library

Data Structures

Conditional Statements

Loops

4. Project Code

```
#include <gtk/gtk.h>

GtkWidget *buttons[3][3];
int board[3][3];
int currentPlayer = 1;
gboolean gameOver = FALSE;
GtkWidget *statusLabel;

static void check_game_status();
static int check_winner();
```

```
static void on_button_clicked(GtkWidget *button, gpointer data) {
    if (!gameOver) {
        int row, col;
     row=GPOINTER_TO_INT(g_object_get_data(G_OBJECT(button),
                                                    "row"));

      col=GPOINTER_TO_INT(g_object_get_data(G_OBJECT(button),
                                                    "col"));

        if (board[row][col] == 0) {
            board[row][col] = currentPlayer;
            gtk_button_set_label(GTK_BUTTON(button),(currentPlayer
                                                    == 1) ? "X" : "O");
            gtk_widget_set_sensitive(button, FALSE);
            currentPlayer = 3 - currentPlayer;

gtk_label_set_text(GTK_LABEL(statusLabel),(currentPlayer=1)
                        ?   "Player X's turn" : "Player O's turn");
            check_game_status();
        }
    }
}

static void check_game_status() {
```

```
    int winner = check_winner();
    if (winner != 0) {
        if (winner == 1) {

gtk_label_set_text(GTK_LABEL(statusLabel),"PlayerXwins!");
        } else if (winner == 2) {

gtk_label_set_text(GTK_LABEL(statusLabel),"PlayerOwins!");
        } else {
            gtk_label_set_text(GTK_LABEL(statusLabel), "It's a draw!");
        }
        gameOver = TRUE;
    }
}

static int check_winner() {
    for (int i = 0; i < 3; ++i) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] &&
                                                        board[i][0] != 0)
{

            return board[i][0];
        }
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] &&
```

```
                                                    board[0][i] != 0) {
            return board[0][i];
        }
    }
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] &&
                                                    board[0][0] != 0) {
        return board[0][0];
    }
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] &&
                                                    board[0][0] != 0) {
        return board[0][2];
    }
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (board[i][j] == 0) {
                return 0;
            }
        }
    }
    return -1; // It's a draw
}

static void reset_game(GtkWidget *button, gpointer data) {
    for (int i = 0; i < 3; ++i) {
```

```
        for (int j = 0; j < 3; ++j) {
            board[i][j] = 0;
            gtk_button_set_label(GTK_BUTTON(buttons[i][j]), "");
            gtk_widget_set_sensitive(buttons[i][j], TRUE);
        }
    }
    currentPlayer = 1;
    gameOver = FALSE;
    gtk_label_set_text(GTK_LABEL(statusLabel), "Player X's turn");
}

int main(int argc, char *argv[]) {
    GtkWidget *window;
    GtkWidget *grid;
    GtkWidget *resetButton;

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window), "Tic-Tac-Toe");
    g_signal_connect(G_OBJECT(window),"destroy",
                                    G_CALLBACK(gtk_main_quit), NULL);
    grid = gtk_grid_new();
    gtk_container_add(GTK_CONTAINER(window), grid);
```

```
for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 3; ++j) {
        buttons[i][j] = gtk_button_new();
        gtk_widget_set_size_request(buttons[i][j], 100, 100);
        gtk_grid_attach(GTK_GRID(grid), buttons[i][j], j, i, 1, 1);
        g_object_set_data(G_OBJECT(buttons[i][j]),"row",
                                                GINT_TO_POINTER(i));
        g_object_set_data(G_OBJECT(buttons[i][j]),"col",
                                                GINT_TO_POINTER(j));
        g_signal_connect(G_OBJECT(buttons[i][j]),"clicked",
                            G_CALLBACK(on_button_clicked), NULL);

    }
}

statusLabel = gtk_label_new("Player X's turn");
gtk_grid_attach(GTK_GRID(grid), statusLabel, 0, 3, 3, 1);

resetButton = gtk_button_new_with_label("Reset Game");
g_signal_connect(G_OBJECT(resetButton),"clicked",
                            G_CALLBACK(reset_game), NULL);

gtk_grid_attach(GTK_GRID(grid), resetButton, 0, 4, 3, 1);
```

```
    gtk_widget_show_all(window);

    gtk_main();

    return 0;
}
```

5. Explanation of the Code

Initialization:

The code starts by initializing GTK using gtk_init. It also creates a main window and sets its title to "Tic-Tac-Toe."

Grid Setup:

A GTK grid is created to arrange the game buttons and other widgets.

Buttons: Inside a nested loop, nine game buttons are created and added to the grid. Each button corresponds to a cell on the Tic-Tac-Toe board. The g_object_set_data function is used to associate the button with its row and column on the grid.

Button Click Handling:

The on_button_clicked function is called when a game button is clicked. It checks if the game is not over, identifies the row and column of the clicked button, updates the game board, and displays 'X' or 'O' based on the current player. It also disables the button and updates the player's turn.

Game Status Checking:

The check_game_status function checks for a win or a draw. It calls the check_winner function, which examines the board for a winning combination. If there is a winner, it updates the status label accordingly, and the game is marked as over.

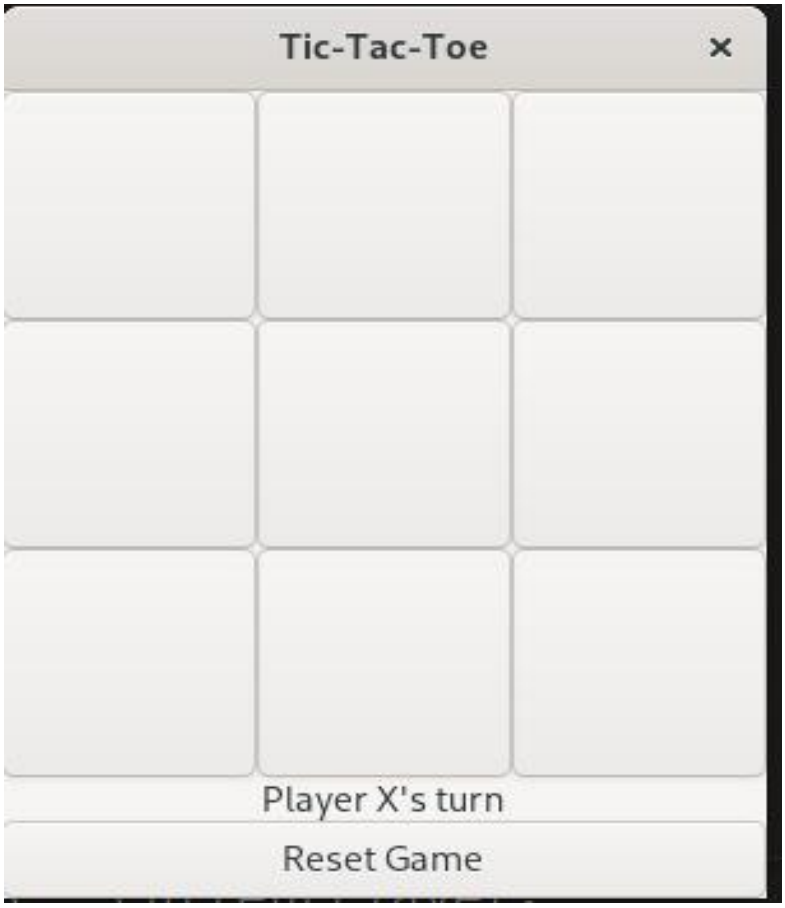
Reset Button:

A "Reset Game" button is added to allow players to start a new game. Clicking this button resets the board, enabling all the buttons and changing the current player back to 'X'.

Main Function:

In the main function, the main window and all the widgets are displayed, and the GTK event loop is started using gtk_main(). This loop handles user interactions and keeps the game running until the window is closed.

6. Result



Here is my output, The game allows two players to take turns marking a 3x3 grid with their respective symbols (X and O) until one player wins or the game ends in a draw. First X turn and then O turn.

All boxes are filled, but there is no 3x3 grid matching so the match is drawn

Here we start the game. First the player X turn he played on 3x2

Now player O turn played on 2x1

Now player X turn played on 3x3

Now player O turn played on 2x3

Now player X turn played on 2x2

Now player O turn played on 1x2

Now player X turn played on 1x1 and the player X wins.

Here the player O wins he made 3x3 grid by his respective symbol.

7. Conclusion

Implementation of the timeless Tic-Tac-Toe game, leveraging the GTK library for graphical user interface development. This code exemplifies a wealth of fundamental programming principles and best practices. It encompasses GUI development by skillfully creating an interactive user interface, featuring various GTK widgets like buttons, labels, and a grid, fostering player engagement. The code adeptly employs event handling through callback functions, enabling responsive interaction with the game. It adeptly utilizes data structures, like arrays, for representing the game board and button states, facilitating efficient game state

management. Game logic is effectively implemented, including player tracking, win condition checks, and game outcome determination, leveraging conditional statements and loops to ensure game integrity. The user interface design demonstrates careful attention to detail, offering an intuitive layout, label updates, and a game reset button for enhanced user experience. Input validation is enforced to guarantee that players make valid moves, contributing to the game's reliability. The comprehensive code covers the entire application lifecycle, from initialization to window creation, handling closure events, and running the main event loop. Game state management is executed seamlessly, including player turns and end game scenarios. The inclusion of a game reset feature adds a layer of user convenience, allowing for repeated play. In essence, this code is an instructive showcase of software design and development principles, making it a valuable learning resource for those exploring C++ and GUI programming. Furthermore, it delivers a captivating and interactive gaming experience, illustrating the practical application of programming concepts in a real-world context.