Game Project Report

crPong

nASIF AHMED

Department of

Computer Science and Engineering

University of Rajshahi

Submitted by

Name: Nasif Ahmed

Roll: 1610776129

Course Title: Software Development Lab-I

Course code: CSE-2142

Session: 2015-16

Part-II, Odd Semester

2015-2016

**Abstract**

The goal of this game project was to make game with GUI using C++ and SFML in a 2D environment. For my project, I decided to recreate Atari’s classic hit, pong, with a few ideas of my own. The game can be played by 2 players on the same computer from local multiplayer support. The game has no particular objective, players can play as much as they want and try to defeat the opposition player or the CPU. The game is designed in and for the Windows environment and written in C++ and SFML (Simple Fast Multimedia Library) which incorporates OpenGL. Paint and GameMaker Studio was used for creating certain resources.

**Contents**

1.0 Introduction 3

2.0 Game Overview 4

2.1 Game Description 4

2.1.1 Paddles 5

2.1.2 Balls 5

2.1.3 Environment 5

2.2 User Manual 6

3.0 Game Design and Tools involved 7

3.1 Ball Movement 8

3.2 Collision Detection 8

3.3 Sound Effects 8

3.4 CPU movement 8

4.0 Conclusion 9

4.1 Future Plans 9

**1.0 Introduction**

Pong, first developed by Atari, one of the first video games to reach mainstream popularity was the first commercially successful arcade video game machine helped to establish the video game industry along with the first game console, the Magnavox Odyssey.

This project aims to bring the fun and simplicity of the classic pong game with some new features. This report describes the process involved in implementing my own version of pong with SFML. Using SFML and C++, I created my own version of pong where the objective for the user is to score higher than the CPU or the opposition for self-satisfaction. This report describes the game overview, including the description and game play and then focuses on game design, describing how the game was implemented and the programming function and libraries used in the design.

**2.0 Game Overview**

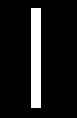
The primary objective of the game is for the user to score as much as they can against the opposition. The game is played between either 2 players or 1 player against the CPU of which the difficulty can be selected as per the user preference. The user(s) controls the paddle(s) and the ball moves on its own. The user(s) try to collide the ball with the paddles so as to not lose when the ball is going to the user’s paddle direction. This section elaborates on the description of the game play and user controls.

**2.1 Game Description**

In the game, player when selected single player controls the left paddle of the game and plays against the CPU or when selected multiplayer, both players play against each other to score as much as they can. To win a round, the user has to make the opposition fail to hit the ball with the paddle and to hit the wall behind opponent paddle. The UI is minimalistic with sprites and texts drawn over a blank screen each frame.

**2.1.1 Paddles**

Minimalistic rectangle shapes that represent the paddles the player(S) and CPU controls. These regular paddle shapes can have textures set up on them but I have left it alone. The player(S) and CPU controls these paddles and tries to hit the incoming balls and tries to win by hitting the ball out of bounds.



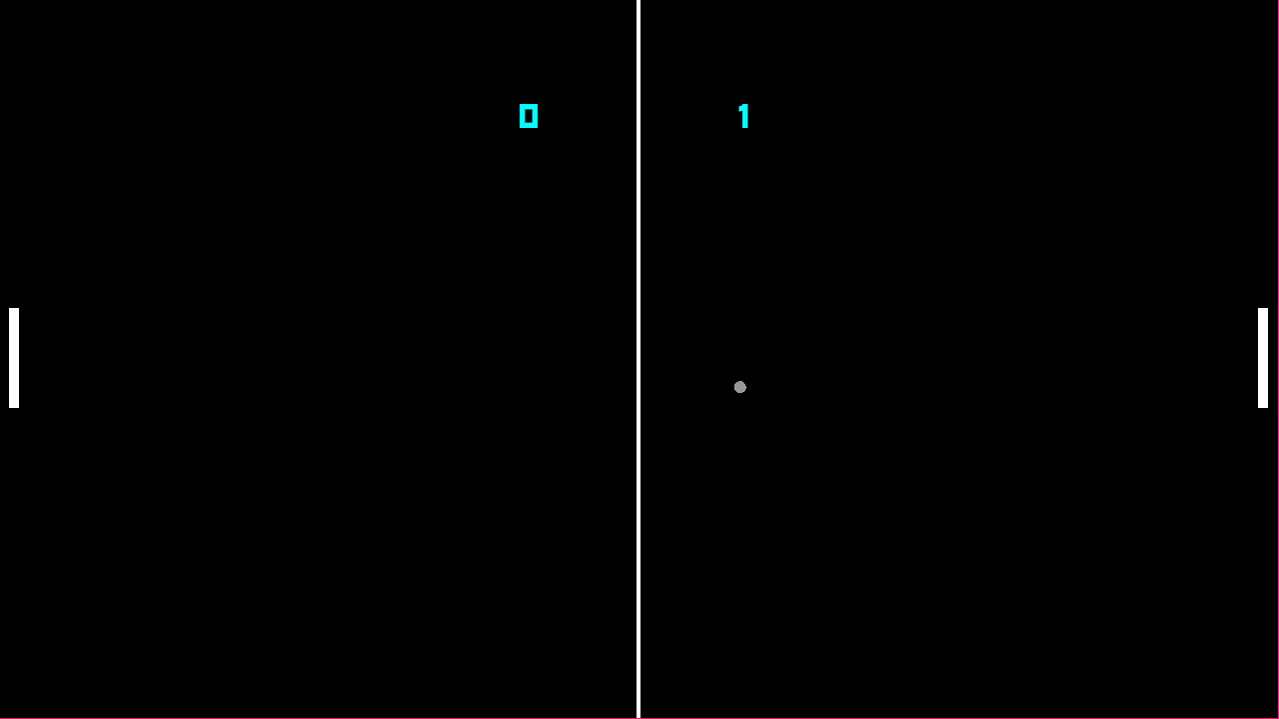
**2.1.2 Ball**

The ball that moves each frame at a certain angle with a certain velocity that player(S) and CPU tries to hit is also a minimalistic version of SFML’s own ball shape which has its texture as a plain one for this implementation. The ball generates a random angle each time it collides with a paddle, and negates its angle when it hits with the horizontal border.



**2.1.3 Environment**

The game starts with an intro and credits page for those who have helped in some way. Whilst playing the game frame is minimalistic with the paddles and the ball are drawn each frame after events have been processed. There’s a middle bar drawn each frame as well but its position is static and its purpose is for only visuals. The game also keeps a track of score each time one player scores a point which resets once the game is reopened.



**2.2 User Manual**

Depending on whether multiplayer or single player is selected, keyboard input for movement differs. Controls are listed for various purposes below.

Menu Navigation:

* Key ‘Up Arrow’: Moves up the current menu selection.
* Key ‘Down Arrow’: Moves down the current menu selection.

Single Player:

* Key ‘W’: Moves the left paddle upwards.
* Key ‘S’: Moves the left paddle downwards.

Multiplayer:

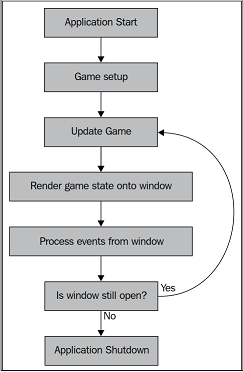
* Key ‘W’: Moves the left paddle upwards.
* Key ‘S’: Moves the left paddle downwards.
* Key ‘Up Arrow’: Moves the right paddle upwards.
* Key ‘Down Arrow’: Moves the right paddle downwards.

Other:

* Key ‘ESC’: Takes the game state to the menu screen regardless which state it is in.
* Key ‘Return’: Navigates to the selected game state or selects an option.

**3.0 Game Design and Tools Involved**

In creating the game, the basic outflow of the game can be considered as below.



Visual studio’s 2015 IDE has been used in developing the source of the game, whereas all the other resources of the game has been made through Paint, Gamemaker, and fonts and libraries downloaded from the internet. SFML library has been used for most part of the GUI. DLLs have been linked dynamically through copying each DLLs of SFML to the executable file’s location. Windows 10 has been chosen as the build platform, which can also be built on linux as the source doesn’t have any windows specific code.

Multiple SFML’s built-in functions have been used in designing the game. This section describes the methods of how the game is implemented.

**3.1 Ball Movement**

The ball moves along a random angle every time it collides with a paddle and inverts the sign of the angle when it collides with a horizontal wall. The ball movement of the game is time independent. It moves a certain amount each frame regardless of how much FPS the user’s computer can produce. The ball moves along the cos and sin values of the angle produced each frame multiplied by a constant factor so as to make it smooth as possible.

**3.2 Collision Detection**

As the ball can collide with either the wall or the paddles, the implementation of collision detection is required to detect when the ball hits the wall, and when the ball hits either of the paddles. For the first type, collision between the ball and wall works when the ball’s area extends across the frame of the game, and when it does, it goes to a new position right below the border. For the second case, the condition is: if there is an overlap between the area consumed by the ball and one of the paddles.

**3.3 Sound Effects**

SFML’s own library was used for the playback of the sounds of the game when the ball collides with either the wall or the paddle. The sound clip is in wav format which are stored and loaded in an array for easy access.

**3.4 CPU Movement**

When playing as single player, user is able to select difficulties while playing. In both difficulties, the computer moves along only the Y axis depending on the ball’s current Y axis position or the predetermined Y axis position offset by a certain value based on the difficulty selected.

**4.0 Conclusion**

In making of the game, a 2D game environment along with static viewing angle was made. The game was built successfully with the help of SFML’s own audio, graphics library and its own collision detection. The game has also been ran on multiple systems each build to make sure it runs and without much errors. All the necessary runtime libraries have been included in the executable file’s directory so the user doesn’t have to go to too much trouble.

**4.1 Future Plans**

Although the game is fine on its own, a few features will hopefully be added so as to give users a better experience. Some of which are listed below.

* Adding acceleration to paddle movement for precise movement.
* Add single player modes for practice against just a wall.
* Keep track of high scores.
* Making the game bug free for underpowered systems.