Design Doc

For CA2 of 2D games Development we were required to improve upon the game of pong.

## What is the game?

The game is fairly simplistic. It is simply a bat and a ball. You must hit the ball with the bat to hit the top of the screen to score points. You only have 3 lives. Once lives reach 0 your score resets.

## Where does the game take place?

The game takes place within a black screen but I modified the background to give it more of a sci fi feel.

## What do I control?

You control a bat and you can slide from left to right to get to the ball. You act as almost like a goalie protecting your goals.

## What is the main focus?

The main focus of the game is to not allow the ball to pass your bat and to hit the ball up to the top of the screen to score as many points as you can before your lives run out.

## Whats different about my game?

In my game the screen is changed from the black to a more sci fi looking background. I also added a sound for when the ball hits something so like the left right top or bottom of the screen. I also added a main menu before the game begins and I increased the balls speed to make it harder.

# Code Modifications:

To start with Ill talk about how I increased the speed of the ball. What I did was set the speed variable to 3000.0f in the ball.cpp file.

Ball::Ball(float startX, float startY, float speed)

{

speed = 3000.0f;

m\_Position.x = startX;

m\_Position.y = startY;

m\_Speed = speed;

m\_Shape.setSize(sf::Vector2f(10, 10));

m\_Shape.setPosition(m\_Position);

}

This sets the speed for the ball and passes it through to the method in the header file for the ball.

Next was the audio. For this I had to import the sfml audio library.

#include <SFML/Audio.hpp>

Then later on in the code I made the Soundbuffer object and called the file from the sound folder for the sound and set the sound object to the soundbuffer object. I created this sound myself using bfxr.

SoundBuffer ballBuffer;

ballBuffer.loadFromFile("sound/ball.wav");

Sound ballSound;

ballSound.setBuffer(ballBuffer);

Next I changed the background for the game. To do this I made a texture object and loaded the jpeg file from the graphics folder. I found this image online it was royalty free. Then I created a sprite to set the texture object to. I set the position to 0,0 so it would cover the whole screen. Then I drew it with the other objects at the bottom of the code.

Texture textureBackground;

textureBackground.loadFromFile("graphics/newBackground.jpg");

Sprite spriteBackground;

spriteBackground.setTexture(textureBackground);

spriteBackground.setPosition(0, 0);

Finally I created the main menu for the game. For this I attempted to create a functioning leaderboard but it kept glitching through the main menu so I had to cut it as we have not learned to create a proper menu yet. What I did manage to do was display 3 strings. Once was “MAIN MENU”. The next was “Play Game - Press enter” and finally “Exit game – Press escape”. To create this is I had to add quite a bit of code to the pong.cpp file. To start I added 2 new Booleans – menu and paused. Menu is set to true by default and paused was also set to true. The reason for the paused Boolean was so that I could start the game in a paused state before it drew any of the game.

bool menu = true;

bool paused = true;

Next I made the Text objects for the menu –

sf::Text menuText1;

sf::Text menuText2;

//sf::Text menuText3;

sf::Text menuText4;

I then set the size of the string, the font and set what the string would say.

menuText1.setFont(font);

menuText2.setFont(font);

menuText3.setFont(font);

menuText4.setFont(font);

menuText1.setCharacterSize(100);

menuText2.setCharacterSize(100);

menuText3.setCharacterSize(100);

menuText4.setCharacterSize(100);

menuText1.setString("Main Menu");

menuText2.setString("Play Game - Press enter");

menuText3.setString("Leaderboard - Press 1");

menuText4.setString("Exit Game - Press escape");

Then I went on to set the position the menu would display in. I did this by using float rect which put the string in a rectangle which I could position around the screen. I used the dimensions of the screen 1920 x 1080 for reference and positioned it around that.

FloatRect textRect = menuText1.getLocalBounds();

menuText1.setOrigin(textRect.left +

textRect.width / 2.0f,

textRect.top +

textRect.height / 2.0f);

menuText1.setPosition(1920 / 2.0f, 1080 / 3.0f);

FloatRect textRect1 = menuText2.getLocalBounds();

menuText2.setOrigin(textRect.left +

textRect1.width / 2.0f,

textRect1.top +

textRect1.height / 2.0f);

menuText2.setPosition(1920 / 2.1f, 1080 / 2.2f);

/\*FloatRect textRect2 = menuText3.getLocalBounds();

menuText3.setOrigin(textRect.left +

textRect2.width / 2.0f,

textRect2.top +

textRect2.height / 2.0f);

menuText3.setPosition(1920 / 2.1f, 1080 / 1.7f);\*/

FloatRect textRect3 = menuText4.getLocalBounds();

menuText4.setOrigin(textRect.left +

textRect3.width / 2.0f,

textRect3.top +

textRect3.height / 2.0f);

menuText4.setPosition(1920 / 2.1f, 1080 / 1.7f);

Next I made an event for the paused state which just has the game paused until the player presses a key.

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::KeyReleased && !paused)

{

// Listen for key presses again

acceptInput = true;

}

}

Here is the if statement for the menu just below the event. So if the menu Boolean is true which it is by default then it will only draw the menu text items and not the game. Once this Boolean is set to false however the game will be drawn and you can play.

if (menu)

{

window.draw(menuText1);

window.draw(menuText2);

//window.draw(menuText3);

window.draw(menuText4);

window.display();

}

Here is the code for that. So once enter is pressed it goes onto the game which is all stored in an if statement for the acceptInput Boolean. It sets menu to false and accept input to true.

if (Keyboard::isKeyPressed(Keyboard::Return))

{

menu = false;

//highScore = false;

acceptInput = true;

}

if (acceptInput)

{

//TIME BAR CODE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8

/\*timeRemaining = 6;

timeRemaining += (2 / 1) + .15;

\*/

// Handle the pressing and releasing of the arrow keys

if (Keyboard::isKeyPressed(Keyboard::Left))

{

bat.moveLeft();

}

else

{

bat.stopLeft();

}

if (Keyboard::isKeyPressed(Keyboard::Right))

{

bat.moveRight();

}

else

{

bat.stopRight();

}

To conclude. I feel I learned quite a lot from this CA. I learned to work within the bounds of my current knowledge to create the most that I can. I ran into some speed bumps with the leaderboard but we will learn how to properly implement a main menu next semester.