

**CSED451 COMPUTER GRAPHICS**

**ASSIGNMENT 1**

2D DRAWING

SUBMITTED BY: ONG WEI HUA (49003139, ongweihua)  
 TAN WEI XUAN (49003140, tanweixuan)

TEAM NAME: SINGAPORE PEOPLE

DEPARTMENT: DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBMITTED TO: PROF LEE SEUNG YONG

SUBMISSION DATE: 15 MARCH 2019

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**2019**

# PROGAMMING ENVIRONMENT

The Programming Languages used are as follow:

* *FreeGlut 3.0.0-2*
* *GLEW 2.1.0*
* *GLM 0.9.9.1*
* *C++*
* *OpenGL & GLSL in Windows*

The Integrated Development used is

* *Microsoft Visual Studio 2007*

We also utilized the Source Control Platform, *GitHub*, and integrated it with Microsoft Visual Studio 2007 to allow us to be able to coordinate work between us and to allow us to easily track changes in our source code during our development process. The Repository can be accessed and view at the following link:

* *https://github.com/jermsinarocket/ComputerGraphics\_Assignment1*

# FUNCTIONALITY OF THE PROGRAM

The program that we implemented is a 2D Volleyball game that is based on the concept of the original “*Pikachu Beach Volleyball*” 2D side-scrolling game.

Our game consists of two characters that are situated on the left/right of the screen with a **net** separating them and determining their respective play zones. One of the characters is a **playable** one, where it can be controlled by the user, while the other one is an **AI** where it will be moved based on the logic and algorithm being implemented by us.

The “*Volleyball*” will be constantly moving and changing directions upon collision with the player/AI, net and game (window) boundaries.

The objective of the game is to reach a predetermined score by letting the ball touched the ground within the opposing side’s play zone.

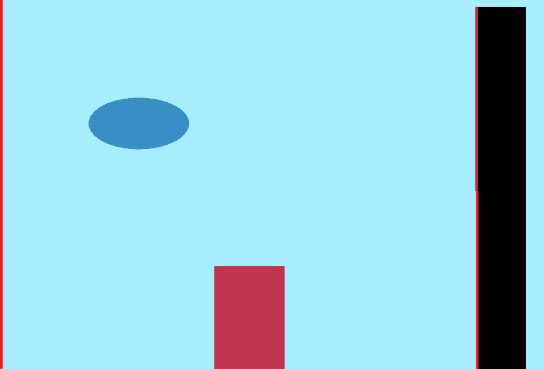
# DESIGN AND IMPLEMENTATION

Our program’s source codes are being split into their respective header and implementation files, with the formal comprising of the class definitions and functions and the latter containing the implementation of that class. This helps us to improve the build time of our program and to enable us to link against code without having the source of the definitions.

Our game is initialized with a viewport of *1440* by *800* pixels and is projected and mapped to the clipping coordinates, , with the origin being fixed at the centre of the window. It is set to be in full screen mode to replicate an actual game and to prevent screen resizes from distorting the components within the game *(although it can be fixed by multiplying the projection matrix with the ratio of the resized width and height, the components within the game will be scaled to become either too large or too small)*.

The Net is centralised on the viewport with reference to the defined origin and it serves as a divider for the play zones of the playable character and AI respectively. It is stationary, and its position will not be changed throughout the entire execution of the game.

The playable character will be the one that the user will be able to control, and it will be centred initially at the left play zone with reference to the left game boundary and the left boundary of the net. It can be controlled/moved through the directional keys inputs on the keyboard (left/right), with the left directional key moving it left along the x-axis *(negating its speed and adding that to its [])* and the right directional key moving it right along the x-axis *(Adding its speed to its []),* with the position of the character between constrained by the factors *(Figure 1).*

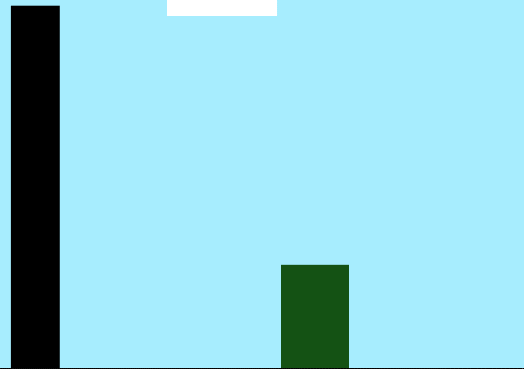


*Left Net Boundary*

*Left Game Boundary, x = -1*

Figure 1

The AI will be centred initially at the right play zone with reference to the right game boundary and the right boundary of the net. Its movement will be automatic and continuous along the x-axis constrained by the factors and the AI’s position will be reversed upon collision with either the right boundary of the net or the right boundary of the game window. Its speed is set to be much lower than that of the playable character as it will be moving on every render, unlike the playable character.



*Right Net Boundary*

*Left Game Boundary, x = 1*

Figure 2

The ball will be centred initially between the top of the net and the top of the game window by referencing its position to the -coordinate of the net and the -coordinate of the game window.

# BRIEF EXPLANATION OF THE PROGRAM

# PROBLEMS FACED

There are several problems that we have faced while we are doing the programming assignment one for this.

Firstly, as we are new to coding in OpenGL, we are not familiar with the syntax of how to code an OpenGL program. Despite the both of us having the programming background knowledge, the languages, C++ and OpenGL programming, are a relatively new concept to us. We faced difficulties in understanding the syntax for OpenGL, and in order to grasp the concept of OpenGL programming, we had to learn through reading and watching online tutorials. Secondly, the drawing of the different shapes of the characters and ball was another problem we faced. We had to make sure the coordinates of each object are at the position that we want it to be at and making sure that is reflected correctly whenever the window is clipped or resized. To learn how to make the right shape correctly, we had to learn from an online tutorial that teaches how to make a right shape for OpenGL programming.

Thirdly, the clipping window function that is required by the assignment was difficult to implement as we are not sure how to do the clipping windows with the window following the ball when it moves. We had to read up and research on how the clipping windows works in OpenGL to get the windows moving with the ball. From the class, we are taught that the clipping windows are only showing a portion of the window without the whole world coordinates. We adapt the method of what we had learnt and coupled it with the tutorial to create the clipping window that is required by the assignment. Lastly, calculating the velocity of the ball moving speed and direction of the ball requires some mathematics calculation. We learnt it by setting a certain speed to the ball and keeping it constant despite hitting the obstacles.

# IMPROVEMENT TO PROGRAM

Firstly, we would like to make the characters have some jump functions without having to wait for the ball to “come” to them to bounce off them. With the jump functions, it can help to make a difference to the game play such as the direction of the ball can be changed accordingly without following a fixed direction of bouncing. Secondly, the score board can be implemented directly in the middle of the window where the score board is more visible than the current one. The current one is that it keeps track the score of each player at their own side of the field. Having it in the middle will allow a better overview of the score and easily for the player to see it. The score board will have a boundary to it so that the ball does not travel into the score board which makes the ball invisible to the player. Next, we could make it more interactive for the AI by having some random speeches when it plays like encouraging words for the player whenever they win or lose a point. The speech will be generated with a pre-defined set of phrases and set to use the phrases at certain point differences is too far or whenever the player scores a goal.

# CONCLUSION

In summary, we have learnt how to use OpenGL with C++ and developed a simple 2D game with OpenGL. As we did not have a lot of programming background with C++, we have learnt how to use C++ along the way with OpenGL through the various video tutorials. We learnt about how to use the various functions in OpenGL from the lectures and tutorials to get our 2D game running. As creating a 2D game from scratch was tough, it was a valuable experience to start coding using OpenGL.

# REFERENCES

http://lazyfoo.net/tutorials/OpenGL/index.php

https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG\_Introduction.html4

http://www.codersource.net/2011/02/06/a-simple-2d-game-with-opengl-ball-with-bat/

https://gamedev.stackexchange.com/questions/23672/determine-resulting-angle-of-wall-collision