# Game Engine Development - Assignment 1

The game that is going to be developed using this game engine, will be based around a 'Brick breaking' type genre of game, where the user will use keyboard inputs to control a character platform to bounce a ball back and forth to destroy target objects. To add a layer of difficulty and challenge to the game, after an increment of levels, Ie 5 levels there will be a ‘boss’ battle. This will require the user to hit objects towards the boss reducing the life of the enemy each time until the bosses life points are depleted.

### SFML VS SDL

In my opinion the game engine will use SFML as the external library and API, SFML has the wider selection of libraries and functionality compared to SDL, while also being automatically hardware accelerated to run processes faster rather than SDL libraries. This is because of the use and access of the GPU rather than SDL's sole use of the CPU. Taking those factors aside, SFML is a more recent API so in turn the updates and bugs are being pushed out much faster than SDL libraries. There is good documentation and online tutorials for SFML whereas due to the lack of activity from SDL development team soon parts of the libraries will be deprecated and not function correctly.

## SFML

### 2D Rendering

2D rendering will be easily managed by SFML. "sf::RenderWindow adds high-level functions to help you draw things easily."(Sfml-dev.org, 2017). This alongside the 'draw' functions allows a multitude of 2D rendering options, Ie. Adding sprites, Text, shapes and vertices. (Milchev, 2015)

SFML uses an easy to access functionality for text, sprites shapes and vertices with the ability to render with different properties, using easy to understand and simplified API language. This helps with keeping the amount of code to compile down and in turn keeping processing speeds high while also leaving it easy to read for future developers and editors increasing the longevity of the engine. (Sfml-dev.org, 2017)

### User Input

The human interface device’s that will be able to be used within the game engine will be, either a wired keyboard that is generally used for Microsoft pc’s, or a wired Microsoft Xbox 360 game pad. The engine will read the devices by polling periodically each frame, checking the status of the hardware’s states at each point and allow for continuous uninterrupted movement of the character. Using Microsoft’s Xinput API uses the same method as about and allows for easy access to the functionality of the Xbox 360 game pad. (Gregory, 2014; Sfml-dev.org, 2017)

The Engine will have menu options available to select between a keyboard or a xbox 360 game pad. Using ‘XINPUT\_VIBRATION’ will allow the game to send outputs to the game pad to use the ‘Rumble’ mechanics of the hardware to give a sense of tactile feedback when an object hits the players character.

### Collision Detection

The game is going to be Requiring collisions between the object(‘Ball’) and the player character, blocks and the ‘boss’ entities, with the ability to bounce of the side, bottom and top walls. Sfml provides a simple way to do this using Bounding boxes (Sfml-dev.org, 2017), due to the simplicity it would create collision hitbox errors and overall look visually unpolished, especially when working with shapes that are not flat i.e. A sphere/circle so using open source pre written functions in SFML Like “*Simple Collision Detection for SFML 2”* (Koirala, 2017) a more high level of functionality and accuracy can be created. Doing this will also create an efficient collision world maintaining all collision information within a private data structure rather than storing collision information within the objects themselves, this increases performance and organization. (Gregory, 2014)

### Physics

The ‘ball’ object will have a constant velocity, whereas the players character will increase in velocity when movement its active, until it reaches a max speed and decrease in velocity when it is required to stop, creating a visual representation of acceleration and deceleration. (Buckland, 2004) This creates a basic form of physics that will not require importing libraries with unwanted functions and wasting computing resources.

Personal Notes

\*\*\*\*Re read through slides and check that I’m explaining the right things.

• AI - I don’t plan to use much AI, But if I do, Talk about my use of quad trees and the procedural use of them to be used as a component rather than a hard coded thing.

• Resource management – SMFL provides good resource management and this is important to keep speeds and reliability up. Also include something about limiting error files and read outs and limiting what users to developers can actually use and see from the game engine.

• Scene management – Not sure on this yet.

# **References**

(Gregory, 2014)

Gregory, J. (2014) *Game engine architecture*. 2nd edn. Boca Raton, Florida: CRC Press.

(Moreira, 2013)

Moreira, A., Haller, J. & Hansson, H.V. (2013) *SFML Game Development,* 1st edn, Packt Publishing, Olton.

(Milchev, 2015)

Milchev, M. (2015). *SFML essentials*. Birmingham, UK: Packt Publishing, pp.24-28.

(Sfml-dev.org, 2017)

Sfml-dev.org. (2017). *Tutorials for SFML 2.0 (SFML / Learn / 2.4 Tutorials)*. [online] Available at: https://www.sfml-dev.org/tutorials/2.0/ [Accessed 8 Nov. 2017].

(Koirala, 2017)

Koirala, N. (2017*). Simple Collision Detection for SFML 2*.[online] Available at: https://github.com/SFML/SFML/wiki/Source:-Simple-Collision-Detection-for-SFML-2: LittleMonkey Ltd. [Accessed 13 Nov. 2017].

(Buckland, 2004)

Buckland, M. (2004). AI game programming by example. Plano, Tex.: Wordware.

# **Bibliography**

<https://www.sfml-dev.org/faq.php#grl-whatis>

<https://www.reddit.com/r/gamedev/comments/44npzz/sdl20_vs_sfml2_in_2016/>

<https://www.reddit.com/r/gamedev/comments/42og4b/what_is_a_good_library_to_2d_games_in_c/>

<https://www.sfml-dev.org/tutorials/2.0/graphics-draw.php>

<https://en.wikipedia.org/wiki/Application_programming_interface>

<https://stackoverflow.com/questions/22886500/how-to-render-text-in-sdl2>

<https://wiki.libsdl.org/FAQDevelopment>

<https://en.wikipedia.org/wiki/Polling_(computer_science)>

<https://www.sfml-dev.org/tutorials/2.1/window-events.php>

<https://answers.unity.com/questions/218084/xinput-how-do-i-access-vibration-on-360-controller.html>

<https://msdn.microsoft.com/en-us/library/windows/desktop/microsoft.directx_sdk.reference.xinput_vibration(v=vs.85).aspx>

Collision Detections:

<https://en.sfml-dev.org/forums/index.php?topic=5704.0>

<https://en.sfml-dev.org/forums/index.php?topic=11655.0>

<https://www.sfml-dev.org/tutorials/2.4/graphics-transform.php>