Simon Technical Design Doc

# Target

The game will run on a college PC with a mouse and sound playback capabilities.

I will be testing on the Development Machine (There will not be a test machine).

SFML libraries need to installed and on the path of The PC.

Windows OS most any version will be testing with 10 and 8.1

# Development Environment

VS2017 community

SFML 2.5.0

NO audio SOftware

NO graphics editing Software

Dev PC Pete’s home PC & Pete’s Office PC and LRC Lab PC’s

# Design methodology

A single game class

A game loop calling process events every loop

Calling update 60 times a second

Calling render every iteration (more than update possible infinite)

# Classes

## Enum GameState

(4 modes)

## Game class

### Members

Array[] for notes

4 Booleans for button clicks from process events to update

mainWindow

4 rectangleshapes for buttons

4 coloured text message for menu

Title and status text objects

4 int timers for buttons

1 int timer for mode change delays

Soundbuffer and 4 sounds

Font

Current gamemode

Int current note, current sequence length, difficulty length

### Methods

**Game()** default constructor, setup window, rectangles and other variables to initial states.

**resetButtons()** reset the booleans to false for the next process events call.

**randomiseNotes()** get a new set of 32 random notes.

**setupButtons()** Load the font & sound and set up the square’s colour location, the text elements font, text and location.

**run()** main game loop.

**processEvents()** check for window events and call game process events.

**processGameEvents(sf::Event& event)** deal with events and set the boolean if there’s a mouse click on one of the buttons.

**update(sf::Time time)** switch to one of the update methods based on gamemode.

**startingUpdate()** update for the main menu, start a gamer or exit.

**countdownTimers()** decrement the colour timers and reset the colour if necessary.

**recievingUpdate(sf::Time time)** check is a button was pressed check if it’s correct or wrong, check if time has run out and play sound and highlight button if pressed. Move on to next note is correct.

**showingUpdate()** play the next note and highlight button after appropriate delay. Check for win

**overUpdate()** play winning or losing tones and wait to go to menu.

**render()** draw buttons and text as appropiate.

# Bit & pieces (code samples for students)

Add highlight

**m\_greenSquare.setFillColor(m\_greenSquare.getFillColor() + sf::Color(64, 64, 64, 255));**

Check mouse button released

**if (sf::Event::MouseButtonReleased == event.type)**

Mouse location

**if (event.mouseButton.x > 350 && event.mouseButton.x < 550)**

Check time lenght

**if (m\_inputTime.asMilliseconds() > 1500)**