LIM CHUAN YONG, ALDEN (A0118870B)
NM2207 Final Project Documentation

IDEA

I designed a colour-matching based game due to a number of inspirations. Firstly, the Emitter exercise done in class gave me the idea of shooting small pellets. Secondly, based on certain recent game concepts such as Splatoon and the mobile game Gyro, I settled on the idea of a paintball shooting game based on the three primary colours: red, blue and yellow.

I chose this mainly because I could leverage on some code already done in class such as the Emitter exercise as mentioned above and from the various assignments. Also, the paintball shooting game allows me to exercise all skills learned in class such as arrays and loops. I also went on to research on the switch case statement and how to attach “setIntervals” individually to objects rather than doing a global “setInterval” statement that looped through every single object.

CODING

To code the game, I broke down the game into various components: the shooting mechanic, the spawning mechanic, the collision detection mechanic and the adaptive difficulty mechanic.

To do the shooting mechanic, I converted the Emitter exercise code to only spawn circles upon mouse clicks. An array is used to easily create multiple bullets using the same variable name. Each element in the bullet array has its own properties such as xpos, ypos, xrate, yrate. Specific to this game, each bullet has to have one of the three primary colours. In order to do that, I added eventlisteners to keypresses on the ‘1’, ‘2’ and ‘3’ buttons on the keyboard. ‘1’ corresponds to red, ‘2’ corresponds to blue and ‘3’ corresponds to yellow. The numerical value itself will then become the colorId of each bullet which is important later on under the collision detection mechanic. The keypresses also changes the current colour string value such that when bullet spawns, they will be the colour that the player selected. The bullet destroys itself when it goes below the playing area to prevent unnecessary usage of resources as it would thus never collide with an enemy. The game state also determines whether mouseclicks

will produce bullets from the player object to prevent bullets from spawning when the game has not started yet.

To do the spawning mechanic, I created a function that dynamically creates enemies using the enemy array. It dynamically creates enemies using a setInterval method such that enemies spawn based on time. Each enemy object also comes with its own properties such as xpos, ypos, xrate, yrate. More importantly, colour is randomized and its respective colorId is attached to the object. Other than the 3 primary colours, additional composite colours formed from mixing the 3 primary colours will spawn as the player successfully progresses through the game. This is done by doing a Math.rand with a range factor to determine the current range of colours. The numerical value is then passed through a switch case function to return the colour string to colour the created enemy.

To do the collision detection mechanic, I created a setInterval function that is attached to both the bullets and the enemy. The bullet detects collision with the enemies whereas the enemy detects collision with the player to determine game over state. Collision is done by measuring the distance of xpos and ypos of each corresponding object. If the xpos and ypos is within each other, a collision occurs and then the corresponding functions gets carried. For the bullet collision, the bullets always gets destroyed even if the enemy colour is not the same. In the event the colour is the same, the enemy gets destroyed as well and increments the score variable. For composite colours, the respective primary colour gets deducted from the colour and the enemy changes to the leftover colour. This is done using a switch case statement for each and every possible combination of bullet colorId and enemy colorId. For example, the purple block (colorId = 4) reacts only when either a red (colorId = 1 / case 1) or blue (colorId = 2 / case 2) bullet hits it. If a red bullet hits the purple block, it turns blue whereas if a blue bullet hits the purple block, it turns red. For the enemy collision, the game over state is called when an enemy touches the player object.

To do the adaptive difficulty mechanic, I used the score variable as an indicator to determine difficulty increases. For every 10 success, the interval of the spawn increases by 10%. This is done by multiplying the existing interval value by 0.9. Also for every 30 success, a new coloured enemy will start spawning up to 3 new colours: purple, green and orange.