Design Overview for Snake Game

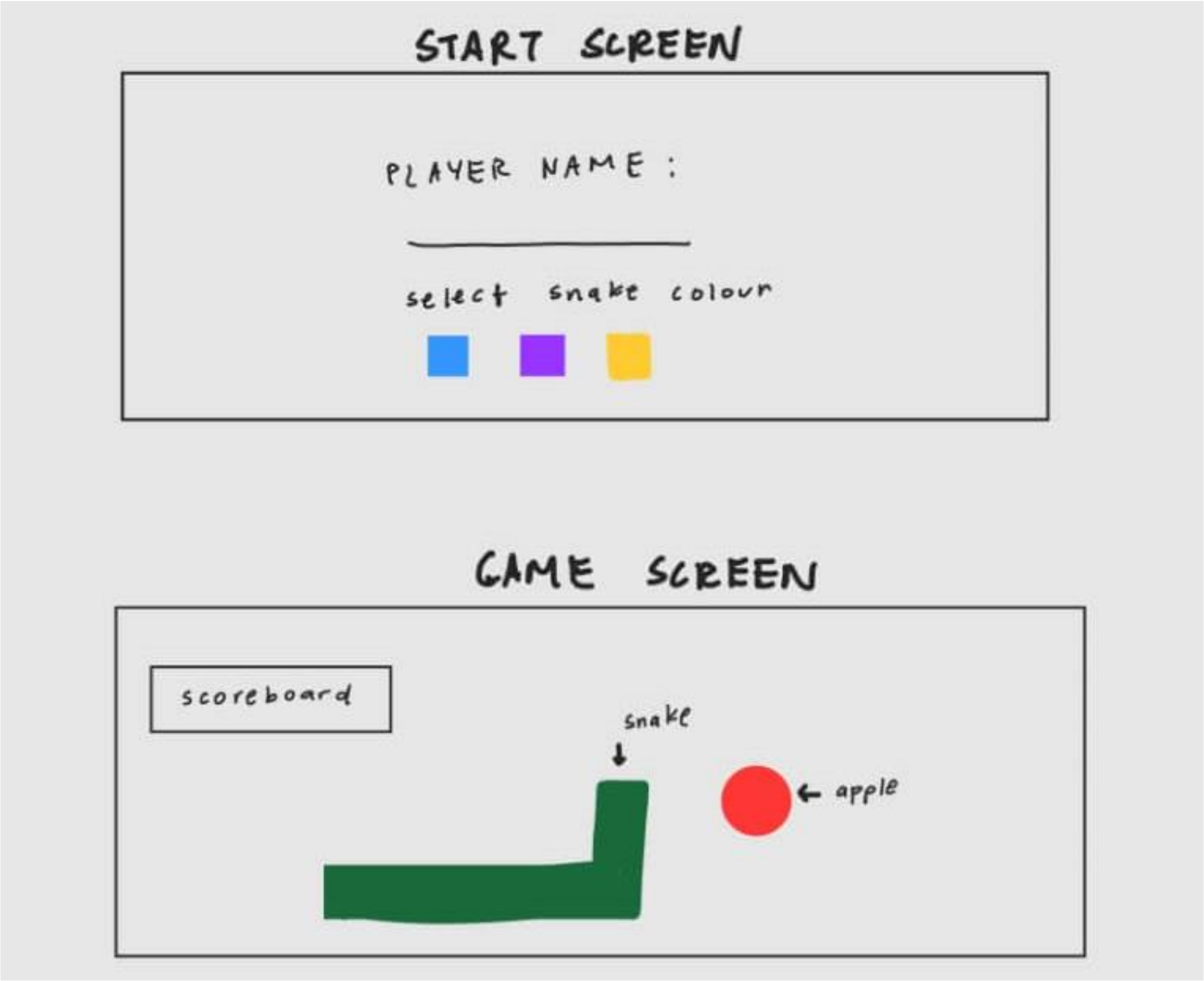
Name: Khor Jing Yin
Student ID: 104491695

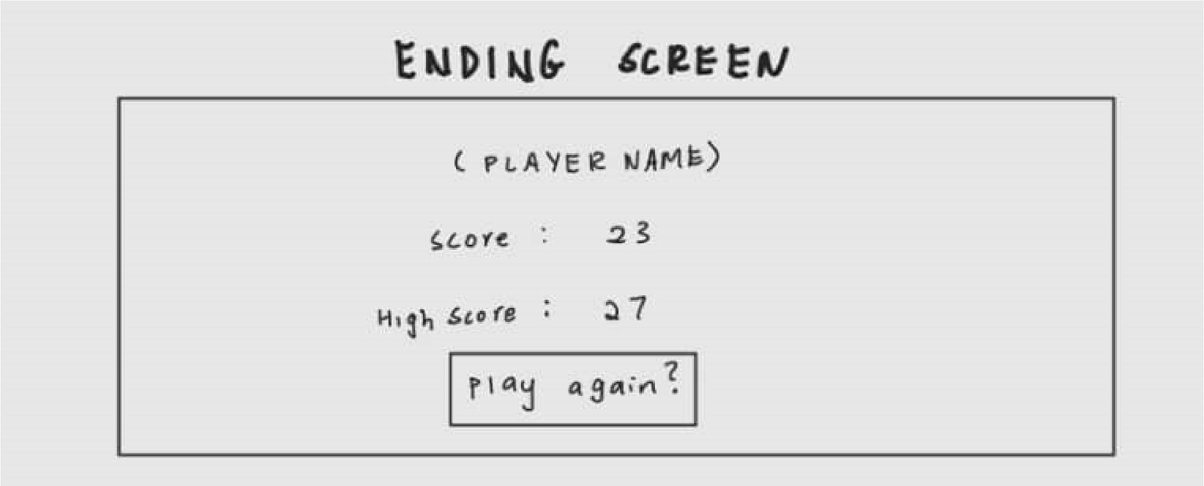
Summary of Program

My program is going to be the classic 'snake' game. Using gosu, I will illustrate a moving snake, whose movement direction can be controlled by the user. Balls will randomly pop up on the screen, and users can eat these to grow the length of their snake. The game ends when the snake collides with itself. The goal of the user is to grow their snake as long as possible.

I will allow users to select the colour of their snake, from a pre-selected list of colours. This will add a bit of customisability to the game for the users. At the end of the game, the users score, as well as their highest score will be displayed for them to see.

To move the snake, I will use the library 'Ruby2D'. This will allow me to get input from the keyboard keys – using the up, down, left and right keys to move the snake.





Required Data Types

Table 1: Snake Control Record

Field Name	Type	Notes
Colour	String	Allow user to select the colour of the snake
Position	Array - Int	This is an array that contains the x and y coordinates for each block making up the snake. Each block is a new array element.
Direction	String	Get input from the keys the user presses. This helps determine the changes to be made to the coordinates in the positions array, to move the snake.
Growing?	Boolean	If true, then add another block to the positions array.

Table 2: Positions Array

Field Name	Type	Notes
X-coordinate	Int	This array is part of the 2D array (positions). This array represents the coordinates for one block.
Y-coordinate	Int	

Table 3: Game Elements Control Record

Field Name	Type	Notes
Ball X-coordinate	Int	X-coordinate of the ball position
Ball Y-coordinate	Int	Y-coordinate of the ball position
Score	Int	Users score that is calculated based on the number of balls they are able to 'eat'
Collision?	Boolean	If colliding with itself, end game. If colliding with the ball, increase snake length and generate a new ball
Game over?	Boolean	Continue game if snake has not collided with itself.

Enumeration for Snake Colour

Value	Notes
Maroon	Default list of colours available for players to choose from for their snake
Teal	
Olive	
Silver	
Orange	

Enumeration for Type of Collision

Value	Notes
Self	When this happens, end the game
Ball	Add one block to snake length, and generate new ball

Overview of Program Structure

Main Functions/Procedures:
There are 3 main functions for my proposed custom program, which are the snake control function, the game control function, and the update procedure.

Snake Control
The snake is made up of an array comprising of the x and y coordinates for every block making up the snake.

1. Snake Grow: This oversees the updating the snake length whenever it collides with the ball, to increase its length. Upon collision with the ball, another set of coordinates will be added to the array, to increase the total length (number of blocks) of the snake.
2. Snake Position: This deals with the x and y coordinates for each block inside the array. First it gets information from 'Snake Direction' to know which coordinates need to be altered. For example, if the snake was moving towards the right, it would be x-coordinate plus 1, and if it was moving downwards, it would be y-coordinate minus 1. This function will also need to remove the last block in the array, and add one to the front, to make the snake 'move'.
3. Snake Direction: Registers input for which direction the user wants to move the snake. To make this function possible, I would need to use the library 'Ruby2D' , so that any keyboard inputs can be registered and used.

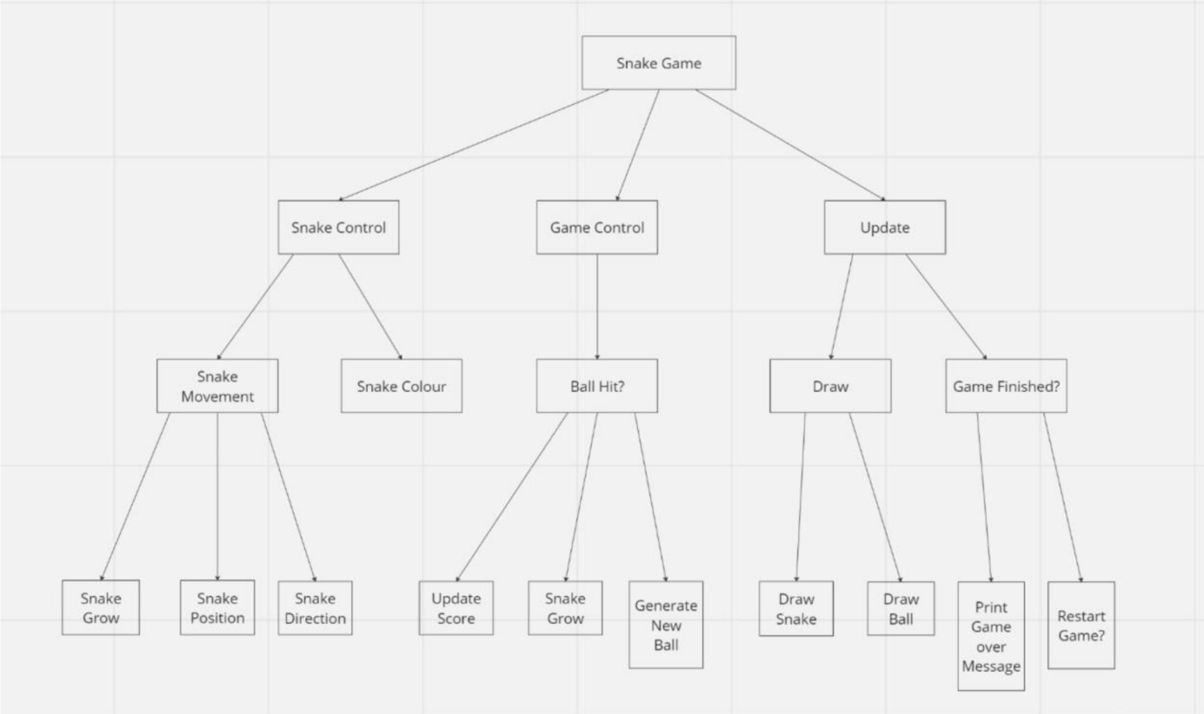
Game Control
The game control function is concerned with checking any collisions between the snake and the ball, and updating any necessary game elements when that occurs.

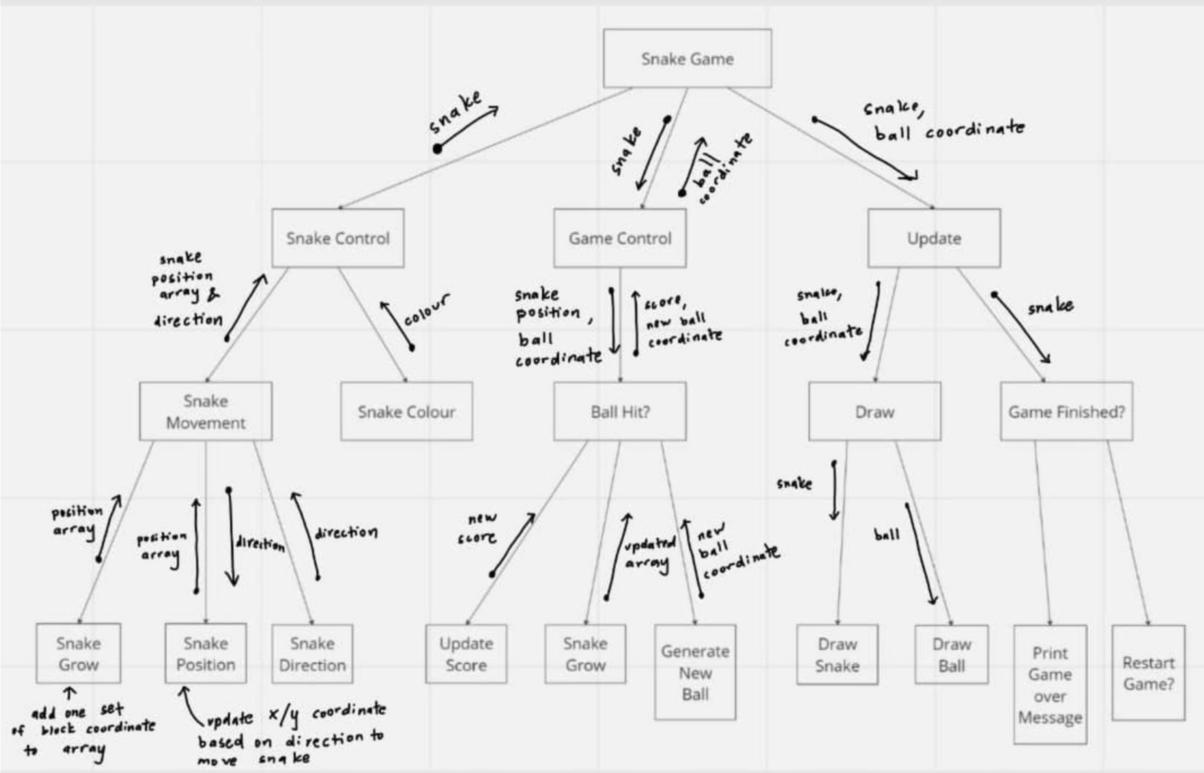
1. Update Score: Adds one to the existing score upon the snake colliding with the ball.
2. Snake Grow: The function will be called upon the snake colliding with the ball
3. Generate new ball: Remove the ball that has been hit, and place a new ball on the game window.

Update
This procedure is meant to constantly iterate the previous functions, so that the snake keeps moving, and to constantly check if any collisions have occurred.

1. Draw: Draw the snake and ball on the game window
2. Game finished?: Checks if snake has collided with itself. If it hasn't, continue looping the procedure, else halt the snake movement and display a message indicating that the user has lost. Prompt user to play another round.

Structure chart


Snake Game
Snake Control
Game Control
Update
Snake Movement
Snake Colour
Ball Hit?
Draw
Game Finished?
Snake Grow
Snake Position
Snake Direction
Update Score
Snake Grow
Generate New Ball
Draw Snake
Draw Ball
Print Game over Message
Restart Game?


Snake Game
snake
snake
ball coordinate
snake, ball coordinate
Snake Control
Game Control
Update
snake position array & direction
colour
snake position, ball coordinate
score, new ball coordinate
snake, ball coordinate
snake
Snake Movement
Snake Colour
Ball Hit?
Draw
Game Finished?
position array
position array
direction
direction
new score
updated array
new ball coordinate
snake
ball
Snake Grow
Snake Position
Snake Direction
Update Score
Snake Grow
Generate New Ball
Draw Snake
Draw Ball
Print Game over Message
Restart Game?
add one set of block coordinate to array
update x/y coordinate based on direction to move snake