MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

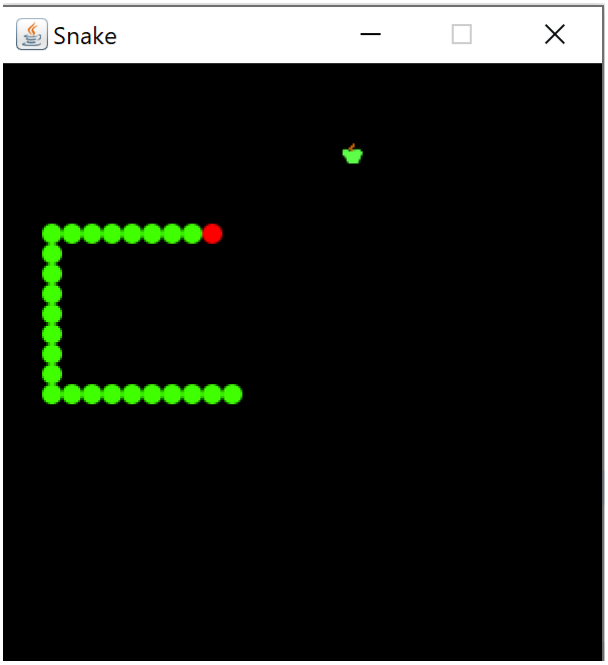
159.261—Games Programming
Assignment 1: Snake Game

Total marks: 100
Course Weighting: 30%

Due Date: *23rd April, 23:59 (China time)*

Assignment Brief:

You are tasked with developing a Java version of the classic game **Snake**[1]. The game will allow a user to control the movement of the snake using the keyboard's cursor controls. The player will initially control a three-circle player object (Snake with red head and green body), which will move one cell at a time on the X or Y axis, representing the game world. As the player consumes the green apples, the snake will in turn grow by 1 (the apple and red and green circles will be provided to you as PNG images).



If the snake collides with itself, then the game is over, and you may prompt the user to play again. If the user consumes an apple, the apple will disappear and the length of the snake will be incremented by one. Apples eaten by the snake will be replaced immediately and are to be placed at random, on another square not currently occupied by the player's snake or another item or obstacle. You can use a grid for the game world, if you wish.

You may or may not choose to use the GameEngine code provided as part of the course.

Snake Movement:

The snake can move into the cells immediately adjacent to itself but it cannot move back on itself. The snake will be moving forward in its current direction at all times.

[1] http://en.wikipedia.org/wiki/Snake_(video_game)

Snake Implementation:

Use an Array or an ArrayList to represent the body of the snake using a maximum of 20 circles (including its head). You will need an extra variable to keep track of the current size of the snake. When the snake moves to a new cell, all the previous snake cells will be shuffled down by one array element. You may define four boolean variables to keep track of the snake movements (leftDirection, rightDirection, upDirection, downDirection).

Your actionPefromed method could look something like this:

```
public void actionPerformed(ActionEvent e) {

    if (inGame) {

        checkAppleLocation();   //if snake has eaten the apple, increase the length of snake body and
                                  recalculate random coordinates for the apple
        checkCollision();       //check collision with snake body or borders of the game and update InGame
        move();                 //move the rest of the body in a loop and then calculate the new x or y for the
                                  snake head, based on user input
    }

    repaint();                  //call paintComponent(Graphics g) method
}
```

You can have a Snake class extending JFrame and initiating an object of a Board class that does the rest. It's up to you if you want to design it differently.

Snake Collisions:

The game will end if the snake collides with itself or with the borders of the game.

Snake Notes:

To implement this game you will need to use (some or all of) these classes to implement the required game functionality.

- JFrame
- JPanel
- ActionListener
- Color
- ImageIO
- Graphics
- KeyEvent
- KeyAdapter

Marking Guide

Make sure your code can run without errors. You will lose some marks, if there are any compilation errors.

Develop and render a game world.

Working Game Architecture Code – 13 Marks
Working Game Code – 5 Marks

Keyboard Input Code – 5 Marks

(Marks 22)

Develop code to respond to user input in the event processing section of the main game loop. The user input should be translated into code to allow the control of the snake player character.

Working Keyboard Input code – 5 Marks
Event Handling and Control of the Player Character – 4 Marks
Working Event Handler Code – 4 Marks
Code to shift all existing snake positions down by one array element – 10 Marks

(Marks 23)

Develop code to allow the snake to grow by one cell when it eats an apple. This code will need to allow the snake to grow by one cell, and will need to ***shift*** all the snake cell positions down by one. The game should compile with no errors or warnings.

Random placement of apples with respect to other game objects – 4.5 Marks
Code to add a new element to the snake array / list – 3 Marks
Working code to add new elements to the end of the snake array – 2.5 Marks
Working code to shift array elements – 10 Marks

(Marks 20)

Additional Features

Produce code for additional features you can think about to enhance the game. Marks will be allocated based on the complexity of the extra features. Examples can include (but not limited to):

Two players playing the snake game. For example, a blue snake is controlled with *"W,A,S,D"* keys and a green snake is controlled with *"Arrow Keys"*
Setting up a start Menu with different options: Play, Help, Quit
If the game ends, the user can be notified of this and prompted to play the game again
Introduce a Game Grid
Introduce four lives for the Snake, so the player is given four opportunities before the game is over
Introduce a food item that will reduce snake's lives, if the snake eats it
Other features you can think of…

(Marks 25)

Presentation

Each student is required to give a class presentation for a maximum of 5 minutes in length. As part of your presentation, cover the design decisions, and the additional features of your game, and a live demo of the game and how to improve it in future. 10% of total marks will be allocated to presentation. A presentation schedule will be released that will have details for your exact presentation day and time.

The presentation must also be submitted on Stream.

(Marks 10)

(Total 100)

Submission Details:

This is an individual assignment. You must work on the individual tasks by yourself and all work on these tasks must be your own. When submitting the work via the Stream, as part of your assessment submission you agree that the work is your work and your work alone. You will be asked to submit via *Turnitin.*

If you are using any online code, please make sure you include the link and acknowledge the source in the comment section.

Your submission should include -

1. the source code
2. a one-page document or a README.txt file explaining how to run the code as well as a list of all additional features you have implemented (if any).
3. presentation file
4. a demo video of the game covering all the features developed (you can upload the video to a cloud server e.g. Tencent cloud and provide a link)