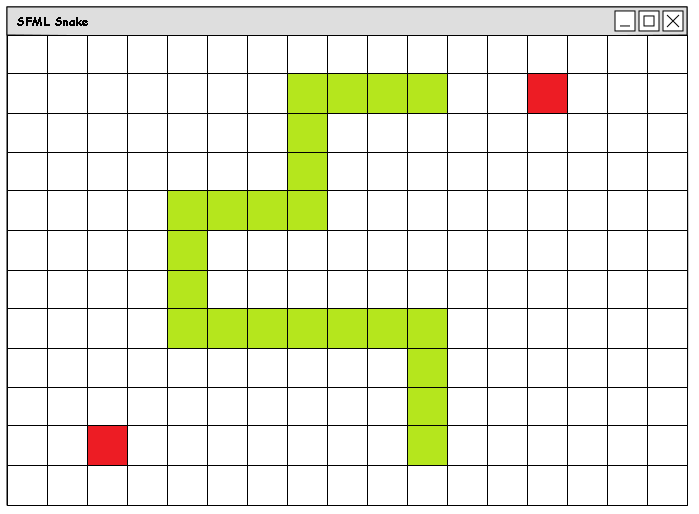
|  |  |
| --- | --- |
|  | **ISCG6442—Game Programming**  **Assignment 1: SFML Snake**  *Total marks: 100*  *Course Weighting: 30%* |

*Due Date: Monday 2nd April, 2018 23:55.*

**Assignment Brief:**

You are tasked with developing an SFML version of the classic game **Snake**[[1]](#footnote-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 single square object, which will move in line with a grid which represents the game world. As the player consumes red squares (apples) the snake will in turn grow.



The game world is defined using a grid of arbitrary size which wraps around on all sides. If the snake collides with itself, then the game is over, and the user will be prompted to play again. If the user, consumes an apple, here denoted by a red square, 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.

**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.

**Snake Implementation:**

Use an array of sf::Vector2i objects to represent the body of the snake using a maximum of 20 elements. You will need an extra variable to keep track of the current size of the snake. When the snake moves to a new grid cell, all the previous snake grid cells will be shuffled down by one array element.

**Snake Collisions:**

The game will end if the snake collides with itself or any obstacles in the environment. If the game ends, the user should be notified of this and prompted to play the game again.

**Snake Notes:**

To implement this game you will need to use these classes to implement the required game functionality.

* sf::Vector2i
* sf::Font
* sf::Text
* sf::RenderWindow
* std::default\_random\_engine
* std::uniform\_int\_distribution
* sf::Event
* sf::Keyboard
* sf::RectangleShape
* sf::Vector2f
* sf::Vertex
* sf::Lines

For more information please see the working prototype on Moodle. Please note that the rules governing direction have not been fully implemented in this prototype and so the snake is able to turn on itself.

# Marking Guide

## Difficulty Level: Low

Produce an Object Orientated game architecture. Develop a grid of lines using sf::Vertex and sf::Line objects. Render the grid of lines using window.display(), having drawn the lines using window.draw() in the main game loop.

~~Object Orientated Game Architecture – 5 Marks~~

Working Game Architecture Code – 5 Marks

~~Game Grid – 5 Marks~~

~~Working Game Grid Code – 5 Marks~~

(Marks 20)

## Difficulty Level: Medium

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.

~~Game Loop – 5 Marks~~

~~Keyboard Input Code – 5 Marks~~

~~Working Keyboard Input code – 5 Marks~~

~~Event Handling and Control of the Player Character – 2.5 Marks~~

~~Working Event Handler Code – 2.5 Marks~~

(Marks 20)

## Difficulty Level: High

Produce code to allow the snake to eat randomly placed apples (grid cells coloured red).

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. New cells will only be added to the front of the snake. The game should compile with no errors or warnings.

~~Random placement of apples with respect to other game objects – 2.5 Marks~~

~~Correct use of <random> standard template library – 2.5 Marks~~

~~Code to add a new element to the snake array / list – 2.5 Marks~~

~~Working code to add new elements to the end of the snake array – 2.5 Marks~~

~~Code to shift all existing snake positions down by one array element – 5 Marks~~

~~Working code to shift array elements – 5 Marks~~

(Marks 20)

## Difficulty Level: Higher

Produce code to allow two players to play the snake game. The game compiles and runs, is free from errors and compilation warnings, the win/lose conditions are checked correctly. The program is a game. The game is fully functioning and works.

Keyboard Input for Second Player – 2.5 Marks

Working Code to handle keyboard input – 2.5 Marks

Event Handling and Control of the Second Player – 2.5 Marks

Working Code to handle player control – 2.5 Marks

Independent data structures to keep track of the position of the segments that makeup the second snake – 10 Marks

Working Code to implement separate data structures for second player – 10 Marks

No errors or compilation errors – 5 Marks

Win/Lose conditions defined correctly – 5 Marks

(Marks 40)

(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 Moodle as part of your assessment submission you agree that the work is your work and your work alone.

This is an individual assignment. You must work on the individual tasks by yourself and all work on these tasks must be your own.

Please sign the statement below to declare that this assignment submission is your own work and hand in the signed statement with your assignment. Failure to sign and include this statement may mean your assignment is not marked.

|  |
| --- |
| **ISCG6442—Game Programming**  Assignment 1  I declare that the individual part of this assignment submission is my own work. Where I have incorporated work by other people, I have correctly acknowledged the source in my assignment.  Student Name …Shivneel Achari... Student ID ….1463570…  Date: …1/04/18…. |

1. <http://en.wikipedia.org/wiki/Snake_(video_game)> [↑](#footnote-ref-1)