# **Snake Game**

**Myles Coleman**

|  |  |
| --- | --- |
| **Action** | **Input** |
| **Upwards Movement** | **W** |
| **Left Movement** | **A** |
| **Downwards Movement** | **S** |
| **Right Movement** | **D** |
| **Play/Retry** | **Space Bar** |
| **Quit** | **Esc** |

## User Guide

## Analysing the Requirements

For this project it was up to us to question our lecturer to help determine what was required for the project. Looking at the specification there was quite a lot of room for interpretation. This allowed to tackle the project in our own creative ways. However, I didn’t stray too far from the specification myself – with how I interpreted it.

* My snake was trapped in a tank,
* The snake was continually moving in its facing direction in steps the size of its head,
* The player could only control the snake’s direction,
* There are two snakes’: a player and an AI,
* Collectable food appears randomly, with a limit of 5, occupying their own location,
* When the collectables are collected the snake’s tail grows by the specified amount – and the snake gains score,
* The snake must surface for air periodically, losing segments and score if it fails to,
* If the snake goes more than one step above the water, it also loses tail segments and score,
* Collisions with the tank walls, itself or other snakes result in death,
* A snake with no segments dies,
* The water level in the tank falls uniformly across ninety seconds,
* And the winner is the surviving snake with the highest score,

Nevertheless, I did take opportunity to inject my own creativity in the areas that were left entirely unspecified. For instance:

* I added a particle system to demonstrate the leaking water,
* Snake’s falling on death so that other snakes can eat them,
* And the fruit growing up to its random location.

## UML Class Diagram

## Reflection

Beyond sharpening my C++ problem-solving skills, this project provided a solid foundation in SFML. This is thanks to the many features I tried to utilise: shapes, sprites, fonts and a particle system.

Whilst I have learned a great deal due to successes, my shortcomings have also informed me for future projects. Throughout this project I never really followed a plan, simply implementing whatever was next asked of. But this caused some problems, namely having to replace the second player with an AI and having to redo the collectable spawning logic to be more memory efficient. But these are just the problems I was able to fix. I fear my code isn’t formatted nor optimised to the best of my ability due to running out of time and rushing ahead without a plan.

Rushing ahead without a plan however didn’t prove to be a total disaster, based on a lecture I implemented an observer pattern for my keyboard inputs. This proved extremely challenging, yet extremely informative. And I was clearly not the only person who saw this challenge, when it came time for the next lecture, I learned my lecturer had ditched this implementation in favour of a simpler method. Whilst this simpler method would have done the trick, it wouldn’t have furthered my studies and informed me in anyway.

So ultimately, I am satisfied with my progress during this project.