Games Programming ICA 2 Report

"Something Like Snake"

The aim of this project was to demonstrate my ability to identify and implement appropriate classes within a game context, and to demonstrate correct use of inheritance, polymorphism, aggregation and composition within this context, as well to demonstrate my ability to use both the Standard Template Library and my own implementation of lists and other such structures if appropriate.

This report will therefore, talk about the problems that led to the choices I have taken in this implementation, as well as any places where I could have improved my program further, or problems I was unable to solve.

Implementation Choices

Snake - Controller Model

I chose to use the Snake-Controller model as opposed to the Snake-AISnake/PlayerSnake model to make changing the behaviour of the snake easier. The Snake-Controller model allows the snake behaviour to be adjusted on the fly, and could allow for scaling AI sets, or different collision or movement rules depending on situations.

Pausing (Space key)

My implementation of pause changes the snake's controller, directing it to move around the edge of the screen, as well as overriding the collision rules so that it doesn't collide with any obstacles. The spawn cycle for items is also paused so that the map doesn't fill with blocks, or warp powerups while the game is paused (the unlimited nature of blocks and warp powerups being a deliberate mechanic due to the dual benefit-hazard nature of an item that can move you into a wall). Collectibles are still consumed if touched however.

Keyboard Control

All keyboard control is managed via the the key namespace and each state referring to it. I chose to use a namespace because of the way I would be using my inputs across all states and this would make it easily remappable across them. I ran out of time during the remapping process however.

Item Controller

For dealing with spawning different kinds of food and powerups, I created an Item Controller, overseeing the generation of all items on the map. This allowed me to create items bound by limits (like the 5 food limit) and different production rates. I chose to do the spawn check during the timer/motion cycle for the sake of speed, as well as to avoid having to use insanely large numbers in my rand() range to avoid item flooding.

Collision Map

To manage collisions I used the STL map template, which holds both a Tile and that tile's location, contained in an x, y pair. This allowed me to keep track of as many or as few objects as I needed, as well as access them, move them or remove them with ease. I used a separate map for the snake and the items to allow for the potential of items or terrain features that could be passed through.

Things I have learned from this project

During the course of this project I have became much more familiar with the STL, and specifically the map and multimap templates, as well as gained a greater understanding of correct linked list implementation. I also became much more comfortable with pointer manipulation.

Potential for expansion

If I were to expand this project I would add controls remapping, more video settings, and stage management.