AC22005 CW1 Broad Design Document

Outline:

The goal of this document is to describe the production plan of the 1st coursework assignment for the Architecture and Operating Systems course for team 3 (Aleksander Barczak, Flynn Henderson and Benjamin Houghton).

The goal as set out by the assessment brief describes that we are to produce a 2D tile-based game using the Microsoft Visual Studio IDE with its built-in features of creating GUIs and to use C# to develop the functionality of the game alongside generating the grid of tiles to use within the game.

The broad description of the game we are proposing is an amalgamation of Snake and Pac-Man, drawing on elements from both games.

Within the game you will control a snake trapped in a maze with several ghosts (possibly change this to rats for thematic purpose?), the victory condition is to have the snake consume all the rats and the loss condition is the death of the snake.

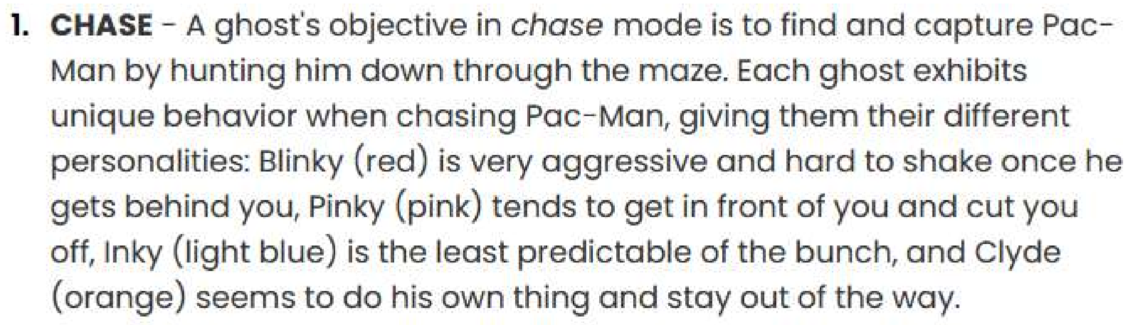
Given a tile grid the maze is the first part of the game to develop, walls and corridors are to each be 1 tile thick and will be designed by hand (possibly extended to computer generation if time permits). The map will be toroidal in traversal, meaning any gaps in the side walls will transport the snake or ghost to the other side of the maze.

The snake is to travel in a straight line until it hits a wall, or the user instructs it to turn at a junction. If the snake hits a wall and the user does not instruct it to move in any direction then the snake will randomly pick a direction of those available, meaning it will carry on as expected in corners and flip a coin to pick a direction at T-junctions.

If the snake's head comes into contact with a ghost, the ghost dies and the snake grows in size, if the snakes head comes into contact with its own body the snake will bite itself at that point, destroying the section of tail that was there, and splitting the snake body in two separate parts which will now both be snakes, if either of the new snakes made is less than 5 tiles long, that snake will die.

This idea stands to be the main gimmick, controlling multiple bodies as best as you can to not make mistakes with them and is subject to removal based on the difficulty of implementation or influence on the fun of the game.

The ghosts (or rats), can start off as being implemented as the four ghosts from Pac-Man, their behaviors are widely documented by many sources like the one shown here:



1. **CHASE** - A ghost's objective in *chase* mode is to find and capture Pac-Man by hunting him down through the maze. Each ghost exhibits unique behavior when chasing Pac-Man, giving them their different personalities: Blinky (red) is very aggressive and hard to shake once he gets behind you, Pinky (pink) tends to get in front of you and cut you off, Inky (light blue) is the least predictable of the bunch, and Clyde (orange) seems to do his own thing and stay out of the way.

https://www.gamedeveloper.com/design/the-pac-man-dossier#close-modal

if a ghost comes into contact with the snakes body it will split the snake in two and the same rules will apply as when the snake was bitten by itself, splitting the snake in two also gives the bonus of giving the snake a better chance of getting away after suffering damage, a sort of health system if you like.

That describes the core elements for the most part, further details will be added as time continues to pass.

In terms of implementation, upon starting the game loop, we'll generate and store a two-dimensional array of form objects representing the tiles on the game grid where the indexes correspond to the x and y positions of the tiles on the screen, to allow easy access to each tile at a particular position.

Ideally, we should store the details of the maze's walls and corridors in the form objects themselves, this could possibly be done using differing form object types for walls and corridors, such as the difference between buttons and labels, alternatively we could extend one of the form object classes into our own class that has all the additional data fields we need. (or reuse something no longer to be used, i.e. hide the text field but have its value set to "wall" or "corridor" for easy checking)

When the maze is in place and displaying, we can draw the snake and ghosts on top of it, the snake and ghosts' position could be stored differently from the wall, we could simply store the absolute position of each and then access the tile at that position and draw over it to indicate it is now a snake/ghost.

For the most part, that just leaves taking user input, since the program is intended to be asynchronous, we could have a main loop that runs as soon as the game loads up run asynchronously to a thread dedicated to taking user inputs, this can easily be done as the forms are asynchronous in terms of pressing on the ui with the mouse, meaning that we can easily start of implementation using the player’s mouse to indicate the direction to head and then move on to also running a thread collecting user keyboard I/O to make the experience more pleasant.

That’ll conclude the first version of the design doc, a little vague in some areas to leave room for discussion and improvement but accurate enough to define the direction we are headed.