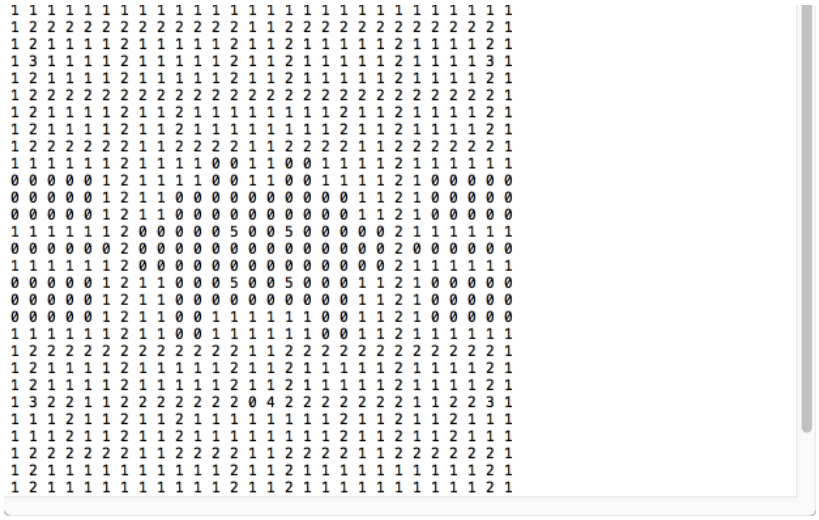
Quinn Ramsay
301141921

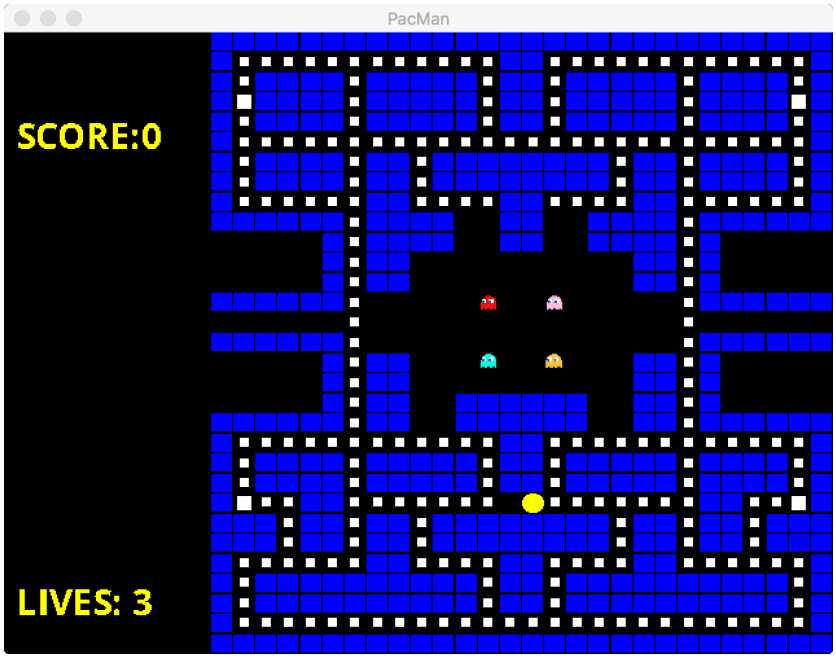
CMPT 365 Project Report

For my project I created a Pacman game using C++ and SDL 2.0, Simple DirectMedia Layer. SDL is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. It is commonly used for 2D game programming, emulators and video-playback applications. My original project proposal was to implement a small interactive game. My idea was to have either a battle type game or implement a remake of Pacman. I would design the project either using C++ and OpenGL/SDL or C# and Unity depending on which game I chose to build. I chose to use C++ and SDL instead of C++ and OpenGL or C# and Unity because it represented the middle ground for simplicity, control and depth of learning. OpenGL would have been a doable project but it is too extensive and complicated for the project, requiring many, many lines of code. Unity was the opposite; as it is a game engine, it includes many items that I would not have learned without doing the project from scratch in SDL. It also would include much less code in their scripts, as a large amount of the abilities are built-in to Unity.

I originally wished to have a colour look-up table that would allow the user to switch colour display modes, but as the ghosts and Pacman are loaded from .jpg and .png images I couldn't determine how to modify the image data to use with a colour look-up table. The block and text colours are chosen by the program, however I thought my efforts would be better used elsewhere instead of including colour changes of just the wall blocks. I did manage to include score and lives in the project as well as user movement for Pac and AI movement for the ghosts. I also originally wanted the project to include basic animations, transitions between menu and scenes, sound effects, and a simple working GUI, however these all were out of the scope of my project. My project includes 5 classes GameBody, Map, Ghost, Pac and LTexture.

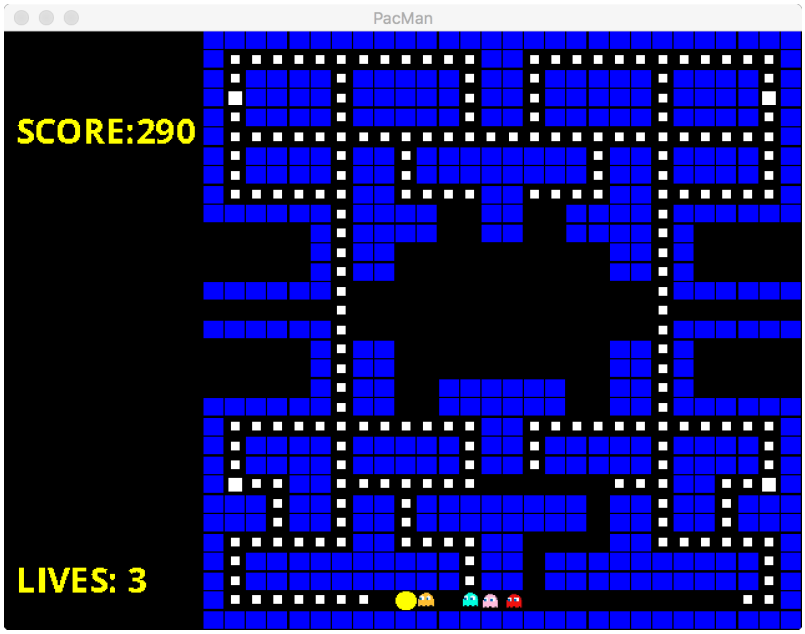
GameBody includes all the information about the object bodies including: tile dimensions, coordinates in index form due to tiling, and object types (floor, wall, points, eatems, ghost and Pac). Each GameBody is reacted with differently; Pac and Ghosts cannot pass through walls but can pass along floor, points, and eatems as well as ghosts can pass through ghosts. When Pac passes over points and eatems (the larger dot that causes Pacman to be allowed to eat the ghosts) he picks them up and when Pac encounters ghosts either he eats them if he is in eatemstate, or he dies. It also includes functions to set and get its properties as well as a function to detect collisions with other GameBodys.

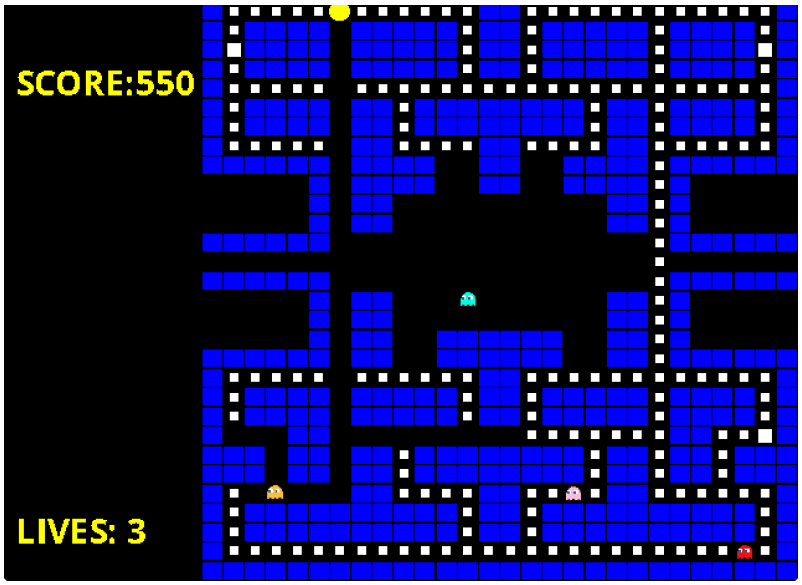




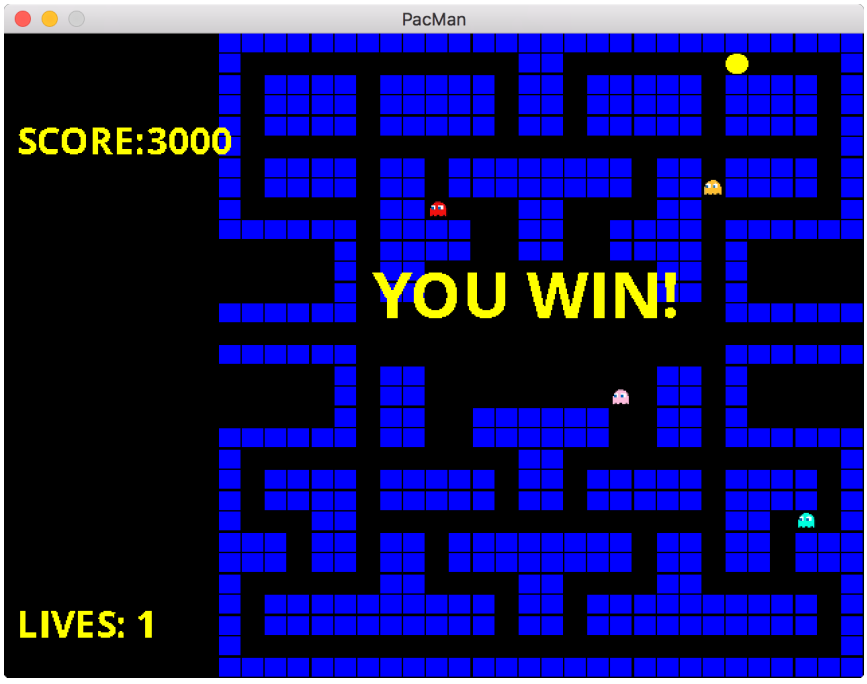
Map consists of a grid of GameBodys that it holds in a 2D vector grid. It loads this grid from file where the first two numbers on their own line represent the dimensions of the map, and the remaining grid of numbers represent objects. 0 represents a floor space, 1 is a wall space, 2 is a point space, 3 is a eatems space, 4 is the start location of Pacman, and 5 is the start location of each of the ghosts. The above images illustrate the loading file and the results. Map holds the point count, which is the number of points/eatems left on the map. When point count runs down to 0, then the game is over and you have won. It also holds the tile dimensions along with offset of the map that allow grid indexes to be transformed into screen coordinates, as well as the ghost and Pacman start coordinates for when their positions need to be reset. It includes functions to draw the map grid, check whether a tile is movable, take objects (like points and eatems on pickup), reset the map for game resets and get and set functions for various properties.

The Ghost class consists of a GameBody, LTexture to display the image, stop counter to determine how long to pause before moving again, a flag to determine whether it is running away, last direction it moved, and sprite clips (rectangle coordinates) to determine the clipping of the image combined with an internal Ghost number to determine which clipped ghost image to display. In terms of functions Ghost has functions to draw the ghost, move the ghost, kill the ghost, get its GameBody, set its stop time, set its run away flag, update the ghost and reset the ghost upon game reset. The function to draw the image utilizes clipping to determine the correct section of the image to display. As can be seen above, ghost number 0 selects the first top left red ghost and 1 selects the pink etc.



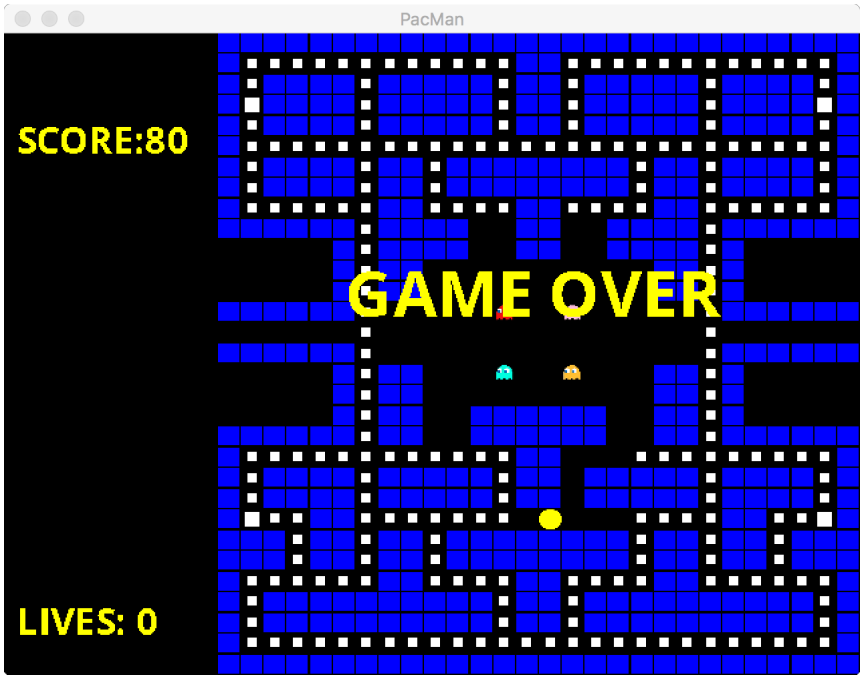


The moving function is the most complex function in the ghost class. It utilizes a basic AI and the run away flag to determine whether to run towards or away from the player object. The moving function ranks the directions of movement based on the dot product between the difference in position of the ghosts and the chosen direction of travel (North, West, South, East or None). If the ghost wants to move towards the player then it ranks the direction that maximizes the dot product as first, and so on. Moving away from the player flips the ranking around. The function then tests these new positions by ranking with reference to the map whether they are eligible moves. If yes then the highest-ranking new direction is chosen. To eliminate the perfection of the ghost in chasing down its target, I gave it a 25% probability of choosing a random direction that will slow it accordingly. At the start of the game the ghosts have an initial countdown of 5 before they move. Upon killing the ghost, the ghost returns to its starting position and sets its stop counter to countdown from 30 before it can move again. The stop time function is only used when the ghost kills Pacman and he needs to reset their initial stop timers and its run away function is set when Pacman picks up an eatem object. The still images above show the ghosts running towards the character, as well as away from the character in addition one of the ghosts was eaten.



Pacman is the main character class which holds your lives, eatemstate flag, score, count until eatemstate runs off, last direction moved, GameBody, and LTextures to display the score, lives, Pacman image and Win/Game Over messages. In terms of functions there are two moving Pacman functions, a function to pickup GameBodys from the map, functions to draw Pacman as well as his score, lives, and game over strings, get his eatemstate, update him, interact with ghosts, kill Pacman, and reset Pacman upon game reset.

The default moving function attempts to move Pacman in the direction of previous travel and the directional move function take a new direction from user input and sets his new last direction and motion as well. The function to pick up objects interacts with the map to picks up items. If it is a point object then Pacman gets an additional 10 score, if it is an eatem object Pacman gets 150 score and sets his eatemstate to active. The Draw Win function displays "You Win!" to be called when all points are gathered, and Draw Game Over displays "Game Over" when lives are 0. The function to interact with ghosts checks for collisions with ghosts, and depending on his eatemstate either kills Pacman or kills the ghost. If Pac is killed then he and the ghosts get sent back to their starting positions with Pacman losing a life. The LTexture class was borrowed and modified from a Lazy Foo Productions tutorial on SDL graphics with C++. It holds the hardware texture to render to the window. The texture can be loaded by file or as a string of rendered text using True Type Fonts and sizing. For all my text I used OpenSansBold.ttf at both size 28 for score and lives or 48 for Game Over and Win announcements.



In addition to the movement input I included user input to pause the game using 'P' and reset the game using 'R' which can be called at any time. I enjoyed working on the project and for the future, I wouldn't mind making the game independent of frame rate, as currently it executes with each frame on 200 ms guaranteed rest, as well as sound effects.