**Anthony Sturdy**

**Fundamentals of Game and Graphical Systems Development**

**Semester 1**

**Pacman**

The Pacman monster can move in four directions and collect the two munchies.

Both munchies will randomly choose a position once it has been collected. The cherry gives 5 points once collected, the other gives 1 point when collected, but there are 20 of the 1 point munchies. The 1 point munchies start in the same place but choose a random position once the player collects them for the first time.

If the Pacman monster hits the edge of the screen, it will wrap to the other edge of the screen on both X and Y axis.

Once you’ve chosen a game after the first screen, you will have to click the game to make it active, otherwise no input will work.

Controls –

* WASD: Movement

**Dungeon Ninja**

Dungeon Ninja is the other game I made for the submission, it’s a simple game in one room where the objective is to kill as many slime enemies as possible.

The game is drawn to appear 3D, but is just multiple 2D sprites drawn one above another to give the 3D effect. I used a software called MagicaVoxel to create voxel models, then a software called Vox2png to convert the voxel models to multiple .PNG files, which are then loaded by the game and drawn one above the other to draw the models. The drawing order of each sprite is decided by the Y position of the sprite. Sprites drawn on the bottom of screen are drawn first, then the higher the sprite the later it’s drawn.

The player can move in 8 directions relative to the camera’s rotation (Can move in any direction if the camera is rotated) and attack. The player uses two types of collision detection. One is based on what level tile the player is currently in, if they walk into a wall tile it sends them in the opposite direction the same amount meaning they stop. The other type of collision uses a radius to the centre of a tile, which means if the player gets too close to the centre of the tile, it does the same thing and stops the player (this is used for the pillars).

The enemies are very simple, they move towards the player and attack when they’re close enough, they do one damage to the player per attack and die in 1 hit. The player gains one point towards their score per kill.

The game has 4 game states. It has a Main Menu, Gameplay, Paused and Game Over/Leaderboard game state.

The Game Over/Leaderboard game state has an online leaderboard using a free service called dreamlo, which lets me get all the scores from the leaderboard and post new scores to the leaderboard using only HTTP GET requests. I used libcurl for the GET requests. The game over screen allows the player to enter their username and tells them where on the leaderboard they will place based on their score. The score is then submitted once the player presses Enter to go back to the main menu. I used a snippet of code from one of the libcurl C++ examples in the leaderboard class, which is what libcurl uses to write the data from the HTTP GET request. I’ve linked the source to the code snippet above the function.

I added a custom text renderer to the game which allows me to use different fonts as S2D doesn’t support other fonts. I did this by downloading a font spritesheet, the program loops through the provided string and renders a specific position of the spritesheet depending on the current character it needs to render. I downloaded my fonts from opengameart.org. (https://opengameart.org/content/the-collection-of-8-bit-fonts-for-grafx2-r2)

Once you’ve chosen a game after the first screen, you will have to click the game to make it active, otherwise no input will work.

Controls –

* WASD: Movement
* Left/Right Arrow Keys: Camera Rotation
* Space: Attack
* Enter: Submit high score (On game over/leaderboard screen)