PRISON ESCAPE

Julia Acosta
Miguel Barrales
Elijah Davis
Mario Hernandez
Miguel Trigueros
Requirements:

Description-

Prison Escape is a 2D game top view that challenges the player to try to escape prison. The game is a one player game. The prisoner is equipped with weapons and a map to try to maneuver its way through out the game. The challenging part is collecting the keys while running away from the security guards. Prison Escape starts with a Menu screen, the options are “Play Game” or “Quit Game”. In order to begin the game, the player clicks “Play Game” and to end the game, the player clicks “Quit Game”. The player is going to start at Prison Cell 3 and in order to move the player around the arrow keys are used to move the player up, down, left, and right. The space bar is used to shoot the guards in order to kill them.

Objective-

The objective of the game is trying to escape prison by collecting the keys that open certain doors and trying to go through the prison without getting caught by the security guards. To win the game, the player needs to open the door in the room between Tower 2 and 4.
Design:
We all decided to make the game into a grid layout so when the player enters through a door, the player is at a different part of the map. The game will only show one room at a time and we designed the game so the player has to enter/exit through the doors in order to see the other rooms.

Before-

At the beginning, we only had a basic menu and for the game we used a box for the player and enemies.

After-

For the final project, we added a picture for the menu, a background for the floor, and images for the characters.
Components of the game-

We equipped the player with weapons and a map in order to escape prison. In order to use the weapons, the player needs to press the numbers 1-5. The player has a red heath bar at the top left of the screen and the security guards health bar are located on the top of them. When shooting the guards, their health will decrease and eventually die.

![Prison Escape Map](image)

Weapons:
- Knife
- Shield
- Crowbar
- Pistol
- Shotgun
- Rifle
- Taser
Game play Flow Diagram

Start

Menu
Play
Exit

Exit

Load Game

Pause

Winner

Return to Game
Check Inventory
Change Settings
Return to Main Menu
Enemies

The enemies were at first created using the same structure as the player. This evolved by producing multiple enemies which wasn’t that hard. The biggest difficulty I had was the fact that the enemies would not be tile dependent, which meant that when the player moved tiles, the enemies would still be where they last were. I achieved this by making the enemies into a triple array instead of a single array. I now look back and could have used a vector to achieve the same thing and it would’ve probably been easier but I was still able to get the enemies to be tile dependent. Enemies also do not reset every time you exit the tile; they stay dead until the player dies in which they are reset. Player detection was not difficult, I take the distance between the player and the enemies and if that is within a threshold the enemies moves towards you otherwise they continue on their way. This also allows for the effect of getting away as when the player gets far enough the enemies can no longer detect him and they may change directions away from the player. Most of my functionality is called within my render function for the enemies, my other functions mostly take in the pointer to game and also take in the current map position and which enemy we are currently checking in terms of
movement, object collision, and player detection. I decided to make the H key activate hard mode which changes the way that the enemies look and also makes them faster and stronger. We originally had the idea of giving the enemies weapons but that was scrapped as the enemies are already powerful enough and the game is not easily beatable. In hard mode it would have made the game impossible. Throughout the process the biggest bug I had was that enemies wouldn’t initialize in one of the map tiles which ended up being the position being out of bounds. I fixed it by just adding 1 to all positions but I now realize that I could have also made the tile start in the middle of my array instead of at the beginning in case we wanted to expand the current map in the -x direction more. It wouldn’t be a difficult thing to add if I decided too but it would require scaling back the enemies in terms of health and speed. In their final form enemies have normal and hard mode. Normal mode gives enemies are blue, have a slower speed, and a smaller range. Hard mode the enemies are turned red, their speed is increased, and the enemies can detect you from farther away. Below are pictures of the two enemy modes.

Normal Mode Enemy
Hard Mode Enemy

Weapons

As a group we first decided to plan on which weapon we wanted to implement into the game. The weapons had to follow are prison scene which the game was place in. The weapons we chose were knife, crowbar, riot shield, pistol, rifle, shotgun, and taser. The weapons were first developed as a single square that was able to shot one particle at time. During later stages of developing the weapon more weapons were add as more shapes using declare object. Then keys for each weapon were implement to the game like 1,2,3,4,5,6,7. The controls for the game are arrow up, arrow down, arrow right, arrow left, and space bar. The arrow keys are for the player’s movement, and space bar is to shoot. The biggest difficulty I had was creating the functionally with the space and the weapon shooting ability. For example, the spacebar had to features to it when it was held it was keep shooting game->n particles in rapid movement and when it was pressed it would only shoot once. I had really difficult time trying to figure out how to shoot only one particle for single shot weapons like shotgun and pistol. It was
fix by using bool and changing game->n particle ability to make particles. For the shotgun the particle were just made more particles by initialize and rendering to center of the shotgun. The weapon movement was created by layering the weapon on top of the player. The weapon was layered by rendering the weapon before the player. The movement was develop by making weapon center equal to player center. In each case if the movement key or the arrow keys were press the center of weapon would have to be changed to equal new center of player. Another feature that had to be change was the size of weapon with the arrow keys to match direction the player face, so the fix that issue the weapon had to change width and height to match the each direction. The melee function for melee weapons such as knife, crowbar, and riot shield was created by changing center and adding a count and reset method for the animation of the melee. After the functionality of weapon were developed textures for weapons and particle were made. I had to make my own texture of top down 2d weapons because I couldn’t find any open source ones expect for 2 the knife and pistol. For my special key function I decide to use the key z. It would make the particle shot out of gun turn to into a texture which would be the kappa face. It would difficult to
make the functionality because I would have to add the texture to velocity and the coordinates of the particle.
Doors-Keys

The keys in the game are scattered around at predetermined locations and are required to progress farther in the game. The keys are implemented as quads with a texture drawn over the top and given a distinctive color to indicated different keys. Once collected the keys are stored as a boolean array and then used to decide if the doors should be opened; with 5 different keys and 3 doors, 1 door requiring 3 keys to finish the game. The doors are implemented much like the walls such that they are stored in an object array that holds all the shapes to be drawn on the current tile. The doors are checked against the key required to open and once the key has been collected the door opens.
UI Elements

One of the things that is vital to a game aside from gameplay and controls is the UI. The first thing that I worked on was to create a main menu, which would become the basis for my pause menu by establishing button checks, drawing objects and text to the screen to represent buttons. Once the main menu button properly functioned by taking the player to the game and stopping the program, it helped set up a precedence for utilizing game states for the pause menu, settings, and even the map. One bug I later encountered with game states is that combined with key checks for the space bar to move to the settings menu from the pause menu changes triggers changing the bool variables for setting settings on and off. As a result, upon entering the settings menu, it turned the hudDisplay setting to off when by default it's on when true upon starting the code as it counted the space bar check to the menu as the check to change the setting on or off. Another notable bug with UI was that textures have to be reset by either unbinding or setting the color to white given the nature of opengl textures. The pause menu displays the current weapon equipped and keys picked up by the player, change the UI settings, and return to the game or main menu.
Collision

Collision with objects is done by taking the edges of the quad each object is contained within and comparing them against the edges of another object and based on the objects that are colliding different code is executed such that the player can't walk through walls, bullets don't travel through players and enemies, and that characters can collide with each other. This allows for our maps to contain the characters and force our pathing on the player. This concept is also used to determine if the keys used for doors have been collected.