Algorithm and Programming

Final Project Report

Name of Lecturer: Jude Joseph Lamug Martinez, MCS

Made by:

Orlando Jonathan Padiman (2702337615)

BINUS UNIVERSITY INTERNATIONAL

JAKARTA

2024

TABLE OF CONTENTS

Chapter 1

PROJECT SPECIFICATIONS

1.1. Project Description

For my final project, I attempted to recreate the first “Five Nights at Freddy” game by Scott Cawthon. Initially, I had my own concept for unique character and game assets, but after experiencing several creative blocks, I decided to use the assets from the original game instead.

The player is situated within a room/office in the middle of the night with two doors to defend themselves from the oncoming animatronics for the next 6 hours. These animatronics will be roaming around the building, slowly getting closer to the room where the player is. The player can track the position of the entities using a camera system which allows them to access the many CCTV cameras around the building. The player can defend themselves by closing the doors when the entities are right outside their room. However, the doors cannot be left shut for too long, as it will drain the building’s power. When the building’s power runs out, it’s game over for the player, as they will be left defenseless.

1.2. Project Link

The GitHub Repository of the project can be accessed through the link provided below:

GitHub Repository

1.3. Essential Algorithm

1. Enemy Movement AI:
    - The enemy has several important variables in deciding their movement:
        - AI Level
        - Movement Opportunity Value
        - Movement Opportunity Intervals
        - Movement Pattern List
        - Room Index

- When the AI's movement opportunity appears, they will randomly generate a number between 1-20. This number is stored as the Movement Opportunity Value.
- This movement opportunity value is then compared to the enemy's current AI Level.
    - If the AI Level is greater than or equal to the Movement Opportunity Value, they will succeed the Movement Opportunity.
    - If the AI Level is less than the Movement Opportunity Value, they will fail the Movement Opportunity.
- If the Enemy succeeds with their Movement Opportunity, the game will take its current position from that enemy's Movement Pattern List and move them one room forward. Else, they will stay in place and wait for the next Movement Opportunity.
- When the enemy is one room before the final room in the list, which is the office. They will appear in the player's left or right door, depending on their assigned doors.
- At this state, what happens next depends on the current condition of the door in front of them. In the event that the enemy succeeds their Movement Opportunity:
    - If the doors are open, they will enter the office and kill the player on their next successful Movement Opportunity.
    - If the doors are closed, they will be forced to go back to the starting position in their Movement Pattern List.
- The enemy's AI Level is increased by 1 for every in-game hour that passes (1 minute in real life). This is done to make it feel like the game progresses in difficulty as time goes on.

2. Power Drain Algorithm:
    - The player starts the night with 100% Power.
    - Some important properties that the Power System has include:
        - Power Drain Interval

        - Power Usage Level
    - The power drain interval is made into a custom PyGame user event with a timer of 7 seconds so that, when checking for PyGame events, it will detect the power drain event every 7 seconds.
    - The Power Usage Level depends on how many utilities the player is currently using. This Usage Level can alternate between 1-4.
    - Several factors that affect the power usage level include:
        - Camera
        - Doors
    - While the player is using one of these utilities, it will add 1 to the usage level. An example of this system in work would look like this:
        - Office at Neutral State → Current Usage Level: 1
        - Camera On → Current Usage Level: 2
        - Door Close → Current Usage Level: 3
        - Camera Off → Current Usage Level: 2
        - Door Open → Current Usage Level: 1
    - Every time the Power Drain event occurs, it will subtract the current power the user has with the usage level.
    - When the power reaches 0% or less, the game will be sent to a blackout state which will eventually bring the player to the game over condition.
3. Camera System Algorithm:
    - The user can toggle in and out of the cameras with the ‘S’ keys on their keyboard.
    - When the camera is up, the player can use the keys ‘1-8’ to view the different cameras around the building to monitor the enemy’s positions.
    - Upon closing the camera, the last viewed camera will be saved so that the user will return to the same camera they last left off in.
4. Time Algorithm:
    - The time algorithm includes two variables that play an important role:
        - Hour Duration
        - Night Win Condition

        - Hours Passed
    - The hour duration is made into a custom PyGame event in order to detect when the one hour interval has passed.
    - When PyGame detects the one hour event, it will add 1 to the integer stored in the hours passed (starts at 0).
    - When the number of hours passed is greater than or equal to the night win condition's number (6), the game will enter the win state, which means that the player wins.
5. Game State Manager Algorithm:
    - Checks for the current game state.
    - There are 5 game states:
        - Night Ongoing - Main Game Loop
        - Night Win - Game Win
        - Dead - Determines being Jumpscared or not
        - Blackout - For Power Outage
        - Game Over - Game Over
6. Door Algorithm:
    - Door's condition is checked so that:
    - If it's open, close the door
    - If it's closed, open the door.
7. Keypress Algorithm:
    - Using PyGame's event.get() function to scan for keypresses and corresponding them to certain actions in the game.

1.4. Modules

1. PyGame:

   The foundation of the entire game. Without PyGame, the game would only be a terminal/text-based game. PyGame allowed me to make the GUI for the game by blitting images onto the screen and using the mixer to play audio in order to make the game more lively. Without PyGame, most of the time-based mechanics would not work either.

2. Sys Module:

   Although not used as much as PyGame, the sys module was only used to close the game with the sys.exit() method. Aside from that, nothing else from the Sys module was really utilized.

3. OS Module:

   The OS module was also not used as much since it was only used in one of my files to get the path to a font for use.

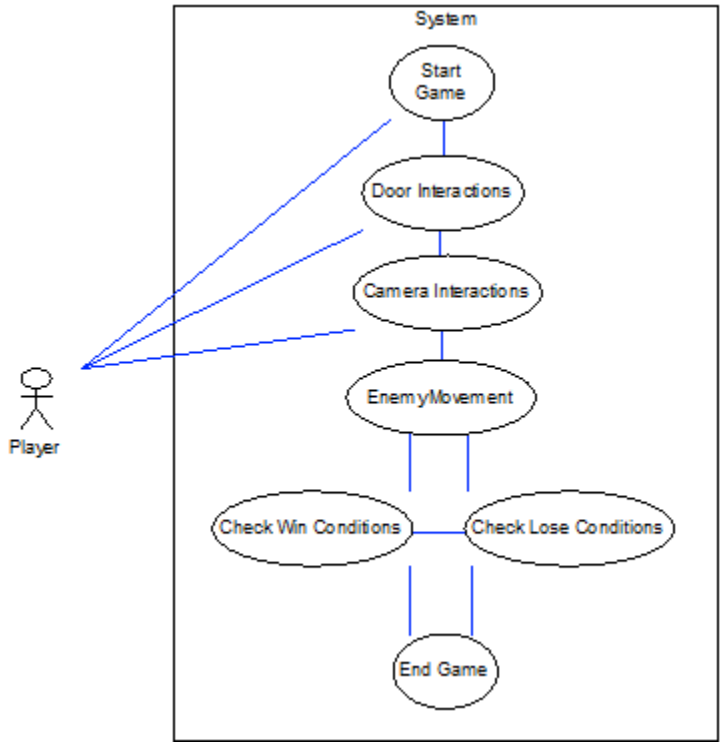
4. Random Module:

   The Random Module plays an important role in this game as it is involved in generating the Movement Opportunity Value for the enemies. It was also used once in picking the screen to be blitted in the blackout sequence with random.choice(). This module makes it so that no two gameplay experiences are the same.

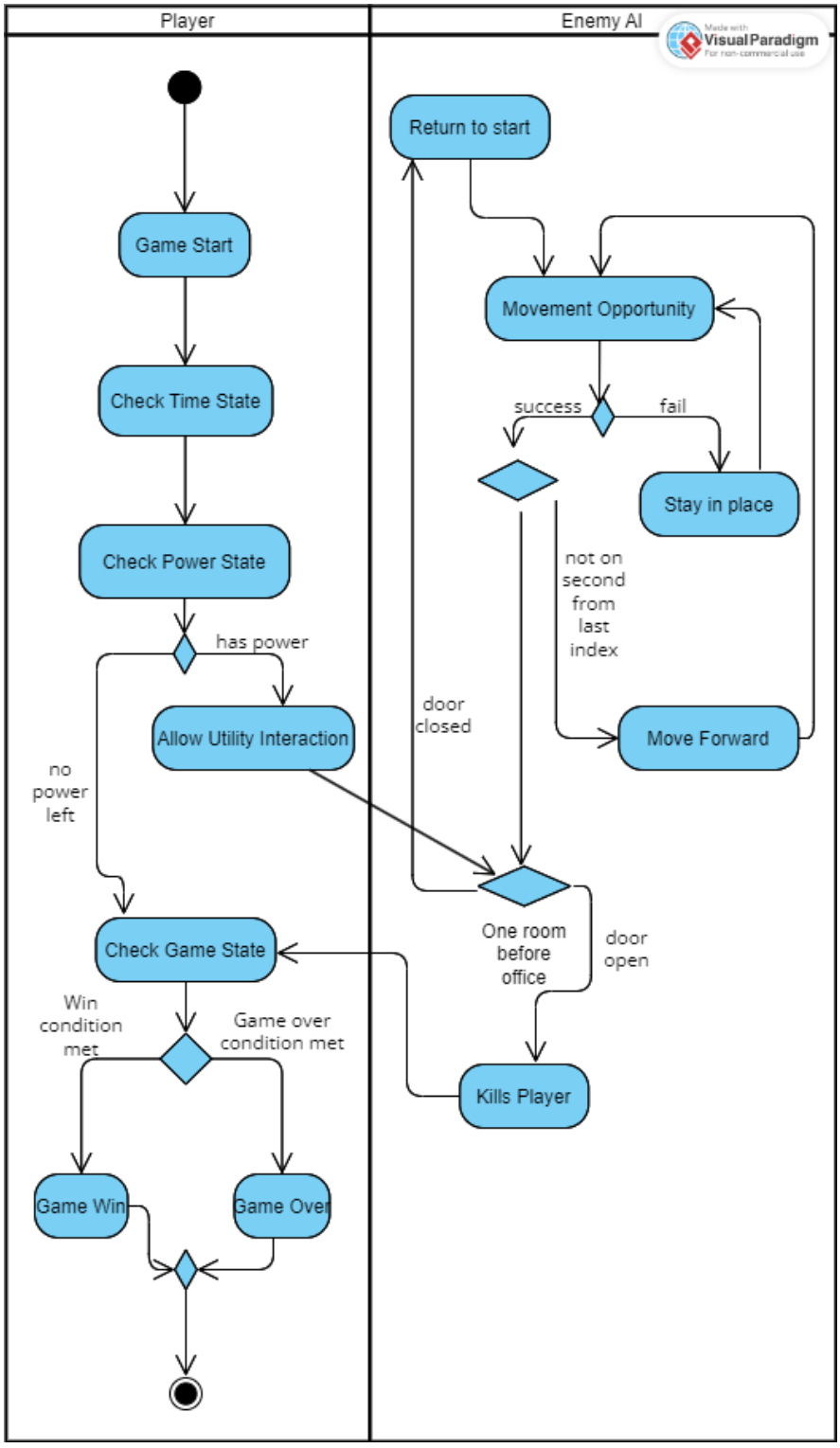
Chapter 2

SOLUTION DESIGN

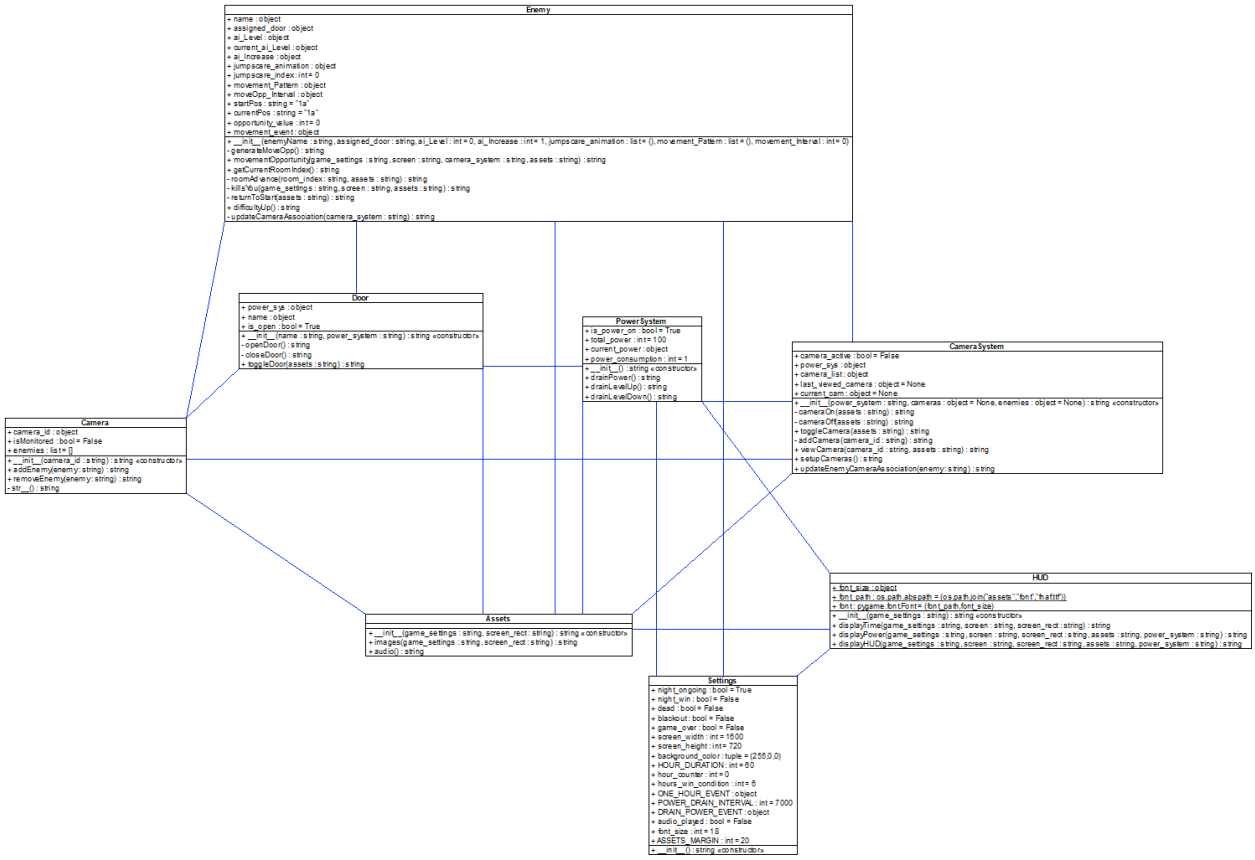
2.1. Use Case Diagram



2.2. Activity Diagram



2.3. Class Diagram

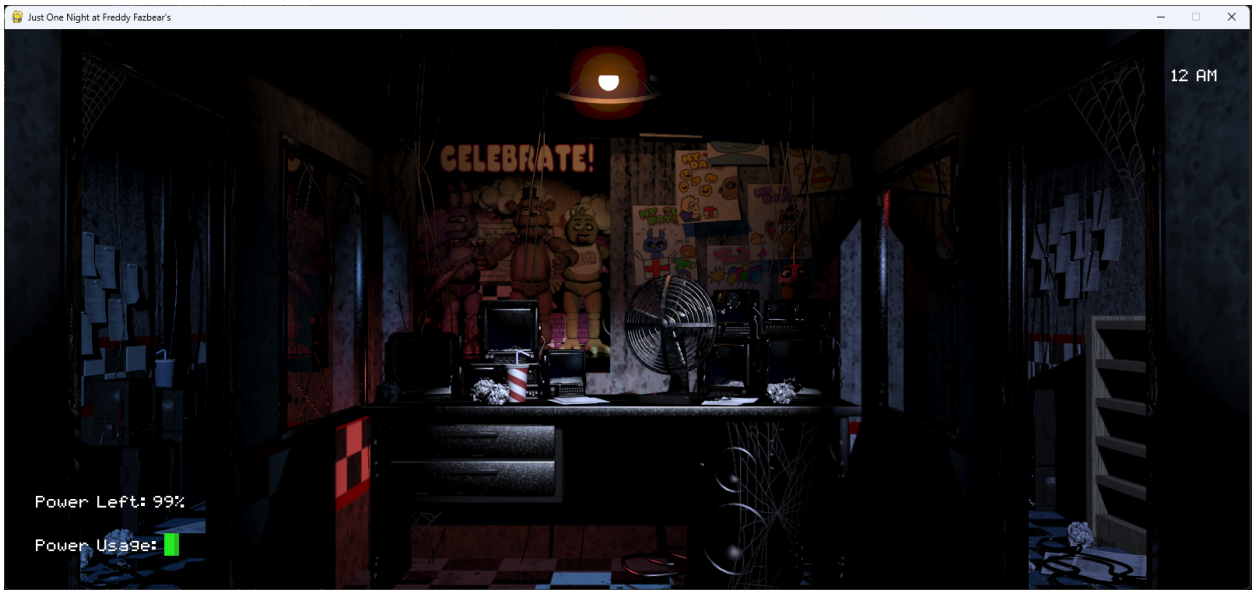


Chapter 3

DOCUMENTATION

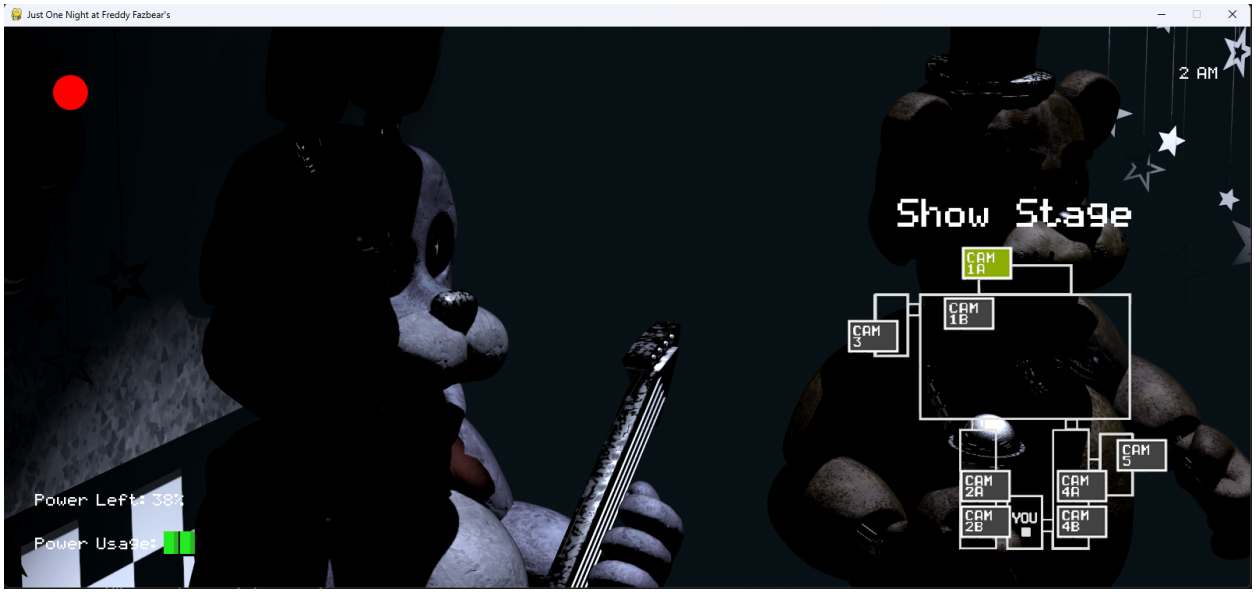
3.1. Screenshots

Office:



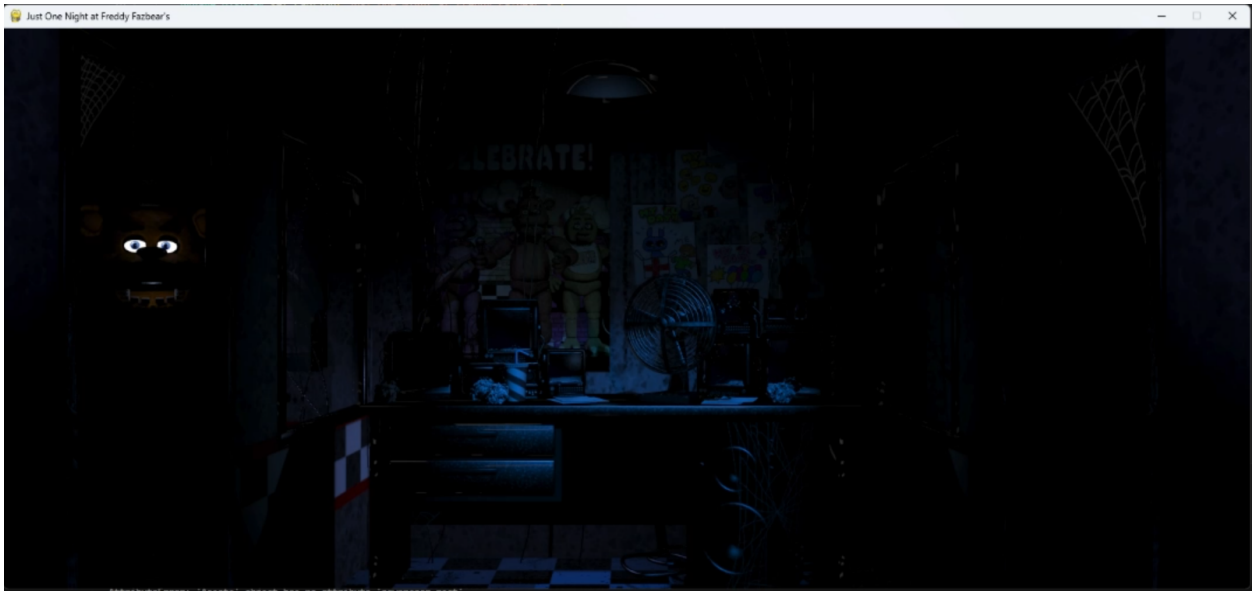
Trapped:

Cameras:



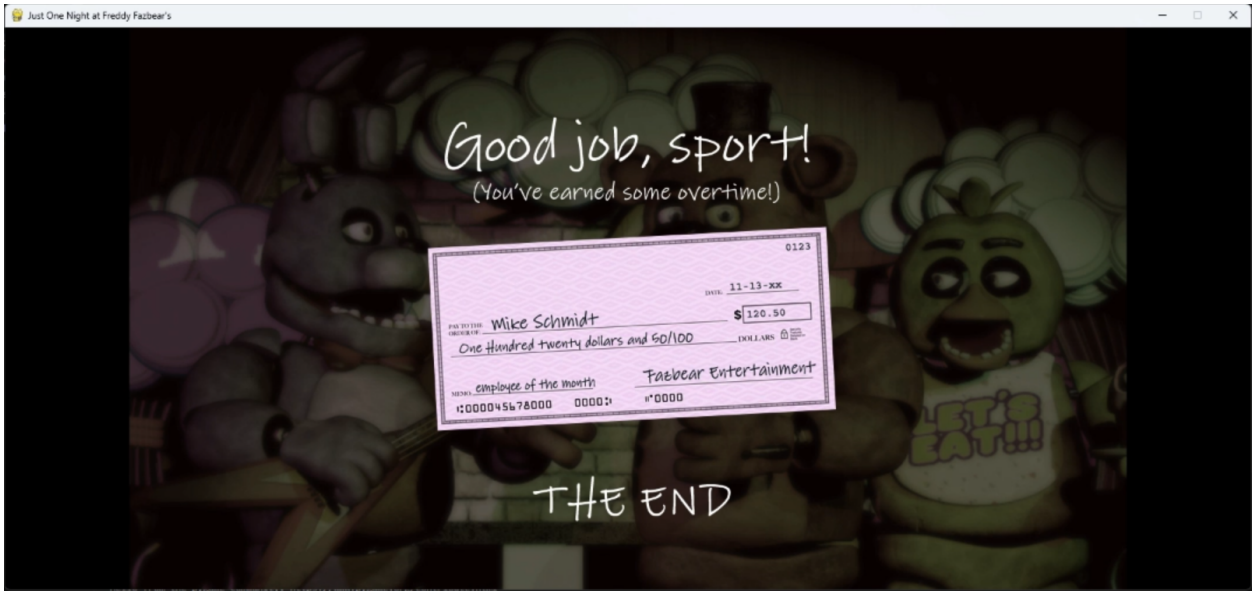
Bonnie Jumpscare:

Blackout:



Game Over:

Game Win:



3.2. Video Demo

I have uploaded the video demo in my Google Drive. The video can be accessed through the link provided below:

Video Link

Chapter 4

EVALUATION AND REFLECTION

4.1. Lessons Learnt

I have learnt a lot of things while working on this final project. Not only did I learn new things about Python as a programming language or PyGame as a library, but I also learnt about new ways to think outside of the box whenever I hit a roadblock in implementing a feature.

Through many sleepless nights I finally learnt and understood the fundamentals of programming games like the basics of game states, making a game state manager, making event handlers, making an AI for the game, and linking them all together to make a fully functioning game.

Aside from this I have also been reminded of the importance of time management. To be honest, I did not make any progress on my project until a week before 2024. I didn't think that I'd face too many challenges while programming since I thought that I already understood the mechanics for FNaF and that implementing them in terms of Python would be easy as I more or less had a grasp on the pseudocode for it. However, I was proven very wrong as I only managed to get the GUI and HUD working a day before my presentation day. Therefore, this project has also served as a lesson for me to not underestimate my work as well as the importance of time management.

4.2. Future Improvements

There are a lot of features in the original FNaF game that I was not able to implement within the given timespan. For example, just as the title states, there are supposed to be 5 nights, whereas my remake of the game only had 1 night. This is one of many possible improvements I could add if I were to improve on my project.

Another thing that I failed to implement in time was the game's point-and-click nature. The original game never relied on the keyboard as much as my version does. However, I wasn't able to implement handling mouse-related inputs in my version as I still had several issues in making it work.

I also did not implement a main menu. The main menu is one of the most important aspects of a game which I skipped entirely as, for some reason I was not able to make it work, even after I tried multiple different methods in handling game states, which was why I didn't implement it.

Some other things that I could've implemented but are only there for the aesthetics are animations. For example, an animation for pulling up the camera monitor, or for the door being closed. There are a lot of things that are meant to be animated within the game's screen. However, I only figured out how to properly get them to work on the last day and therefore, I was only able to implement several of the more important animations like the jumpscares.

Overall, there is still so much more room for improvement for my project. But even knowing this, I am very satisfied with what I was able to make within such a tight time frame, even if it is only a remake of an already existing game and not at all an original idea with original assets.