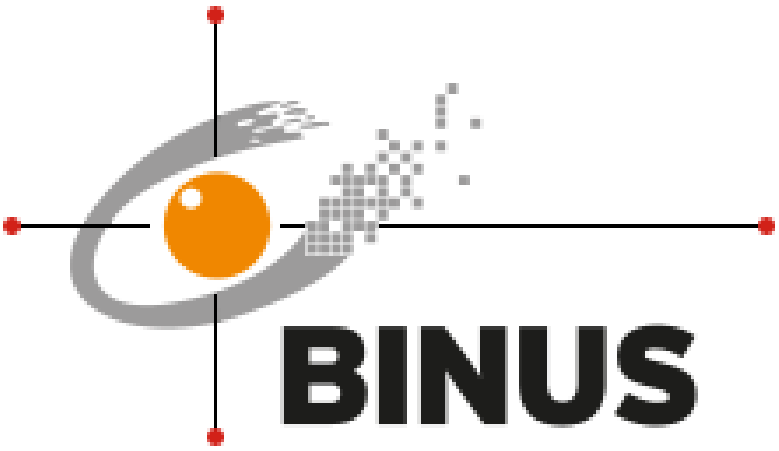
Algorithm and Programming
Final Project Report



BINUS
UNIVERSITY
INTERNATIONAL

Name of lecturer: Jude Joseph Lamug Martinez

Made by:
Irene Angelina (2802501060)

BINUS UNIVERSITY INTERNATIONAL
JAKARTA
2025

TABLE OF CONTENTS

Chapter 1
PROJECT SPECIFICATIONS

1.1. Project Description

For my final project, I developed a package delivery simulation game where the player takes on the role of a courier and has to deliver packages to houses each day. This project was inspired by one of my original characters who was a mail carrier. Originally, I wanted to create my own assets but decided on using free assets to save time.

At the start of each in-game day, the player receives a random number of packages and each package is randomly assigned to one of the houses on the map. When interacting with a house or door, the game provides feedback in the form of one of these three messages:

- "Package delivered!" — when the player delivers the package to the correct house.
- "Package already delivered!" — when the player attempts to deliver a package to a house that has already received one.
- "Wrong house!" — when the player tries to deliver a package to an incorrect house.

The game features a day/night cycle, where the screen gradually gets darker. Once night falls (indicated by specific color threshold), a smooth fade-to-black transition occurs, signaling the end of the day. The game then resets the level, generating a new set of packages and delivery locations, allowing for continuous gameplay.

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. Package Delivery Algorithm:
    - The player is assigned a random number of packages each day, determined by the random module.
    - For each package, the game randomly assigns it to a house.
    - When player interacts with a house or door:
        - If the house is the correct house, the package is marked as delivered and the counter will decrease.
        - If the house has already received a package, the game will notify the player with the message “Package already delivered!” and the counter will not decrease.
        - If the house is incorrect, the player will get the text “Wrong house!” and the counter will not decrease.

2. Day/Night Cycle Algorithm:
    - The day/night cycle sets an overlay on the screen that slowly changes.
    - Day Phase:
        - At the start, a white overlay is applied over the screen to simulate daytime.
        - Overtime, the color will change from white to a darker color. This is done through color interpolation (RGB values shift from white to a deep blue).
    - Night Phase:
        - As the transition completes, the screen reaches its final color (deep blue), signaling nighttime.
        - Once it reaches the threshold for night, the game triggers a fade-to-black transition with text saying “The Next Day...”, resetting the cycle with a new set of packages and houses for the next day.

3. Keypress Algorithm:
    - The game uses pygame.key.get_pressed() to detect keys that are being pressed by the player in real time.

- Key Actions:
    - “M”: Opens the map which is an image of the level.
    - “F”: Interacts with the houses or doors whenever the player is nearby.
    - “Esc”: Used to exit out of the map or dialogue.
    - WASD: Moves the player around the map. Depending on which keys are pressed, the player moves in the corresponding direction.

1.4. Modules

1. PyGame:

    Serves as the core framework for the entire game. Without it, the game would only be a terminal/text-based game. PyGame enabled the creation of Graphical User Interface (GUI) by allowing images to be rendered on screen. Moreover, many of the game’s time based mechanics rely on PyGame to function properly.

2. Sys Module:

    Only used to handle a clean exit from the game when the player closes the window or presses the “X” button. This is done through the sys.exit() method. Other than that, nothing else from the Sys module was utilized.

3. OS Module:

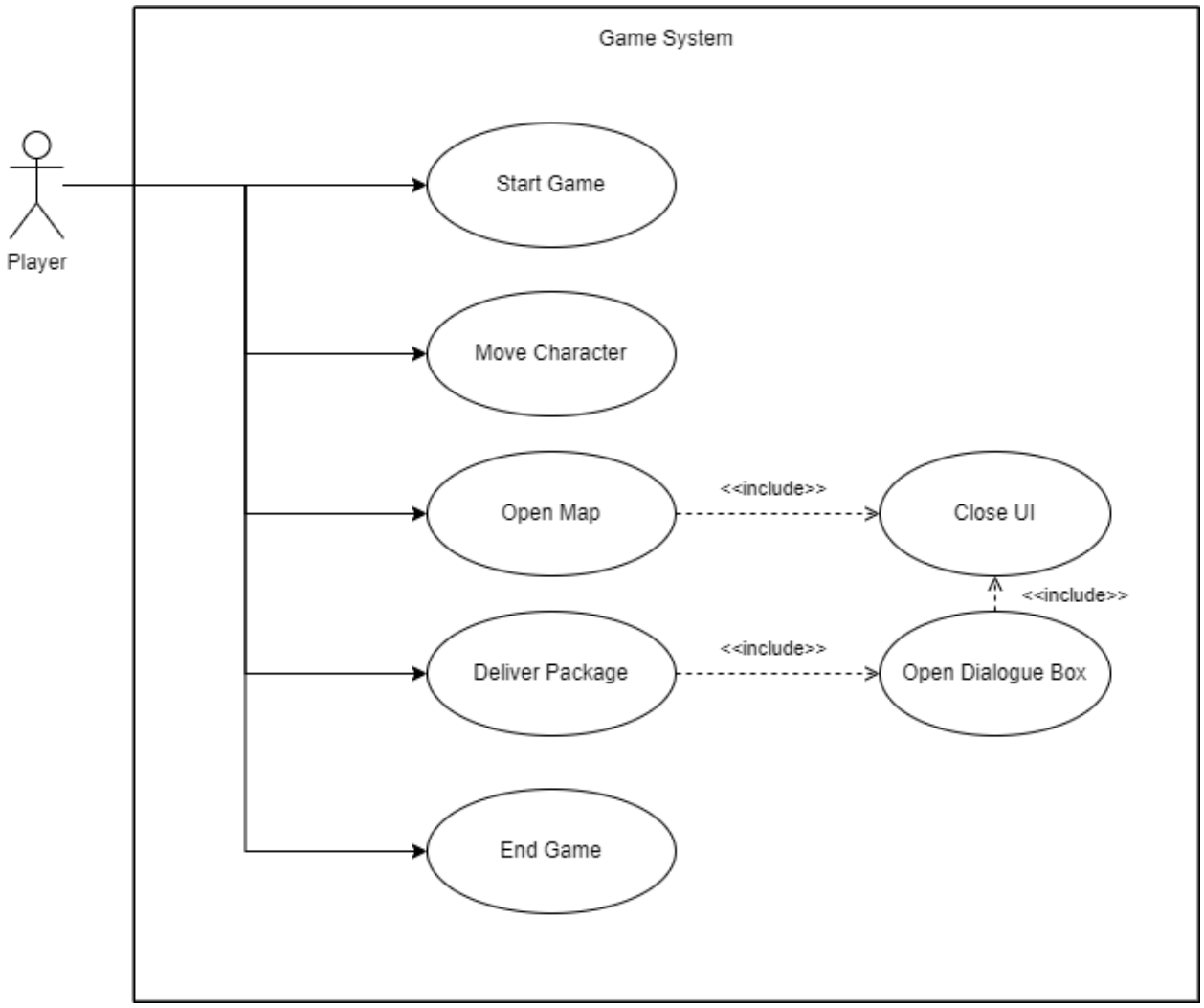
    The OS module was mainly used so I could import folders containing my sprites (for each player state) into a dictionary. It was also used to get the file path to a font for use.

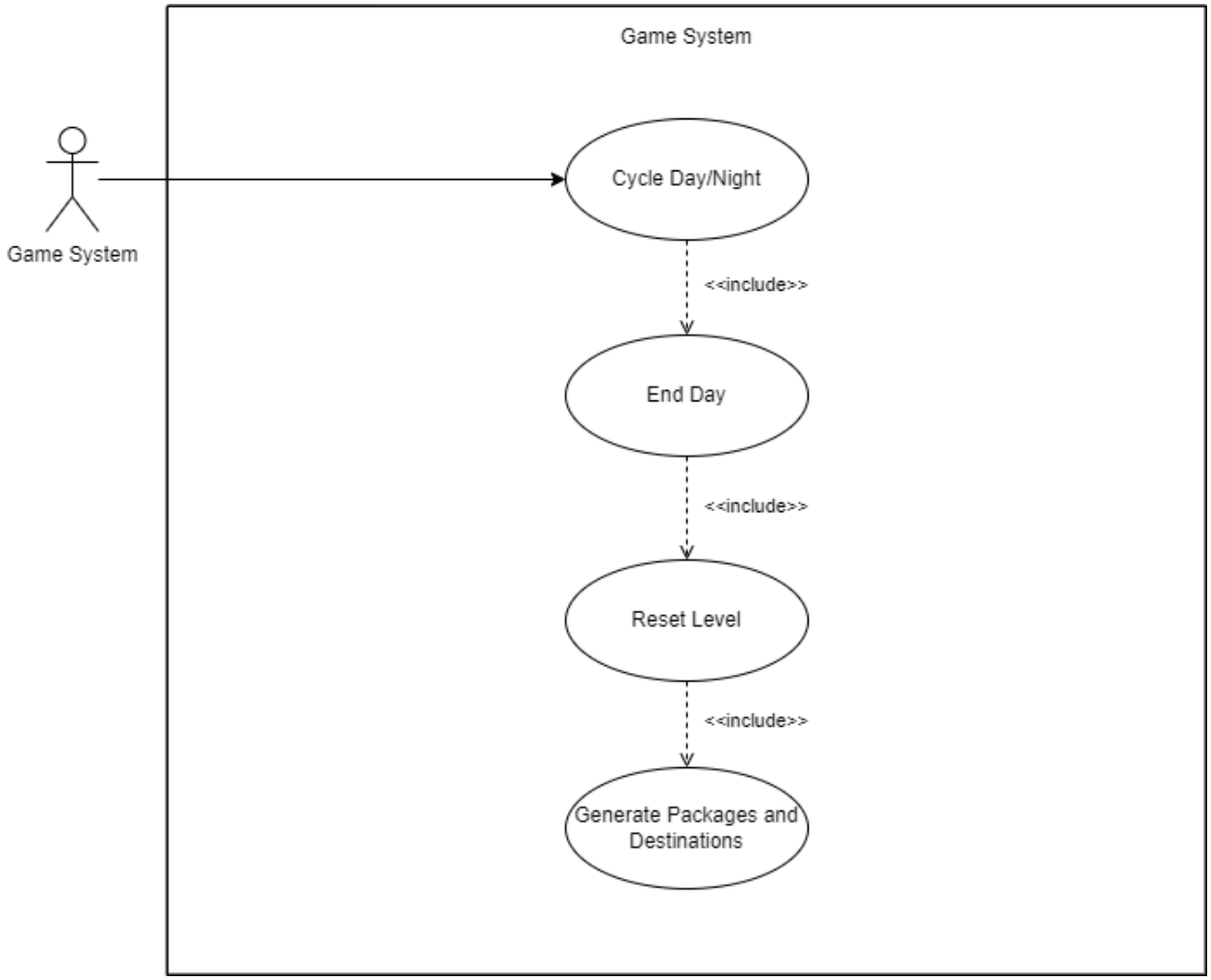
4. Random Module:

    The Random module adds unpredictability to the gameplay by randomizing the number of packages the player receives each day and assigning them to different houses. It ensures no package is delivered to the same house in consecutive runs. Additionally, it controls the weather, generating a number between 1 and 10, with values above 5 triggering rain and lower values resulting in clear weather.

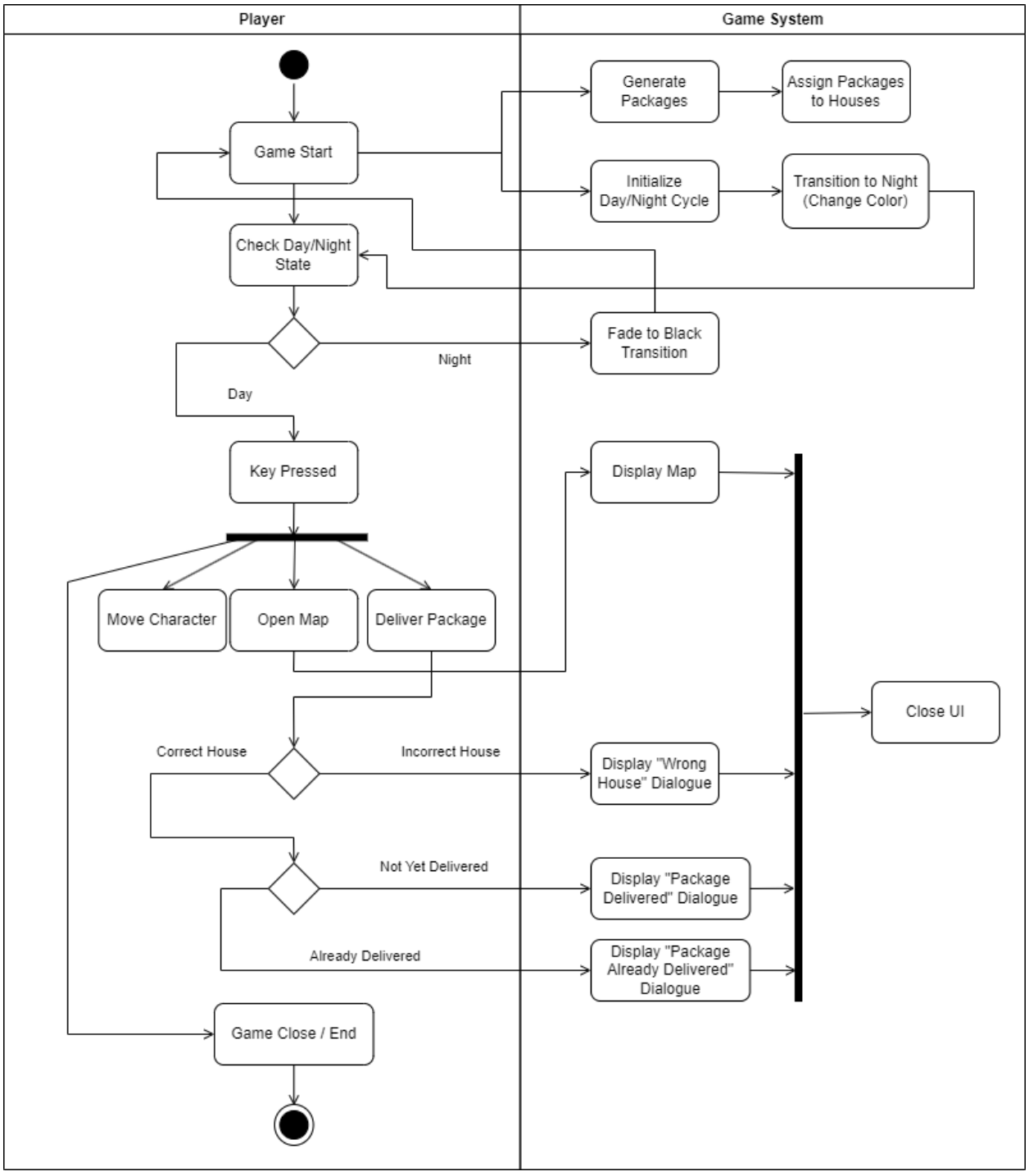
Chapter 2
SOLUTION DESIGN

2.1. Use Case Diagram

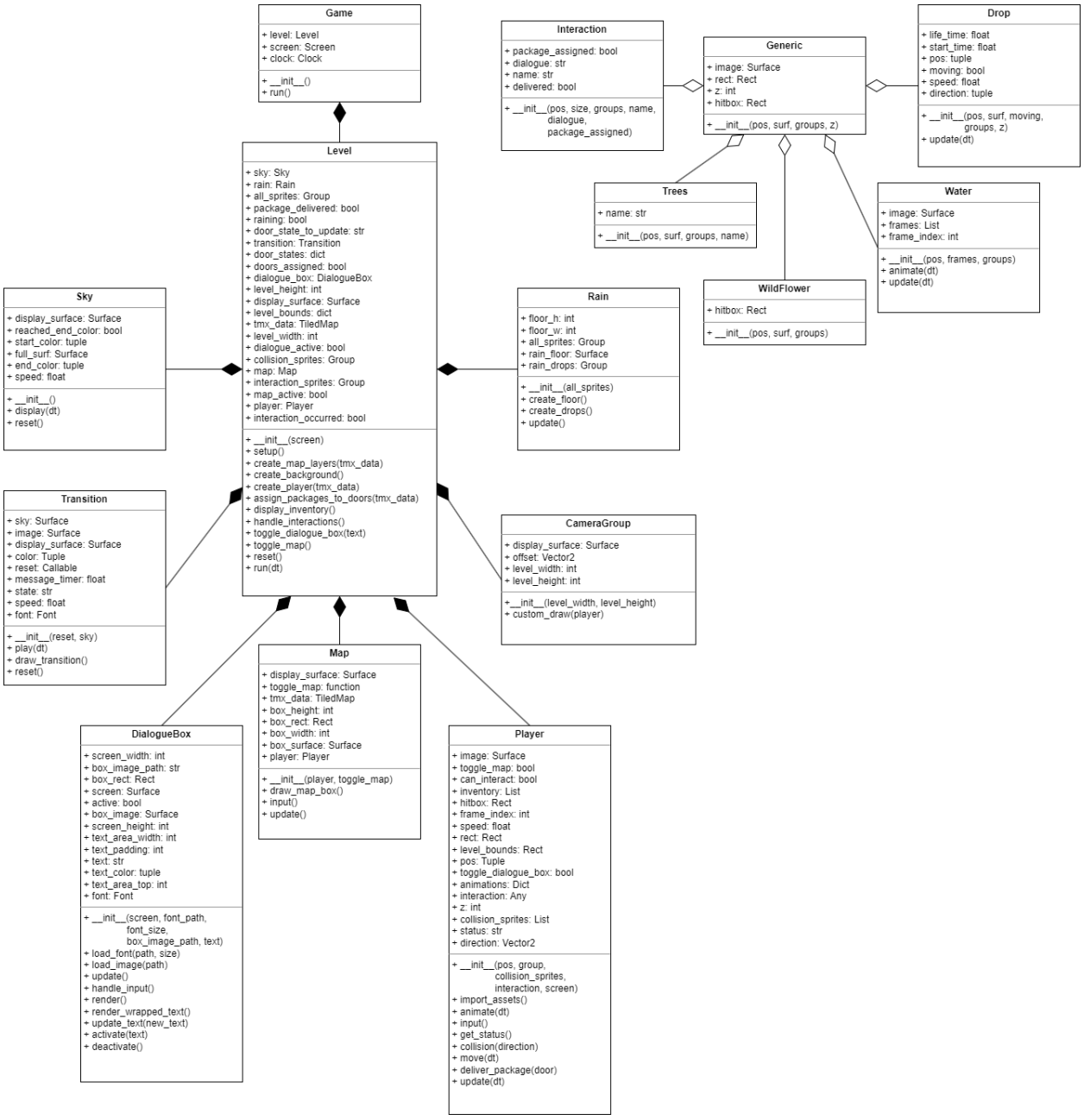



Game System
Game System
Cycle Day/Night
<<include>>
End Day
<<include>>
Reset Level
<<include>>
Generate Packages and Destinations

2.2. Activity Diagram



2.3. Class Diagram



Inheritance / Aggregation: White / Hollow Diamond
Composition: Black / Filled Diamond

Chapter 3
DOCUMENTATION

3.1. Screenshots

Clear Weather:

Rainy Weather:

Package Delivered Message:

Package Already Delivered Message:

Wrong House Message:

Day Phase:

Night Phase:

Transition:

3.2. Video Demo

I have uploaded the video demo to my OneDrive. Here is the link:

Video Demo

Chapter 4
EVALUATION AND REFLECTION

4.1. Lessons Learnt

Working on this project has taught me a great deal. I expanded my knowledge of Python as a programming language and PyGame as a library but also new ways to think outside the box when facing challenges in implementing features.

I gained a solid understanding of essential game development concepts like developing event handlers — even some concepts that didn't make it into the final product (e.g., managing game states) — and integrating these components to build a fully functional game.

Aside from this I was also reminded of the importance of proper time management. Being completely honest, I only started making the game about one or two weeks before the presentation deadline. I had thought I understood the mechanics needed for the game and that implementing them in terms of Python would be easy. However, I was proved very wrong as I spent almost a whole week trying to fix the day/night cycle as well as figuring out how to assign each package to each house depending on the number of packages the player receives at the start. My problems only increased from there on as each new function and method I added created issues with previous existing ones (I hadn't managed to fix the dialogue to display "Package already delivered!" properly until right before the presentation). Therefore, this project serves as anecdotal evidence to remind me not to underestimate the scope of my work.

4.2. Future Improvements

There were a lot of initial features that I was unable to implement within the given timespan. The original idea was to have different menus (e.g., title screen, pause menu, end-of-the-day screen, victory/lose screen) but it was scrapped at the last minute because they caused major issues that broke the game.

I also intended to include a story sequence at the beginning to explain the character's background and why they became a courier. This would have

been presented in a visual novel style with images and dialogue boxes, where players could progress the story by pressing a button. However, I realized this feature was far beyond my current abilities, as it felt like developing a second game and trying to merge it with the existing one.

Furthermore, I had planned to add small details like sound effects and an arrow to highlight which houses each package belonged to, but I removed these features out of concern they might further disrupt the game's stability.

Overall, I recognize there's still a lot of room for me to improve my Python skills and manage my time better to avoid procrastination. Despite all that, I am proud of being able to make a game even if it is very simple in premise and execution.