Fatal Impact

URL:
https://github.com/comp195/senior-project-fall-2022-project_apocalypse/wiki

Eesa Khan: e_khan2@u.pacific.edu
Kevin Nguyen: k_nguyen159@u.pacific.edu

September 18, 2022

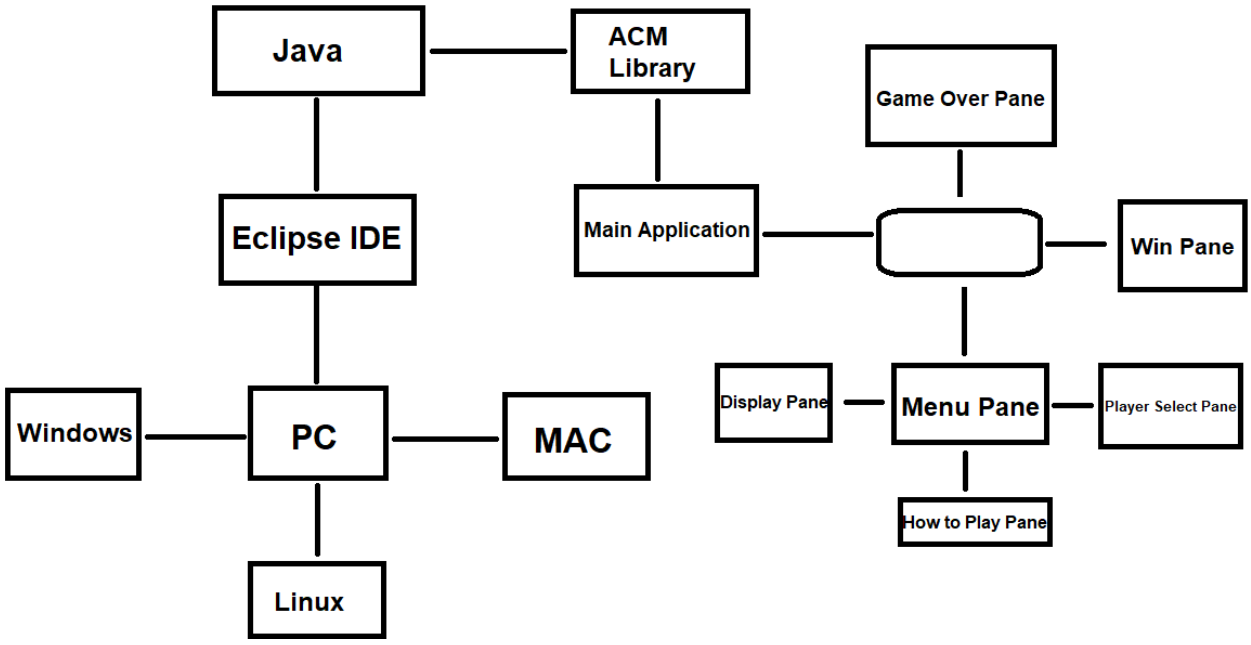
Project Introduction and Scope

Fatal Impact is about a zombie apocalypse survival game. The player will try to survive hordes of zombies that intend to attack him. There will be items that he can pick up to help him survive: melee weapons and long range weapons to fend off attacks; food and water to fend off hunger and thirst. The player can explore the world and find a zombie boss to defeat and win the game. Alternatively, the player will have to find a way to survive each day, while having the option to hide and sleep through the night.

The inspiration for this project comes from similar zombie apocalypse survival games, such as H1Z1 and Project Zomboid. Some of the intricacies in game mechanics for Fatal Impact will be based off of some design features from H1Z1. For example, an article from gamespot highlights a key feature in zombie spawn points of a particular build of H1Z1: “Biased zombie spawns toward areas with more player heat, and more items, so that cities, camps, and player firefights have the highest rate of zombie spawns, and the wilderness areas have less” (Makuch, 2015). Like H1Z1, Fatal Impact will have zombie spawn points based on environmental factors, such as buildings with a lot of items, and based on the distance of the player.

Likewise, the intricate details of eating and sleeping in Project Zomboid is something Fatal Impact will live up to. Project Zomboid implemented a realistic sleep feature that affects the player: “The player typically requires between 6 to 12 hours of sleep per night. If the player hasn't slept for a long time, the sleeping time can drastically increase depending on the severity of the exhaustion” (*Sleep - PZwiki*, n.d.). Project Zomboid implemented this feature to add a level of difficulty: exhaustion and drowsiness due to the lack of sleep. Fatal Impact will have a similar feature in which the player will need to sleep in order to complete tasks efficiently. If not, the player’s movement speed will decrease.

System Architecture



Hardware Components: Windows, PC, Mac, or Linux. These are the operating systems (OS) required to run the Eclipse IDE.

Software Modules: Java – programming language used in game, ACM library – supports creation of object oriented graphic displays, Eclipse IDE – software development, runs/debugs code.

User Interfaces: Main Application, Game Over Pane, Win Pane, Menu Pane, Display Pane, How to Play Pane, and Player Select Pane. These panes are all Java classes responsible for displaying their respective user interfaces.

- The Main Application is responsible for switching between each pane using the switchToScreen() method.
- The Game Over Pane will display when the player dies and has to restart the game to continue.
- The Win Pane will display if the user has won the game by defeating the main zombie boss.
- The Menu Pane will display when the user first boots the game up. The user will have the opportunity to switch between the Display Pane, Player Select Pane, and the How to Play Pane.
- The Display Pane is responsible for running the actual gameplay. If the user selects the menu option “Play”, he will be able to start the game and it will switch to the Display Pane.

- The Player Select Pane will allow the user to select his character. He will be able to select a long-ranged character or a short-ranged character.
- The How to Play Pane will allow the user to read up on the controls of the game.

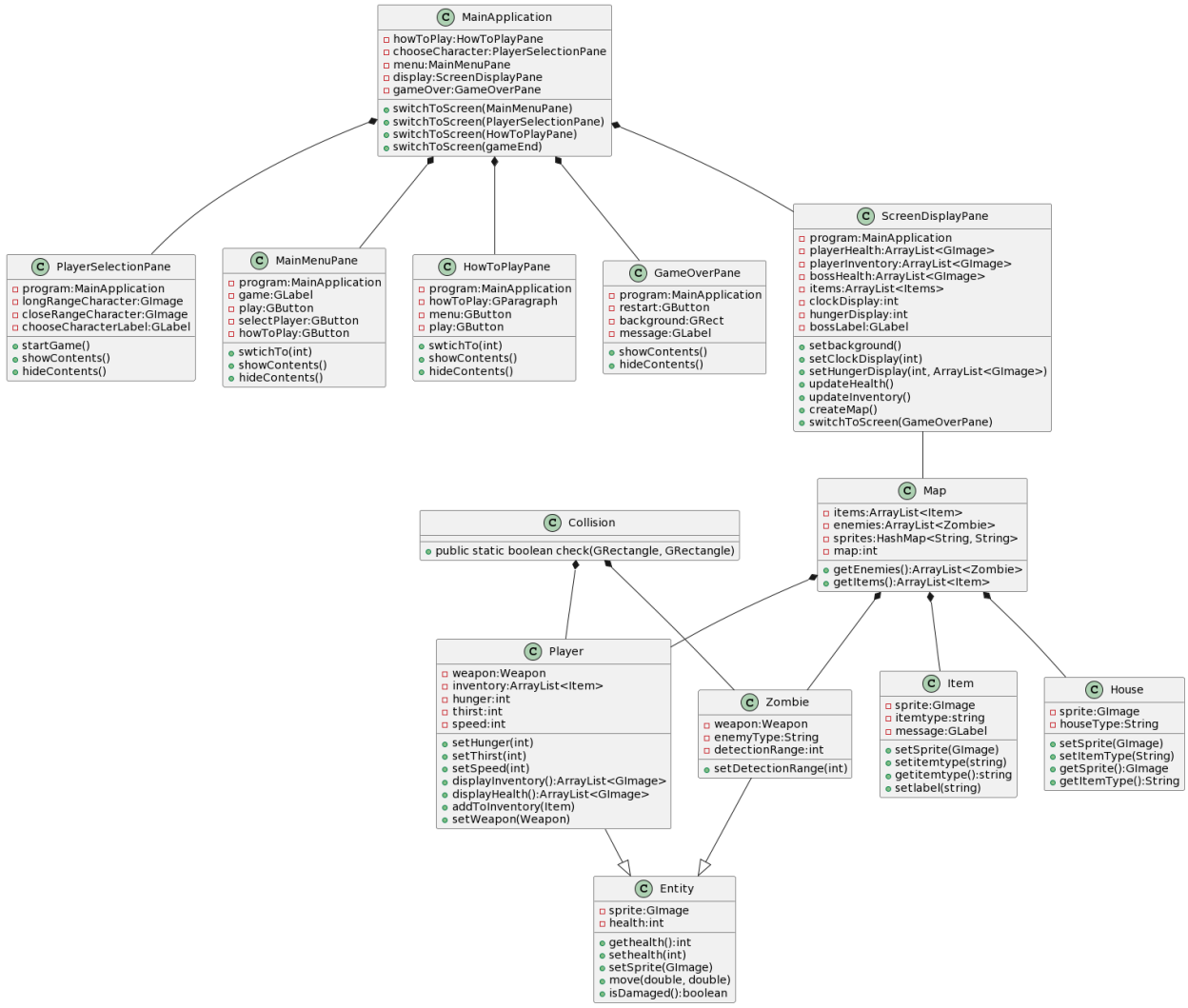
Hardware, Software, and System Requirements

- Hardware requirements: minimum of 3.20GHZ CPU speed for standard performance, 8 Gigabyte RAM, and minimum of 100 mbps of internet speed.
- Software requirements: Eclipse IDE 2019 and up, Java version 17 and up.
- System requirements: Microsoft Windows OS, Apple Mac OS, or Linux OS.

Software Design

NOTE: Zooming into the Class Diagram and the Interaction Diagram will allow you to see the text clearly.

Class Diagram



Class Specifications

MainApplication: Responsible for managing the switches between all the panes.

PlayerSelectPane: Responsible for displaying a closed-ranged character and a long-ranged character for the user to select. The option to start the game is also here.

MainMenuPane: Responsible for allowing the user to select “play”, “select character” and “how to play” options. Each option switches to its respective pane.

HowToPlayPane: Responsible for allowing the user to read up on the controls of the game.

GameOverPane: Responsible for displaying a “game over” message and for allowing the user to restart the game.

ScreenDisplayPane: Shows the background image of the map; the health bar for both the players and the enemies; a separate boss health bar; creates and displays the map, and a win/lose screen based on the display type. It’s also responsible for displaying the time in digital format and the player’s hunger/thirst status. The player’s inventory is also displayed here.

Map: Responsible for handling enemies and their sprites. Also responsible for setting the background image.

House: Responsible for managing images for different kinds of houses; Allowing the user to enter and explore the interior.

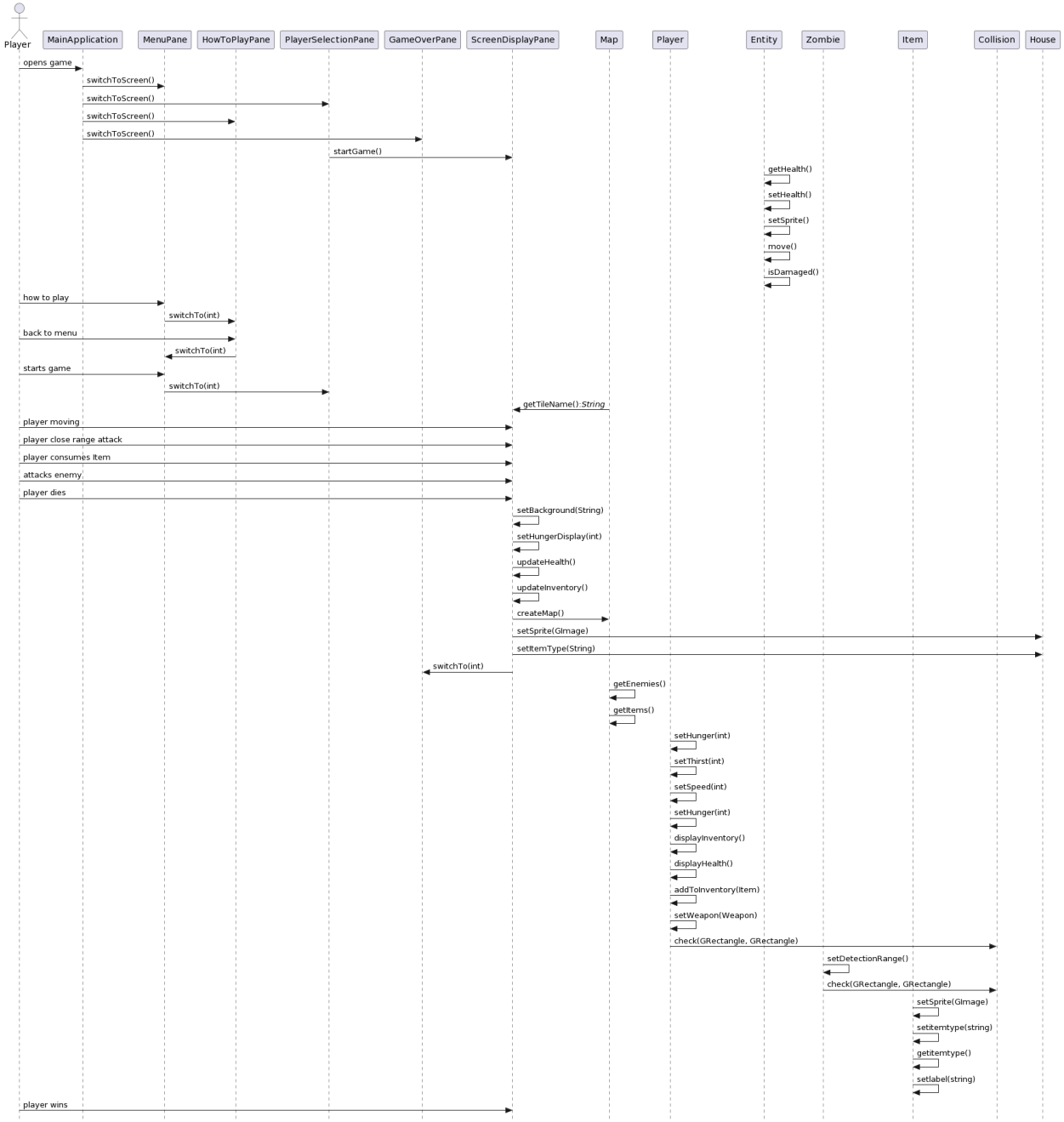
Entity: Responsible for basic functionality of Player, Zombie, and ZombieBoss. Allows these entity types to move.

Player: Responsible for handling the player’s hunger, thirst, inventory, and weapon.

Zombie: Responsible for moving toward the player and attacking him.

Item: Responsible for sprites and item types for every type of Item.

Interaction Diagram

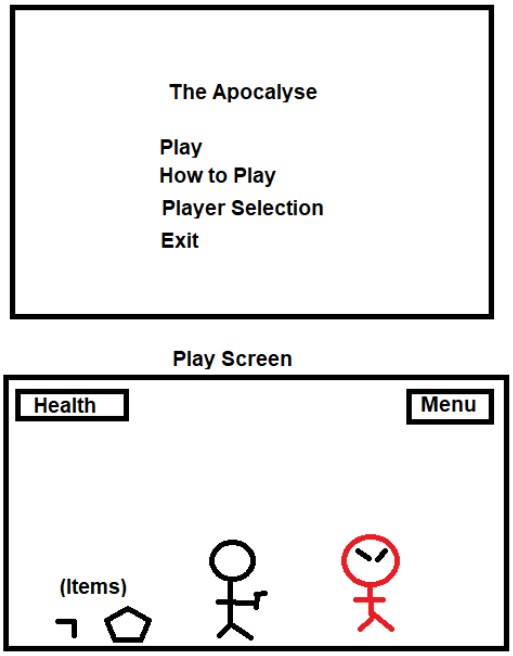


Design Considerations

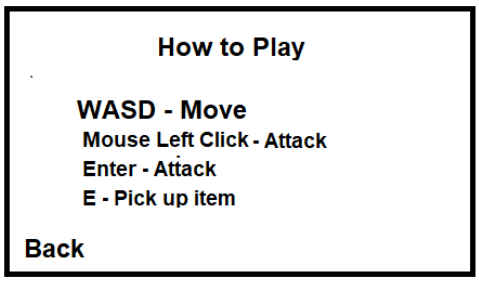
In the initial design of these classes, there was a single Item class for Player to interact with. However, it became abundantly clear that that single Item class was too vague to describe every item type. Hence, Weapon, PickupItem, Chest, and Door became their own classes inheriting from Item. This means that Item acts as the parent class.

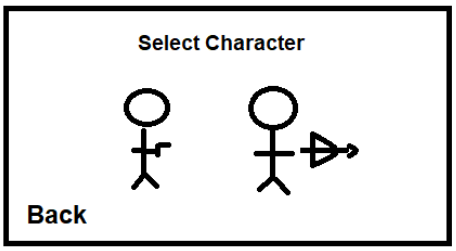
The Entity class was also its own class until realizing, much like with a single Item class, that a single Entity class was too broad to describe both the player and the enemies. So Player and Zombie were created as class children to Entity. ZombieBoss was also created and it inherits from Zombie.

User Interface

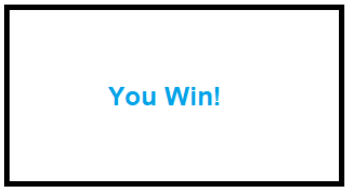


- Menu: The user has the option to to select “Play”, “How to Play”, “Audio”, and “Exit”.
- Play: If the user selects “Play” from the menu, he will start the game.
    - Items: The user should be able to pick up items on the ground.
    - Enemies: The user will have to fight enemies in his way.
- Exit: If the user selects “Exit”, the game will shut down and stop running the program.





- How to Play: The user can see what the game controls are and learn how to play the game. He also has the option to back out of this pane to the menu pane.
- Select Character: The user has the option to select a short-ranged character or a long-ranged character to play the game. He also has the option to back out of this pane to the menu pane.



- Win: If the player completes the winning objective, the win screen will be displayed.
- Game Over: If the player dies, the game over screen will be displayed and will need to restart the game to continue.

Glossary of Terms

Application - A computer software package that performs a specific function directly for an end user or, in some cases, for another application.

Class Inheritance - A class that inherits all the methods and properties from a parent class.

CPU speed - Refers to how optimal a personal computer is.

Enemy - An NPC that attempts to cause a Game Over for the player.

Game Over - A situation wherein the player is defeated, and brought back to the main screen.

IDE (Integrated development environment) - software application that provides comprehensive facilities to computer programmers for software development.

Inventory - A space to store items.

Item - A collectible object the player in a video game can use in specific cases.

Java ACM Library - Supports creation of Java object oriented programming with accompanying graphical displays.

Map - A diagrammatic representation of an area of land or sea showing physical features such as cities and roads.

Melee: All forms of close combat in a video game.

Menu - What the game displays when first opened or when the user presses the pause key during gameplay. It serves as the main method for moving among different screens and selecting tasks to perform.

Operating System - The software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

Pane: A rectangular area within an on-screen window that contains information for the user.

Parent Class (AKA base class) - The class being inherited from.

RAM (Random Access Memory) - A component that all modern personal computers have. It is the short-term memory where the data that the processor is currently using is stored.

Spawn: The live creation of a character, item, or NPC in a video game.

Sprite - A character image or an object image in a video game.

User Interface - The means by which the user and a computer system interact, in particular the use of input devices and software.

Video Game - A game played by electronically manipulating images produced by a computer program on a television screen or other display screen.

References

Makuch, E. (2015, April 2). *H1Z1's "Big Damn Patch" Out Now*. GameSpot. Retrieved September 18, 2022, from https://www.gamespot.com/articles/h1z1-s-big-damn-patch-out-now/1100-6426360/

Sleep - PZwiki. (n.d.). Retrieved September 18, 2022, from https://pzwiki.net/wiki/Sleep