Living Document

Team Info

Team Members:

- Douglas Sandford - Developer
- Kaden Allen - Developer/QA Engineer
- Joseph Liefeld - Designer / Developer
- Jonathan Hotchkiss - Developer, Tester
- Graham Glazner - Coordinator, Developer
- Wyatt Fujikawa - Developer / Designer
- Owen Wickman - Developer/QA Engineer

Github

https://github.com/LitchDoctor/winter2025-group2-rpg.git

Trello

https://trello.com/invite/b/6781aa06f4132bde26a088ae/ATTI281728df6d053a134dd1f0f5fcf09ab6E74188A9/pt-2-group-2

Communication

Discord and Text Group chat

Rules

- Check Discord regularly
- Respond within 8 hours if pinged on discord
- Text Group chat for high priority messages

Project Description:

Protect Humanity RPG

Abstract

Our project is a role-playing game where the player plays a robot who travels through a dangerous, apocalyptic world where humans have mutated into horrific creatures, and your job is to escort the last living human(Jason) to safety. The map includes enemy encounters that enter the player into a turn-based battle. For each turn, the player must choose an action to keep the human alive. The human will also act, though they may not always be helpful.

Specifications

The robot and human spawn in a desert wasteland far away from the launchpad they need to reach in order to save the last living human on Earth. Visually, the robot leads the human. They travel toward the end of the stage to the right, encountering enemies, points of interest and items. At the end of the first two stages are mini-bosses which block the path to the next stage. To initiate combat, move until the robot collides with an enemy. A pop-up comes up to choose whether to initiate combat. If you select yes, the fight starts after a transition. If you select no, there is a random chance that the human will initiate combat anyway. Colliding with an item allows you to select whether to pick it up. Colliding with a point of interest shows a block of text and allows you to close the pop-up.

Combat:
The player has 5 options every turn: defend, taunt, empower, heal or stun. The enemies have two attacks: weak or powerful. They either target the robot or the human. The enemy shows a visual effect hinting at whether it will use a strong or weak attack. The enemy's visual may also indicate who it will attack. The human has a percent chance to do one of its abilities which include attack, special attack and do something stupid. The human has a reduced chance to choose the same option twice in a row.

Items:
Computer chips for the robot: move improvement items.
Perhaps there are items which unlock robot options.

Goal

We are trying to provide the large video gaming community with a novel and entertaining game.

Current Practice

Today's RPGs are typically developed with game engines such as Unity or Unreal Engine. These games often feature vast open worlds and real-time combat. Many RPGs also integrate online multiplayer and dungeon raids with a group of players.

Novelty

Unlike previously made RPG's, our game will have the player be protecting another entity, which differs from other games where the focus is just on the player's survival.

Effects

In our day and age, the video gaming industry is at an all-time high demonstrating how highly our society values entertainment. The game we propose will inject itself into the said industry and provide the community with a new and valuable source of entertainment.

Use Cases

Use Case 1: As a player, I want to enter an enemy battle encounter so that I can progress the game.

Author: Graham Glazner

1. Actors
    a. The Player
2. Triggers
    a. The Player moves their Character into the same square as an enemy.
3. Preconditions
    a. The enemy is visible on the screen. The character is not currently in a battle. The character has a clear path to the enemy.
4. Postconditions (success scenario)
    a. The character is in a battle with the enemy they ran into. The enemy, the character and the human npc are displayed. The character’s battle options are shown.
5. List of steps (success scenario)
    a. The player moves their character into the enemy’s square.
6. Extensions/variations of the success scenario
    a. The transition to battle animation plays.
7. Exceptions: failure conditions and scenarios
    a. The Character goes through the enemy sprite without entering a battle.
    b. Player can't reach NPC due to error

Use Case 2: As a user, I want to heal the “pet” during an encounter so that they do not die

Author: Jonathan Hotchkiss

1. Actors
    a. User
2. Triggers
    a. User selects to heal “pet”
3. Preconditions
    a. “Pet” is alive (hp > 0) and not at max hp
    b. Heal cooldown has expired
4. Postconditions (success scenario)
    a. Heal cooldown increased
    b. “Pet” hp has increased
5. List of steps (success scenario)
    a. User enters into an encounter with an enemy
    b. At least one round of fighting takes place to reduce “pet” health
    c. User selects heal when battle prompts are shown
6. Extensions/variations of the success scenario
    a. The user chooses to defend or attack and wait another round before healing
7. Exceptions: failure conditions and scenarios

    a. The user does not heal the “pet” and it dies

Use Case 3: Player movement
Author: Owen Wickman

1. Actors
    a. The player
2. Triggers
    a. The player presses any of the movement keys W, A, S, or D
3. Preconditions
    a. The player is in the main world (not in a menu or encounter)
    b. The players movement is not restricted
4. Postconditions (success scenario)
    a. The player moves in a direction depending on the key pressed
        i. W -> Up
        ii. A -> Left
        iii. S -> Down
        iv. D -> Right
    b. If two keys are pressed, the player will move in the average direction
        i. W and D for example will cause the player to move up and to the right
5. List of steps (success scenario)
    a. The user presses any movement keys
        i. The player moves in the correct direction ex: W -> Up
6. Extensions/variations of the success scenario
    a. The user presses multiple movement keys simultaneously
        i. The player moves in the average direction ex: S and D -> Down and Right
7. Exceptions: failure conditions and scenarios
    a. The player fails to move when movement keys are pressed or the player moves in the wrong direction

Use Case 4: Defeating an Enemy (Author: Joseph)
Actor: User playing the game

Trigger(s): Using an attack that reduces the enemy’s health points to less than or equal to zero

Precondition(s):

- In battle state
- Enemy has remaining health (hp > 0)

Postcondition(s):

- Sprite disappears from screen
- “Enemy defeated” text box is displayed

List of Steps:

- User enters a fight
- User uses abilities that deal damage to enemy

Extension/Variation:

- User dies before defeating enemy

Exceptions (failure conditions + scenarios):

- Enemy's HP is reduced to <= 0, yet the enemy remains one screen and continues taking turns

Use Case 5: As a user, I want my character to level up as a reward for killing an enemy so that I have a reason to kill enemies and get stronger.
Author Wyatt
Actor: player

Trigger(s): passing or meeting the experience threshold required to level up

Precondition(s):
Player must defeat an enemy or gain enough experience to pass the amount of experience required to level up
Postcondition(s):
Pop up message saying "You have leveled up!"
Experience data is saved and added to player class in case player doesn't level up
Music queue indicating level up?
Random increase in all player stats (1-5)
Not sure which stats we want a player to have
HP, MP, strength, resilience, speed, etc. (open to ideas)
More experience is required to level up again
Experience is "reset" after every level up

List of steps (success scenario)
Player defeats entity or gains experience another way
Meets the experience requirements
Player levels up and a pop up message is generated
Another pop up is generated showing statistical increase
Example: 1 → 4
Pop up closes and user is returned to the main scenario
Extensions/variations of the success scenario
Learning new abilities?
Skipping levels if enough experience is gained leading to greater stats increase, new abilities, etc.
Can both the player and the robot level up?

Exceptions: failure conditions and scenarios
Experience isn’t properly saved to the player class after an enemy is defeated
Messages don’t pop up when player levels up
Player stats aren’t properly updated or saved
No new abilities are learned even if they’re supposed to be (if we add new abilities)

Use case 6: Game over conditions
Author Kaden
Actor: Player

Triggers: Player character or AI companion reaches 0 hp

Preconditions: Player is in combat and an allied character reaches 0 hp.

Postconditions: Depending on which entity dies, either combat continues without user input(player death) or the game over screen displays(AI death)

List of steps (success scenario)
Player enters combat with an enemy
Player is damaged by enemy
Player’s Hp falls to 0
Death message displays
Combat continues without player input. (Until AI dies or battle is won)

Extensions/variations of the success scenario
If the AI companion dies instead of the player character, combat ends with a game over screen.
If the player character is revived mid combat, they should be able to act again.

Exceptions: failure conditions and scenarios
Combat continues after the AI companion dies
Player remains able to act after reaching 0 hp
Game over screen fails to display
Combat is treated as a victory even if the player loses

Use case 7: Talk to an NPC
Author Douglas
Actor: Player

Triggers: The player character is near an NPC with an icon above their head and starts an interaction with them

Preconditions: The player is able to walk around the world and not in battle

Postconditions: The NPC dialog is starting allowing the player to read what the NPC is saying

List of steps (success scenario)
The player moves to the NPC
The player can progress the dialog with a click or a button
The dialog starts

Extensions/variations of the success scenario
the player receives the next quest or item

Exceptions: failure conditions and scenarios
The NPC is not able to be interacted with
Player can't reach NPC due to error

Non-Functional Requirements

- Each time the player lands on a battle, the battle animation/screen should load up within 3 seconds.
- Player movements must occur within .5 seconds of the user providing input.
- The game should maintain a consistent frame rate of at least 30 FPS.
- The game should offer colorblind options.
- The system should be highly modular making it easy to implement new content.

External Requirements

- The program will be able to withstand reasonable user input errors and will respond accordingly.
- The game must be downloadable and playable for users on their own devices.
- Software used for the entire program should be accessible to others so that they can develop their own versions. As a result, the program should be well-documented and understandable by others.

Team Process Description

- Specify and **justify** the software toolset you will use.
    - Unity includes a 2D preset that we can build off of and is designed for game creation. Our art requirements are not large, so we should be able to make everything that we need in Piskel.

- Define and **justify** each team member’s role: why does your team need this role filled, and why is a specific team member suited for this role?
    - Kaden: Developer/ QA Engineer
        - Our team needs many developers since there is a lot of different types of development work to be done. I have played many different video games so I should be able to identify quality problems before they become ingrained in our systems.
    - Douglas: Developer/ Combat Engineer
        - Our team needs people to design each part of the game. I have experience in playing multiple different kinds of turn-based games. I should be able to efficiently make changes based on testing.
    - Jonathan: Developer/ Tester
        - Including developers in the team is crucial to being able to create a working program. Without the developers, there would be no running program. Testers are also important to ensure the program will run without errors and each system component will function properly. I am suited to take on these responsibilities because I have experience developing programs and small games throughout previous classes. I also have some experience with testing, and I understand the importance of carefully testing each portion of the program.
    - Graham: Coordinator/ Developer
        - Our team needs a coordinator so that we can keep everyone on the same page. The coordinator will communicate across the development team to make sure people know what they need to do. I am fit for the position because of my communication skills and awareness. I will also be contributing to the development of the top-down exploration portion of the game.
    - Joseph: Designer / Developer
        - Our team needs several developers to work on various parts of the game, to ensure that no part falls behind schedule. I am fit for the role of Designer because I have played a wide variety of video games that all have their own unique traits and design choices, and I understand how developers use certain game mechanics to teach and engage with the player.
    - Wyatt: Designer/Developer
        - Our team needs multiple developers in order to address multiple areas of focus required to build the game we’re envisioning. As someone who has played several games where you customize your own character, build your own teams, and create your own buildings, I am fit for the role of designer. I also pay close attention to the niche details most people skip over when designing characters and npc’s which will be a helpful trait when designing characters and the map throughout this project.
    - Owen: Developer / QA Engineer

        - Our team requires a diverse set of developers to handle the various types of development work. Having played numerous games, including several RPGs, and with experience developing an RPG in Unity, I bring relevant expertise to my role as both a QA Engineer and developer.

- Provide a schedule for each member (or sub-group) with a *measurable*, concrete milestone for each week. "Feature 90% done" is not measurable, but "use case 1 works, without any error handling" is.
    - Combat Team (Doug/Kaden)
        - Week 1: Explore different ideas for the combat system
        - Week 2: Formalize combat requirements
        - Week 3: make characters classes
        - Week 4: Create an ability structure
        - Week 5: Simple combat system
        - Week 6: Add in-game over system
        - Week 7: Create an Item system
        - Week 8: add the ability to use the items
        - Week 9: Bug hunting/polish game
        - Week 10: Ensure successful product launch

    - Overworld Team (Graham/Joseph)
        - Week 1: Form the team.
        - Week 2: Connect unity + github.
        - Week 3: Player Movement Use Case works, no error handling.
        - Week 4: Obstacles stop the player's movement.
        - Week 5: Map design is complete; The Player can only move within the set bounds of the map.
        - Week 6: Encounter Use Case works without error handling. At least 4 combat encounters are included on the map.
        - Week 7: Game over resets the player's position to the start.
        - Week 8: Talk to NPC use case works without error handling. At least 3 NPCs are included on the map.
        - Week 9: Use cases above work with error handling. At least 2 Points of interest are added to the map.
        - Week 10: All sprites, animations and assets are loaded in. Use cases still work as expected with error handling.

    - Player Progression (Wyatt)
        - Week 1: Flesh out a basic character design

        - Week 2: Establish an experience system and player class with stats that will be used
        - Week 3: Create basic UI including messages and a level up screen
        - Week 4: Work with other teams to determine specific animations and/or sprites for levelling up
        - Week 5: Assign multiple player classes to all entities that can level up
        - Week 6: Begin implementing and testing in actual game to find errors
        - Week 7: Error checking
        - Week 8: continue to error check and work with other groups to fix issues
        - Week 9: IF done, consider adding more functionality (skills, abilities, etc.)
        - Week 10: wrap up

    - Sprite/Animation Team (Jonathan/Owen)
        - Week 1: Formalize team assignments and determine key steps to complete for each week
        - Week 2: Determine the number of characters that will be necessary and develop 2-3 ideas for the main user character
        - Week 3: Create a basic working character design to allow other teams to test working capabilities
        - Week 4: Create a working animation for entering and leaving battle scenes without testing UX Design components
        - Week 5: Develop characters for enemies that will be fought throughout the game progression
        - Week 6: Create all “bot”/NPC characters that will be throughout the map
        - Week 7: Review all characters created to this point. Test each character to make sure they are properly implemented.
        - Week 8: Make adjustments to characters not working/appearing properly
        - Week 9: Test battle scene animations, and create an appealing animation that keeps users engaged
        - Week 10: Review all created characters/animations. Correct any remaining errors that are affecting the game flow.

- Specify and explain at least three major risks that could prevent you from completing your project.
    - None of our group members have worked on a project like this before, so if there is not a relevant tutorial that we can reference, we may not be able to create the features that we want.
    - Creating art may take too much time so we may have to cut corners on looks.
    - Different teammate schedules may keep us from meeting consistently.

- Describe at what point in your process external feedback (i.e., feedback from outside your project team, including the project manager) will be most useful and how you will get that feedback.
    - External feedback will be most helpful once we get the majority of our systems online towards the beginning of beta testing. This feedback can be given by the TA or anyone not involved in the project. We will mostly be interested in game feel and player experience. This info should allow us to tweak our final product to be more suitable for a wider audience.

Technical Approach

We will create the game using the Unity game engine and C #. The sprite assets will be created using pixel art and the online sprite editor Piskel.

Risks

Our team has limited experience with Unity, and when it comes to RPG's it is easy to overestimate what can be realistically completed in the timeframe. We will combat these risks by focusing on learning Unity early on as well as building the most integral features.

Features

Major

- Top Down Exploration
- Aesthetic Visuals
- Satisfying Encounter-Based Combat System
- Exciting Player Progression
- Quest Oriented Progression
- Item System

Stretch Goals

- Save/Load Game
- Items & Inventory
- Multiplayer
- Story
- Music/Sound Effects
- Celebration Animations

Core Elements

- Combat
    - Player Decision Menu
    - Enemy/Human Decision making(RNG, Decision tree, whatever we decide)

        - Battle Graphics
        - Descriptive Text (e.g. “*The enemy uses laser attack!*”,”*The human trips on a rock!*”, etc.)

- Top Down Exploration
    - Player Movement
    - Player Movement Animation
    - World Design

- Player Progression(leveling up)
- Sprite Designs

Secondary Elements
- Items/Player Inventory
- Save files
- NPC dialogue encounters -> overarching story
- World interactables(use axe to cut down tree)
- Quests

Instructions (Project Architecture and Design specifications) - DELETE AFTER

Add three new sections to your living document (Software architecture, Software design, and Coding guidelines) and revise the process description section. Additionally, *incorporate the feedback that you have already received*. **Add the updated version of your living document to the main branch of your repository by 02/04/25 11:59 PM.**

1. Software architecture

- Software Components:
    - Combat Scene
        - Enemy AI
    - Overworld Scene
        - Maps
    - Player class
        - Movement
    - AI class
        - AI AI
    - Movement
- Specify the **interfaces** between components.

- Overworld passes information about the player and AI to Combat so that it can properly render the fight. (as well as which enemy to render)
- Movement passes information to Overworld so that the player can move.
- AI and player class holds stat information for Combat.

- Describe in detail what **data** your system stores, and how. If it uses a database, give the high level database schema. If not, describe how you are storing the data and its organization.
    - Player class
        - Player stats (HP, resists, damage, etc)
        - Player abilities
    - AI class
        - AI stats
        - AI abilities (and chances for each)
    - Enemy Database
        - Stats for each encounter
        - Attack patterns

- If there are particular **assumptions** underpinning your chosen architecture, identify and describe them.
- **Alternatives**
    - Central Database
        - Holds all information about Player, AI, and Enemies in one, easily accessed location
    -

2. Software design

Provide a detailed definition of each of the software components you identified above.

- What **packages, classes, or other units of abstraction** form these components?
- What are the **responsibilities** of each of those parts of a component?

3. Coding guideline

www.geeksforgeeks.org/c-sharp-coding-standards/

The coding guidelines from the above link will be used for the project. These specific guidelines were chosen because they enhance the readability of the code. They also will force the developers to keep a clean code appearance that allows someone joining the project to easily understand what is going on.

To enforce these guidelines throughout the project, members of the team will review the code. One plan is to have coding “buddies” where members will get together and work on developing the program. During these meetings, one member will be coding, while the other watches and assists. This will improve the consistency of the code and ensure that the code will follow the guidelines.

4. Process description

Expand your process description to address the following five topics:

i. Risk assessment

Identify the top five risks to successful completion of your project.

For each, give:

- Likelihood of occurring (high, medium, low);
- Impact if it occurs (high, medium, low);
- Evidence upon which you base your estimates, such as what information you have already gathered or what experiments you have done;
- Steps you are taking to reduce the likelihood or impact, and steps to permit better estimates;
- Plan for detecting the problem (trivial example: running automated tests to determine that a file format has changed);
- Mitigation plan should it occur.

Explicitly state how this has changed since you submitted your Requirements document.

ii. Project schedule

Identify milestones (external and internal), define tasks along with effort estimates (at granularity no coarser than 1-person-week units), and identify dependences among them. (What has to be complete before you can begin implementing component X? What has to be complete before you can start testing component X? What has to be complete before you can run an entire (small) use case?) This should reflect your actual plan of work, possibly including items your team has already completed.

To build a schedule, start with your major milestones (tend to be noun-like) and fill in the tasks (tend to start with a verb) that will allow you to achieve them. A simple table is sufficient for this size of a project.

iii. Team structure

Make sure to update your team structure, if necessary, and provide more details about team organization and team members' roles and responsibilities.

iv. Test plan & bugs

Describe what aspects of your system you plan to test and why they are sufficient, as well as how specifically you plan to test those aspects in a disciplined way. Describe a strategy for each of unit testing, system (integration) testing, and usability testing, along with any specific test suites identified to capture the requirements.

We require that you use GitHub Issues

Links to an external site.

to track bugs that occur during use and testing.

v. Documentation plan

Outline a plan for developing documentation that *you plan to deliver with the system*, e.g., user guides, admin guides, developer guides, man pages, help menus, wikis, etc.

Clarifications

What principles should our software architecture and design follow?

- Strive for modularity and testability.
- Use encapsulation.
- Keep related data and behavior in the same place.
- Emphasize cohesion, limit coupling, and do not pollute public interfaces.
- Emphasize separation of concerns: avoid "god classes" that do too much.
- Avoid overengineering and complexity: avoid insignificant or irrelevant classes that do too little.
- Make sure you can describe the purpose and functionality of each class concisely and clearly.
- Keep the data model decoupled from the UI (view).
- Allow for features to be developed in parallel as much as possible.

Any other writing hints?

In evaluating your work, we will check whether your living document

1. addresses all the necessary elements of defining the architecture and design of your system,
2. provides reasonable justifications for decisions made,
3. provides a reasonable schedule and set of tasks, and
4. shows general improvements, in particular to the process section.

Be brief. Your living document should address the required points briefly and clearly. The staff will deduct points for overly long, poorly organized, or poorly explained documents.

Do not include **content-free filler** in your living document. This includes anything that could appear in identical form in another group's materials. For example, don't merely state as a risk, "we might fall behind schedule" (another example is "we don't know the tools"). What factors do you think are most likely to cause schedule slippage for your project? Why do you think those are the parts of the schedule with the greatest potential variance? What have you done, or what do you plan to do, to learn more about how long those particular tasks will really take?

Your living document should clearly describe (and possibly visualize) the system architecture. For example, if you are using the Model-View-Controller architecture, your living document should make this clear and provide references to the design section, which in turn clearly describes which classes implement the model, the view, and the controller of your system.

Avoid duplicated information and make good use of cross-references.

DELETE THIS SECTION ONCE FINISHED