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. Humanity is escaping to Mars. All of the humans left on Earth have mutated into horrific creatures, except for one. Your job is to escort the last living human, named Jason, safely to the launchpad.

The robot and human spawn in a desert wasteland. Three stages away is the launchpad. Each stage includes more difficult encounters. The robot leads the team of two with Jason following behind. They travel toward the end of each stage to the right, encountering points of interest, enemies, and items. Colliding with a point of interest shows a blurb about the world with a button for the player to close the pop-up. Colliding with an item allows the player to select whether to pick it up. On the side of the screen there is a button to open inventory. When the player picks up an item it is added there. The player can equip only a couple of items at a time.

The map includes enemy encounters that enter the player into a turn-based battle. To initiate combat, the player moves until the robot collides with an enemy. A pop-up comes up to choose whether to initiate combat. If the player selects yes, the fight starts after a brief transition. If the player selects no, there is still a chance that the combat will initiate anyway. For each turn, the player must choose an action to keep Jason alive. The player starts with 3 options every turn: defend, taunt, or empower. The enemies have two attacks: weak or powerful. They either target the robot or Jason. 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. Each round, Jason will also act, though they may not always be helpful. He has a percent chance to do each of his abilities which include attack, special attack and do something stupid. Jason has a reduced chance to choose the same option twice in a row. During a battle, a text box shows the actions taken by each participant. The robot gains a new ability after each of the first two bosses. The abilities gained will be “heal” and “stun”. Every few encounters you win, your team(robot and Jason) level up and their stats automatically increase. Stats include: Health(both), Skill Recovery(robot), Speed=Turn order priority(both), and Damage(human). If Jason dies, you must start the game over from the beginning. If the robot dies in combat, but Jason manages to defeat the enemy, a new identical robot replaces the old one.

At the end of the first two stages are mini-boss enemies which block the path to the next stage. At the end of the last stage is the final boss. Defeating the final boss allows Jason to get on the spaceship to mars, joining the other survivors.

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 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

Software Toolset

    - 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.

Team Structure

- 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

- **Top-Down Exploration**
    - Player Movement
    - Player Movement Animation
    - World Design
- **Combat System**
    - Satisfying Encounter-Based Combat
    - Player Decision Menu
    - Enemy/Human Decision Making (RNG, Decision Tree, etc.)
    - Battle Graphics
    - Descriptive Combat Text (e.g., "The enemy uses laser attack!", "Jason trips on a rock!")
- **Player Progression**
    - Leveling System
    - Sprite Designs
- **Item System & Inventory**
    - Items & Player Inventory
    - World Interactions (e.g., use axe to cut down a tree)
- **Story & Worldbuilding**
    - NPC Dialogue & Encounters
    - Overarching Story
- **Stretch Goals**

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

Software architecture

- Software Components:
  - Combat Scene
    - Enemy AI
    - Combat system
    - A dialogue system that explains what attack each person is using
    - Stat systems (xp, and strength, health)
  - Overworld Scene
    - Maps
    - NPC system to allow you to talk to NPCs
    - Interactable environment (use an axe to cut down a tree)
  - Player Component
    - Movement
    - Player Class (stats/items)
      - inventory
  - AI class
    - AI pathfinding
    - Stats
    - inventory
    - Dialogue when talked to
  - Movement
    - Movement speed

- Specify the **interfaces** between components.
  - Overworld Scene to combat scene
    - Passes stats, AI stats, abilities
    - Triggers the combat UI and loads the combat scene
  - Movement to Overworld scene
    - Updates the position of the player based on the movement imputed
    - Triggers collisions to make a pop-up to allow the user to choose whether to initiate combat
  - Ai and player classes to combat system
    - Stores stats, abilities, and inventory so you can use it in combat

    - Dialogue system
        - Gets the dialogue based on the encounter and NPC
    - Interactive environment
        - Allows for gathering items from the environment

- 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, xp, level, speed)
        - Player abilities list
        - Player inventory
            - Uses an array to store objects within unity
    - AI class
        - AI stats (HP, resists, damage, speed)
        - AI abilities list (and chances for each)
        - Possible dialogue
    - Enemy Database
        - Stats for each encounter (types, stats)
        - 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
            - Pros: Everything is stored in one place
            - Cons: Need access to database at all times
    - Event-driven approach
        - Information stored in memory and tracked through event listeners and updated in real time
            - Pros:
                - Improves performance by reducing data fetches
                - Makes it modular and easier to expand
            - Cons:
                - More complex to implement due to event dependencies

        - Requires more management to prevent memory leaks

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?
    - Combat Scene
        - The combat scene takes information from the overworld scene in order to render the correct enemy encounter.
        - Combat relies on the Player and AI classes to know what both the player character and the AI Human are allowed to do at that point of the game.
    - Overworld Scene
        - The overworld is formed by textures and sprites to create the environment that the player will walk through, which will be found using the Unity Asset Store. The overworld will also use colliders to create boundaries, ensuring the player cannot walk outside the design of the map or through objects on the map.
    - Player Component
        - Several classes
    - AI Class
        - Follows player class
    - UI
        - Packages include TextMeshPro
        -
- What are the **responsibilities** of each of those parts of a component?

Coding guidelines

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:

1. 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. There is a low likelihood that a relevant tutorial will not be found. Even though our game is an original idea, it utilizes many of the features that a classic RPG has and even the idea of creating another entity that can't be controlled and must be protected isn't difficult to find. If we are unable to find a tutorial, it will only have a low impact on the completion of the project because of the number of people collaborating to complete the project. These estimates are based on what was already found by doing simple searches on YouTube. The tutorial videos found were relevant and usable for the project and easy to find and understand. This is also only what was found on YouTube and since there are so many other sources that can be used to find tutorial videos and answer questions on how to utilize Unity, it seems that the likelihood of not being able to find a relevant tutorial is very low. Steps that are being taken to reduce the likelihood or impact to permit better estimates include bookmarking videos that seem useful, regular group meetups for better collaboration, and setting realistic goals each week to give a clear idea of what needs to be found, worked on, and finished each week in order for the project as a whole to be completed on time. Should we be unable to find a relevant tutorial video, the group will meet up and collaborate on the feature that is attempting to be created. If there is still struggle, consulting the TA is the next option and asking for further assistance from either the professor or other TA's for more resources or advice on how to code the feature.
    a. Low probability
    b. Low impact

    c. Relevant tutorials were easy to find through a simple search on YouTube. Based on the ease of that search, it's realistic to assume that if a more specific search is conducted, the desired information will be found.
    d. Bookmarking relevant information, group collaboration, setting goals to give a clear idea of what needs to be searched for in order to achieve goals.
    e. Looking in advance for videos that might pertain to the project beforehand. This allows for more time to plan out how to program the desired feature.
    f. Group collaboration and if the group is still unable to figure out how to code the desired feature or resolve an issue, consult the TA's and professor for advice or recommendations on resources to troubleshoot.
2. Creating art may take too much time so we may have to cut corners on appearance. There is a pretty high probability that this might happen, but the impact should be rather low. While timelines are still adjusting slightly, we have based the timelines on our progress to this point. As of now, we are expecting about 1.5 weeks per map that we create, however, this may shorten as we get more comfortable with the software. To reduce the likelihood of this risk having a large impact, map creation was started early on to give sufficient time. It can be difficult to determine exactly how long the art will take, but keeping similar designs across maps will help us get a better estimate for the other maps we plan to develop. This problem is relatively easy to detect by looking at the progress of the map every couple of days and seeing how much is left to complete. Another plan to detect issues with art is to run automated tests for character movement interacting correctly with art features. Should there be a need to make adjustments, we have determined a range for how many maps are necessary, allowing us to remove maps or adjust the sizes to increase production time.
    a. High probability of occurring
    b. Low impact
    c. While timelines are still adjusting slightly, we have based the timelines on our progress to this point
    d. To reduce the likelihood of this risk having a large impact, map creation was started early on to give sufficient time. It can be difficult to determine exactly how long the art will take, but keeping similar designs across maps will help us get a better estimate for the other maps we plan to develop
    e. This problem is relatively easy to detect by looking at the progress of the map every couple of days and seeing how much is left to complete. Another plan to detect issues with art is to run automated tests for character movement interacting correctly with art features

    f. Should there be a need to make adjustments, we have determined a range for how many maps are necessary, allowing us to remove maps or adjust the sizes to increase production time.
3. Different teammate schedules may keep us from meeting consistently. There is a high probability of this occurring. While group members can sometimes join discord if they're unable to attend in person, it's difficult to find a time when 7 different people can collectively meet for an extended period of time with no other engagements. Currently, the impact of not being able to meetup is low but as the deadline for the project approaches, the impact of not being able to all meet up as a group will increase gradually as the urgency for the work to be done increases. While the group has scheduled several meetings outside of class time, There have only been 2 out of 6 occurrences where all 7 group members were able to meet at the scheduled time. However, work was still done and progress on the project is still being made at a steady pace. Depending on the difficulty of the feature being programmed, the frequency of meetings might need to be increased and attendance of these meetings becomes more and more important as the deadline approaches as mentioned before. In order to combat this issue, group meetings should be planned as early in advance as possible and for those who are unable to attend, they should notify the group as soon as possible so that the group can adjust accordingly. If there is no chance that a group member can meet consistently, they should update the trello board as often as possible and keep the group informed online of the progress that they're making and if they need any assistance in what they're working on.
    a. Low → High probability
    b. Low → High impact
    c. Out of the 6 total meetups that have been conducted outside of class time, only 2 meetups had all 7 group members present. Group members were brought up to speed during mandatory meetings and work was still done in a timely manner but it might be more difficult as bigger deadlines approach.
    d. Planning to meet far in advance, keeping the trello board with goals and tasks to do updated, letting team members know in advance if they're available or not and how they're doing on the completion of their respective tasks.
    e. Everyone states what days during the week that they're free and if they're unable to attend a meeting in person, their online availability. If they're unable to attend online, what they plan to work on and communicate that with the group.
    f. If no one is able to meetup, a discord message should be sent stating this as well as what they intend to work on.

4. Github pushing errors
    a. Medium likelihood
    b. High impact
    c. Github has had issues pushing the correct files to the repository. Sometimes it adds unnecessary log files that take up required space. This can cause us to hit the cap size for a github repository.
    d. We have updated our .gitignore to tell it not to push these files, but sometimes it still does.
    e. The problem is easy to detect as whenever you push to source, you can see the file types that are being pushed.
    f. If any .log files are present or other unnecessary files, they can be deselected.
    g. This risk has been present from the start but didn't quite make it into our Requirements document.
5. Github merging errors
    a. Medium likelihood
    b. High impact
    c. When 7 people are working on the same project, we are likely to run into merge conflicts when branches are merged into main.
    d. We plan to pull into main as often as possible to reduce potential merge conflicts.
    e. The problem is easily detectable as github automatically shows any merge conflicts when a merge is attempted.
    f. Our plan is to look through the code and figure out what the problems are in an attempt to fix the conflict. If this is not possible, we may need to revert to an older version of the project.
    g. This risk has become much more likely since we submitted the Requirements as we have started using Github more.

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.

Major Milestones:	Tasks to Achieve Milestones:
Map Creation	● Create Paper Prototypes for Maps ● Gather assets (textures, sprites) ● Create tilemap for floor + edges ● Create tilemap for sprites (items, interactable objects,) ● Add physics to objects ● Add pop-up menus for item pickups, points of interest, enemy interactions
Player Movement	● Create player movement script ● Create terrain collision logic ● Create collision logic for triggering encounters
Player Inventory	● Create the inventory script to create the actual inventory ● Create interactable items that can be stored in the inventory ● Assign effects to the given items ● Gather assets (sprites) ● Create inventory UI ● Refine and test all features
Combat Scenes	● Create combat animations ● Develop
Player Progression	● Create the abstract item object ● Create public player stats ● Make player stats scale off of xp ● Make player xp increase on encounter completion ● Make abilities unlock on level completion

iii. Team Structure

This has been updated in the Team Process Description section earlier. Not much has changed this week.

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

For our game, we will develop a brief guide book to inform the player about the mechanics of the game: How combat works, how to progress through stages, how to use items, etc. This likely will be written once a playable version of the game is ready.