Project: Procedurally-generated 2D Role-playing Game

Marcos Sanson, Jerod Muilenburg, William Krol, Brendon Do, Ely Miller

CIS 350: Winter 2024

Table of Contents

Project Information

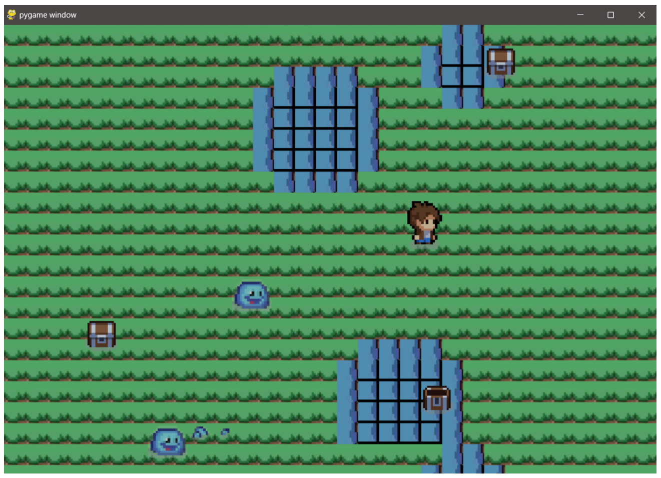
This project consists of the development of a 2D role-playing game (RPG). Our team consists of 5 people. In the game, players will assume the role of a customizable character navigating through diverse procedurally-generated terrains filled with interactive objects and water bodies. The game will feature a storyline with encounters, random events, and quest-driven gameplay, allowing players to shape their own adventure. Players will engage in combat with various creatures and monsters inhabiting the game world, utilizing their character's abilities and equipment upgrades. Access to tutorials, game manuals, and in-game help guides are also available. The game will provide regular updates and patches to enhance gameplay mechanics and address any issues.

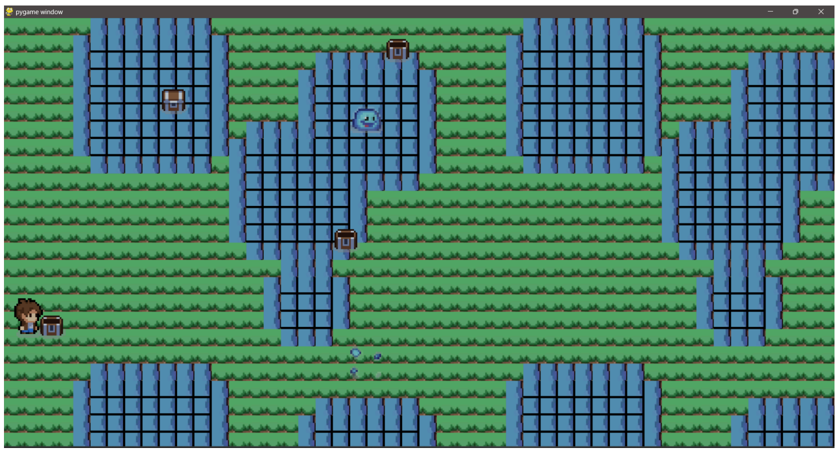
Features

Has the following features for release 1:

- Character movement with arrow keys or WASD
- Collision detection with water and objects.
- Randomly generated terrain with clusters of water.
- Randomly generated objects with collision detection.
- Randomly generated enemies with collision detection, health systems, and defeat animations.

Screenshots





Project Plan

Our project plan follows the waterfall model with a prototype as customization to better address iterative development needs. We initially focused on gathering requirements and designing the game mechanics, including character customization, terrain generation, and object interactions. Then, we started with developing a prototype of the game using Python and game development packages/tools/resources. Testing and verification will be conducted later in the process (in accordance with the waterfall model) to ensure functionality and identify any bugs or issues. Through testing, we'll verify that the game meets all requirements and is ready for release. Finally, we'll deliver the finished product to players and provide ongoing support and updates as needed to enhance the gaming experience.

Requirements & Definition

In the requirements and definition phase, we brainstormed ideas to come up with user needs and project scope. We identified key requirements, such as character customization, interactive terrain elements, and combat mechanics. These requirements were put into the use case diagram we have further down in this document. Following the document is our traceability matrix covering the cross of functions and the requirements they accomplish. Additionally, we created physical drawings and outlines to visualize and refine these concepts. This phase laid the foundation for the development process, guiding the next stages of design and implementation.

Development

For the development and implementation phase, we used Python as our primary programming language, along with relevant packages such as Pygame for game development. We divided the project into manageable tasks and assigned them to team members based on their expertise and interests. Throughout this phase, we followed coding standards (e.g., docstrings) and best practices to ensure code quality and maintainability. Regular code reviews and testing were conducted to identify and address any issues promptly. Collaboration tools such as version control systems (Git) were used to manage collaboration between team members.

Verification

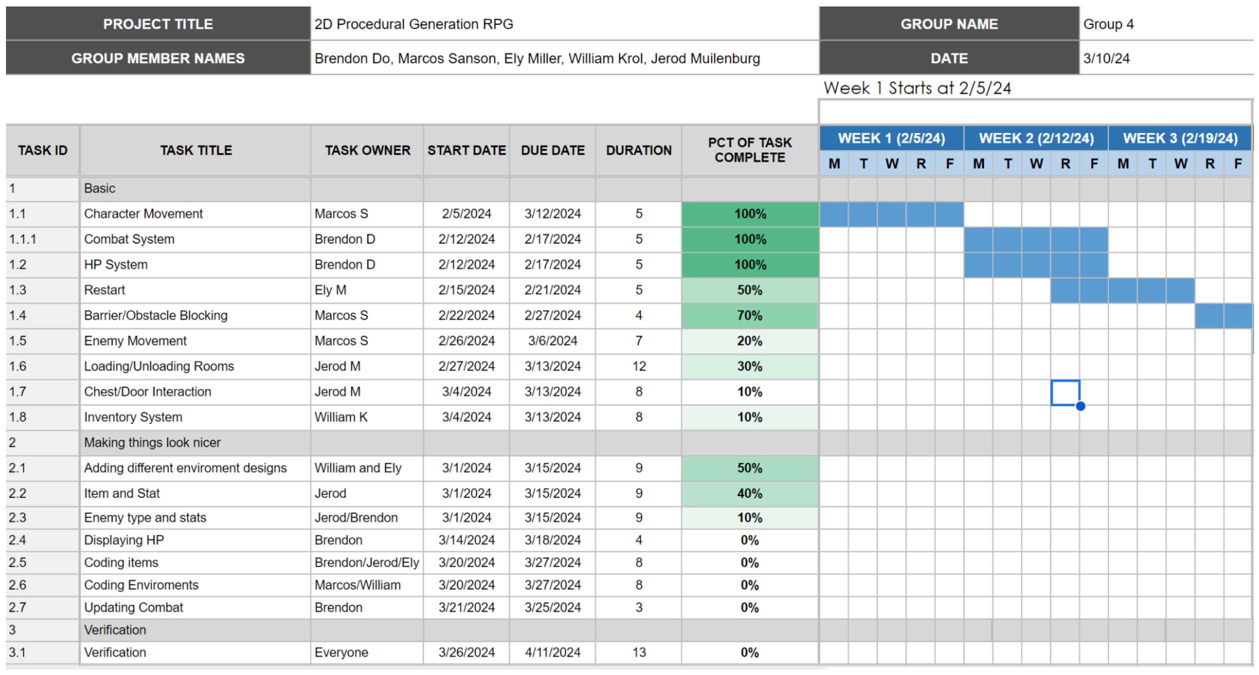
Verification will be done between the first release and the second release displayed in the gantt chart. We plan to start the verification process during Week 8 on Tuesday the 26th in March. We have tasked all team members to do verification of our own tasks within the project.

Maintenance

For maintenance, we will address any problems found by the professor or during updates for the next release. We'll prioritize fixing any bugs or issues reported by users to ensure the game operates smoothly. Additionally, we'll regularly update the game with new features and improvements based on feedback from players. We will also fix any issues identified by individual members of our group, if they discover problems. For tracking the issues we are internally discussing any issues that arise in our discord so we all can prioritize the fixing process. We will also look at the github issues page for any bugs that are reported.

Umbrella Activities

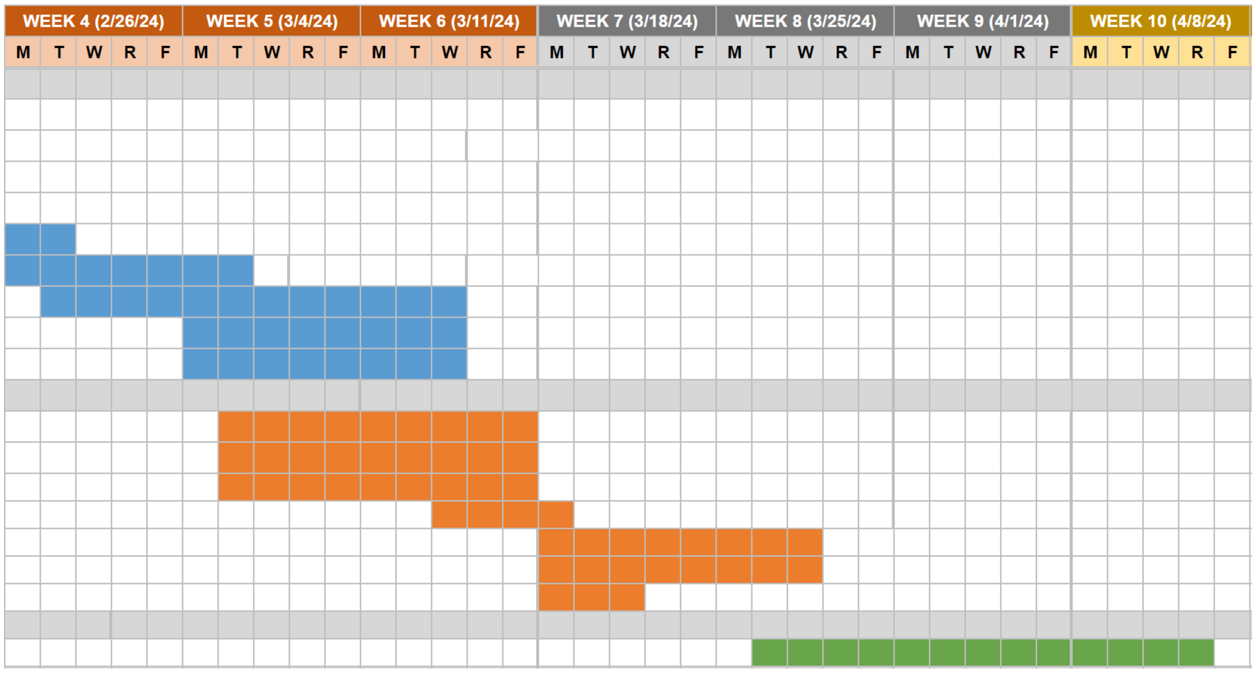
For project management and status tracking, we will hold regular meetings to discuss progress, address any issues, and make decisions regarding the project's direction. We also constructed a Gantt Chart to help keep track of tasks needed to be done. We currently meet and talk over Discord every week. Meetings involve all team members to ensure everyone is informed and aligned. During these meetings, we review the current status of tasks, identify any obstacles, and determine the next steps to keep the project on track. This will help us to manage the project efficiently and successfully complete the project.



PROJECT TITLE	2D Procedural Generation RPG	**GROUP NAME**	Group 4
GROUP MEMBER NAMES	Brendon Do, Marcos Sanson, Ely Miller, William Krol, Jerod Muilenburg	**DATE**	3/10/24

Week 1 Starts at 2/5/24

TASK ID	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	WEEK 1 (2/5/24)					WEEK 2 (2/12/24)					WEEK 3 (2/19/24)				
							M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	Basic																				
1.1	Character Movement	Marcos S	2/5/2024	3/12/2024	5	100%	■	■	■	■	■										
1.1.1	Combat System	Brendon D	2/12/2024	2/17/2024	5	100%						■	■	■	■	■					
1.2	HP System	Brendon D	2/12/2024	2/17/2024	5	100%						■	■	■	■	■					
1.3	Restart	Ely M	2/15/2024	2/21/2024	5	50%									■	■	■	■	■		
1.4	Barrier/Obstacle Blocking	Marcos S	2/22/2024	2/27/2024	4	70%														■	■
1.5	Enemy Movement	Marcos S	2/26/2024	3/6/2024	7	20%															
1.6	Loading/Unloading Rooms	Jerod M	2/27/2024	3/13/2024	12	30%															
1.7	Chest/Door Interaction	Jerod M	3/4/2024	3/13/2024	8	10%															
1.8	Inventory System	William K	3/4/2024	3/13/2024	8	10%															
2	Making things look nicer																				
2.1	Adding different enviroment designs	William and Ely	3/1/2024	3/15/2024	9	50%															
2.2	Item and Stat	Jerod	3/1/2024	3/15/2024	9	40%															
2.3	Enemy type and stats	Jerod/Brendon	3/1/2024	3/15/2024	9	10%															
2.4	Displaying HP	Brendon	3/14/2024	3/18/2024	4	0%															
2.5	Coding items	Brendon/Jerod/Ely	3/20/2024	3/27/2024	8	0%															
2.6	Coding Enviroments	Marcos/William	3/20/2024	3/27/2024	8	0%															
2.7	Updating Combat	Brendon	3/21/2024	3/25/2024	3	0%															
3	Verification																				
3.1	Verification	Everyone	3/26/2024	4/11/2024	13	0%															



Responsibilities

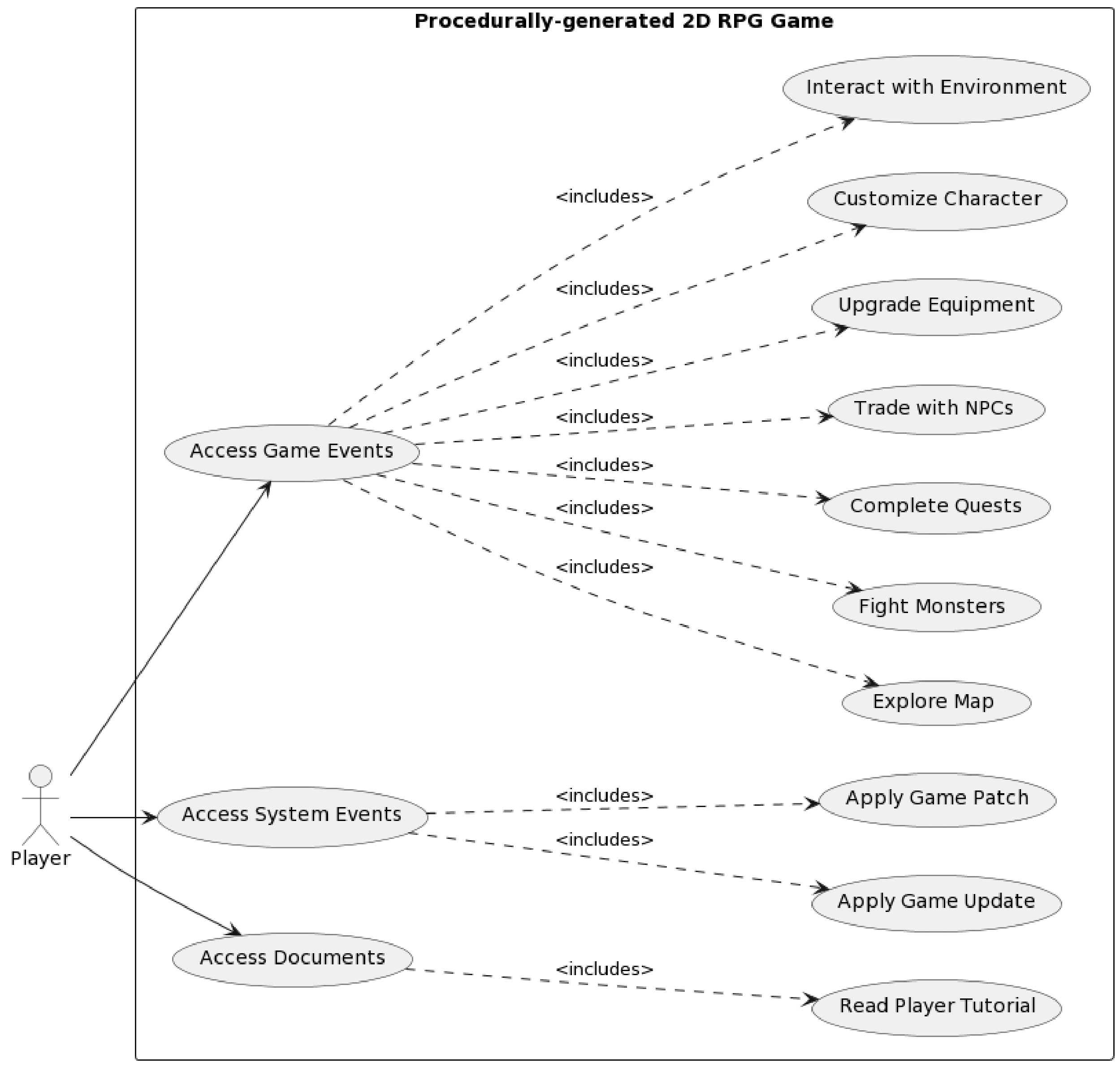
Brendon Do - Developing HP and Combat system
Marcos Sanson - Developing basic movement and terrain, enemy, and object procedural generation algorithms
Jerod Muilenburg - Developing weapons, enemies, NPCs, and all their associated stats/sprites/animations, along with descriptions and use with procedural generation
William Krol - Creating new, initial terrain features, inventory system and help with procedural generation
Ely Miller - Designing different room environments and obstacles to be procedurally generated

We break down the work by each team member's interests, and communicate over Discord and GitHub to collaborate effectively and ensure our code works with each other's designs.

Requirements & Specification

We used multiple different methods in our requirements and specification, including a use-case diagram, use case description, natural language requirements, and a traceability matrix.

Use-Cases



Use Case Descriptions:

Use Case: Access Game Events
Actors: Player

Description:
1.) The Player experiences various in-game events that affect gameplay, progression, and narrative.
2.) These events may include scripted encounters, random encounters, and story-driven plot developments.
3.) The Player's interactions and decisions during game events influence their gameplay experience and shape the outcome of the game.
Pre-Condition: Player must have started the game.

Use Case: Access System Events
Actors: Player
Description:
1.) The Player receives notifications and updates regarding system-related events and processes within the game.
2.) These events may include game updates and patches.
3.) The Player's awareness of system events ensures they stay informed about updates.
Pre-Condition: Player must have started the game.

Use Case: Access Documents
Actors: Player
Description:
1.) The Player accesses various documents and resources provided within the game for reference and information.
2.) These documents may include player tutorials, game manuals, and in-game help guides.
3.) The Player can read documents to gain insights, strategies, and background information about the game world and its mechanics.
Pre-Condition: Player must have started the game.

Use Case: Interact with Environment
Actors: Player
Description:
1.) The Player interacts with various elements and objects within the game environment, including interactive scenery, environmental puzzles, hidden secrets, and interactive objects.
2.) The Player's can discover hidden treasures, unlocking shortcuts, or triggering events that affect gameplay.
3.) The Player can interact with NPCs, objects, and elements using contextual actions such as talking, examining, or using items.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Customize Character
Actors: Player
Description:
1.) The Player personalizes and customizes their character's appearance, attributes, and abilities according to their preferences.

2.) The Player selects character classes, modifies physical features, chooses skills and talents, and allocates stat points to customize their character.
3.) The Player may unlock additional customization options through gameplay achievements or progression.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Upgrade Equipment
Actors: Player
Description:
1.) The Player enhances their character's equipment, weapons, and gear to improve combat effectiveness and performance.
2.) The Player upgrades weapons, armor, accessories, and other items using in-game currency, materials, or resources.
3.) The Player may need to meet certain requirements or fulfill specific conditions to perform equipment upgrades.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Trade with NPCs
Actors: Player
Description:
1.) The Player interacts with non-player characters (NPCs) to engage in buying, selling, and trading items, equipment, and resources.
2.) The Player can acquire valuable goods, obtain rare items, or sell surplus inventory to NPCs encountered throughout the game world.
3.) The Player can negotiate prices with NPCs during trade interactions.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Complete Quests
Actors: Player
Description:
1.) The Player starts various quests and missions offered within the game.
2.) These quests may involve tasks such as defeating specific enemies, retrieving items, escorting NPCs, or exploring hidden areas.
3.) The Player earns rewards, experience points, and progresses the storyline by completing quests.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Fight Monsters
Actors: Player
Description:
1.) The Player engages in combat with hostile creatures and monsters inhabiting the game world.
2.) The Player utilizes their character's skills, abilities, and equipment to defeat enemies and progress through the game.

3.) The Player may use items such as potions or buffs during combat to enhance their performance.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Explore Map
Actors: Player
Description:
1.) The Player navigates through different areas of the game world, discovering new locations, landmarks, and points of interest.
2.) The Player encounters various terrains, biomes, and environmental elements during exploration.
3.) The Player may encounter non-player characters (NPCs), monsters, and other entities while exploring.
Pre-Condition: Player must have started the game and accessed game events.

Use Case: Apply Game Patch
Actors: Player
Description:
1.) The Player receives notification of an available game patch.
2.) The Player initiates the patching process from within the game menu.
3.) The game downloads and installs the patch, addressing specific issues, bugs, or balancing adjustments.
Pre-Condition: Player must have started the game and accessed system events.

Use Case: Apply Game Update
Actors: Player
Description:
1.) The Player receives notification of an available game update.
2.) The Player initiates the update process from within the game menu.
3.) The game downloads and installs the update, applying changes to the game environment, mechanics, and content.
Pre-Condition: Player must have started the game and accessed system events.

Use Case: Read Player Tutorial
Actors: Player
Description:
1.) The Player accesses the tutorial section from the game menu.
2.) The Player navigates through tutorial topics covering game controls, mechanics, and features.
3.) The Player gains knowledge and understanding of gameplay concepts and strategies through tutorial instructions.
Pre-Condition: Player must have started the game and accessed Documents.

Natural Language Requirements

Natural Language Requirements:

Req1: The game system shall present in-game events to the Player (including scripted encounters, random encounters, and story-driven plot developments) when the Player is actively engaged in gameplay.
Req2: The game system shall allow the Player to interact with in-game events when the events occur within the Player's current game session.
Req3: The game system shall ensure that the Player's decisions during game events impact their gameplay experience during the current playthrough.
Req4: The game system shall notify the Player of system-related events and processes within the game when the Player is logged into the game.
Req5: The game system shall provide the Player with updates to ensure they stay informed about system events when the Player is logged into the game.
Req6: The game system shall grant access to various documents and resources within the game for reference and information when the Player accesses the game's menu or help section.
Req7: The game system shall include player tutorials, game manuals, and in-game help guides among the accessible documents when the Player accesses the game's menu or help section.
Req8: The game system shall allow the Player to read documents to gain insights, strategies, and background information about the game world and its mechanics when the Player accesses the game's menu or help section.
Req9: The game system shall enable the Player to interact with various elements and objects within the game environment when the Player explores the game world.
Req10: The game system shall allow the Player to customize their character's appearance, attributes, and abilities when the Player accesses the character customization menu.
Req11: The game system shall enable the Player to upgrade their character's equipment, weapons, and gear using in-game currency, materials, or resources when the Player accesses the equipment upgrade menu.
Req12: The game system shall facilitate trading between the Player and non-player characters (NPCs) for items, equipment, and resources, allowing buying, selling, and trading interactions when the Player interacts with NPCs within the game world.
Req13: The game system shall allow the Player to start various quests and missions offered within the game when the Player interacts with quest NPCs or quest boards within the game world.
Req14: The game system shall reward the Player with experience points, rewards, and storyline progression upon completing quests when the Player completes quest objectives.
Req15: The game system shall enable the Player to engage in combat with hostile creatures and monsters inhabiting the game world when the Player encounters these enemies during exploration.
Req16: The game system shall provide the Player with the ability to explore different areas of the game world, discovering new locations, landmarks, and points of interest when the Player moves through different regions within the game world.

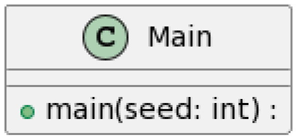
Req17: The game system shall notify the Player of available game patches and enable the Player to initiate the patching process from within the game menu when the Player is logged into the game and a patch is available.
Req18: The game system shall download and install game patches, addressing specific issues, bugs, or balancing adjustments when the Player initiates the patching process.
Req19: The game system shall notify the Player of available game updates and enable the Player to initiate the update process from within the game menu when the Player is logged into the game and an update is available.
Req20: The game system shall download and install game updates, applying changes to the game environment, mechanics, and content when the Player initiates the update process.
Req21: The game system shall provide the Player with access to player tutorials from the game menu, covering game controls, mechanics, and features when the Player accesses the tutorial section from the game menu.
Req22: The game system shall enable the Player to navigate through tutorial topics to gain knowledge and understanding of gameplay concepts and strategies when the Player accesses tutorial topics from the game menu.

Traceability Matrix

Use Cases / Requirements	**Req1**	**Req2**	**Req3**	**Req4**	**Req5**	**Req6**	**Req7**	**Req8**	**Req9**	**Req10**	**Req11**	**Req12**	**Req13**	**Req14**	**Req15**	**Req16**	**Req17**	**Req18**	**Req19**	**Req20**	**Req21**	**Req22**
Access Game Events	X	X	X																			
Access System Events				X	X																	
Access Documents						X	X	X														
Interact with Environment									X	X	X											
Customize Character										X	X											
Upgrade Equipment											X	X										
Trade with NPCs													X	X								
Complete Quests															X	X						
Fight Monsters																	X	X				
Explore Map																			X	X		
Apply Game Patch																		X	X			
Apply Game Update																				X	X	
Read Player Tutorial																						X

Design

In our current code, we only use a single “main” function and no classes. Shown below is an extremely simple class diagram in UML format that represents this main function.



Function Description:

main(seed: int): This function initializes the game environment, including setting up the Pygame display, loading game assets (such as character images, terrain tiles, and enemy sprites), and generating the game world (terrain with water clusters and object/enemy positions). It then enters a continuous loop where it handles player input, updates game state (including character movement and collision detection), and renders the game elements on the screen. The function also manages events such as window resizing and quitting.

We are planning to implement a game class and create separate functions/methods to better manage the game system, scale everything accordingly, and also use object-oriented programming and design.

Development

For this project, we are primarily relying on the Pygame library as it is one of the easiest and biggest game libraries that allow us to do what we want to do for this project. Pygame is used extensively in our code to handle multiple aspects of game development, such as creating the game window, loading and displaying images, handling user input (keyboard events), managing the game loop, and rendering game elements on the screen. It specifically provides functionalities like loading image assets (pygame.image.load()), drawing images on the screen (screen.blit()), handling keyboard events (pygame.key.get_pressed()), and updating the display (pygame.display.update()).

We also use Asyncio. which is a Python library used for asynchronous programming. Asyncio is used to run the main game loop asynchronously (our main method, the async def main() function, runs asynchronously). The “await asyncio.sleep(0)” statement inside the game loop ensures that everything runs concurrently, which maintains smooth gameplay without blocking the event loop.

We also use the random module in Python, which provides functions for generating pseudo-random numbers. This is the basis of our procedural generation algorithms. The random module is used to seed the random number generator (random.seed(seed)) at the beginning of the program to ensure reproducibility of random sequences, similar to other video games or procedural generation algorithms. It’s also used throughout our code to generate random values for various game elements, such as the number of clusters of water, the size and position of terrain clusters, the number and position of objects, and the number of enemies.

Code Standards

This is the result from running the package Pycodestyle to check our code against pep8:

```
.\main-hp-combat.py:25:1: E302 expected 2 blank lines, found 1
.\main-hp-combat.py:35:80: E501 line too long (96 > 79 characters)
.\main-hp-combat.py:42:80: E501 line too long (95 > 79 characters)
.\main-hp-combat.py:48:80: E501 line too long (85 > 79 characters)
.\main-hp-combat.py:50:80: E501 line too long (85 > 79 characters)
.\main-hp-combat.py:51:80: E501 line too long (114 > 79 characters)
.\main-hp-combat.py:52:80: E501 line too long (114 > 79 characters)
.\main-hp-combat.py:57:80: E501 line too long (86 > 79 characters)
.\main-hp-combat.py:58:80: E501 line too long (84 > 79 characters)
.\main-hp-combat.py:59:80: E501 line too long (116 > 79 characters)
.\main-hp-combat.py:75:80: E501 line too long (111 > 79 characters)
.\main-hp-combat.py:76:80: E501 line too long (108 > 79 characters)
```

```
.\main-hp-combat.py:83:80: E501 line too long (132 > 79 characters)
.\main-hp-combat.py:87:80: E501 line too long (101 > 79 characters)
.\main-hp-combat.py:90:80: E501 line too long (85 > 79 characters)
.\main-hp-combat.py:94:80: E501 line too long (88 > 79 characters)
.\main-hp-combat.py:105:80: E501 line too long (93 > 79 characters)
.\main-hp-combat.py:116:80: E501 line too long (91 > 79 characters)
.\main-hp-combat.py:123:80: E501 line too long (80 > 79 characters)
.\main-hp-combat.py:124:80: E501 line too long (108 > 79 characters)
.\main-hp-combat.py:153:80: E501 line too long (111 > 79 characters)
.\main-hp-combat.py:158:80: E501 line too long (136 > 79 characters)
.\main-hp-combat.py:172:80: E501 line too long (103 > 79 characters)
.\main-hp-combat.py:179:80: E501 line too long (101 > 79 characters)
.\main-hp-combat.py:190:80: E501 line too long (125 > 79 characters)
.\main-hp-combat.py:191:80: E501 line too long (123 > 79 characters)
.\main-hp-combat.py:194:80: E501 line too long (110 > 79 characters)
.\main-hp-combat.py:195:80: E501 line too long (108 > 79 characters)
.\main-hp-combat.py:197:80: E501 line too long (96 > 79 characters)
.\main-hp-combat.py:198:80: E501 line too long (110 > 79 characters)
.\main-hp-combat.py:199:80: E501 line too long (108 > 79 characters)
.\main-hp-combat.py:212:80: E501 line too long (86 > 79 characters)
.\main-hp-combat.py:213:80: E501 line too long (91 > 79 characters)
.\main-hp-combat.py:217:80: E501 line too long (86 > 79 characters)
.\main-hp-combat.py:218:80: E501 line too long (91 > 79 characters)
.\main-hp-combat.py:230:80: E501 line too long (115 > 79 characters)
1       E302 expected 2 blank lines, found 1
35      E501 line too long (96 > 79 characters)
```

Some of the errors generated from pycodestyle are from comments, while the other lines that go past the max length amount are from pure code.

Static Analysis

This is the result from running Pylint to see if there are any static analysis errors.

```
************* Module main-hp-combat
main-hp-combat.py:35:50: E1101: Module 'pygame' has no 'RESIZABLE' member (no-member)
main-hp-combat.py:121:29: E1101: Module 'pygame' has no 'QUIT' member (no-member)
main-hp-combat.py:123:31: E1101: Module 'pygame' has no 'VIDEORESIZE' member
(no-member)
main-hp-combat.py:124:69: E1101: Module 'pygame' has no 'RESIZABLE' member (no-member)
main-hp-combat.py:129:16: E1101: Module 'pygame' has no 'K_LEFT' member (no-member)
main-hp-combat.py:129:39: E1101: Module 'pygame' has no 'K_a' member (no-member)
main-hp-combat.py:131:16: E1101: Module 'pygame' has no 'K_RIGHT' member (no-member)
main-hp-combat.py:131:40: E1101: Module 'pygame' has no 'K_d' member (no-member)
```

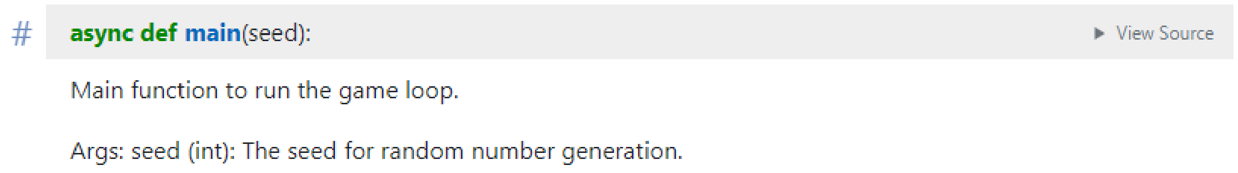
```
main-hp-combat.py:133:16: E1101: Module 'pygame' has no 'K_UP' member (no-member)
main-hp-combat.py:133:37: E1101: Module 'pygame' has no 'K_w' member (no-member)
main-hp-combat.py:135:16: E1101: Module 'pygame' has no 'K_DOWN' member (no-member)
main-hp-combat.py:135:39: E1101: Module 'pygame' has no 'K_s' member (no-member)
main-hp-combat.py:137:16: E1101: Module 'pygame' has no 'K_q' member (no-member)
main-hp-combat.py:139:16: E1101: Module 'pygame' has no 'K_z' member (no-member)
```

Pylint says there are errors with pygame not having those keyboard presses as a member, though the game controls still work when running the code.

Code Documentation

Link to generated documentation report:
https://github.com/Marcos-Sanson/CIS-350-Semester-Project/blob/Release-1/Prototype-HP-Combat/html/main-hp-combat.html
To view the report, download the HTML file to view the documentation generated from pdoc.

There are classes in our current Python code which are commented out that pdoc is aware of, but there is no documentation associated with the classes as they are incomplete. Our current code consists of one primary function, as outlined previously: the main(seed) function.


```
async def main(seed):
```

Main function to run the game loop.

Args: seed (int): The seed for random number generation.

(See link to report for more details and documentation)

Configuration Management

Our configuration management relies on branching in our GitHub repository. The way we are tracking releases is creating branches after the release that are not to be modified after the release. Currently for release 1, we have a branch named release-1 that shows the state of the Git repository during the first release.

Verification

Verification will be started on March 26, 2024 as stated by the gantt chart in the umbrella activities section.

Integration Tests

Verification will be started on March 26, 2024 as stated by the gantt chart in the umbrella activities section.

Unit Tests

Verification will be started on March 26, 2024 as stated by the gantt chart in the umbrella activities section.

Code Coverage

Verification will be started on March 26, 2024 as stated by the gantt chart in the umbrella activities section.

Requirements Coverage

Verification will be started on March 26, 2024 as stated by the gantt chart in the umbrella activities section.

Postmortem

Group:

- When we began we were excited by the possibilities our 2D procedurally generated game could accomplish. That sense of accomplishment led us to have many ideas for use cases and mechanics in the game that overall led to many tasks to be completed for our first prototype. This caused some of the tasks to be completed easily while others were more difficult resulting in those tasks not being completed by our first deadline. As a group, we had a more relaxed approach to the work assigned and now seeing the hill of work to be done, we now have a better understanding of how we are going to proceed going forward with our project. Overall, however, we have successfully developed a functional prototype with many of the systems we planned to have, including random generation for a variety of game aspects.

Individual:

- William: Going into this project I knew that this wouldn't be any easy task to finish in a couple days or so. That being said, looking back at my involvement with the project, I definitely see that I have not dedicated as much time as I would have liked to. I feel that my fellow group members have put more effort into their assignments than I have of my own. To them I have nothing but complements in their work. I am glad we have a working prototype to use as my biggest struggle is getting things off the ground. With a clear path forward I can be more resourceful and useful then I was prior. In doing so, I have seen how my choices are affecting the progress and I am determined to dedicate more of my available time to the project in any way that can further our progress.
- Marcos: I believe that the project has gone well so far. At this point, we have a functional game that takes user inputs smoothly and has legitimate procedural generation, which is the primary goal of this project. At this point, the game is already unique and varied with each seed entered by the user, which is fantastic. There are definitely more features to be added, but these build on our current features and game. The procedural generation algorithms function and they create decently realistic environments, although there are still bugs, such as being "trapped" in an area blocked off by water. Overall, while there were definitely more features I (and our group) could have implemented previously, overall the project has gone well and these features can all be reasonably added by the project deadline.
- Ely: I think we've made good headway on our project as a group, but in hindsight, I've fallen into bad work habits so far. Mostly, I felt that while I did some work on the project, including a task I was assigned to work on from the start, I didn't accomplish as much as I had set out to do. Part of this was uncertainty about the development process (GitHub, workflow, deciding on a design/structure for our codebase), but the lost time from not being sure what to do next could have been avoided sooner with more clear communication, which I've made a top priority for myself for the final release. By working more directly with my group and making sure the work is cut out clearly for me and the other team members, I think we can make quick progress on our goals from here on out.

- Brendon: Progress on the project could be better, though I am proud of what we have done so far. While we are behind on the gantt chart, the tasks ahead can still be done in a timely manner. I feel that I am pulling my weight and doing what I am tasked to do within the time frame, though it could have been improved with animations and HP being shown to the player on the screen. Overall, I think we have made good progress on the project, and can finish it in the time remaining.
- Jerod: I think we made a lot of progress toward our final goal. We have a good structure and a working prototype that can be expanded to what we are looking for. I struggled with knowing what I should be working on or how to contribute and getting github to finally work correctly but I think I have a good idea of the path forward and my role in the project.

Earned Value

According to the gantt chart, we currently have 3 tasks completed, and by the first release, we wanted to have 8 tasks completed. With this in mind, we know that we are considerably behind schedule for the project.

Variances

In addition to being behind schedule according to the Gantt chart, we have encountered some other technical challenges, such as debugging issues with the procedural generation algorithms. Like what was mentioned previously, we still need to solve some issues with areas being blocked by randomly-generated water, so that the gameplay isn't inhibited at all. This essentially will require adjusting our timeline to resolve these specific problems within the larger tasks that need to be completed.

Lessons Learned

One lesson that some of us may have learned more than others is that any small, periodic, progress and work on the project really helps more than relying on big, multi-hour work periods, especially if we're getting close to the end. The constant small work allows you to see the problems or obstacles that will have to be overcome and be able to brainstorm ideas over the course of time rather than trying to come up with a solution all at once.
Another aspect we have taken away from is the importance of our communication. Apart from our initial meetings after class, we set up a Discord group chat to communicate, but that was used less than would be ideal in an industry setting (compared to using Slack channels, in-person meetings, etc.). Now closer to the end of the first prototype we have been utilizing it much more to effectively communicate our contributions and progress so that everyone is up to date in the loop and not having to wait till we meet in-person in class.

References

References:

DeVries, Byron. (n.d.). Original sprite movement code. (Retrieved February 6, 2024).
Game Endeavor. (n.d.). Mystic Woods downloadable asset pack [Sprites, images]. Retrieved from https://game-endeavor.itch.io/mystic-woods (Retrieved February 6, 2024).
ChatGPT. (February 13-23, 2024). Originally created use case diagrams, use case descriptions, and natural language requirements. Modified by Marcos Sanson (February 13-23, 2024).