Development Plan
SFWRENG 4G06 - Capstone Design Project

Team #7, Wardens of the Wild
Felix Hurst
Marcos Hernandez-Rivero
BoWen Liu
Andy Liang

Table 1: Revision History

Date	**Developer(s)**	**Change**
2025-09-21	All	Created Document
2025-09-22	Marcos, Felix	Fixed some missing sections
2025-09-22	Marcos, Felix	Fixed issues with grammar, punctuation

Contents

1 Confidential Information?

There is no confidential information to protect.

2 IP to Protect

There is no intellectual property to protect.

3 Copyright License

For licensing, our team is using LGPL-2.1 license.

4 Team Meeting Plan

Our team plans to meet on our Discord server every Sunday and via server messaging channels. We will stay in contact with our supervisor on Microsoft Teams, where we will share updates of our development progress, and may invite them to our Sunday meetings or dedicated meetings as needed, usually occurring on Thursdays if a meeting is necessary.

The meetings, which are mostly virtual, will generally last 30 minutes to 1 hour and will cover topics on a prewritten agenda. Each meeting will have a chair that will keep the discussion on track and ensure all agenda topics are covered.

Meetings are recorded on our GitHub repository as issues. The issues contain the agenda as well as the meeting minutes.

5 Team Communication Plan

Our team will communicate primarily using Discord, where we will plan, ask questions to each other, and give each other feedback. With our webhook, team members are notified of all updates made to the GitHub repository within the Discord application. Commits are pushed as pull requests which are then reviewed and pulled by at least one other team member.

GitHub issues are used to track meetings, major feature development, bugs, and other key tasks.

6 Team Member Roles

We anticipate the following roles and responsibilities:

- Team Leader - Keeps the team informed and reminded of deliverable deadlines, communicates with instructors, and has final say on topics the team cannot agree on.
- Art Director - Determines the aesthetic direction of the game's artwork.
- Character Artist - Draws sprites of the player character.
- Environment Artist - Draws assets of the environment in the game world.
- Programmer - Writes the game code.
- Music Director - Determines the aesthetic direction of the game's soundtrack.
- Composer - Composes music for the game.
- Meeting Chair - Keeps the team on track during meetings and ensures all agenda topics are discussed.
- Reviewer - Reviews pull requests and accepts or rejects them.

7 Workflow Plan

Each individual, or group of individuals, is assigned a specific feature / bug fix / task to implement in a branch. When implementation is complete, a merge request for that branch will be opened and reviewed by another team member. All pull requests will also undergo unit testing via CI/CD.

We will use given template issues for team meetings, TA meetings, supervisor meetings, lectures, and peer reviews. We will use our own format of issues for features, bugs, and other miscellaneous tasks.

8 Project Decomposition and Scheduling

September - December Recess

- Technical Environment of the game (Destructible environment, player character, player tools, Slime mold traversal)
- V&V Plan Revision 0
- Design Document Revision 1
- Proof of Concept Demonstration

Start of January - Mid January

- Level Design
- Revision 0 Demonstration

January - End of February

- Integration
- Design Document Revision 0

January to End of March

- Bug fixing
- V&V Report and Extras Revision 0
- Final Demonstration (Revision 1)

We will be using a Kanban board on GitHub projects to keep track of our current tasks and what we have yet to work on, what we are currently working on, and what we have already completed. It will be populated with issues that may have one or more collaborators, but always have one person responsible to make sure it gets completed. The issues will be created as tasks are identified and each task will be something small that the assigned individual(s) can do within a relatively short amount of time.

GitHub Project available HERE.

9 Proof of Concept Demonstration Plan

The main risk for the success of our project, which we will demonstrate during our proof of concept demonstration, will be the behaviour of our organism that is modelled by the slime mold algorithm.

Compounding errors due to effects on performance of our procedural and physics based features interacting with each other (pixels are in random shapes and quantities which in conjunction with the non-deterministic mold traversal would result in significantly different results each time), if the sandbox level has all features working well in conjunction with one another, then it proves the feasibility of level design centered around these technical libraries.

The proof of concept will be a specially curated sandbox level with a player controlled character and sample environment which contains objects that highlight the functionality of player's tools and mold traversal. In terms of our demonstration of the slime mold, it will be represented by a segmented game object, such as a plant or fungus. The player can manipulate the terrain to unearth some water. The slime mold will then react by traversing closer to the water source. Other similar environmental manipulations may be involved in this demonstration, such as heat and light sources, which also cause the slime mold to react in different ways.

10 Expected Technology

Git and GitHub will be used as a version control system.

The game will be primarily developed in Unity 6.1 game engine. Our team expects to use C# as our main scripting language and libraries used are developed within our team. Use of license-free material may be used in a transformative manner within designs and artwork. The main technology focuses of our project are as follows:

- Procedural destructable environment
- Intelligent level traversal of segmented game object
- Visible player inventory

Continuous integration is implemented with updates on each member's area of specialization (libraries, and functions) with team code standups and code reviews.

In terms of testing, our team will be extensively testing each technology separately in specially designed unit tests after each update. Continuous integration will involve unit test, security, and formatting checks.

11 Coding Standard

We will use Pascal case for files, camelCase for functions and classes, and Snake case for variables. Code formatting will be done in Kernighan & Ritchie style.

Appendix — Reflection

Team

3. Thankfully, our team had no disagreements during this deliverable and were all on the same page, so this will serve as a response to everyone's Q3.

Andy Liang

1. Creating our development plan was crucial for several reasons specific to our ambitious project. First, our game involves complex technical challenges - procedural destructible environments, intelligent slime mold traversal, and physics-based interactions that could compound into performance issues. Without a clear plan, we could easily get lost trying to solve these problems simultaneously. The plan helped us identify our main risk early: ensuring the slime mold behavior works as intended while maintaining performance when combined with our voxel-based destructible environment. By recognizing this upfront, we can focus our proof of concept demonstration on exactly this integration challenge. Additionally, with our diverse team roles (Art Director, Character Artist, Environment Artist, Programmer, Music Director, Composer), coordination is essential. The plan establishes clear communication channels through Discord and GitHub, defines our workflow using pull requests and code reviews, and sets expectations for CI/CD implementation. Without this structure, our different specializations could easily work in isolation and create integration nightmares later. The scheduling aspect also forces us to think realistically about deliverable deadlines and break down our complex technical goals into manageable milestones.

2. Early Issue Detection: Given our concern about compounding errors between procedural systems and physics, automated testing can catch integration problems before they become major headaches. Team Coordination: With multiple people working on different systems (art, code, audio), CI/CD ensures everyone's work integrates properly and nobody breaks someone else's features. Code Quality Assurance: Our plan includes unit testing, security checks, and formatting verification, which is essential when working with C# and Unit.y Performance Monitoring: Since performance is a key risk with our procedural and physics systems, automated performance testing can flag issues early.

BoWen Liu

1. Creating a development plan prior to starting the project is essential in aligning the team's goal, and workflow in order to have an realistic and feasible starting point and roadmap on how to proceed in this project.

2. CI/CD improves traceability and accountability in one's work both in terms of intra/inter team development as well as for upper management in

a business context. The disadvantages to using CI/CD could be low quality of work to meet rigorous and sometimes unrealistic weekly milestones as well as adding unnecessary overhead when committing deliverables.

Felix Hurst

1. Creating a development plan prior to starting a project ensures many aspects of proper organization. Everyone in the team knows what tasks they are responsible for, so different team members do not end up trying to do the same work, and know who to contact to ask questions about specific modules. The team has expectations set, including activity, quality, self-imposed deadlines, and meeting schedules. The team is ultimately guided by the development plan in nearly everything they do while working on the project. Without this kind of structure, team members would be spending a lot more time asking questions, causing delays in development. Or, they may underperform compared to the other team members' internal expectations. It is important that everyone is on the same page to minimize the need for future questions and minimize the possibility of conflict within the team.

2. The advantages of using CI/CD include:

   - Pull requests could be verified to meet specified tests. This ensures poorly written code is not accepted into the repository.
   - New code could be automatically built into a testable version of the project, making it faster to test.

   The disadvantages of using CI/CD include:

   - It takes time to set it up and write tests, especially those that are intended to be universal across all newly accepted code.
   - It may slow down the process of merging pull requests for minor changes that don't need extra testing.

Marcos Hernandez-Rivero

1. Creating a development plan before starting a software engineering group project is essential because it provides a clear roadmap for the team, defining goals, scope, roles, and timelines to keep everyone aligned. It helps prevent confusion, overlap, or missed tasks by assigning responsibilities, establishes coding and documentation standards for consistency, and outlines milestones to manage time effectively. A development plan also anticipates risks to project success, and ideally sets strategies to address them.

2. CI/CD allows teams to integrate code frequently, and catch many errors early and automatically, which in the long run improves software quality and reduces the amount of bugs either on release or later down

the production workflow. Some notable disadvantages though, are that it requires setting up and maintaining the CI/CD system, which can be time-consuming and/or confusing to many individuals. Additionally, any automated tests need to be thorough so that they act as a reliable tool to ensure code quality.

Appendix — Team Charter

External Goals

Our teams goals are to:

- Create a project worthy of a good grade, and one that we can all be proud of. i.e. Aim for Level 4 on the rubrics.
- Learn and develop new technical skills.
- Further develop interpersonal and teamwork skills

We are making a video game as our project, because most of us find that highly interesting and useful for our own personal careers in the future.

Attendance

Expectations

Each team member is expected to:

1. Attend all scheduled meetings, both on Discord and in-person. This applies to Team Meetings, TA Meetings, Supervisor Meetings, and any other kind of meetings involving our team.
   If you have a valid excuse for missing a meeting, you must inform the team prior to the meeting. This helps us plan with your absence and start the meeting faster instead of wasting time waiting for you to come.
2. Reply to questions asked on Discord within a period of 24 hours.
3. Contribute around 20-25% of commits, opened/closed issues, and other administrative duties. This percentage will be a rough estimate as contribution ratios vary a lot throughout the project lifespan, but this is the ideal split.

Acceptable Excuse

Acceptable excuses for temporary inactivity / missed meetings include:

- Illness/Injury
- Family issues
- Unexpected complications with other school work (explained to team)
- Other similar serious issues

Unacceptable excuses include:

- Oversleeping

- Double booking
- Forgetting
- Other similar preventable complications

In Case of Emergency

If a team member has an emergency, they will be excused from their responsibilities until their emergency is resolved. The rest of the team is not expected to pick up the slack in the meantime. If this emergency causes the team to submit an unfinished deliverable, the situation will be discussed with the instructor. Otherwise, the team member who had an emergency is expected to catch back up on their work as best they can.

Accountability and Teamwork

Quality

Team members should be properly briefed on the agenda of meetings, and deliverables assigned to them should be complete and ready for code review before the deadline. Any questions or concerns should be prepared and brought up to the team. All team members are responsible for properly testing their code with the appropriate endpoints and in the "Sandbox Environment".

Attitude

Team members should listen and be respectful of all members' ideas and code commits, and an open environment should be facilitated to encourage feedback on coding practices and logic improvements. Any disagreements will be resolved by team polls, and all team members will adhere to McMaster's and the Department of Engineering Code of Conduct.

Stay on Track

Discord events and/or Google/Outlook calendar events may be used to keep track of meeting schedules.

Activity will be tracked through:

1. GitHub issues marking meeting attendance
2. Discord chat history
3. GitHub commit history

If team expectations are not met and cause problems for the team, we'll first give a gentle reminder. If the rules are broken again, a more firm warning will be given. Finally, if rules continue to be broken and harm the team after the firm warning, then team members in good standing will take the issue up with the TA and/or instructor.

Team Building

A team building activity we will partake in is play testing our game for fun and coming up with cool and unique levels in the sandbox. We will have a channel dedicated to showcasing our creations and fun moments in our Discord server.

Decision Making

Our team will make decisions primarily by majority vote. Everyone is encouraged to suggest a solution that will be voted upon. Then, we will use a poll system, such as built-in Discord polls, to collect everyone's votes. The solution with the most votes wins. Ties may be broken by reducing the options to select from to only the ones involved in the tie. If there are two options that are still stuck in a tie, then we may decide either by:

- Further discussion and arguments of both sides that may persuade a team member to change their vote.
- Flipping a coin.

If there is still strong disagreement even after a decision is made, the Team Leader will settle disputes.