Development Plan
SFWRENG 4G06

Team 9, dice_devs
John Popovici
Nigel Moses
Naishan Guo
Hemraj Bhatt
Isaac Giles

Table 1: Revision History

Date	Developer(s)	Change
2024/09/19	Hemraj Bhatt	Added content to sections 7, 8 and the appendix. Created reference page
Date2	Name(s)	Description of changes
...	...	...

[Put your introductory blurb here. —SS]

1 Team Meeting Plan

2 Team Communication Plan

3 Team Member Roles

4 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classificaiton, etc.?

5 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

6 Expected Technology

[What programming language or languages do you expect to use? What external libraries? What frameworks? What technologies. Are there major components of the implementation that you expect you will implement, despite the existence of libraries that provide the required functionality. For projects with machine learning, will you use pre-trained models, or be training your own model? —SS]

[The implementation decisions can, and likely will, change over the course of the project. The initial documentation should be written in an abstract way; it should be agnostic of the implementation choices, unless the implementation choices are project constraints. However, recording our initial thoughts on implementation helps understand the challenge level and feasibility of a project. It may also help with early identification of areas where project members will need to augment their training. —SS]

Topics to discuss include the following:

- Specific programming language
- Specific libraries
- Pre-trained models
- Specific linter tool (if appropriate)
- Specific unit testing framework
- Investigation of code coverage measuring tools
- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done
- Specific performance measuring tools (like Valgrind), if appropriate
- Tools you will likely be using?

7 Coding Standard

The coding guidelines mentioned below will allow for an unfirm codebase and proper collaboration between developers. The guidelines set are universally accepted, which ensures these guidelines meet industry standards.

- Use "int" rather than unsigned types. `int` is commonly used in C# and interacts more easily with other libraries. Exceptions apply only for documentation specific to unsigned data types. [1]
- Class and method names should always be in Pascal Case
  - Ex: PascalCase
- Method arguments and local variables should always be in Camel Case
  - Ex: camelCase
- Employ abstraction, encapsulation, inheritance and polymorphism
  - Abstraction: Model the relevant attributes and interactions of entities as classes to define an abstract representation of a system. Use interfaces or abstract classes to provide a blueprint for other classes without revealing implementation details. [2]
  - Encapsulation: Hide the internal state and functionality of an object, allowing access only through a public set of functions. Use private fields to hide an object's internal state. Provide public properties or methods (get and set) to access or modify the private fields in a controlled way, ensuring safe interaction with the object. [2]
  - Inheritance: Create new abstractions based on existing abstractions. Reuse common functionality from the base class while allowing the derived class to add or override methods or properties. [2]
  - Polymorphism: Implement inherited properties or methods in different ways across multiple abstractions. Use interfaces or abstract classes to define common methods that can be implemented differently in each class. [2]
- Write code with clarity and simplicity in mind
- Avoid overly complex and convoluted code logic
- Comments should be consice and to the point
- Code should be formatted appropriately
- Hard coded constants should be avaided, employ symbolic names and use configuration files
- Any code taken from external sources must be sufficiently cited through comments
  - Include links to webpages for any external resources
  - Include input and output for any AI-aided code

8 Project Scheduling

The team has decided to employ a Gantt chart to schedule the project. A Gantt chart provides a way to depict the duration, completion percentage, and leads of tasks in a visual manner. Due to these benefits, deadlines are met due to the duration of a task being enforced by start and end dates. It also allows for effective communication as others know who to refer to if there is an issue pertaining to a certain task due to the leads being listed. All in all, a Gantt chart provides great facilities to effectively plan out a project.

The team will also implement a backlog for our development which will be featured in our readme file on github to aid potential contributors. This will allow them to understand which features have been implemented and which are still in development. This will be in addition to utilizing GitHub project boards.

Gantt Chart: Link

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. **Why is it important to create a development plan prior to starting the project?**

   A development plan is essential when taking on any sort of project. It allows for the team to set a clear set of goals and milestones while also defining the project's scope. It creates a baseline of expectations for each team member and reduces any confusion about the team's plan as a high-level plan is detailed within the development plan. Without a development plan, a direction for the team is not set which in both the short and long term will cause issues that will eventually lead to missed deadlines and poor quality of work.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**

   In our scenario, the team is creating a game using Godot, an open-source game engine. Therefore, there are some issues that present themselves. One is, performing automated testing, as it is more complex due to the need for graphical tests and user input simulations, which aren't easily automated in standard CI/CD pipelines. Another issue with CI/CD is that they are complex to set up and maintain as the system must remain updated, secure, and functional.

   However, there are still benefits to employing CI/CI pipelines as it promotes collaboration. Everyone is always working on the most up-to-date version which reduces commit conflicts and new features not working due to code logic being altered in an unpulled commit. It also allows new features and fixes to be pushed to production more quickly and reliably which enhances software quality and user satisfaction.

   Overall, implementing Continuous Integration (CI) and Continuous Deployment does provide a sizable benefit to most projects. However, as mentioned above, it cannot be used in certain circumstances. It may be

ideal but it is not feasible. In the team's case, CI/CD may not be beneficial and impractical but we can still use aspects of CI/CD in our methodology.

3. **What disagreements did your group have in this deliverable, if any, and how did you resolve them?**

   When constructing the development plan, the team faced no disagreements. Due to the area of the project being outside of our expertise, the team was open to suggestions on how the project could be developed. Due to close to zero experience in game development, when researching important tools that would be used to develop the game such as what game engine to use we were able to go off separately on our own and research the pros and cons of certain game engines such as Unity and Godot. Doing this allowed each member to understand the pros and cons of different game engines and were then able to conclude which game engine to use without much issue.

Appendix — References

[1] B. Wagner, "C# Coding Conventions," *learn.microsoft.com*, Available: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions.
[2] B. Wagner, "Object-Oriented Programming C#," *learn.microsoft.com*, Available: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop.