Reflection and Traceability Report on SFWRENG 4G06

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

[Reflection is an important component of getting the full benefits from a learning experience. Besides the intrinsic benefits of reflection, this document will be used to help the TAs grade how well your team responded to feedback. Therefore, traceability between Revision 0 and Revision 1 is and important part of the reflection exercise. In addition, several CEAB (Canadian Engineering Accreditation Board) Learning Outcomes (LOs) will be assessed based on your reflections. —TPLT]

1 Changes in Response to Feedback

[Summarize the changes made over the course of the project in response to feedback from TAs, the instructor, teammates, other teams, the project supervisor (if present), and from user testers. —TPLT]

[For those teams with an external supervisor, please highlight how the feedback from the supervisor shaped your project. In particular, you should highlight the supervisor's response to your Rev 0 demonstration to them. —TPLT]

[Version control can make the summary relatively easy, if you used issues and meaningful commits. If you feedback is in an issue, and you responded in the issue tracker, you can point to the issue as part of explaining your changes. If addressing the issue required changes to code or documentation, you can point to the specific commit that made the changes. Although the links are helpful for the details, you should include a label for each item of feedback so that the reader has an idea of what each item is about without the need to click on everything to find out. —TPLT]

[If you were not organized with your commits, traceability between feedback and commits will not be feasible to capture after the fact. You will instead need to spend time writing down a summary of the changes made in response to each item of feedback. —TPLT]

[You should address EVERY item of feedback. A table or itemized list is recommended. You should record every item of feedback, along with the source of that feedback and the change you made in response to that feedback. The response can be a change to your documentation, code, or development process. The response can also be the reason why no changes were made in response to the feedback. To make this information manageable, you will record the feedback and response separately for each deliverable in the sections that follow. —TPLT]

[If the feedback is general or incomplete, the TA (or instructor) will not be able to grade your response to feedback. In that case your grade on this document, and likely the Revision 1 versions of the other documents will be low. —TPLT]

1.1 SRS and Hazard Analysis

1.2 Design and Design Documentation

1.3 VnV Plan and Report

2 Challenge Level and Extras

2.1 Challenge Level

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. —TPLT]

2.2 Extras

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. —TPLT]

3 Design Iteration (LO11 (PrototypeIterate))

[Explain how you arrived at your final design and implementation. How did the design evolve from the first version to the final version? —TPLT]

[Don't just say what you changed, say why you changed it. The needs of the client should be part of the explanation. For example, if you made changes in response to usability testing, explain what the testing found and what changes it led to. —TPLT]

4 Design Decisions (LO12)

[Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions? Discuss each of these separately. —TPLT]

5 Economic Considerations (LO23)

[Is there a market for your product? What would be involved in marketing your product? What is your estimate of the cost to produce a version that you could sell? What would you charge for your product? How many units would you have to sell to make money? If your product isn't something that would be sold, like an open source project, how would you go about attracting users? How many potential users currently exist? —TPLT]

6 Reflection on Project Management (LO24)

[This question focuses on processes and tools used for project management. —TPLT]

6.1 How Does Your Project Management Compare to Your Development Plan

[Did you follow your Development plan, with respect to the team meeting plan, team communication plan, team member roles and workflow plan. Did you use the technology you planned on using? —TPLT]

6.2 What Went Well?

[What went well for your project management in terms of processes and technology? —TPLT]

6.3 What Went Wrong?

[What went wrong in terms of processes and technology? —TPLT]

6.4 What Would you Do Differently Next Time?

[What will you do differently for your next project? —TPLT]

7 Reflection on Capstone

[This question focuses on what you learned during the course of the capstone project. —TPLT]

7.1 Which Courses Were Relevant

[Which of the courses you have taken were relevant for the capstone project? —TPLT]

Our project consisted of utilizing Godot, a game engine, and writing our scripts using GDScript, Godot's in-house language. Due to this, none of the courses the team has completed are directly relevant. However, the method in which the team developed the game in the back end and in the front end was influenced by courses the team has taken in the past. The following are some of the courses that the team took learnings from and integrated into the project.

- **SFWRENG 3RA3:Software Requirements and Security Considerations:**
  - This course taught us how to formulate both functional and nonfunctional requirements properly. Consequently, it also taught us how to construct an SRS document properly. The team followed the guidelines taught in the class in order to verify whether the functional and nonfunctional requirements set by the team were correct. Additionally, when creating the SRS document, the team was able to complete it without much issue. Overall, this course aided in our documentation being readable and professional while also preparing us to be able to formulate requirements correctly.
- **SFWRENG 4HC3:Human Computer Interfaces:**
  - This course focused on how interfaces should be designed optimally, delving into how this can be achieved using different methods. The course highlighted how important multiple cues for something can be helpful in making a UI more intuitive, whether that is done through symbolic, visual, or audio cues. Therefore, we made sure our UI was designed in a way that would be easy to traverse and understand. One example is the inclusion of both visual and audio cues when the user performs an action. The visual cue is the button being pressed, usually shown as being dark and recessed, while the audio cue is a sound playing when a button is clicked. This allows the user to both see and hear their action being processed.
- **SFWRENG 2AA4:Software Design I - Introduction to Software Development:**
  - This course introduced the team to industry-standard coding practices. Due to GDScript being an object-oriented language, we employed the basic principles of encapsulation, inheritance, polymorphism, and abstraction. These principles were taught in this course, and with this knowledge, the team was able to develop code that had order and was not all over the place. One such example is the modularization of the code; most logic is modularized, which allowed

the team to create a game where nearly all aspects of the base game could be modified without causing issues to other aspects. Overall, this course allowed the back-end development of the game to be done in a sustainable manner. This was proven further in development when the adherence to such coding practices allowed for changes to be made quickly and efficiently.

7.2 Knowledge/Skills Outside of Courses

[What skills/knowledge did you need to acquire for your capstone project that was outside of the courses you took? —TPLT]

Our project was developed using the Godot game engine, with GDScript as the programming language. The team had no prior experience with game design or development and had not taken any related courses. As a result, we had to learn both the engine and the language from scratch. To do this, we relied on Godot's documentation and numerous educational videos on Godot and GDScript. These resources helped us build a solid foundation, allowing us to gradually develop our game. As our understanding grew, we used debugging tools and trial and error to resolve issues and refine our work.

- **Game Design:**
  - The team had to learn game design and understand how to create a game that was not only fun but also engaging and balanced. The team watched a few educational videos on game design and what made for good game design, but a large portion of it came from firsthand experience. During various development phases, the team conducted internal testing, where the members played the game and tested out features. This feedback was then taken into account during development. When the game was at a stage where it could be playtested by others, usability testing played a crucial role, allowing the team to gather feedback from real players to identify pain points and improve the user interface and overall game experience. Through this, the team gained a better understanding of how a game should be designed and how resources should be managed during development.
- **Godot:**
  - The team had to learn how to navigate and use the Godot engine. The team was able to learn the basics within a few days through educational videos and documentation. As development progressed and the team continued working on the platform, they became more and more familiar with the platform and all the tools it offered, which made development smoother and faster.
- **GDScript:**

– The team had to learn how to use GDScript from scratch, as it was a language that the team was unfamiliar with, as it is only used in Godot. Due to it being an object-oriented programming (OOP) language, the team was able to pick it up quickly because of their prior knowledge of other OOP languages. The team mainly used educational videos and Godot's official documentation to delve deeper into the language's nuances. Additionally, as development went on, familiarity with the language increased, and now the team can code in GDScript just like any other language they knew prior to the project.