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 Instructor Feedback

- **Issue:** Team Contributions
  - **Feedback:** Commit numbers are not even across group members, others appear to be contributing in other ways but either commit numbers need to be more even, or alternate contribution methods must be better documented
  - **Response:** Team members have worked hard to ensure that commit numbers throughout the last third of the course have been much more consistent across the board, and work that is not captured by commits has been documented in project management contributions files

1.2 Supervisor Feedback

- **Issue:** Grey Out Opponent's Dice Numbers
  - **Feedback:** Confusing which dice UI represents your own dice and which represents the opponents
  - **Response:** Added grey overlay over opponent dice UI and made UI box smaller to indicate it is not interactable
- **Issue:** Pip Too Dark
  - **Feedback:** The 1 pip on the D6 is too dark
  - **Response:** Light emission added to improve D6 face visibility
- **Issue:** Check Scoring
  - **Feedback:** I want to know what score I will get for each hand before I click it
  - **Response:** Added a score preview for each hand when hovering

- **Issue:** Grey Out Roll Selected
  - **Feedback:** Grey out the roll selected button until dice have been selected
  - **Response:** The roll selected button now gets greyed out until dice have been selected
- **Issue:** Raise Stakes Should Hide Opponent's Rolls
  - **Feedback:** Raising the stakes should hide the opponent's dice rolls
  - **Response:** After opponent raises the stakes, their rolls get hidden on all subsequent rolls
- **Issue:** Pips or Numbers
  - **Feedback:** Consider deciding on using strictly pips or numbers on all dice to ensure consistency
  - **Response:** Base versions of dice will be updated to use numbers (**planned**)
- **Issue:** Specify Yahtzee
  - **Feedback:** Finalize rules for the Yahtzee hand and implement it
  - **Response:** Yahtzee hand rules are planned and is being implemented (**planned**)
- **Issue:** 1 & 7 Look too Similar
  - **Feedback:** The 1 and 7 look too similar for viewing
  - **Response:** New fonts on dice chosen to improve readability (**planned**)

1.3 TA Feedback

- **Issue:** Interactive Lobby
  - **Feedback:** Request for social hub to facilitate matchmaking
  - **Response:** Added to post-MVP road-map since matchmaking was pushed out of scope
- **Issue:** Team Contributions
  - **Feedback:** Commit numbers are not even across group members, try to even out commits more
  - **Response:** Team members have picked up open issues when nothing is assigned, to ensure work is consistently available for all members. All members have worked hard to ensure that commit numbers throughout the last third of the course have been much more consistent across the board

1.4 Usability Testing Feedback

Also refer to our Usability Testing extra for more details

1.4.1 UI/UX Improvements

- **Issue:** LineEdit Text Validation
  - **Feedback:** No input length restrictions
  - **Response:** Implemented SMS-length chat (160 chars) and connection code limits
- **Issue:** LineEdit Text Overlap
  - **Feedback:** Text overflowed UI borders
  - **Response:** Adjusted margins and text wrapping
- **Issue:** Scoreboard Size Changes
  - **Feedback:** Scoreboard changes size when hovering to view points
  - **Response:** Fixed layout constraints to avoid sizing issues
- **Issue:** Blurry UI Scaling
  - **Feedback:** Pixelation was noticed with window scaling
  - **Response:** Implemented nearest-neighbor scaling for bitmap elements
- **Issue:** Checkbox Formatting
  - **Feedback:** Unclear state indicators (check boxes) used throughout game
  - **Response:** Redesign with better labeling (**planned**)

1.4.2 Gameplay Enhancements

- **Issue:** Round Transition Feedback
  - **Feedback:** Unclear progression from last roll of hand to next round
  - **Response:** Add visual/audio cues and round summaries (**planned**)
- **Issue:** Scoring Explanations
  - **Feedback:** Needed in-game hand guides to explain scoring
  - **Response:** Created interactive tutorial with dice examples
- **Issue:** Bonus Threshold Clarity
  - **Feedback:** Unclear requirements for attaining the bonus
  - **Response:** Add live tracking and tooltips (**planned**)

1.4.3 Multiplayer Features

- **Issue:** Opponent Name Integration
  - **Feedback:** Generic "Opponent" labels get used in chat
  - **Response:** Implement actual profile name usage in chat (**planned**)
- **Issue:** Chat During Waiting Screen
  - **Feedback:** Inaccessible chat during time when waiting for other user to roll
  - **Response:** Layer reordering for UI access to chat feature throughout round (**planned**)
- **Issue:** Interactive Lobby
  - **Feedback:** Request for social hub to facilitate matchmaking
  - **Response:** Added to post-MVP road-map
- **Issue:** Connection Status Clarity
  - **Feedback:** Not sufficiently clear when client player connects to the host lobby
  - **Response:** Add SFX and status indicators to show connection (**planned**)

1.4.4 Technical Improvements

- **Issue:** Profile Persistence
  - **Feedback:** Profile data is not persisting across multiple openings of the game
  - **Response:** Implemented new file storage location that allows for data persistence when the game gets packed into an executable
- **Issue:** Round Count Mismatch
  - **Feedback:** D4 preset has 1 too many rounds and gets stuck on last round with no playable hands left to select
  - **Response:** Reduced number of rounds associated with preset
- **Issue:** Debug Module
  - **Feedback:** Needed logging system to better explain issues appearing during usability testing
  - **Response:** Created auto-loaded debug manager

1.4.5 Art/Animation

- **Issue:** Tavern Animation Sync
  - **Feedback:** Tavern theme animations are out of sync
  - **Response:** Re-sync tavern animation for dice bouncing
- **Issue:** Dice Contrast
  - **Feedback:** D6 dice UI not contrasted enough, creating poor visibility
  - **Response:** Updated color schemes to improve visibility
- **Issue:** D4 Model Replacement
  - **Feedback:** Confusing layout of current D4 model
  - **Response:** Add alternate D4 model that is easier to interpret (**planned**)

1.4.6 Audio Improvements

- **Issue:** Music Variety
  - **Feedback:** Music gets too repetitive
  - **Response:** Added more music variety to game to reduce the frequency at which users hear the same audio

1.4.7 Tutorial System

- **Issue:** Tutorial Tabs
  - **Feedback:** Tutorial section gets hard to navigate for different game variants
  - **Response:** Add tabs to tutorial section for each variant of the game (**planned**)
- **Issue:** Update Tutorial Theming
  - **Feedback:** Tutorial does not match current game theme
  - **Response:** Updated tutorial to match most recent theming of the game

1.4.8 Network Features

- **Issue:** Client Name Transfer
  - **Feedback:** Client name does not get passed to host when connecting
  - **Response:** Should be updated to use client's name that gets passed (**planned**)

- **Issue:** End Game Chat Visibility
  - **Feedback:** Chat should not remain visible on end screen
  - **Response:** New functionality added to hide chat when game ends

1.4.9 Scoreboard System

- **Issue:** Scoreboard Scroll Clarity
  - **Feedback:** Unclear that scoreboard can be scrolled, and how far the scrolling goes
  - **Response:** Enhanced scrollbar theming to make scrollbar and handle more prominent (**planned**)
- **Issue:** Scoreboard Hover Explanations
  - **Feedback:** Scoring explanation should be added for each scoring option when hovering
  - **Response:** Opted to create a scoring quick guide that explains each scoring option

1.4.10 Player Differentiation

- **Issue:** Opponent Dice Visibility
  - **Feedback:** Confusing which dice UI represents your own dice and which represents the opponents
  - **Response:** Added grey overlay over opponent dice UI and made UI box smaller to indicate it is not interactable
- **Issue:** Tab Screen Overlay
  - **Feedback:** Pressing tab should show both player's scores as an overlay
  - **Response:** Pressing tab show's both player's scores as an overlay (**planned**)

1.5 SRS and Hazard Analysis

Refer to our GitHub: SRS Edit History, Hazard Analysis Edit History for details about how these files were changed.

Also refer to our SRS and Hazard Analysis revision histories for more details about how these files were changed.

1.6 Design and Design Documentation

Refer to our GitHub: MG Edit History, MIS Edit History for details about how these files were changed.

Also refer to our MG and MIS revision histories for more details about how these files were changed.

1.7 VnV Plan and Report

Refer to our GitHub: VnV Plan Edit History, VnV Report Edit History for details about how these files were changed.

Also refer to our VnV Plan and VnV Report revision histories for more details about how these files were changed.

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]

General

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]

Our team worked on a 'Usability Testing' extra which consisted of a usability testing plan and an associated report. We also worked on a 'User Instructional Video' as one of our extras which we completed to help users better understand how to play and navigate our game.

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]

3.1 Early Phase Design Decisions

In creating this game, our team had plans in the form of requirements, but the exact execution that would lead to the fulfillment of these requirements was not completely clear. As such, we worked towards creating a base model that would satisfy the requirements that had been laid before us, with the most minimal implementation possible. From there we were able to establish a modular structure for developing the project, and an art style to follow throughout development.

3.2 Design Decisions From Feedback

As the project took roots and began to materialize itself as a tangible game, feedback started coming back. First from within the team and from close friends, and later as the game got pushed to larger groups for testing, feedback started trickling in from many different users. Feedback came largely in the form of bug reports, and UI suggestions for improved visibility and clarity. There were some suggestions for better scoring explanations, and more music variety. All of this feedback has been provided in Section 1 under 'Usability Testing Feedback'. All of the feedback that we received was tracked through issues, and most of these issues were resolved with updates to the game, as explained in section 1. Feedback that has not yet been implemented remains available in open issues for future design iterations. It was important to us to fix bugs that got reported for obvious reasons, but when it came to suggestions about UI visibility we took this very seriously because we knew that as developers of this game it would be easy for UI issues to go unnoticed by us. We needed fresh eyes to point out areas of concern, where contrast is not high enough, or when there's too much on screen, or when a UI menu is too complicated, because we may be less aware of these things. So we trusted users a lot when it came to their opinions on what features of the games were confusing, or hard to locate, or difficult to interpret. And this is why on all suggestions of this nature we made changes in every case, or at least have changes planned.

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]

4.1 Effects of Limitations on Design Decisions

Some of the limitations in working on this project included time, technical abilities of the team, and limitations of the engine that we chose to work with. Beginning with time as a limitation, we were cognizant of the fact that game development is something that is never truly finished. We knew that there would always be more features that we could add, or ways to improve existing functionalities. This is why we were very clear on what specific goals would

be central to developing this project, and we then set forth to achieve those goals before anything else. This means that we started with dice mechanics, so that dice could be rolled and read. Then we implemented player connections so that games could be initiated, and this led straight into scoring mechanics. We focused on creating all the most central components of the game so that we could best use our limited time to create a game that would meet the goals we set before us. From there we could add polish with whatever time remained. Next we were limited by technical abilities; for example nobody had the level of skill needed to quickly build and animate professional looking 3D models in the time that was given to us for this project. In response to this we made the decision that we would use open source assets so that we could better focus our time on game functionality, and we made similar decisions on the front of music. Lastly engine limitations also played a roll. Engines can be glitchy sometimes, so we always developed the game with this in mind, and we relied on user testing feedback to catch engine bugs that may have been missed in initial development. For example, dice physics could be glitchy at times, but with our understanding of engine limitations, we were able to account for these issues by re-rolling dice that refused to settle properly.

4.2 Effects of Assumptions on Design Decisions

The two critical assumptions that were made at the beginning of this project were that the Godot game engine would be stable and function correctly, and that all external libraries would function correctly since they had already been tested in mass. Now as previously mentioned, we knew that our game engine would have some limitations such as minor physics glitches, but we made the assumption that the engine would be quite stable overall. The game would operate overwhelmingly similar between each play of the game, and crashes would be rare. We structured our design decisions around this assumption; that we would not at any point have to fight with our game engine, and we could develop our game only needing to focus on the functionalities we were implementing, and not on how any specific additions might cause issues in our engine. Fortunately this assumption panned out for our team and we were able to place all of our focus on developing features for our game, without needing worry about whether or not our engine of choice could handle each successive feature. Our next assumption about the correctness of external libraries also held up. We centered some of our design decisions around external libraries enabling things like testing, and player to player connections. These were key parts of this project and we designed around these libraries as if they were trustworthy black boxes. This removed a level of complexity from our lives and turned out to be a positive design decision.

4.3 Effects of Constraints on Design Decisions

Constraints played a pretty simple roll in our design decisions. We set up some system constraints such that the game would have development occurring only

within the Godot game engine, the code base must be written using GDScript, and the system needed to support peer-to-peer and multiplayer capabilities. This meant that all design considerations took Godot as our engine into account, and we ensured that all features and code structuring could be implemented with GDScript. Furthermore we knew that Godot has built in functionality to help support peer-to-peer and multiplayer so this constraint required little thought.

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]

Duel of the Eights is a game that features two primary user groups. The first is the same user group that plays traditional Yahtzee, and the second is composed of mostly young people, who already like playing video games and may be interested more specifically in dice and/or gambling games. Since there are no games quite like our own when looking for 3D yahtzee adjacent games online or on game hosting platforms, there is definitely market space available. And with the two afore mentioned user groups, our team could have a real chance of capitalizing on this market space.

To actually market this product we can first begin with a qr code at the capstone expo for interested onlookers to sign up for a mailing list to be notified when the game is released to the market. Furthermore, the game could use some self marketing techniques, in which our team could create social media accounts including a YouTube channel so that a following is more likely to develop among users that may be interested in watching gameplay with commentary.

To produce a sellable version of the game we would need to pay the initial fee of $100 to get the game listed on steam, and then subsequent payments of roughly $15 a month for running servers with an initial user base, but this number would be expected to grow as user counts increase. We likely would not charge anything for access to the game itself, but we would like to add more cosmetic customization to the game which could be monetized quite easily. This may include loot crates with unique dice skins of different rarities where individual loot crates could be purchased for anywhere between $1-$50 depending on the drop rates associated with the crate. To make money we would have to sell pretty much any non-zero quantity of in game assets since our production costs are remarkably low.

For this game we estimate that the user base could be very large. By looking at other popular games on steam that are centered around dice or cards,

we can see that these games get anywhere from 50k monthly active users on the lower end, to as many as 800k monthly active users on the higher end. We would hope to capitalize on this demographic of users, and then hopefully some of these users would reach out to older family members and friends who already enjoy Yahtzee but don't play a lot of video games, and recommend this online variation of the classic game.

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]

Our project management did not stray too massively from our original development plan but there were some changes. For example our team meetings were planned to happen once every week, but in our second semester we changed the day that these meetings were taken on, and we occasionally skipped meetings when progress was slow during exam time, or added meetings near deliverable submissions. The team roles remained largely the same as planned, and the team's communication planned also did not change. Sometimes the workflow plan was not adhered to exactly, but this was largely due to growing trust among teammates who became very proficient as the project went on, and frequent testing that mitigated negative effects caused by small bugs in merged code changes. This means that sometimes pull requests would be merged without approving reviewers, but test suites still needed to pass for merging so the effects were never problematic. We adhered to our plans with naming PR's, creating issues, working in separate branches, and tracking development through GitHub projects and these aspects of our development plan all proved to be grossly beneficial to our organization and efficiency throughout the course.

6.2 What Went Well?

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

Naming PR's using unique emojis to express what type of change the PR represented, creating issues with clear labels, working in separate branches, and tracking development through GitHub projects all allowed this project to flow very smoothly. These were easily some of the best decisions we made to keep our project organized and on track. Furthermore, meeting every week helped

keep the team in line, and ensured that everyone always knew what was being worked on, and what needed more attention.

6.3 What Went Wrong?

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

Game development is not an easy thing to test, and setting up linter's or code coverage metrics turned out to be nearly impossible in Godot. We certainly struggled to setup a strong test suite, and automate this testing as part of our CI/CD infrastructure. We had initially planned to use linter's and code coverage/quality tools but this later proved to be something that simply was not worth the trouble.

6.4 What Would you Do Differently Next Time?

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

For any future projects, the quantity of documentation that our team needed to focus on was a bit overwhelming and in large minimally beneficial. We would cut most of the documentation that was done so that the actual project could progress faster and more continuously. Things like reflections could be captured in team meetings instead of being meticulously written down, and this would free up significant time to focus on the project itself. With more time to focus on the project, we would better divide the work such that every member of the team gets enough work to keep them occupied, while ensuring that dependencies between each members tasks are minimized as much as possible. This would prevent conflicts, reduce time spent waiting on dependencies to be delivered, and keep contributions even.

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 non-functional 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.