**Discussion Draft**

**5.1 – Overview**

This section will discuss and analyse how successful the project was in achieving its’ aims. The focus will be on the following sections:

- An analysis of how well the level generation model was in the proposed design stage.

- An analysis of the success of implementing this design into the final project.

- An analysis of the testing methods used, as well as how the feedback received by the testers proved the success or failures of the project.

The discussion of the design stage will be used to highlight how well the project was planned initially, including how strongly it related to the project’s aims and objectives, as well as how reasonable the scope of the project was for the timeframe provided.

The discussion of the implementation will highlight how well the final application matched the original design, as well as any issues or shortcomings found whilst implementing the program.

Finally, the discussion of the testing will highlight how effective and accurate the testing method was, including the quality of participants chosen and the effectiveness of the questions used in relation to the feedback expected. It will also discuss how the feedback from the participants was useful in evaluating the overall success and failures of the project in relation to aims, objectives and the research question outlined.

**5.2 – Design State Discussion**

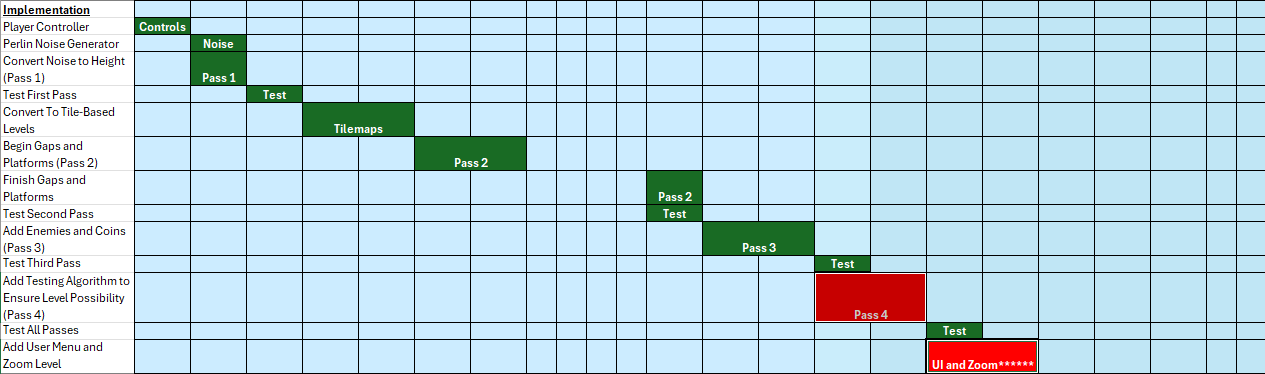
As a whole, the application’s design was successful and carried the overall project well. Importantly, the scope was kept reasonable, and at each point in the project the features to be added were reconsidered.

By designing the application to use passes, which were effectively stages layering on top of one another, this ensured that the project was always in a playable state, which is an important step in ensuring the application could be tested at every stage, with iterations to previous stages being done as required.

Having the features laid out before starting implementation was also a key step, as this helped to prevent feature creep, which may have resulted in the project going outwith the scope, resulting in development not being completed by the deadline. It also ensured there was always a solid next step for development, which made sure each implementation session was efficient and had a clear goal.

The decision to use Markov Chains and Perlin Noise was good, as they were extensively researched prior to beginning the application, and this helped to give a reference to look back on when evaluating implementation methods to ensure the application output was matching what was expected from the design.

A key document which was created during the design stage, and turned out to be extremely helpful throughout development, was a Gantt Chart, shown below:



This helped to structure the order of development, and allowed for a visual representation of how well the project’s time management was being kept. The chart was reviewed at the beginning of each week, and a decision was made as to whether development could continue as planned, or if any non-essential features needed to be removed to ensure the project would be completed on time. As visible in the chart above, features were marked in green as they were completed, and features which were eventually removed were marked as red. The removed features will be discussed in more detail during the Future Work section of this paper.

**5.3 – Implementation Discussion**

Overall, the implementation of the project was a success. All features which were required to classify a successful level, as outlined in the Literature Review section, were successfully generated in the project, therefore the model can overall be classed as successful from an implementation view.

The implementation used an iterative coding method, where regular testing was carried out during the implementation, and would sometimes lead to reconsidering some design choices. For example, the original design included a fourth pass, which would have an AI player model run over the level before starting the game and would only allow the level to be used if it managed to complete it. After some time spent attempting to implement this however, it was found to be too complex for the project’s scope and time allowance, therefore it had to be cancelled. As a result of this, an iteration was required which would more tightly refine how certain aspects of the level was generated, such as adding a constraint to ground generation which only allowed the Y-level of the ground to change by one block at a time. This improved the chance of generated levels being possible, and therefore somewhat omitted the need for a fourth pass.

The biggest benefit of the iterative coding method was having the ability to constantly go back and edit past stages as required, since often adding a new stage would highlight issues in the previous stage.

Of the features outlined in the design stage, only three major additions were eventually omitted:

* The enemy pool was reduced from three to two.
* The fourth pass involving an AI tester was removed.
* The ability to zoom and move the camera around the level was removed.

The reasoning behind these features being removed will be outlined in the Future Work section.

**5.4 – Testing Stage Discussion**

Overall, the testing phase conducted was very effective in receiving useful feedback from the participants, in relation to the project’s overall objectives, specifically:

* To research how procedural level generation could be used in future 2D platformers to give the player a unique experience each time they start a new game.
* To evaluate the level and length of player engagement in a classic linear platformer when compared to one featuring procedural level generation techniques.
* To explore the links between new, unique levels and player engagement levels.

**5.4.1 – Participants**

Participants were found by requesting help through friends and family, as well as fellow students. This allowed the project author to consider who would be suitable to participate in the project as they were known prior to carrying out testing, which is shown from the results of question one of the surveys, where all participants outlined that they had prior experience in 2D platformer games. This was a benefit as the project required a comparison between the project itself and other 2D platformer games, and by ensuring all testers had prior experience, it would prevent outlying results where testers may not have had any other games which they could compare the project to.

This method of recruiting testers did come with some drawbacks however, primarily limiting the survey to a very small sample size. In the end, only five testers took part in the survey, and whilst this was enough to get a good idea of the success within the project’s scope, having more testers would have given a more reliable source of feedback. Another potential issue was that whilst all participants did have experience with 2D platformers, it was not known which specific games this included, and due to the wide range of games within the genre, the games may not have been as closely related to the project as expected. For example, a player who played Super Mario Bros [Nintendo] may not have experienced as many PCG levels as someone who played Spelunky [\*\*\*\*\*REFERENCE].

**5.4.2 – Survey Effectiveness**

In general, the survey was well received by participants, with each tester choosing to answer every question, despite none being mandatory. This can be attributed mainly to the careful consideration during the planning of the questions included, where there was a focus on making them as easy to answer as possible. The questions mainly required the tester to read a statement, then decide how strongly they agree or disagree with the statement.

By using Microsoft Forms, this also made it easy to send out links to the survey to participants, and setting the survey visibility to public meant the testers were not required to sign in prior to completion, making it easy again for testers to give their feedback. Forms also format answers automatically into bar charts, which allowed for an easy overview of feedback results for evaluation.

One issue with the question design was found during the evaluation of results. At the end of the second section, users were asked to compare how many levels they generated, and how many they completed. The issue came when two users said they left some levels incomplete. This caused a problem because there was no question which asked the testers why they failed to complete some levels, so it is unknown whether they generated impossible levels, or if there was another reason why they did not finish the level.

**5.4.3 – Feedback Evaluation**

**5.4.3.1 – Tester Past Experience**

As mentioned in 5.4.2, the first question showed that all users would agree that they have got experience in playing 2D platformers. This was the expected result, and it was important as it ensured testers would be able to make a reasonable comparison to these games to answer questions later in the survey.

The second question showed a more divided split, where three users said they do not often play platformers more than once, whilst two said that they do. This was a good outcome because it allowed for getting the point of view of two types of players; those who feel current platformers are not currently replayable, and those who feel some might have a good replayability aspect. This meant the project would be considered both as adding replayability to a genre which does not currently have it, as well as how it possibly improves and build upon features which already do.

The final question of this section showed that four of the five testers felt the games they played had varied and interesting levels, whilst one said they did not feel this way. Similar to the prior question, this was helpful as it allowed for a split between players considering if the project improves further upon this idea, or if they feel it fixes an issue found in current games.

**5.4.3.2 – Project Quality**

The first question of this section asked testers if they felt the model generated varied and interesting levels, and was met with an overwhelmingly positive response, with all users agreeing that it did. This shows that the project succeeded in the third objective outlined in the introduction, specifically “to utilise programming techniques such as Perlin Noise and Markov Chains to implement a procedural level generator using the Unity game engine which effectively generates interesting and varied levels”, as the feedback showed the project managed to achieve this very effectively.

The next question asked testers if they found the same level twice at any point, and all testers agreed they did not. This showed the first objective “To research how procedural level generation could be used in future 2D platformers to give the player a unique experience each time they start a new game” was a success, as the generator was able to generate a new level every time, so it can be presumed the player would not encounter the same level across multiple playthroughs.

The final two questions of this section asked users how many levels they generated, compared to how many were completed. Across all users, a total of forty-four levels were generated, forty of which were completed. Whilst this was an excellent result, as mentioned in 5.4.2, it is not confirmed why four were left completed. From these results, it can be concluded that players felt engaged enough to complete the vast majority of levels, however knowing why some were not completed would have helped with the strength of results evaluation. As a result of this oversight, it cannot be confirmed whether the project requirement “all levels should have a possible path to the goal zone” was achieved.

**5.4.3.3 – Project Comparison**

The first question of the final section asked testers to compare the project levels to other platformers. The response was entirely positive, which shows all users felt that the model did manage to make levels which were more unique than current platformer games. This shows that, as mentioned in the introduction under research question, “By creating terrain generated using Perlin Noise, as well as populating the levels with objects using Markov Chain decisions, this will create unique and diverse levels” was a success.

The next question asked if users felt the model improved their engagement compared to other 2D platformers, This was again met with positive responses, which shows the project did succeed in one of its’ primary objectives, which was “To evaluate the level and length of player engagement in a classic linear platformer when compared to one featuring procedural level generation techniques”, as it did improve the level of player engagement by procedurally generating levels.

The final question asked if players felt the model would encourage them to play through a platformer multiple times. This was met with primarily positive responses, with four of the testers agreeing, however one tester said they somewhat disagree. Due to the small sample size of testers, it is difficult to tell if this was a single outlying result, or if increasing the number of testers would show a more common trend of players disagreeing, but from the results provided it can be assumed that procedural levels would largely improve replayability. From here, it can be concluded that the project achieved the second part of the aim “consider how this could be used to lengthen the time a player can stay engaged with the game”, as it shows most players would be engaged for longer using this method of generation.

**5.4.3.4 – Summary**

In summary, from the feedback the research question “How can procedural level generation be used to lengthen players’ playtime and improve player engagement in 2D platformer games?” has been answered. The first section, considering player playtime, was answered in the survey’s final question, where the majority of testers agreed the project’s implementation would help to encourage them to replay 2D platformers. The second section is answered in the penultimate survey question, where testers all agreed that the procedural generation model did improve their engagement over their past experience in 2D platformers.