The Melody Slayer Product Initiation Document

1. Introduction

The Melody Slayer is a single-player rhythm-action game in which you play a warrior trying to kill powerful creatures throughout the land using the power of music. In this game, the player will need to hit notes that come towards them along a "note highway" in front of them by moving to where the note is falling and hitting the attack key, in order to deal damage to the boss. Successful hits will deal damage while missing notes or incorrectly timing attacks will damage the player instead, which can fail the level if the player takes too much damage. Successfully hitting notes, however, will regenerate some health. Hitting one of these notes slightly early or slightly late will still count as a successful attack (within a degree) but count for fewer score and accuracy.

The gameplay itself would consist of moving along a horizontal line at a fixed speed dependant on the bpm of the song. This line is lane based, meaning notes will only fall at specific points on this line and the player will need to move to the section and hit the attack key at the right time to successfully hit the note. These notes will be moving at the pace of the song however, and fully mastering a song will be intended to be difficult on higher difficulties. There are also different types of notes, such as hold notes (where the player must continually hold the attack key and remain in the same lane to achieve maximum score) and Swipe Notes (where the player must move quickly across the lanes to successfully hit the note).

The game world will consist of individual levels that each contain a unique song and boss relevant to that level, meaning each level is the duration of the song. These levels would have a range of difficulties, with more score available for higher difficulties. This allows for replayability as mastering the hardest difficulty will require skill, and only a 100% perfect run will achieve the maximum score.

2. Background/Motivations

The Melody Slayer is inspired by rhythm games, and how they test a range of skill. While playing a rhythm game, there is a big feeling of accomplishment in being able to fully master a song, as almost all of them have a range of difficulties and can be played by almost anyone. While some people do not enjoy these kind of games, there are many that do and playing along to the beat of a song can be exciting and rewarding. Beating a boss in a difficult game also provides the same feel. Bosses are designed to test the player and ensure they have learned the gameplay enough to pass to the next area, or to obtain rare rewards.

While I have attempted experiments with both these genres, I have never combined them. I feel there is a unique and fun experience to be found in a game that allows you to defeat bosses to a song, where the game itself is easy to learn but hard to master. If implemented well, players will be playing difficult levels over and over trying to get the maximum score, and feeling exhilarated when finally defeating the boss.

The main references used for this idea are Thumper (http://store.steampowered.com/app/356400/Thumper/) and AudioSurf

(http://store.steampowered.com/app/12900/AudioSurf/), as well as the common rhythm gameplay seen in something such as Frets on Fire (https://en.wikipedia.org/wiki/Frets_on_Fire). These three games helped shape the overall idea and gameplay of The Melody Slayer.

3. Project objectives

1) Having the player move along a horizontal lane, and havin notes will fall from a track in front of them where the player must move to the lane that the note is falling from and hit the correct key at the right time in order to successfully hit each note.
2) Score, health and progress meaning the player can achieve score by hitting notes well, and health is increased or decreased based on the performance of the player. The progress bar acts as the health of the boss and is depleted when the song ends.
3) Several types of notes: Single, hold, slide. Hitting a note provides a corresponding score based on the accuracy of the hit.
4) 5 songs minimum with 3 difficulties each (easy, medium, hard). Each song will have a boss unique to that level.
5) The complete experience of starting a level, picking the difficulty and then completing the level, with a main menu and a hub menu.

4. Initial scope

The initial scope contains each aspect of the game that will need to be developed for both the minimum viable product and the features that may be implemented, but are not needed.

The MVP will require at the minimum, in the order that they are expected to be developed in:

1) Development of the core gameplay mechanic that allows the player to move and hit notes. This will involve:

--the player being able to move along a horizontal axis but within limit.

--the prototype of the highway where the notes fall, including 8 lanes.

--the code that will cause notes to fall along the track and past the player, being removed when out of view.

--The first song being implemented and code to allow synchronisation with the songs bpm. In other words, having the notes fall at the same speed as the song. This part will be improved over time, with the initial part being a simple test map to ensure the mechanics are working.

--A temporary hard-coded map, with some development of the framework to allow easy manipulation of notes and their timing. The framework is a complex aspect of the game, and may be removed if it proves to be taking too much time to implement.

--An end state when the song is finished

--The mechanic for the player to be able to hit a note and determine the accuracy. This will complete the full game mechanics and add a win/fail state to the game.

2) The MVP will also need certain aspects, such as:

--The UI that holds score, health and progress/health bar for the boss, working with the values implemented beforehand and displaying them correctly.

--The mechanics that cause score to increase, health to decrease and the progress/health bar working. This requires the accuracy mechanic to be implemented.

--Basic sound elements for hitting a note perfectly, good and missing. Extra sounds include UI elements and boss sounds, but the main priority is the sounds played during the level.

--The main menu with level selection and statistics for the level.

--The boss for the first song, along with some sort of introduction

--Time here to patch up the mechanics, map and boss to work and flow together. This includes modelling a proper note highway, a proper player model to suit the theme of the game, ensuring the health/score work with the song

--Find/Commission the remaining 4 songs for the remaining levels and bosses

--The normal difficulty for each song, and design a boss unique to each level. This also includes designing mechanics that can change up the song/level and fit the theme of the boss (5 maps)

--Finish the full set of difficulties for all songs, totalling 15 different maps (5 songs * 3 difficulties)

3) The Full implementation of the game may also include:

-- The framework to create a map for the game, such that it is no longer hard coded and can be freely done outside of code. This would involve creating an editor and saving maps outside of the game to be loaded in.

--Scenery behind the main game for the levels. The main game only takes place within the highway and the boss, meaning the rest of the space will be bland at this point. Having some scenery would improve the experience and the feel of the game, making the player truly feel as though they are fighting a boss in their area. While some scenery will be done as part of the MVP, this task involves creating a complete level unique to each song.

--A tutorial that covers the basics of how to play, ideally as a separate map with a single difficulty. The MVP will have a written tutorial, but the full implementation will be its own separate song and map.

5. Resources and dependencies

The game itself will not need many resources, but the music and sound effects to be outsourced as for a game such as this, sound design is very important and I will not be able to produce a soundtrack that would be good enough. As such, finding copyright free music or commissioning it myself will be necessary for the final product.

Unity3D and Visual Studio will be used to create the project, but both of these will be available to me.

6. Method of approach

This game will be developed in Unity3D with C# as the language using Visual Studio to write the code. The project itself will be managed using feature driven development built week by week using Trello as the management tool. The general plan will be drafted and then updated based on what gets completed.

7. Initial project plan

This is the draft of the initial project plan, detailing what is expected at each interval.

Stage	Expected Start Date	Expected Completion Date	Products/Outcomes/Deliverables
Initiation		2nd Feb	PID
Investigation and Initial prototyping (Highlight 1)	2nd Feb	8th Feb	Repo creation, implementation of basic mechanics (movement, note falling) and analyze other genres/ideas
Highlight 2	8th Feb	15th Feb	Add UI elements and finish movement, hitting of notes and the basic map. First song will be added here.
Highlight 3	15th Feb	22nd Feb	Refinement of basic mechanics and adding score/accuracy and draft of the main menu and hub menu. Add the different types of notes.
Highlight 4	22nd Feb	1st March	Design and create the first Boss, and work on completing the first level. Basic sound effects created.
Highlight 5	1st March	8th March	Develop the framework that will allow easy addition of maps and note placement for future levels. UI improvements
Highlight 6	8th March	15th March	Design the remaining 4 bosses and levels, along with the 4 songs. More refinement of mechanics and menus.
Highlight 7	15th March	22nd March	Add the remaining difficulties, complete the MVP (full sound effects, scenery etc)
Easter Vacation	26th March	13th April	Small refinements and personal reflection
Testing and Demonstration	13th April	27th April	Organise the demonstration and ensure the project is up to

			standard. Draft report dring this time
Final report	27th April	17th May	Finalize reflection report and add any of the full implementation ideas if there is time

7.1. Stage Management

The project has 4 main stages:

1) Development of core mechanics and gameplay. This contains Highlights 1 to 3.
2) Designing the boss and completing the UI/game experience. This contains Highlights 4 to 5
3) Designing and adding the remaining 4 levels/bosses as well as refining the game mechanics and adding complete sound effects. This contains Highlights 6 to 7
4) Finalizing the report and demonstration. This is the final submission stage.

7.2. Control plan

The following PRINCE2 control techniques will be employed:

1. Highlight reports as dictated by the PRCO304 module

2. Review meetings with project supervisor as dictated by the PRCO304 module; additional ad-hoc meetings as are necessary

3. Risk management (see Section 8); communication plan (see Section 7.2); quality plan (see Section 9); exception reports and plans as necessary

7.3. Communication plan

The communication plan will involve weekly meetings with supervisor, in line with the control plan.

8. Initial risk list

Schedule overrun – weekly highlight reports and meetings will mitigate the risk of a schedule overrun. A minimisation plan shall also be developed with the supervisor to provide a fallback in the case that some features may not be complete by the end of the week and overrun becomes a risk.

Difficulty with the technology – while I have plenty of experience with Unity3D and the C# programming language, some aspects of this game are significantly more abstract and difficult than what I'm used to, the aspect being the creating and loading of songs and difficulties. Being able to create a map easily without hard coding the type of note and the exact time in relation to the song requires a framework that allows the data of each note and timing to be stored and easily modified. As such there is a risk this part of the project takes longer than anticipated. As such, a fallback plan of hard coding the maps will be implemented in the case this feature does not work in time.

Late changes to requirements – as with many projects, changes can be made to the initial scope when certain techniques become available or others become too difficult or unfeasible. The staged approach to this project should allow for some leniency in regards to requirement changes. There

are also fallback plans developed in the case some mechanics become too difficult to implement, such as the framework of creating maps easily.

9. Quality plan

Requirements will be checked with the supervisor to ensure they are correct and relevant with regards to being implemented and demonstrable. Design validation will be ensured as the game will be tested and designed to ensure there are minimal problems. System V&V to be completed at the end of each relevant increment (2 and 3)//

10. Legal, social, ethical and/or professional issues

Most of the project lacks any of these kind of issues, and an ethical report will not be necessary. However, the use of music and sound effects in this game may have some legal implications as I will be aiming to use copyright free music, or commissioned music. Since I lack the skill to compose songs that would be fun to listen and play the game to, this path is inevitable. As such, there is the risk that a song is used that is copyrighted and would cause legal issues with the owner when the game is released. This is something that needs careful consideration.

Keywords:

Map = refers to a level or a stage, consisting of the song and it's notes that need to be hit. Each difficulty is a separate map.