CS 4850 – Fall 2023

# SP-22: Dynamic Rhythm Game

Bethany Cadman, Willie Doyle, Angel Magallanes, Katlin Scott

Sharon Perry

December 3, 2023

https://dynamicrhythmteam.github.io/

https://github.com/DynamicRhythmTeam/Dynamic-Rhythm

## Table of Contents

# 1.0 Introduction

## 1.1 Overview/Abstract

Have you ever listened to a song and wished you could play that particular song in a rhythm game? Or maybe even wanted more of your own songs in a rhythm game without having to wait for a new DLC pack to be released? Our Dynamic Rhythm Game will do just that, creating procedurally generated ‘rhythm game levels’ (also known as ‘beatmaps’) as songs are selected. Our game will utilize AI technology to detect various beats and inflections within the music and use them to generate new beatmaps for players within minutes. The music will be sourced from the largest music library in the world, Spotify, giving us access to over 100 million songs and near-endless possibilities for our users.

## 1.2 Collaboration Tools & Platform

### 1.2.1 Communication

For our group communication, we mostly use Discord for a majority of meetings and general communication. However, we do talk briefly in class for general communication, task assessments, and reminders.

### 1.2.2 Collaboration Tools

As for collaboration tools, we have a Google Drive folder that has any and all documents we want to collaborate on at the same time, such as our Google Docs for our deliverables, Google Sheets for our Gantt chart, and Google Slides for our slides for our presentation(s). In the beginning, we attempted to use MS Word Online for deliverables, but it is not as good for collaboration as Google Docs is. Plus, Google Drive can be used to share entire folders with others, making it more convenient for everyone to use.

### 1.2.3 Version Control

For version control, we use GitHub, which could also be considered a collaboration tool. We have a GitHub organization with all of us in it (called “DynamicRhythmTeam”) that has multiple repositories for different reasons, such as a website repository to host the website, the main GitHub project for collaboration and to keep track of edits and versions, and potentially other temporary repositories for different aspects of the project. However, GitHub cannot host all of our Unity project files at once due to the size and requiring a paid GitHub subscription.

## 1.3 Definitions and Acronyms

**Beatmap* / Notechart** = The full progression of notes to be pressed on a single song

**DRG** = Dynamic Rhythm Game

**Lane** = a column notes are separated into, usually connected to a specific button

**Note** = a basic single step in a notechart/beatmap

**Player** - a person who plays any type of game, whether it be video games, board games, roleplaying games, etc.

**Spotify** = digital music/audio streaming and media service provider

**Stage** = the area where the notes move, and the player interacts with said notes

**NOTE**: * = Term is preferred in this document

## 1.4 Assumptions

We are assuming that the system that will be used to run the game has the sufficient specifications that will be needed for the game to properly run on the proper operating system (as specified in the Functional Requirements section). We will not assume that the user will have played our rhythm game before nor has played other rhythm games. In other words, we will assume that our player will have no prior experience with rhythm games. Overall experience with games is a plus for the user, but it will not be assumed to be required.

## 1.5 Constraints

### 1.5.1 Environment

Unity allows for a wide range of applications to be created and many possibilities, but there are some tradeoffs. Platform compatibility is something to keep in mind. If we were to have decided to have cross-platform compatibility (Windows, macOS, Android, etc.), we need to ensure it works correctly and efficiently on each of them. However, for the purposes of this report, we have not added cross-platform compatibility at this time. Performance is another constraint. We need to ensure efficiency and playability on a wide range of systems. That includes a framerate that is both smooth and consistent and has efficient memory usage, allowing the system to properly function within our game.

### 1.5.2 User Characteristics

User accessibility considerations for rhythm games are essential to ensure players with different abilities can enjoy the game. This includes providing comprehensive tutorials and practice modes to help players learn the game mechanics and improve their skills at their own pace, along with different difficulty levels for a wide range of players. Other considerations are:

- Customizable controls - Provide options for players to customize controls, including key bindings, button layouts, and game speed.
- Adjustable Timing Windows - Allow players to adjust the timing windows for hitting notes. This feature can accommodate players with different reaction times, motor skills, and habits.
- Colorblind Modes and Inclusive Design - Implement colorblind-friendly modes that use color combinations suitable for players with vision deficiencies, along with an interface easy for everyone to read.
- Warnings - Photosensitivity warnings for those with epileptic disabilities

Some of these features may not be fully implemented in the final product as part of this report, but it is a must to include them as parts of our thought process going into working on our game.

## 2.0 Functional Requirements

In order for our game to function on a base level, we will need to utilize Unity's game creation software to create the gameplay, Spotify’s API to import the songs, and MuG diffusion to help generate our beatmaps. Therefore, for the user to be able to utilize our game, the following functional requirements will need to be followed:

1.0 Display Main Menu

    1.1 Play Button redirects to beatmap selector screen
    1.2 Beatmap Creator redirects to beatmap creation screen
    1.3 Settings Button redirects to settings screen
    1.4 Quit Button exits the game

2.0 Beatmap Selector Screen

- 2.1 List of beatmaps displayed
- 2.2 Search bar and category filter/sort
- 2.3 Beatmap Creation selection
- 2.4 Back Button redirects to main menu

3.0 Stage

- 3.1 User interaction with notes
- 3.2 Feedback based on the success of hitting notes
- 3.3 Results screen

4.0 Beatmap Creation

- 4.1 Adding a song from the Spotify library
- 4.2 Selecting a difficulty

5.0 Settings Screen

- 5.1 Customize Settings
    - 5.1.1 Graphics
    - 5.1.2 Beatmap color customization
    - 5.1.3 Screen Size
    - 5.1.4 Audio/Volume Controls
    - 5.1.5 Controls Customization
    - 5.1.6 Accessibility
        - 5.1.6.1 Color Blindness Options
        - 5.1.6.2 Game Subtitles
- 5.2 Timing Window Adjustment
    - 5.2.1 Offset (Beat Speed)
    - 5.2.2 Difficulty

6.0 Pause Screen

- 6.1 Resume
- 6.2 Restart the level
- 6.3 Options/Settings
- 6.4 Return to Song Selection
- 6.5 Return to Main Menu

According to Unity, the user's PC/Mac graphic and operating system requirements require the user to have:

MINIMUM:

OS: Windows 7

Processor: Dual-Core 2.00GHz

Memory: 2 GB RAM

Graphics: Intel HD 520

Storage: 7 GB

RECOMMENDED:

OS: Windows 10

Processor: i5

Memory: 8 GB RAM

Graphics: GeForce GTX 660

Storage: 10 GB

## 3.0 Non-Functional Requirements

### 3.1 Accessibility

Accessibility holds immense significance in any software or game, especially when catering to a wide audience. Our rhythm game, DRG, prioritizes offering a range of in-game options to enhance gameplay quality and improve the overall experience for players during their time in the game because everyone deserves a chance to play.

Accessibility features that we have thought of for our game:

- Stage color preference setting (lighter colors, darker colors, neutral colors, pastel colors, etc.)
- Alternative coloring for colorblindness (for deuteranomaly, protanomaly, protanopia,
- deuteranopia, tritanomaly, tritanopia)

- Subtitles
- Dynamic (according to settings) button labels for the lanes
- Stage/Lane color variations/skins

### 3.2 Capacity & Scalability

Originally, the plan for the game was to have the availability for storage expansion in case Phase 2 (discussed later in the document under "5.1 Goals and Guidelines") were to have been completed. Each beatmap should take a minimal amount of space but should also allow for songs of different lengths and sizes. The game should also allow for a virtually infinite number of songs in its' storage, only limited by the user's system.

### 3.3 Usability

The game will be easy to learn, engaging, efficient, effective, and error tolerant. The player does not need to have prior experience with playing rhythm games to play our rhythm game. The game will be simple enough to be able to learn through our game, having feedback such as sound cues when the player hits a note. Along with that, the plan is to make it pleasant and satisfying to play through the responsiveness, accuracy, and capability to configure settings in the game.

## 4.0 External Interface Requirements

### 4.1 User Interface Requirements

In order for the user interface to be easy to use for any user the Fonts and Icons should be similar throughout the final product. This will help to create a cohesive experience for the user as they traverse the different menus and displays of the game. The controls that are most commonly used should stay consistent throughout the final product. There should be clear indications on which text items are buttons and which are just general texts, along with easy-to-understand and navigate game screens. The color schemes should avoid visually jarring combinations. The game menus and screens should be appealing and not lead to eye strain due to the colors and/or animations. There should be accommodation for the visually impaired (coloring for colorblindness). The game should also take into effect general gameplay keyboard shortcuts for specific functions (Ex: Esc for pausing).

## 4.2 Hardware Interface Requirements

The game runs on the supported device types of PC and Mac. For the game to run the minimum requirements for the Windows PC, the specifications are as follows:

MINIMUM:

OS: Windows 7

Processor: Dual-Core 2.00GHz

Memory: 2 GB RAM

Graphics: Intel HD 520

Storage: 7 GB

As for MAC/OS the minimum requirements for gameplay are as follows.

MINIMUM:

OS: High Sierra 10.13+

Processor: Dual-Core 2.00GHz

Memory: 2 GB RAM

Graphics: Metal-capable Intel and AMD GPUs

Storage: 7 GB

### 4.3 Software Interface Requirements

This product will make use of the Spotify API and Spotipy, a Python library for the Spotify Web API. This functionality will already be imported into the Unity gameplay menus, so no additional steps will be needed by the user in order to utilize the API.

### 4.4 Communication Interface Requirements

This product will require an internet connection when wanting to create new beatmaps.

## 5.0 Design Considerations

### 5.1 Goals and Guidelines

In our project, we have added an additional phase beyond the two given to us initially. We have defined what each phase corresponds to below.

- Phase 1 is absolutely necessary for our game.
- Phase 2 is quality of life aspects that would improve the game's performance massively.
- Phase 3 is absolutely optional and not required for the game to function.

| | **Goal** | **Description** |
|---|---|---|
| Phase 1 | Base game platform | The base interface |
| | API Spotify implementation | Having Spotify connectivity with songs |
| | User interaction design | To be able to have the user be able to interact with the stage and beats with game input (“Perfect!” “Super!” “Miss,” etc.) |
| | AI development to detect beats | To make the AI detect beats |
| | Playable game prototype | |

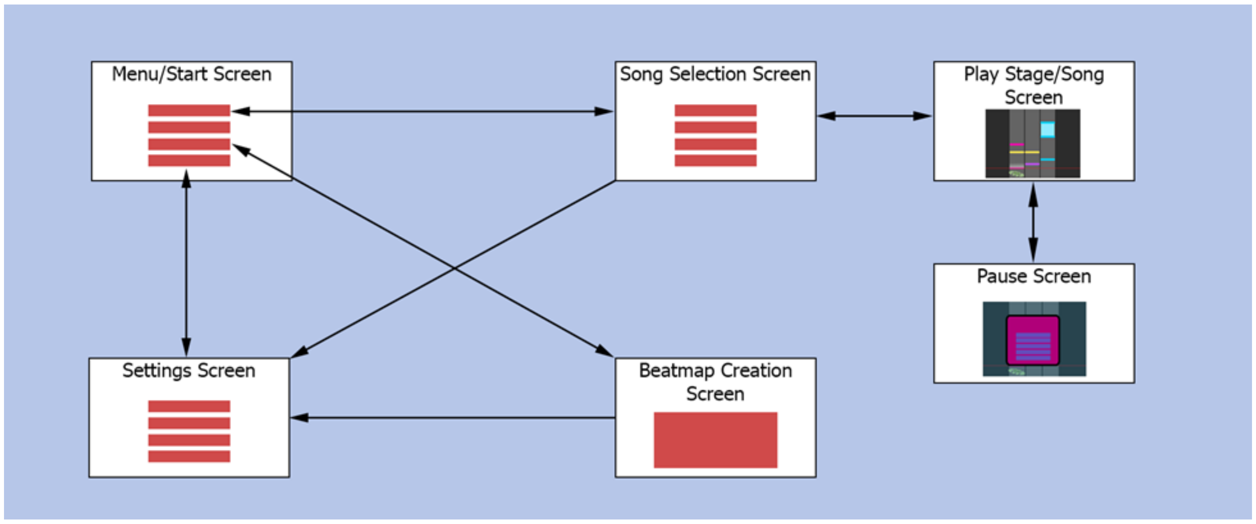
|  |  |  |
|---|---|---|
| Phase 2 | Optimization | Making the game run smoother |
|  | Design review and rework | Relooking at the game |
|  | Tutorial | Will have an in-game interactable tutorial or a video in the game |
| Phase 3 | Storage of (pre-)generated songs | Stores the AI-generated songs (or other) in a database or file |
|  | Beatmap editor |  |
|  | Auto-play feature | Have maps get played through automatically by the computer |
|  | Possible in-game story |  |
|  | Mobile Application Development Build |  |

As of currently, we are only able to complete Phase 1, which is just the main game without the quality-of-life aspects. However, because of the general scope of the project, this was entirely expected due to time constraints.

# 6.0 System Design

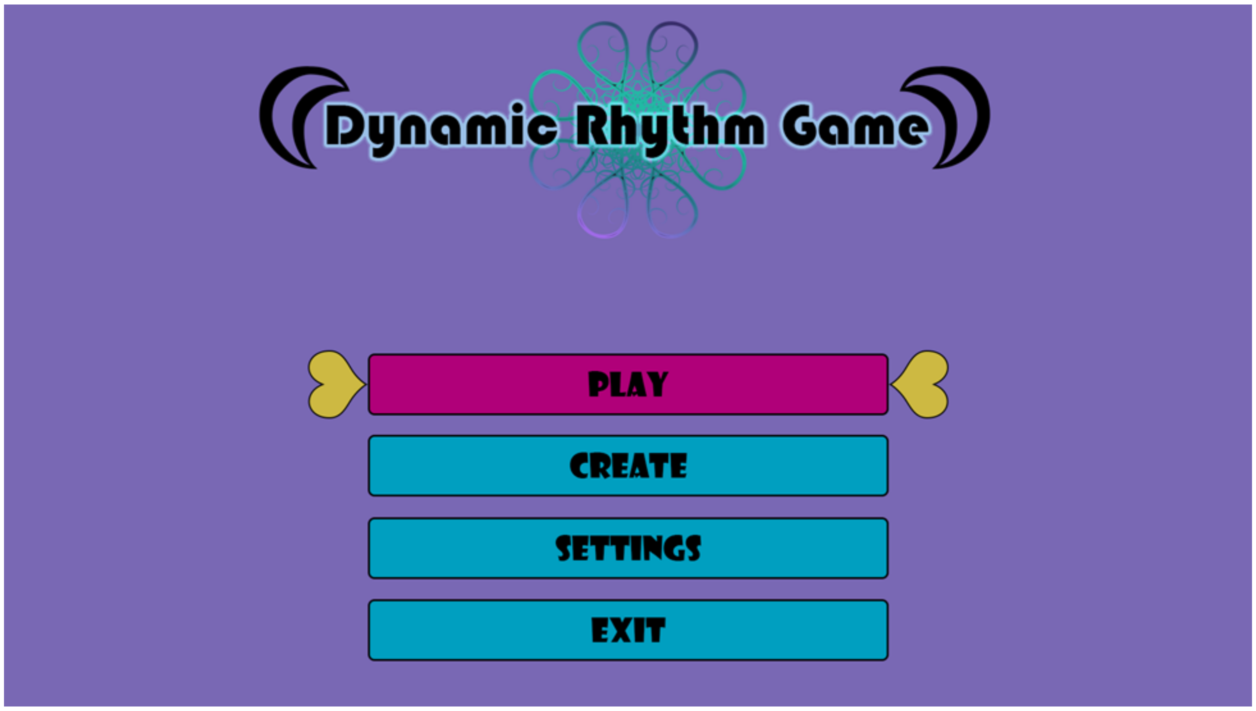
## 6.1 Mockups

### 6.1.1 UX Process Flow Diagram

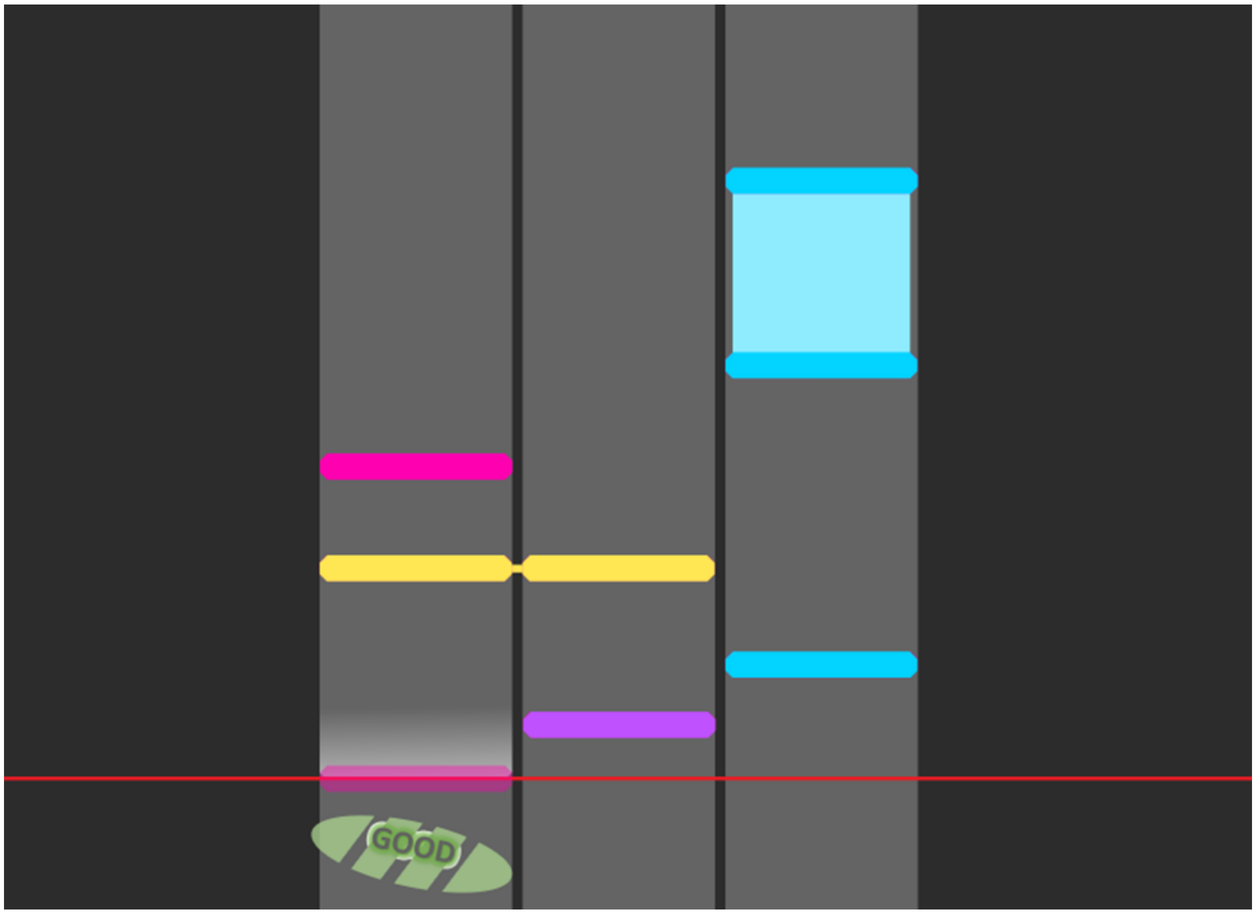


## 6.1.2 User Interface Design Diagrams

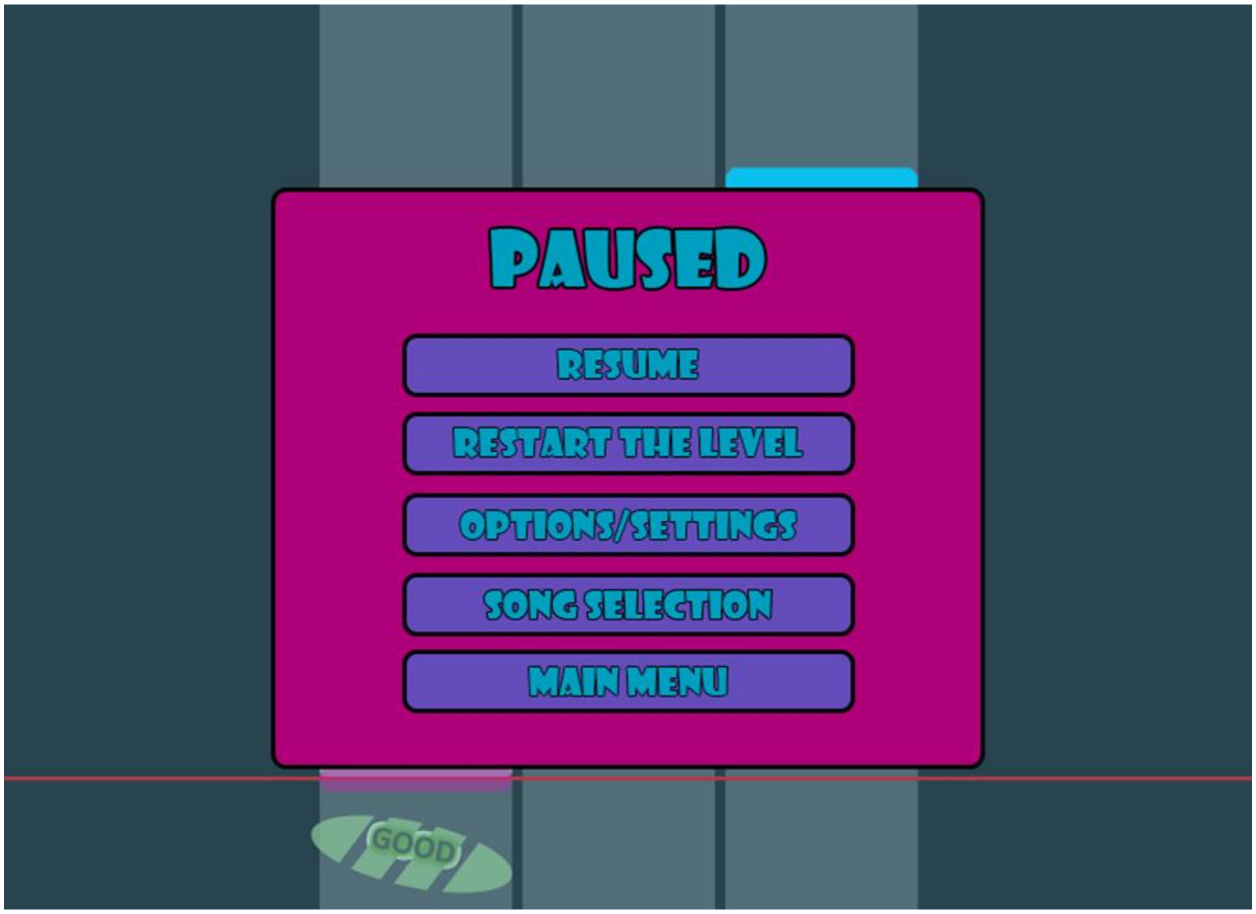
### *6.1.2.1 Main Menu Screen*



### *6.1.2.2 Song Play Screen*

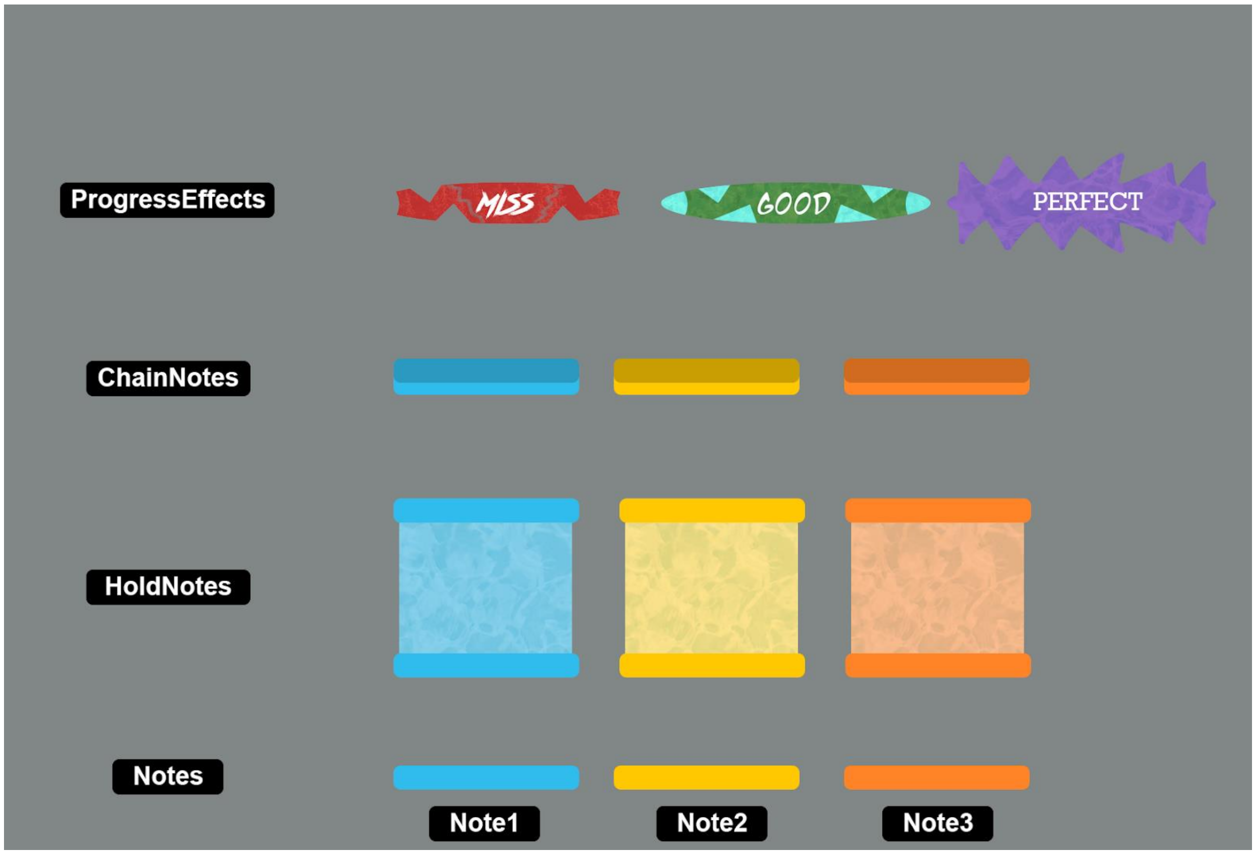


### 6.1.2.3 Pause Menu



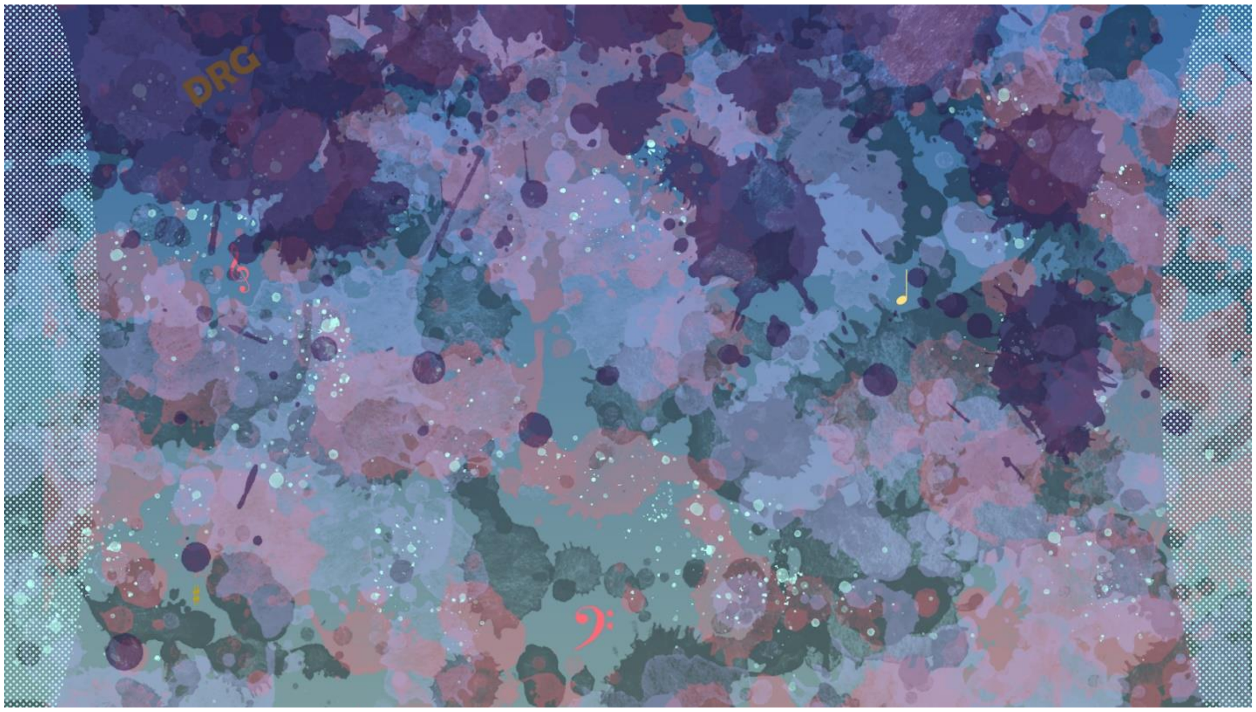
## 6.2 Game Assets

### 6.2.1 UI Game Elements



The assets above were created by Katlin Scott for the sole purpose of our DRG.

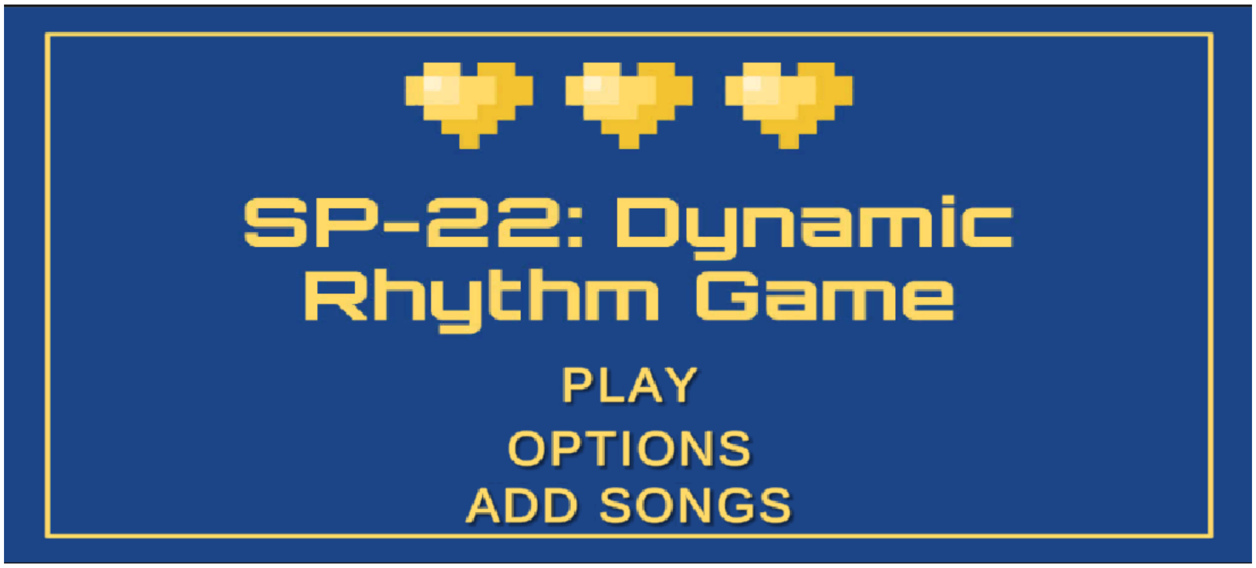
### 6.2.2 Game Background



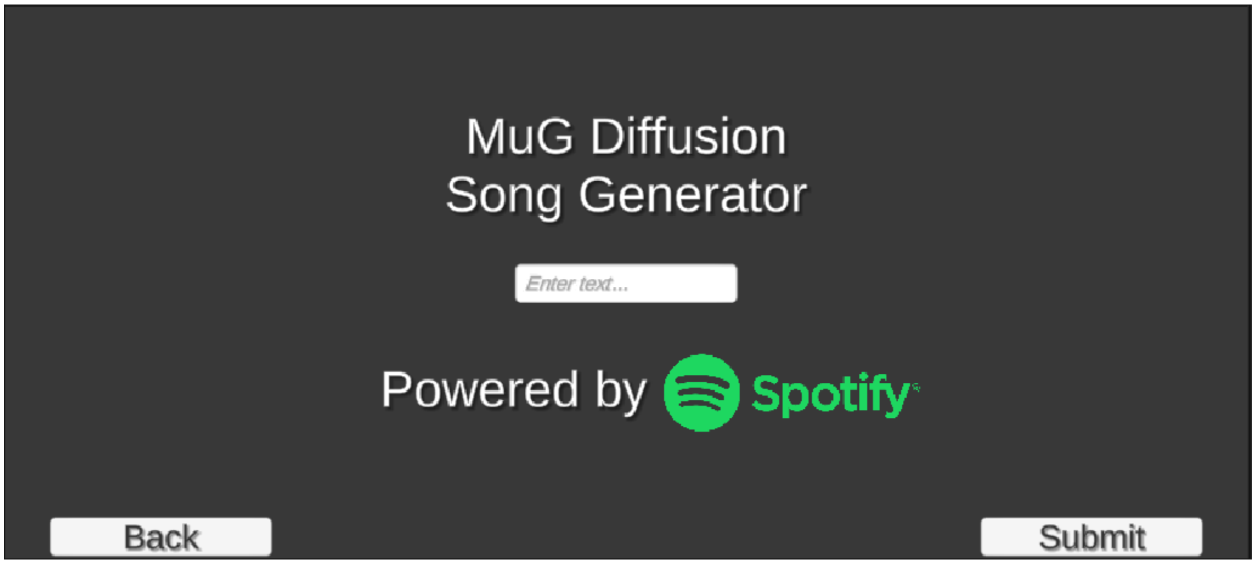
The assets above were created by Katlin Scott for the sole purpose of our DRG.

## 6.3 Screenshots of Program

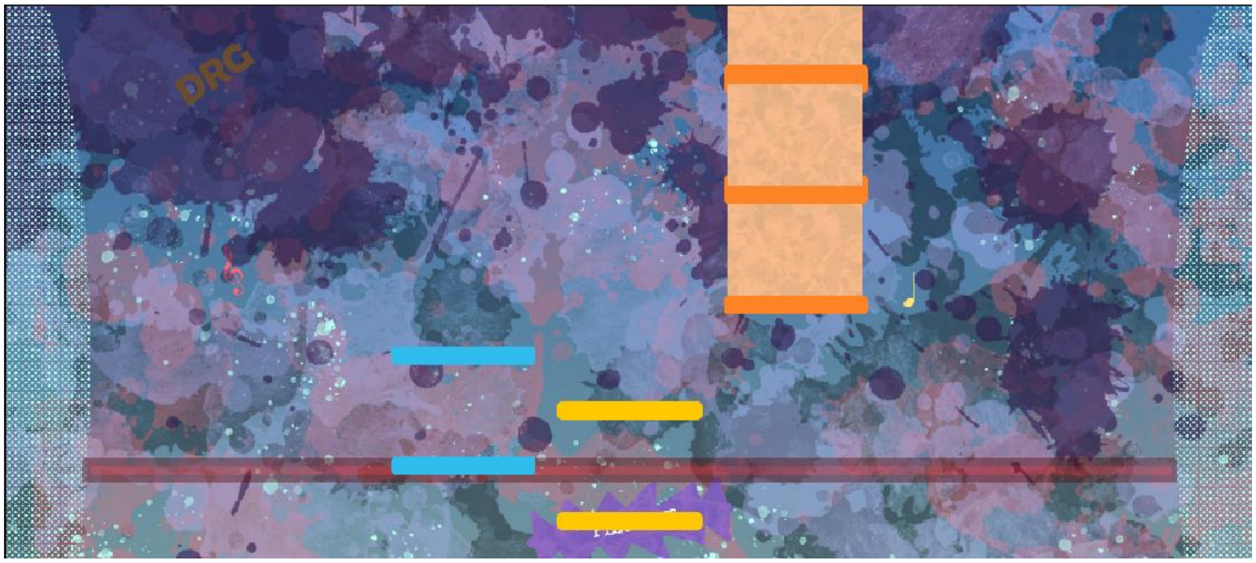
### 6.3.1 Main Menu



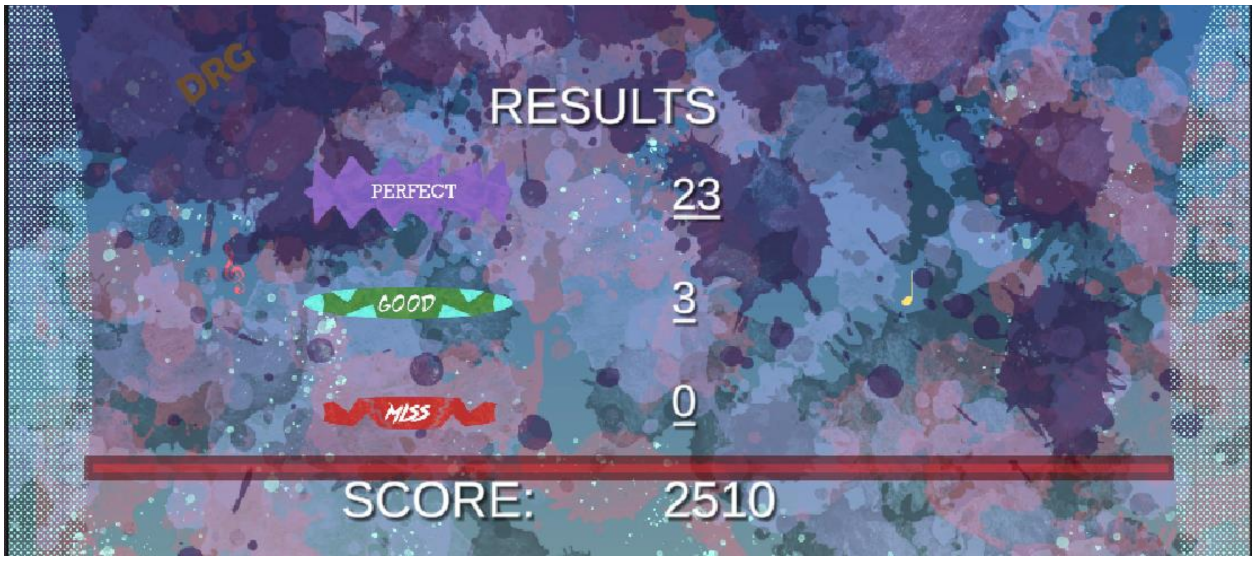
### 6.3.2 Add Song Page



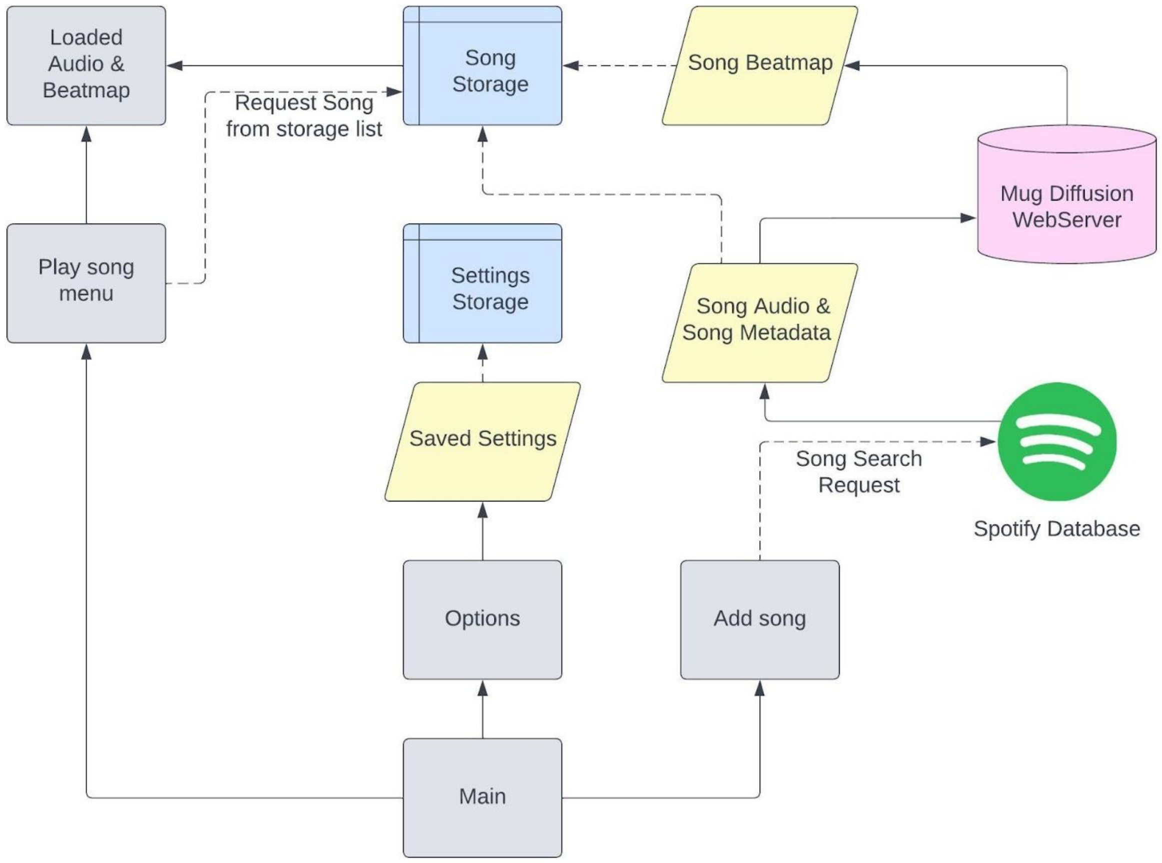
### 6.3.3 Gameplay



### 6.3.4 Results



## 6.4 Architectural Drawings



## 7.0 Narrative Discussion About Project

The Dynamic Rhythm Game project emerged from a shared passion for video games within our group. As avid gamers, we couldn't help but notice a distinct absence of diverse musical genres in the realm of rhythm games, particularly the overwhelming focus on electronic dance music (EDM). This observation sparked a robust discussion among us, leading to the recognition of an untapped potential for rhythm games to transcend their genre limitations.

A primary concern that fueled our curiosity was the lack of beat maps for mainstream and newly released songs. Traditional rhythm games often concentrate on established tracks, leaving enthusiasts craving engagement with the latest hits or hidden gems. This realization acted as the catalyst for our ambitious endeavor – to create a platform that not only defies genre constraints but also empowers users to transform any song of their choice into a dynamic and engaging beat map.

The journey of the Dynamic Rhythm Game unfolded across three pivotal pillars: the development of the game itself, seamless integration with Spotify, and the infusion of artificial intelligence (AI). However, before delving into the intricacies of these technical aspects, we acknowledged the paramount importance of establishing a robust collaborative framework. To this end, we implemented a comprehensive suite of collaboration tools, leveraging Google Workspace for documentation and GitHub for version control and website hosting. This strategic groundwork laid the foundation for a cohesive and efficient development process.

In my role as the head developer steering this innovative project, my primary focus was on the nuanced intricacies of gameplay design. We initiated this phase by translating conceptual ideas into tangible features, with a strong emphasis on accessibility. Learning from our collective experiences with rhythm games that occasionally fell short in terms of user-friendly settings or visual aesthetics, we sought to introduce a level of customizability that would set our game apart.

The decision to opt for a three-lane gameplay structure was intentional, representing a conscious departure from the prevailing trends of four or five lanes in other rhythm games. Our commitment to inclusivity and user comfort manifested in an array of accessibility features. These encompassed stage color preference settings, catering to brightness adjustments, as well as colorblind settings for a demographically diverse player base. Furthermore, we empowered users to customize button configurations, ensuring a tailored experience based on individual preferences. The variability in note and lane colors further underscored our commitment to aesthetic flexibility and visual accessibility.

The transition from conceptualization to execution encountered challenges inherent in navigating the Unity platform, especially given our team's relative newness to its intricacies. Nevertheless, overcoming these learning curves marked a significant achievement, propelling us towards the subsequent phase of Spotify integration.

The incorporation of Spotify functionality presented its own set of challenges, as the platform offered multiple APIs, each with distinct functionalities. Initially drawn to the Web Playback SDK, which functions as a full-fledged music player with Spotify Connect capabilities, we encountered legal constraints associated with recording songs — a vital component of our envisaged integration. This obstacle necessitated a pivot to the Web API, offering access to metadata and 30-second sample versions of songs in MP3 format. To bridge the gap between these diverse elements, we seamlessly integrated SpotiPy, a Python library, and Spotify API-NET, a C# client.

The synergy between the game, Spotify integration, and the AI component represented a holistic approach to our project. The AI, a modified version of MuG diffusion, showcased our commitment to pushing technological boundaries. Rooted in stable diffusion, MuG diffusion employs a Variational Autoencoder (VAE) to train a denoising model. This model, influenced by audio input, prompts, and a meticulously curated dataset of 3600 OSU! maps, undergoes a progressive refinement process over approximately 1000 steps per song. The introduction of Gaussian noise and subsequent denoising produces a final beat map that aligns seamlessly with the rhythm game's rules and the musical intricacies of the input song.

Delving deeper into the technical underpinnings, diffusion models, of which MuG diffusion is an exemplar, operate by introducing noise to training data and subsequently learning to reverse this process. This approach generates coherent outputs, a crucial facet of rhythm game beat mapping. The iterative steps, guided by audio cues and prompts, exemplify the convergence of machine learning principles with the creative realm of music and gameplay.

The integration of MuG diffusion into our project exemplifies the confluence of cutting-edge technologies with the artistry of rhythm games. Beyond the realm of gaming, diffusion models have found application in diverse tasks such as image generation, denoising, inpainting, outpainting, and bit diffusion. The transformative potential of generative models, from GANs to VAEs, is evident in their ability to generate new data based on training data, albeit with unique limitations.

MuG diffusion's methodology aligns with this broader paradigm, utilizing a dataset of OSU! Maps to train a VAE, subsequently employing a denoising model influenced by audio inputs and prompts. The gradual introduction of noise over a predetermined number of steps, approximately 1000 in our case, culminates in the generation of a full Gaussian noise. The subsequent steps involve using the denoising model to refine the beat map, providing a nuanced and accurate representation of the input song. This meticulous process, while demanding, stands as a testament to our dedication to delivering a high-quality and innovative gaming experience.

Navigating the challenges of implementing MuG diffusion was an endeavor requiring collaboration and the assimilation of knowledge from stable diffusion implementation projects within the Unity community. This collaborative spirit and the willingness to learn and adapt ultimately led to the successful integration of AI into our rhythm game project.

Beyond the technical intricacies, the project also underwent a critical phase focusing on the visual aesthetics of the game. Katlin Scott, leveraging her proficiency with Adobe tools, undertook the responsibility of infusing the game with a visually captivating and user-friendly design. This phase involved a meticulous process of incorporating our accessibility functions seamlessly into the overall graphic design, ensuring that the visual elements complemented the overarching goal of inclusivity.

Despite our comprehensive efforts and achievements, certain developmental goals eluded realization. Notably, the creation of a tutorial for our game proved to be a challenging endeavor. This aspiration remains on our horizon as we continue refining and expanding our project, cognizant of the importance of guiding users through the intricacies of our innovative rhythm game.

In conclusion, the Dynamic Rhythm Game project stands as a testament to our collective vision, technological prowess, and commitment to pushing the boundaries of gaming innovation. From addressing genre limitations to seamlessly integrating Spotify and employing AI for beat mapping, our endeavor encompasses a multifaceted approach to deliver a unique and engaging gaming experience. As we navigate the ever-evolving landscape of game development, our commitment to inclusivity, customization, and the fusion of cutting-edge technologies with creative gameplay remains unwavering. The Dynamic Rhythm Game is not merely a game; it is an embodiment of our collective passion for gaming, music, and the relentless pursuit of pushing the boundaries of what is possible in the dynamic realm of technology and entertainment.

## 8.0 Challenges, Assumptions, and Risk Assessments

The general assumption for the DRG is that it should function smoothly, offering an engaging and overall enjoyable rhythm game experience. It should be designed to be a game that players would want to return to, aiming for overall functionality and enjoyment.

Throughout this project, our primary challenges have mostly revolved around time constraints. As dedicated full-time students, it was a bit hard for us to focus exclusively on our project due to the number of projects, assignments, etc. each of us had with other courses and outside commitments. Beyond the time constraint issues, we have had some general issues related to the general inexperience all of us have, which required the necessary time to research things we weren't as familiar with, such as the AI and GitHub-Unity difficulties. With GitHub and Unity, we have had some issues properly connecting Unity to our GitHub due to the way Unity organizes its files, which impeded our beginning progress a bit. We also had issues testing due to the size of our Unity Project exceeding what GitHub offers for projects for free without a paid account/organization.

There are no risks associated with this project.

# 9.0 Testing

## 9.1 Test Plan

For the testing of our game, we plan to conduct several test cases that are meant to test the main functions of the game, such as navigation of the menus and gameplay, because without these functions the game would not work properly. There will also be some test cases for other functions of the game to ensure all is working as intended. All of these tests will be conducted manually to verify that everything is working correctly. All of the results that occurred during the test cases will be recorded, whether they were the expected result or otherwise. They will be then given a status of pass or fail according to whether they met the expected result or not and be given a severity rating if it was a failure. If it is ruled a failure, we will attempt to fix it within our given timeframe. Then, after the testing is conducted, the results will be discussed on how we feel about them and if we are satisfied.

## 9.2 Test Report

| Description | Test Step | Expected Result | Actual Result | Status | Severity |
|---|---|---|---|---|---|
| Ensure that buttons on Main Menu work as intended | Click each button on Main Menu then click back button to go back to Main Menu | Each button clicked should take user to the menu corresponding to the button clicked | Buttons clicked worked as expected. | **P** | None |
| Ensure that buttons on Settings screen work as intended | Click "Settings" button on Main Menu then click/adjust each button on Settings screen | Each button clicked/adjusted should do what it says (Ex: If there is a volume slider it should reduce the volume) | Buttons clicked worked as expected. | **P** | None |

| | | | | | |
|---|---|---|---|---|---|
| Ensure that buttons on Song Selection screen work as intended | Click “Play” button on Main Menu then click buttons on Song Selection screen | The button clicked should do what it says (Ex: Clicking a song to play should take you to play that stage for the song) | Buttons clicked worked as expected. | **P** | None |
| Ensure that the generation of the beatmap works properly | Click “Play” button on Main Menu then Song Selection screen. Then choose any song and generate a beatmap. Once the beatmap is generated then play the beatmap. | Once the beatmap is generated, it should be playable and functioning as intended | Beatmap generation works but takes a while. | **P** | Normal |
| Ensure that buttons on Beatmap Creation screen work as intended | Click “Create” button on Main Menu and then click buttons on Beatmap Creation screen to make sure they function | The button click should do what it says | Buttons clicked worked as expected. | **P** | None |
| Ensure that the controls on the Stage screen work | Click “Play,” then select a song to play and then once on the Stage screen, play the stage for a while to make sure all controls work | While on the Stage and the right buttons are clicked at the right time, the game should tell the user whether the user hit the note (as in “Perfect” “Good” “Miss”) as this means the controls work as intended | Pre-generated and Base game songs are playable with small issues from AI generation. | **P** | Normal |
| Ensure that the Stage Complete screen appears once the user completes the stage | Click “Play,” then select a song to play, and then play the stage to completion | Once the stage is over, the results screen should appear to show the user how they did | Completion screen appears at the end of gameplay with game stats. | **P** | None |

| | | | | | |
|---|---|---|---|---|---|
| Ensure that the Pause function works once on the stage | Click "Play," then select a song to play, and then once on the stage click the button that is mapped to the pause function | Once the button for pausing is clicked, the pause menu should appear | Button clicked worked as expected. | **P** | None |
| Ensure that the buttons on the pause menu work | Click "Play," then select a song to play, and then once on the stage click the button that is mapped to the pause function, and then click each button on the pause menu (may have to repeat going from main menu to stage if the button takes user elsewhere) | Each button should do what it says it will do (Ex: Restart the level button should cause the stage to restart from the beginning) | Buttons clicked worked as expected. | **P** | None |
| Ensure that users are able to access the settings screen from all the screens wherever they are | Click the settings screen on the main menu, then go back to the main menu. Then "Play," then select a song to play, then once on the stage click the pause button, then click settings, etc. | The button should lead the user to the Settings menu to click, whichever of them they are. | Buttons clicked worked as expected and settings menu is accessible from all gameplay screens. | **P** | Normal |

Overall, we are satisfied with the results of our testing because we did not run into major issues that immediately impeded the players' ability to play our game.

## 10.0 Conclusion/Summary

Our Dynamic Rhythm Game was created to be an exciting and refreshing experience for anyone who decides to play it. In order to achieve this goal, we have had to outline our games' creation, and through the outlines, we have been able to demonstrate how much thought and planning went into the creation of the game. Within Unity, we were able to use the Spotify library API, with access to millions of songs, and apply an AI component to our project to generate a beatmap from the song selected for our rhythm game. Also, with the help of services such as Discord, Google Drive, Google Docs, Google Slides, and GitHub, we were able to manage our communication, collaboration, and version control.

We have also provided lots of planning of our requirements for our Dynamic Rhythm Game. This included our functional requirements, which described which screens were needed and what they did, our non-functional requirements, which included important elements of our game such as accessibility and usability, and our external interface requirements. We also have outlines of what our goals were for the semester, and while we did not meet all of them, we are glad to have completed Phase 1 of our goals which were an absolute necessity for our game.

Additionally, we provided screenshots, mockups, and diagrams of different elements of our games. These give an insight into how it was created and how it looks when played. Then, a large narrative discussion of the game was included. This covers the very technical details of our game, how it was created, and what was used in order to make it functional, written by our lead developer, Willie Doyle. We also covered some of the challenges and assumptions that we ran into/made while working on our game, such as our trouble when we attempted to connect Unity to GitHub. Finally, we have the testing results of our game and the many test cases, outcomes, and a report of the testing that was conducted to make sure that the users playing the game have a pleasant experience with it. All of this has been the result of our hard work and dedication to creating an enjoyable experience for everyone involved with our Dynamic Rhythm Game.

## Appendix A: Glossary

**AI = Artificial Intelligence**

Artificial intelligence uses computers and machines to mimic the problem-solving and decision-making capabilities of the human mind. To read more: https://www.ibm.com/topics/artificialintelligence#:~:text=Artificial%20intelligence%20leverages%20computers%20and,capabilities

%20of%20the%20human%20mind

**Algorithm**

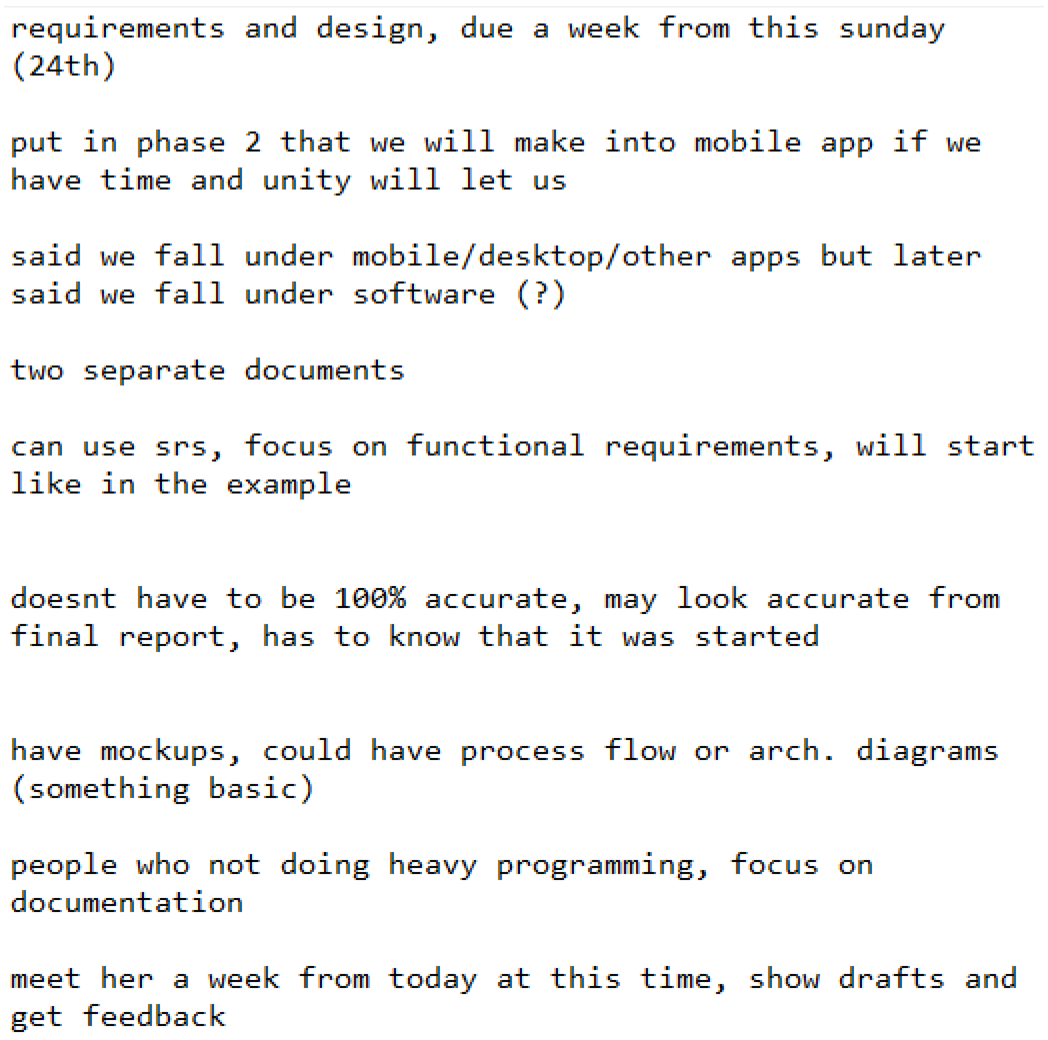
A procedure used for solving a problem. Algorithms act as a list of instructions that help to conduct specified actions step by step in either hardware or software-based routines. To read more: https://www.techtarget.com/whatis/definition/algorithm

**PC = Personal Computer**

A PC is a computing device that utilizes a microprocessor and is typically designed for use by one person. To read more: https://computer.howstuffworks.com/pc.htm

# Appendix B: Meeting Notes

## Meeting with SP (09/14/2023)



requirements and design, due a week from this sunday (24th)

put in phase 2 that we will make into mobile app if we have time and unity will let us

said we fall under mobile/desktop/other apps but later said we fall under software (?)

two separate documents

can use srs, focus on functional requirements, will start like in the example

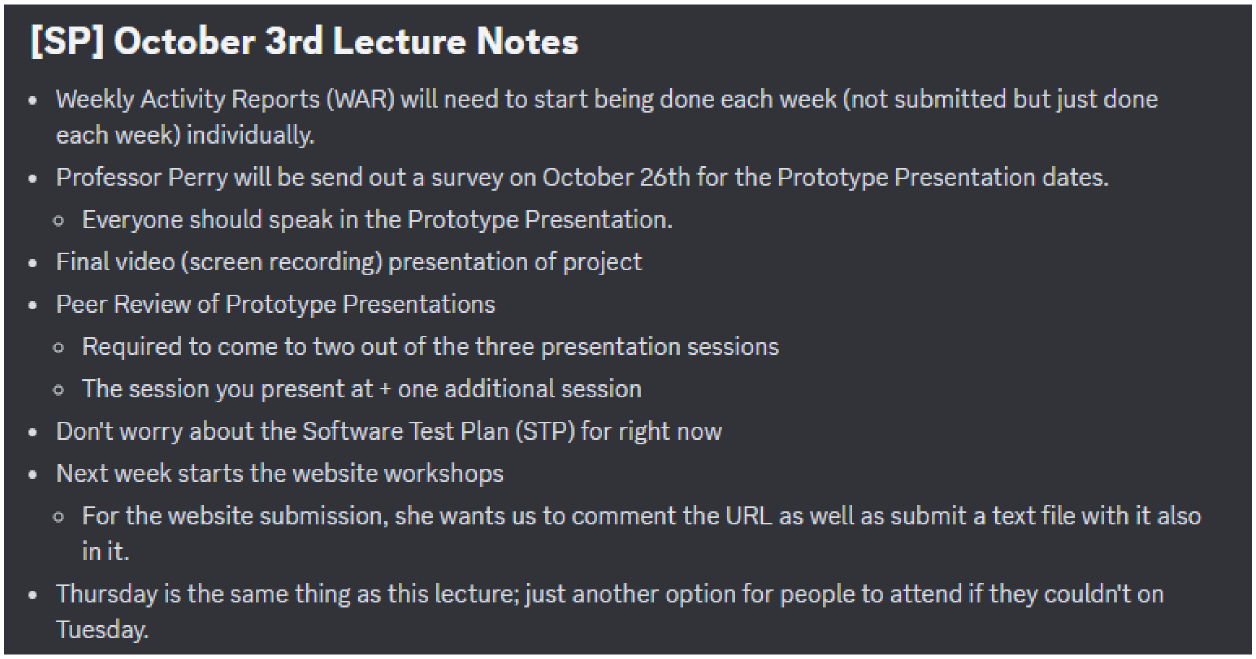
doesnt have to be 100% accurate, may look accurate from final report, has to know that it was started

have mockups, could have process flow or arch. diagrams (something basic)

people who not doing heavy programming, focus on documentation

meet her a week from today at this time, show drafts and get feedback

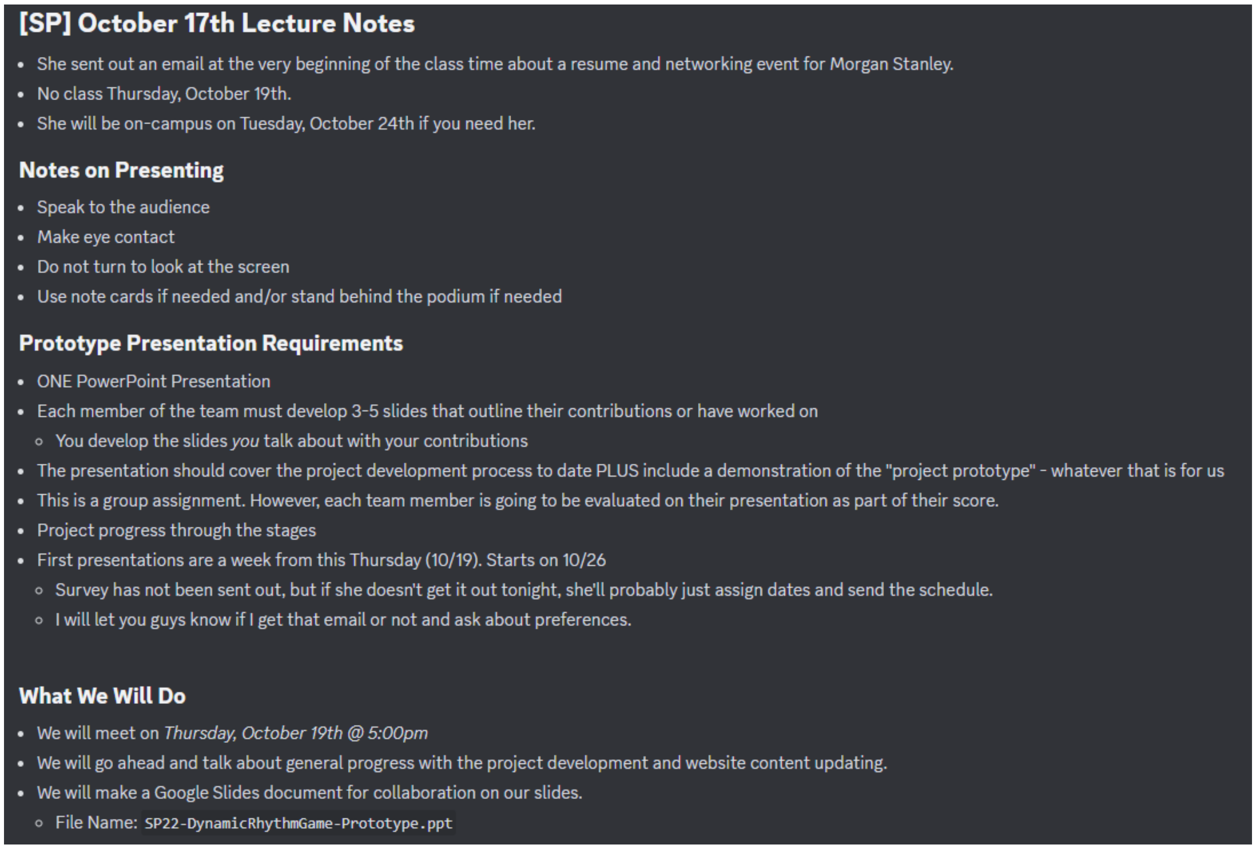
## Meeting with SP (10/03/2023)



### [SP] October 3rd Lecture Notes

- Weekly Activity Reports (WAR) will need to start being done each week (not submitted but just done each week) individually.
- Professor Perry will be send out a survey on October 26th for the Prototype Presentation dates.
  - Everyone should speak in the Prototype Presentation.
- Final video (screen recording) presentation of project
- Peer Review of Prototype Presentations
  - Required to come to two out of the three presentation sessions
  - The session you present at + one additional session
- Don't worry about the Software Test Plan (STP) for right now
- Next week starts the website workshops
  - For the website submission, she wants us to comment the URL as well as submit a text file with it also in it.
- Thursday is the same thing as this lecture; just another option for people to attend if they couldn't on Tuesday.

## Meeting with SP (10/17/2023)



### [SP] October 17th Lecture Notes

- She sent out an email at the very beginning of the class time about a resume and networking event for Morgan Stanley.
- No class Thursday, October 19th.
- She will be on-campus on Tuesday, October 24th if you need her.

#### Notes on Presenting

- Speak to the audience
- Make eye contact
- Do not turn to look at the screen
- Use note cards if needed and/or stand behind the podium if needed

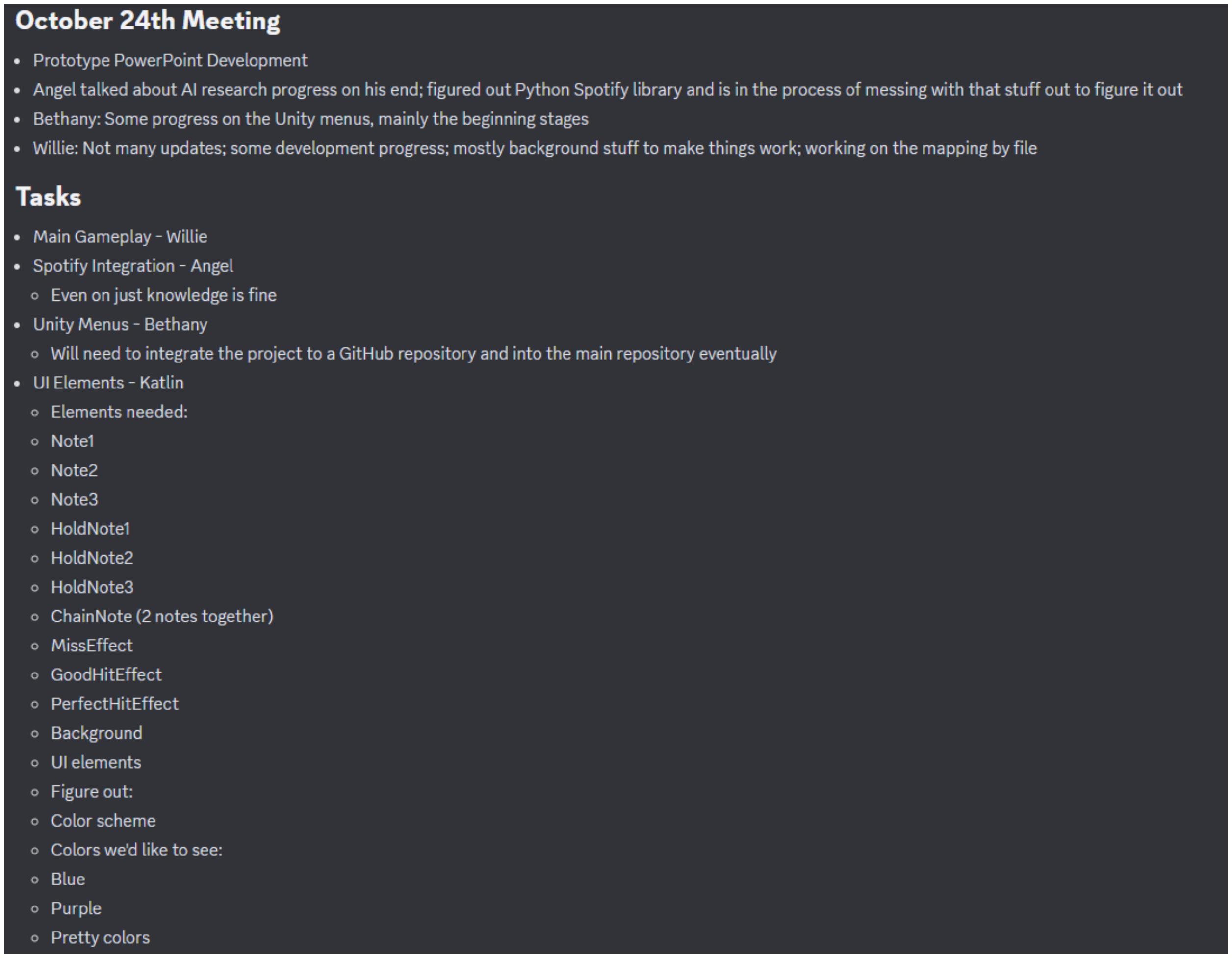
#### Prototype Presentation Requirements

- ONE PowerPoint Presentation
- Each member of the team must develop 3-5 slides that outline their contributions or have worked on
  - You develop the slides *you* talk about with your contributions
- The presentation should cover the project development process to date PLUS include a demonstration of the "project prototype" - whatever that is for us
- This is a group assignment. However, each team member is going to be evaluated on their presentation as part of their score.
- Project progress through the stages
- First presentations are a week from this Thursday (10/19). Starts on 10/26
  - Survey has not been sent out, but if she doesn't get it out tonight, she'll probably just assign dates and send the schedule.
  - I will let you guys know if I get that email or not and ask about preferences.

#### What We Will Do

- We will meet on *Thursday, October 19th @ 5:00pm*
- We will go ahead and talk about general progress with the project development and website content updating.
- We will make a Google Slides document for collaboration on our slides.
  - File Name: `SP22-DynamicRhythmGame-Prototype.ppt`

## Meeting (10/24/2023)

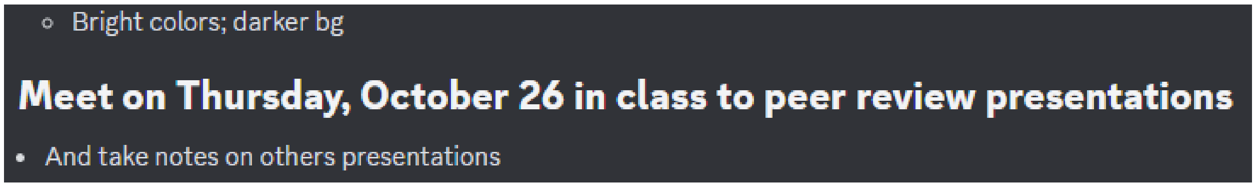


### October 24th Meeting

- Prototype PowerPoint Development
- Angel talked about AI research progress on his end; figured out Python Spotify library and is in the process of messing with that stuff out to figure it out
- Bethany: Some progress on the Unity menus, mainly the beginning stages
- Willie: Not many updates; some development progress; mostly background stuff to make things work; working on the mapping by file

### Tasks

- Main Gameplay - Willie
- Spotify Integration - Angel
  - Even on just knowledge is fine
- Unity Menus - Bethany
  - Will need to integrate the project to a GitHub repository and into the main repository eventually
- UI Elements - Katlin
  - Elements needed:
  - Note1
  - Note2
  - Note3
  - HoldNote1
  - HoldNote2
  - HoldNote3
  - ChainNote (2 notes together)
  - MissEffect
  - GoodHitEffect
  - PerfectHitEffect
  - Background
  - UI elements
  - Figure out:
  - Color scheme
  - Colors we'd like to see:
  - Blue
  - Purple
  - Pretty colors



  - Bright colors; darker bg

### Meet on Thursday, October 26 in class to peer review presentations

- And take notes on others presentations