|  |
| --- |
|  |

**Authors**

Bradley Aiken and Brandon Henry

**Title**

PrismaBeat

**Overview**

PrismaBeat is a rhythm game that uses a color switching gimmick to differentiate itself from other similar rhythm games. The player will select a song and difficulty as a level to play and hit notes that correspond to the rhythm of the song. Players will interact with several objects, tap note objects require the player to hit a corresponding button at the time of the beat in the song, long note objects, for which the player has to hold down the button, and colored objects, for which the player must toggle the colors of the lanes that the notes come down on to match the notes’ colors and get scored.

*Setting:* The game has no real setting, all gameplay takes place within a user interface. When playing, a track with four lanes is displayed to signal when and how the player should perform their inputs.

*Perspective:* A 2-dimensional interface.

*Interaction model:* Inputs can be performed with a keyboard or with a controller that is mapped to keyboard presses. Keys W, E, I, and O perform inputs that correspond with the four lanes, and Q and P toggle the colors of each half of the track.

*Challenges:* Players must react to the objects that come down on the screen so they can perform inputs correctly and at the right time to receive points. There are many songs that can be chosen, and each song has either one or a variety of difficulties that are represented on a scale from one to ten. To “win,” a player must clear a song with an effective rate over 69%, the effective rate is displayed on the screen under the score. When a player hits a note on time and with the correct color, the effective rate gauge increases, and if not, it decreases.

**Core Features**

*Score system:* The notes on each chart are assigned point values calculated by the game before it is played. If all notes are hit at the correct time and with the correct color, a score of 999,999.

*Rhythmic timing based scoring:* The game keeps track of the position of the song that is played during the main game and records the time of each keypress so that the player is rewarded a different amount of points depending on how accurate they are. To get 100% of the points for each note, a player must perform the input within a 30 millisecond time window, and any more than 90 milliseconds will not count the note at all.

*Ranking system:* In addition to the score and completion status, the player receives a letter grade based on the score. In order from best to worst, a player can receive S, A, B, C, D, or F.

*Combo counter:* This does not affect the score, but it keeps track of how many notes a player hits correctly in a row.

*Effective rate:* During each playthrough of a song, separate from the score and rank, the player has an effective rate that starts at 0% and is displayed as a bar under the score. This rate will go up every time a note is correctly inputted and down when a note is missed. If the player finishes a chart with an effective rate over 69%, the song is cleared and the player “wins” that particular level.

*Assets:* There are graphics, audio, chart data, and font assets for different parts of the game.

*Music:* There are different songs for each level that get loaded from the Game class.

**Graphics**

The game has a pixel art style and all assets are created for a 200x200 grid and sized up to 800x800 to fit the game window. Sprites are used for all elements in the game such as the notes, lanes, song info panel, etc. Some sprites such as the life gauge utilize transparency so a bar can be placed behind it. Sprites such as the single note and hold note are originally white and are recolored in libgdx to be the corresponding colors for each song. A custom font is used for the score, song info, menus, etc.

Additionally, a shape renderer is used in many of the screens to make different colored backgrounds and other lighting effects. In the title screen, song selection screen, and results screen, a shape renderer is used to change the color of the background according to the beat of the title screen music. During the playing screen, different colored rectangles are drawn under the track during gameplay, and the colors can change based on the players inputs. The colors are also used to represent which color state each side of the track is in so that the player can match the color of the notes while playing. The shape renderer is also used to fill up the effective rate meter and song position meter.

**Game States**

There are several game states that the player can navigate through with the menu. The game state is stored as a string and in the render function there is a control structure to control what happens in each state. The states in the game include the title screen, help screen, song select screen, playing screen, and results screen.

**Game Balance**

The game is balanced in a way that players of any skill level can play and enjoy the game. Each song has charts with a corresponding difficulty which ranges from 1-10. If a player is new to the game, they can play a lower difficulty, or if they have played a lot of rhythm games before, they can play a higher difficulty.

**Victory Conditions**

A player passes a song if they finish with their life gauge over 70%. The life gauge starts at 0% and for every note hit, the gauge moves up slightly. If the player misses notes, the gauge will move down. Based on the accuracy of each note hit, the player will receive a certain amount of points (less accurate = less points). The amount of points received by the player corresponds to a grade that is given on the score screen at the end of the song (as stated before, the grades are S, A, B, C, D, or F). There is no way to “beat” our game because the player can always try to achieve better scores on each song and improve their skills.

**Interface Design**

The interface of the game is relatively simple and keeps the pixel art style as mentioned before. On the title screen, there are the options to play, view a help screen, or quit the game. If the player chooses the play option, the game goes to the song select screen. On the song select screen, the player can scroll up and down in the list of songs to choose which one to play. The song select screen also has the option for the player to change the scroll speed of the notes, in case it is too slow or too fast. After changing settings and choose a song, the game goes to the play screen, where there are the lanes on the left and song information on the right. The lanes have indicators for each button that the player can press which light up when pressed and an indicator of which color each half of the lanes are currently. On the song info panel, there is the album art, title, artist, beats per minute, difficulty, and progress of the current song. Above the song info panel, there is the life gauge which shows how close the player is to passing the song and the score indicator which displays the current score. After finishing playing the song, the game goes to the results screen which displays information about the performance, such as the score, grade, max combo, and hit accuracy. On this screen there is an option to press a button to go back to the song select screen to pick another song to play.

**Future Additions**

PrismaBeat TODO List:

Black - work in progress

Orange - unsure how to implement

Blue - done

IMPORTANT (to complete project):

* Use NoteReader class to import all settings of a song
* Remove speed from the Note class
* Add menus
* Include graphics
  + Graphics for score labels (graphics done)
  + Lanes
  + Life gauge and score counter
  + Song info panel
  + Color indicator under judgement line
* Add scoring and ranking system
* Allow user to select songs, change difficulty, speed

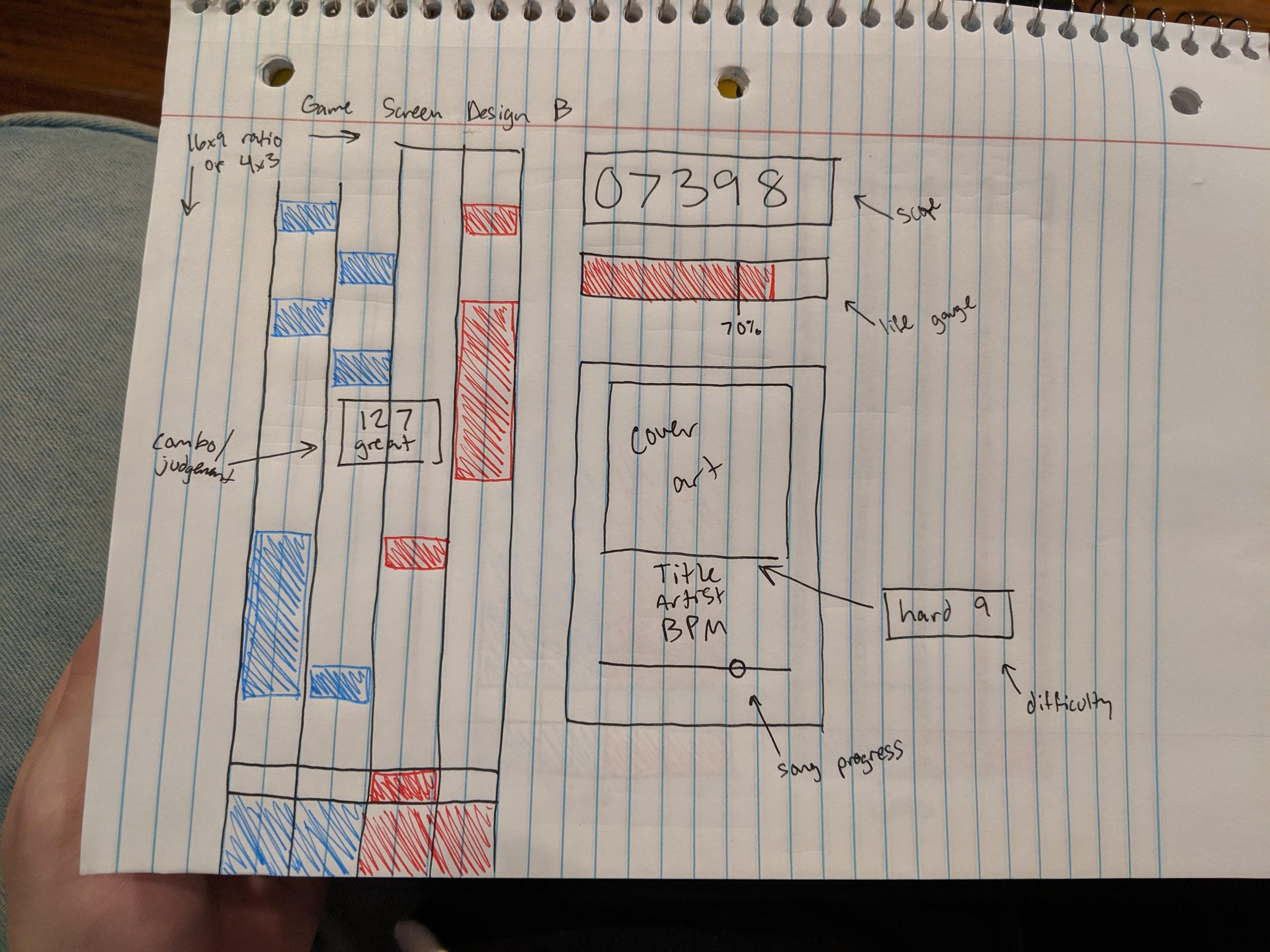
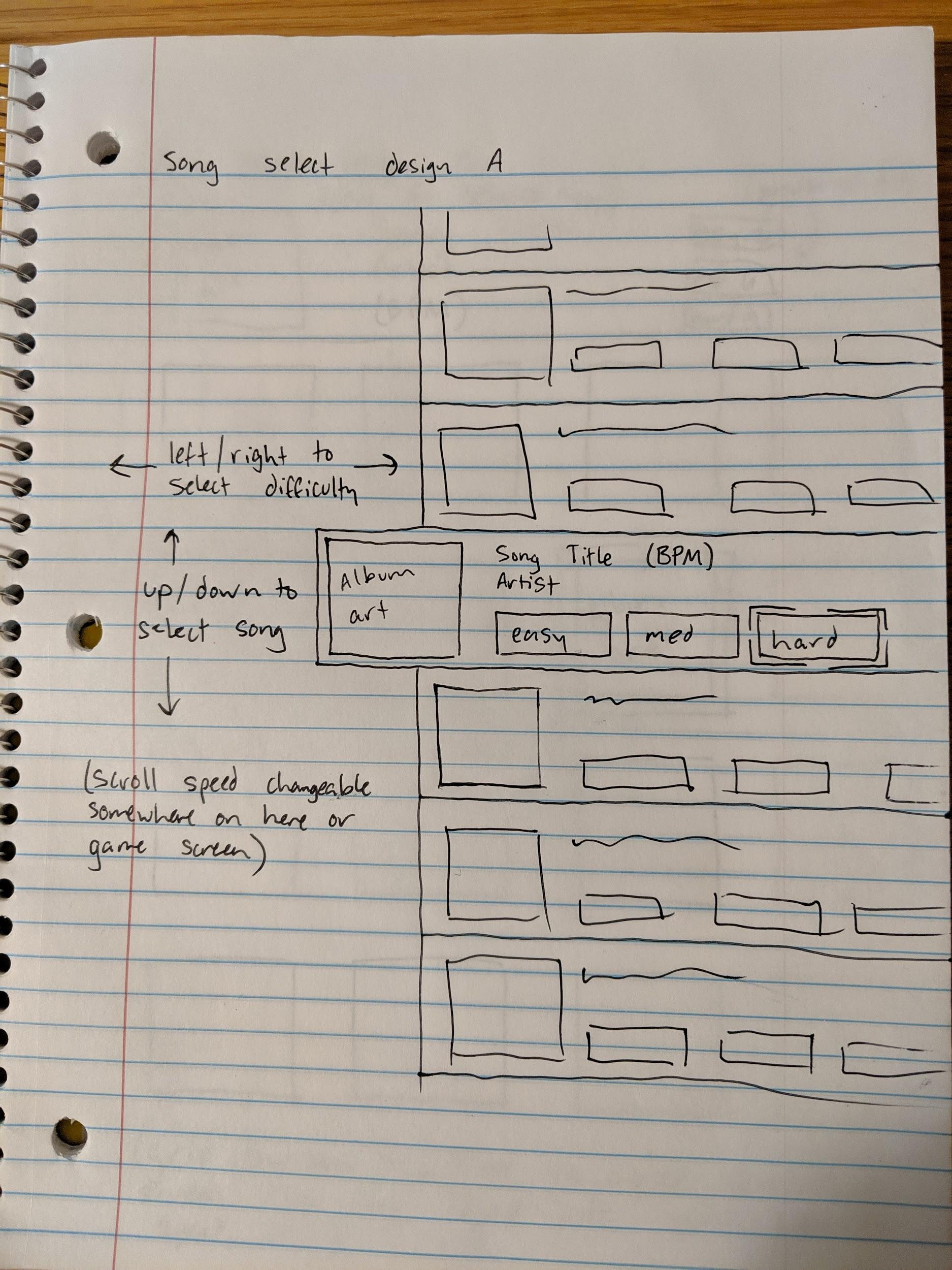
EXTRAS: (during or after project completed)

* Visual head and tail for hold notes
* Add warning for color switch (graphics done)
* Rework entire Lane graphical system to be more dynamic
* Add gameplay mods to increase or decrease difficulty
  + Hidden notes
  + Reverse scroll
  + Auto color switch
* Generate text file to save player info such as settings and scores

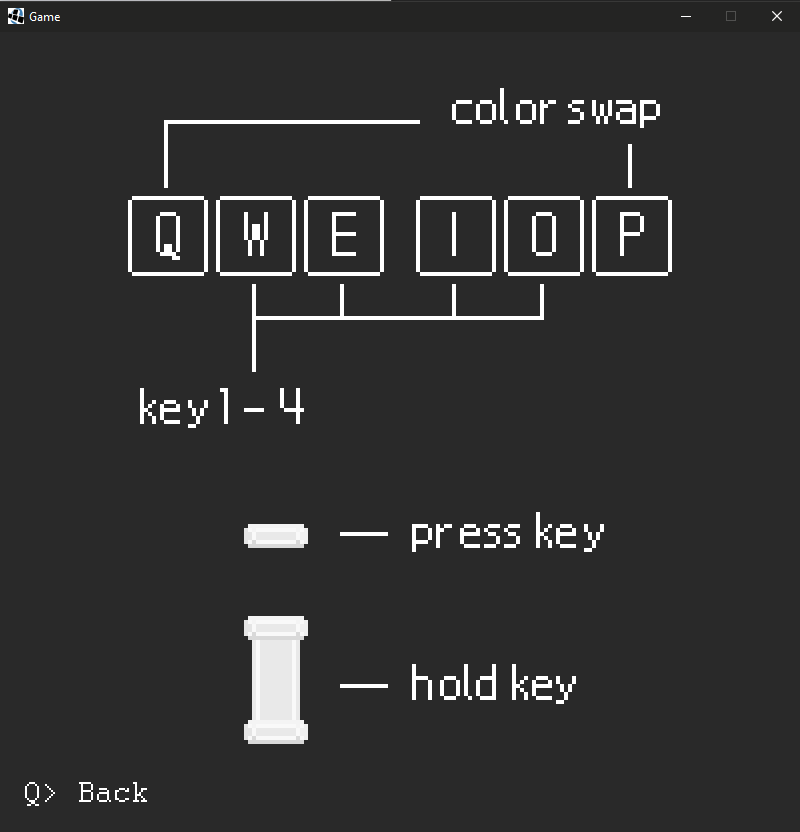
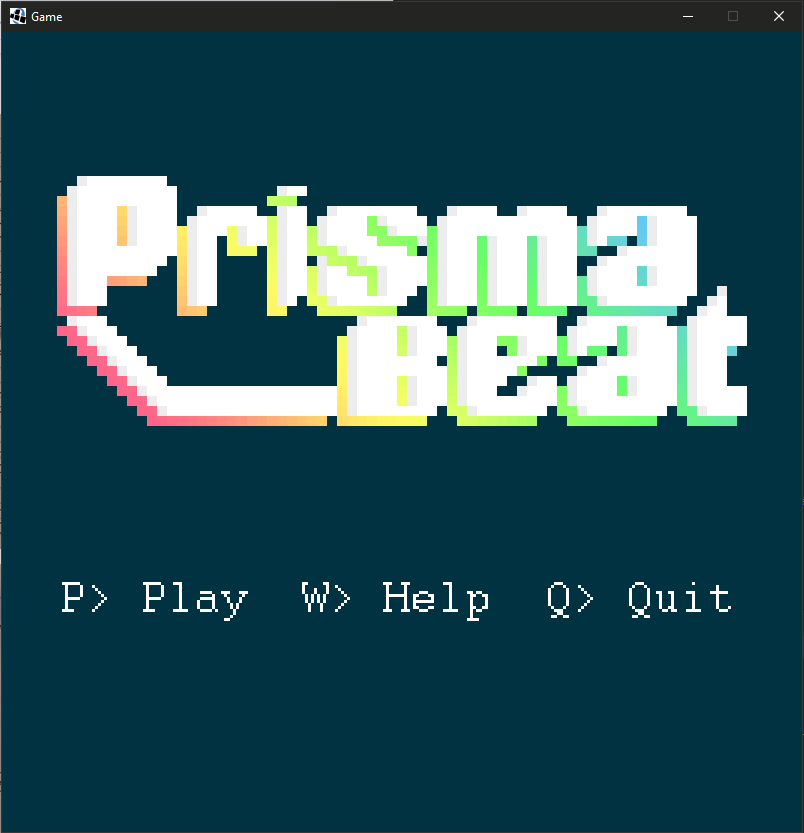
OTHERS:

* Remove possible memory leak with Linux

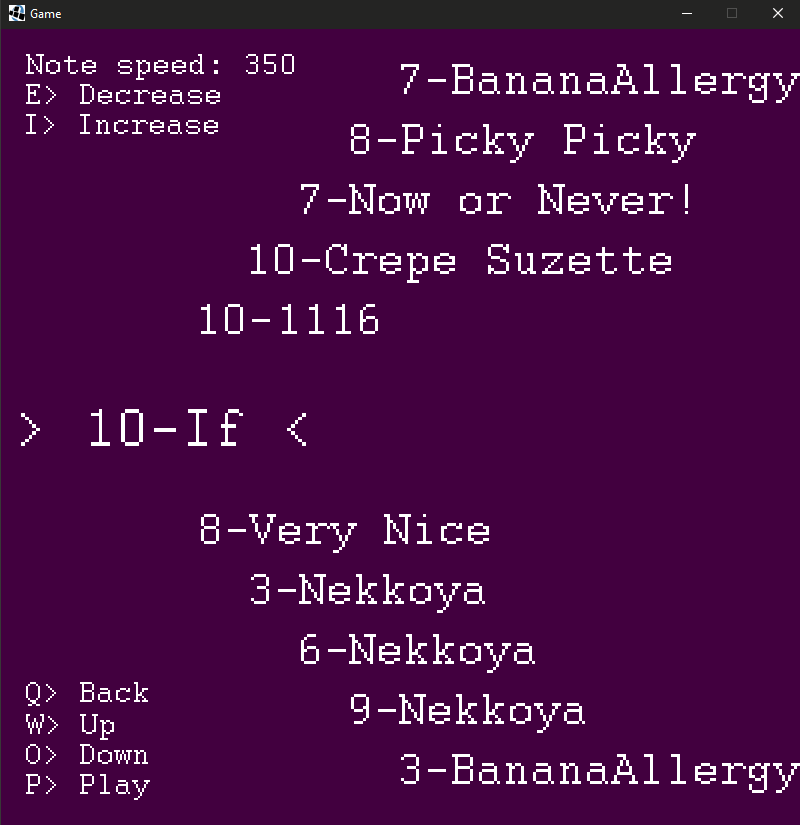
**Concept Art**



**Screenshots**

****

*Title Screen Help Screen*

****

*Song Select Screen Results Screen*

**Gameplay**

Below is a link to gameplay footage which includes the playing game state.

[Gameplay.mp4](https://drive.google.com/open?id=1JVBlfm17ovDPPyhFm7IMID1EacO0mXOF)