Dosu Final Report
by JB Ladera

Changes in game concept

The game now is much less a game and more so an editor for an existing game. Difficulty and lack of time resulted in this change, but fortunately code written prior could easily be reused.

The game is a beatmap editor for the rhythm game osu!. More specifically, it's an editor for one of its four game modes, osu!taiko. A beatmap or map is essentially a level in the game that contains objects that are timed such that when they are hit accurately, they are in sync with the beats of the song. In addition to these objects, there are things like the map background, song metadata, difficulty settings, etc. that are also associated with every map. My editor allows the user to edit all these things and is flexible enough to accommodate most, if not all, songs. Furthermore, the generated files will be compatible with the original game allowing for users' maps to be imported into the game and be playable.

What went well

Making changes to the scene in response to updates to the current beatmap in the editor was fairly easy due to the use of signals in Godot. For example, adding a circle to a place would update the top timeline to show that there's an object at that time. I managed to fit in a lot of features in a short amount of time that would help the user in creating a beatmap. The features, together, helps improve user experience overall.

What could have been better

I should've started early. I would've realized how difficult my original idea was much sooner and possibly could've made a better product than what I have now. Managing my time this project was not very good. Due to family activities, I sometimes missed several consecutive days (3 consecutive days at most) resulting in me falling behind. As a result, on days where I would work on this project, I would often spend 6 or more hours working on this.

In regards to the implementation, I tried to keep the structure clean and put things where they belong, but as time went on, it became less organized. In addition, there are several inefficient searches, insertions, and deletions. I aimed for something that worked well enough, so I spent less time on making things efficient if it didn't impact the user experience greatly.

Sound from the objects is sometimes skipped, too early, or too late. This is because of how sound is determined to play. In my implementation, sound is played from an object when

the song position is within five milliseconds of the object's set time. Because the song position reported is not entirely accurate, it will sometimes skip over an object's set time and so the sound will not play. Even with these accounts for error, my solution is not perfect. It would probably be best to have each object have a timer to play their sound, but this was the simplest solution I could do within the given time.

Other information
The README will contain additional information and links on how to use and other relevant things.

Features Implemented

- Main Menu
    - Can open the game's files where they can find the beatmaps and exported files
    - Switch to the edit screen to see current beatmaps
    - Can exit the game
- Edit screen
    - Can go back to the main menu
    - User can drag an mp3 file onto the game screen, and the game will create a new beatmap and switch to the editor scene
    - User can choose what maps they want to edit if they have multiple
        - Clicking on a beatmap will highlight it. The user must click the select button to go to the editor with that beatmap's data.
        - Background updates to the associated background when clicking on a beatmap. Defaults to a fixed background if the beatmap has no background.
    - Can refresh/reload the beatmaps
        - This is useful if the user edits the game's songs directory directly and wants to see the changes while the game is ongoing. Otherwise, the user must restart the game if they modified the directory directly to see the changes.
    - Can filter for beatmaps
        - The search query compares every beatmap's metadata; specifically, the title, title unicode, artist, artist unicode, creator, version, source, and tags.
- Editor screen
    - Bottom bar
        - Has a bottom timeline with a playhead to indicate song position

                - User can click on the timeline to move to a certain place in the song
                - Shows where existing timing points (red lines) are located in the song
            - Displays the current time in the song and percentage of the song covered
            - 3 Buttons (left-to-right)
                - Seek and start - goes to the beginning of the song and starts playing
                - Toggle pause - switches between pausing and playing the song
                - Seek and stop - goes to the beginning of the song and does not play
    - Top bar
        - Menu bar
            - File
                - Open directory - opens the beatmap's folder directory using the operating system's appropriate program
                - Export - exports the beatmap to an .osz file (requires 7zip in the command line) which can be used to import the user's map to osu!
                - Save - saves the user's changes to beatmap file (.osu)
                - Exit - exits the editor screen and returns to the edit screen
        - Tabs - alternate what's displayed on the screen by switching tabs
        - Has a top timeline to display objects, such as circles and timing points, and ticks
            - Ticks reflect the bpm of the most recent timing point and help indicate where objects could be placed
            - The user can hover over the top timeline, hold Alt, and use the scroll wheel to increase/decrease the timeline zoom (space between ticks).
        - Buttons to the left to adjust timeline zoom
        - Beat snap divisor to easily place different rhythms, such as 1/4 or 1/3 rhythms (quarter notes vs triplets).
    - Middle Body
        - Song setup - user can edit song/map metadata and adjust difficulty settings
        - Compose

- Shows the play area where objects will be placed
- Circle button - places a circle at the current song position. Overrides any existing circle at that time, if any.
- Delete button - deletes the circle, if any, at the current song position
- Right buttons affect the hit sound of the object during gameplay and the color of the object
    - Clicking the button will apply changes to the circle, if any, at the current song position
    - Has no change to sounds in my editor
    - None, Finish - red circle, (none by default)
    - Whistle, Clap - blue circle

- Timing - shows all timing points in the map
    - User can click on a timing point (left) to select it and display its information on the main body (right)
    - Buttons on the bottom left
        - Add button - adds a timing point at the current song position. Overrides any existing timing point at that time, if any.
        - Delete button - deletes the currently selected timing point
    - Main body (right)
        - Time - displays the current time in milliseconds the current selected timing point is set to
        - Use current time button - sets the time of the timing point to the current song time
        - BPM - set the bpm of the timing point
        - Time signature - set how many beats occur in a measure

- General behavior
    - User can use the mouse scroll wheel to move forward or backwards in regards to song position