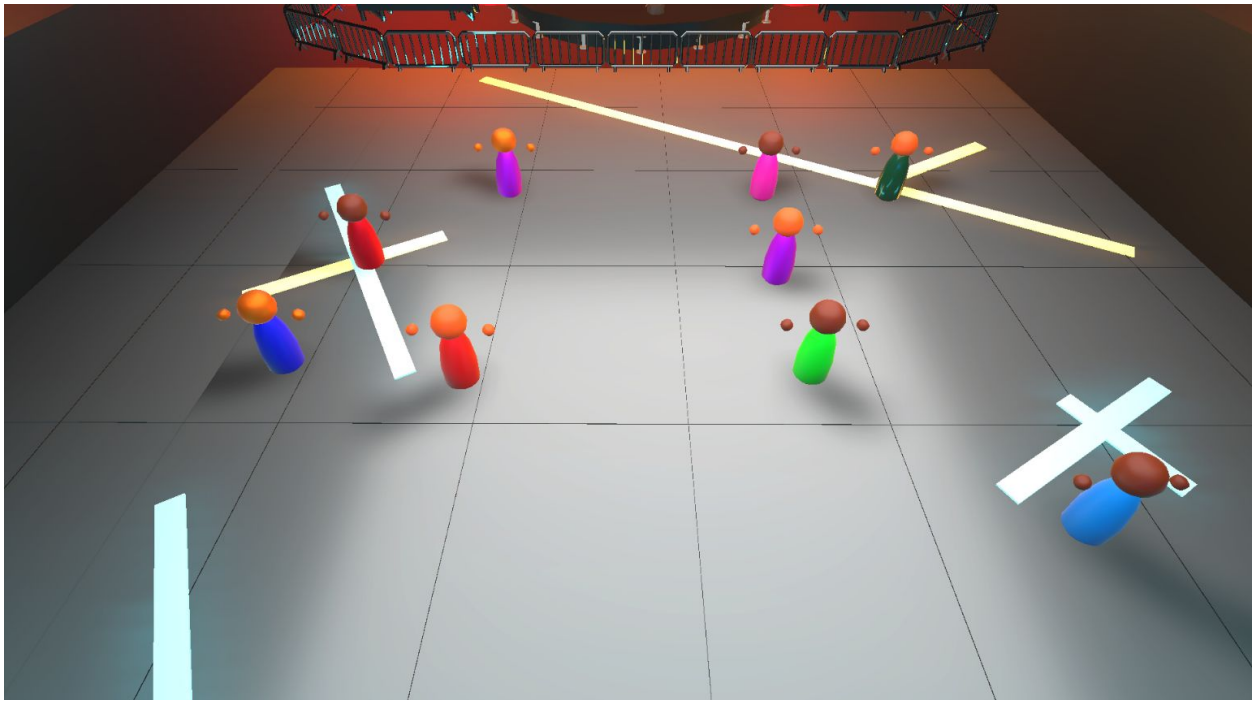
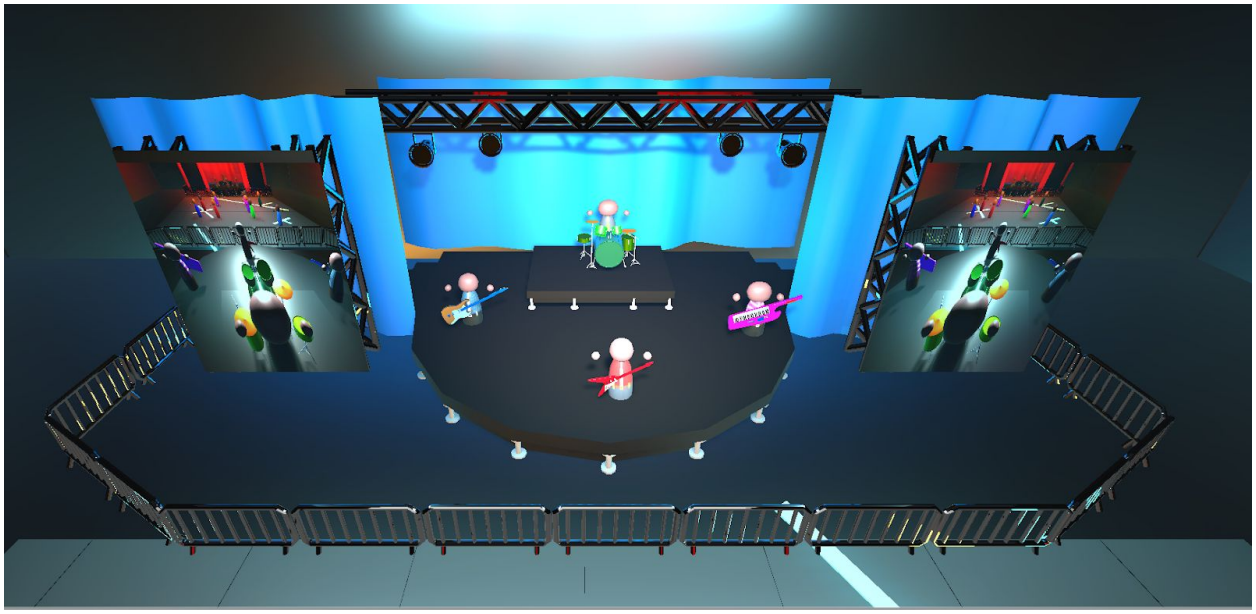
Portfolio for Battle of the Bands

DES502 - Game Design and Development

Rohan Menon - 1901120





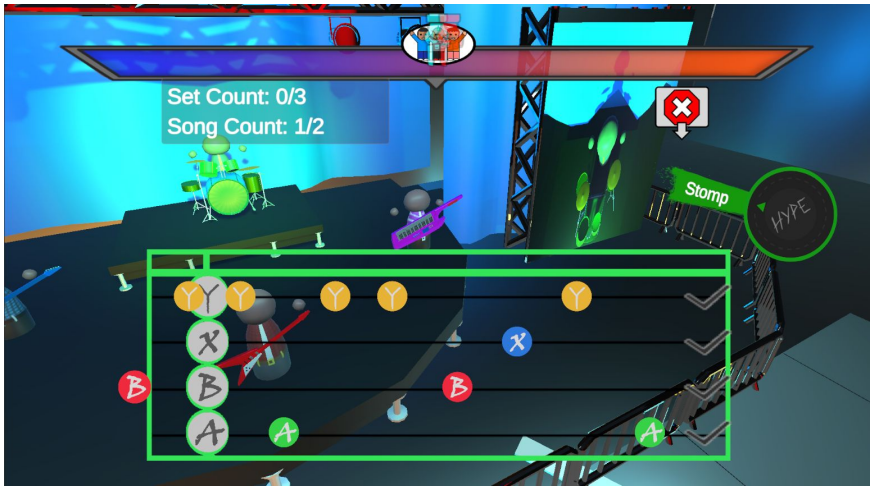
Game

You play as a band competing against other bands locally and nationwide in a tournament to win a record deal. The player can explore towns, complete side quests, and recruit additional band members. They can customise equipment as they win more renown

Prototyped Concept:

- Button prompt A corresponds to “Space” on PC and “B” corresponds to “ESC”
- The player starts off in the world and on pressing ‘E’ can enter a band battle scene.
- The player can select an instrument (From guitar, bass, keytar or drums) using the left joystick or “W/S” and “Up/Down”
- Then a corresponding move for each instrument with the same controls.
- Some moves are purely to gain additional crowd interest while some moves help enable special effects like disabling the next band’s turn, getting bonus moves or adding to the “Hype!” meter.
- The “Hype!” meter being full allows you to use an attack during player selection that doesn’t count as an additional move and allows you to play 2 songs together.
  Try to get better crowd interest and beat the enemy band.
- Not playing correctly (Pressing random buttons or missing notes) reduces crown interest.
  - Missing notes will tune out the sound for the selected instrument temporarily.
  - Pressing random/incorrect buttons will distort the sound of the selected instrument and also reduce hype meter slightly.
- The mini game is about precision and hitting the notes at the correct time to get OK, GOOD and PERFECT scores with corresponding crowd interest and scores. And also about using the correct moves in a strategic manner.
- The note collectors are controlled by 1,2,3,4 on PC and A,B,X,Y on your controller.
- To play the hyped move when the hype meter is full, press “ Left Cntrl” on PC and L2+ R2 on controllers.

Video link: https://youtu.be/I31cKd9qFGo

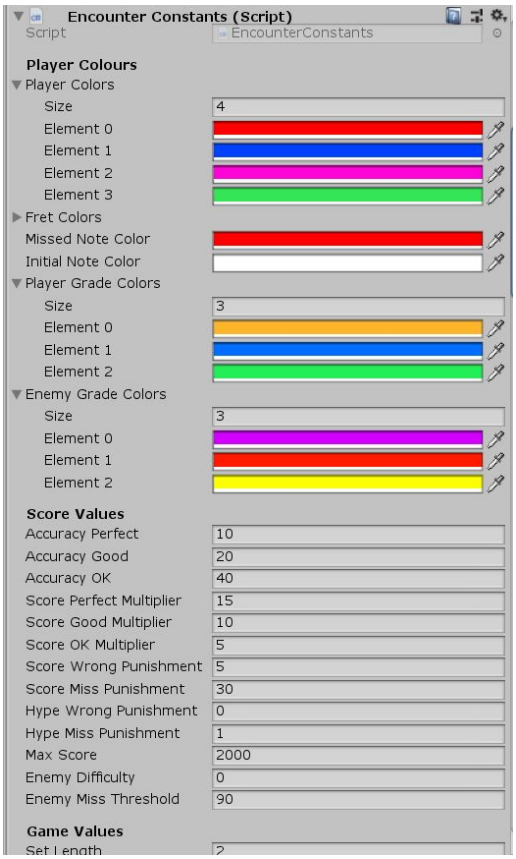


Scripts

1. EncounterConstants

The class that is used to define an encounters parameters like:

a. Colours (UI, Knob, fret board, crowd interest bar, move selector)
b. Band moves
c. Band special effects for the encounter scene
d. Crowd count and movement parameters
e. Enemy Band Difficulty
f. Accuracy reward



2. System between Encounter scene and Exploration Scene

An encounter scene is loaded with default values in it's "EncounterConstants" class, but the "EncounterConstants.cs" script can be modified to change things like difficulty when it is loaded from an Exploration scene. This will initialize the current members of the band, their moves and which all effects they can use and also set the difficulty of the enemy band being faced.

3. Finite State Machine Managers

The encounter scene uses 2 State Machine Managers:

- **Encounter Game manager:**

  This is what handles overall gameplay

  - Generates notes, return them to a game object pool
  - Calls functions for Lights and UI elements
  - Handles behaviour of move status effects (skip move, add bonus points, etc)
  - Also handles when to stop the move status effects
  - States:
    i. StartGame
       - Starts overlay animation
       - Creates connections between game elements

    ii. StartGameUI
       - Clears the UI and focuses on friendly band

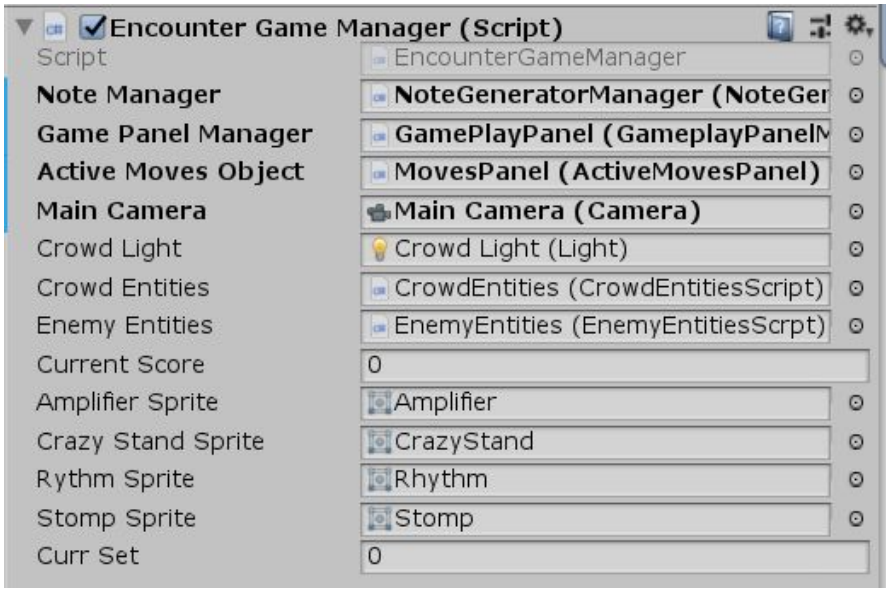
    iii. TurnIntro
       - Starting of a set
       - Focuses on player and shows a knob selector UI
       - This is when Notes Game Manager takes over

    iv. TurnPlay
       - State when player selection is over and player is completing a set

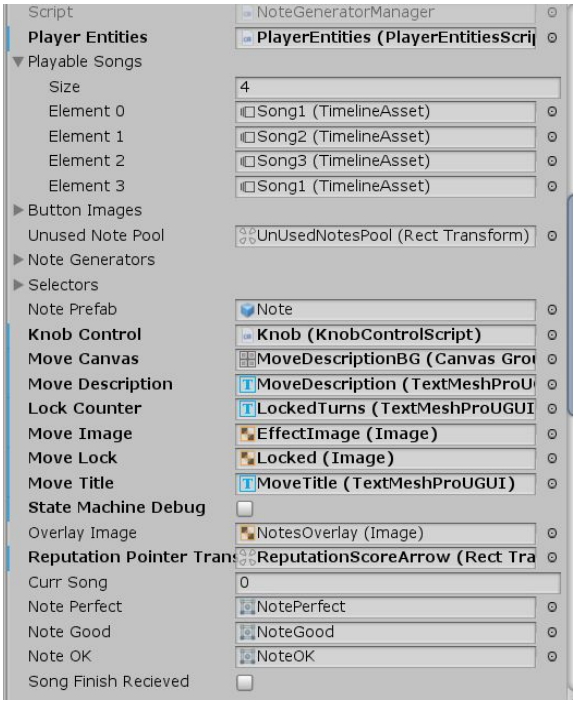
    v. TurnPlayOut
       - Once the game play of Note Game Manger is completed, the crowd reacts and moves to the correct locations for the turn
       - If the current number of sets is greater than total sets, go to EndGameUI state or go to TurnIntro state.

    vi. EndGameUI
       - Show winner labels and end game UI

    vii. EndGame
       - Go back to Exploration Scene

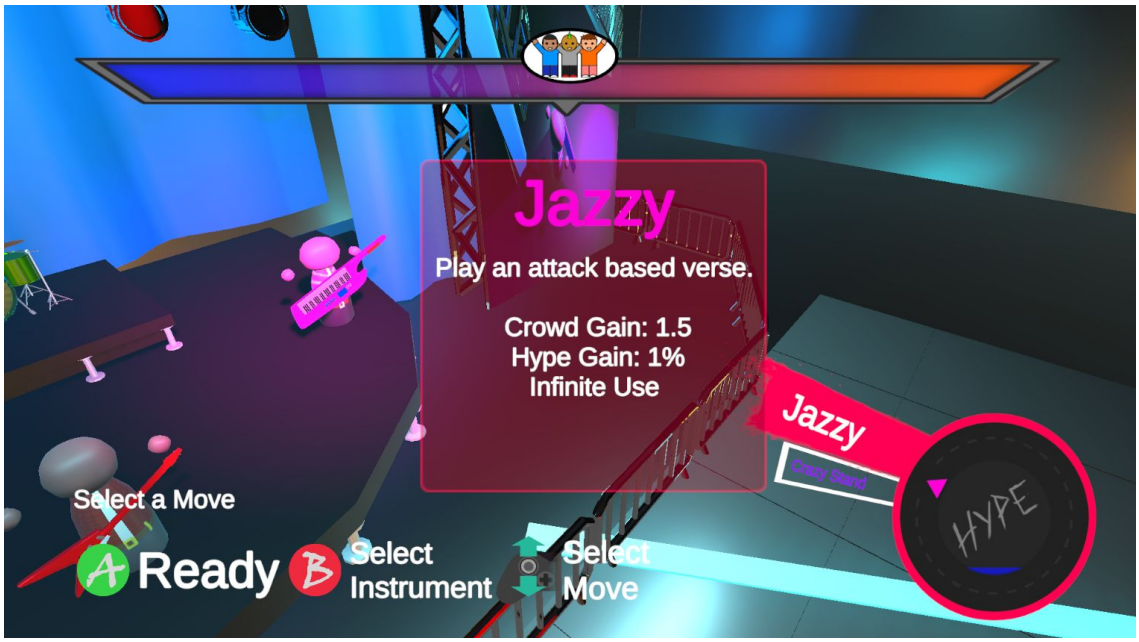


- **Note Game Manager:**
  This is what handles the notes minigame
  - States:
    i. Start
       - Start of note generator UI
    ii. Intro
       - Show Instrument/Player selection
    iii. MoveSelect
       - Show moves selection after player is selected
       - Allows going back to Intro
    iv. Playing
       - When the user is playing the notes game
       - When the notes are finished generating, go to enemy intro
    v. EnemyIntro
       - Turn and introduce enemy,
       - If turn is being missed, go to Intro or go to Enemy
    vi. Enemy
       - When the enemy band and UI is shown
       - If the current song is greater than the set song length, go to EndSet, else go to Intro.
    vii. EndSet
       - Set has ended and it moves Encounter Game Manager to TurnPlayOut
    viii. Hyped
       - State when hyped song sequence is being played(both sets)
    ix. Idle
       - Idle states when notes aren't being shown



4. Move Sets

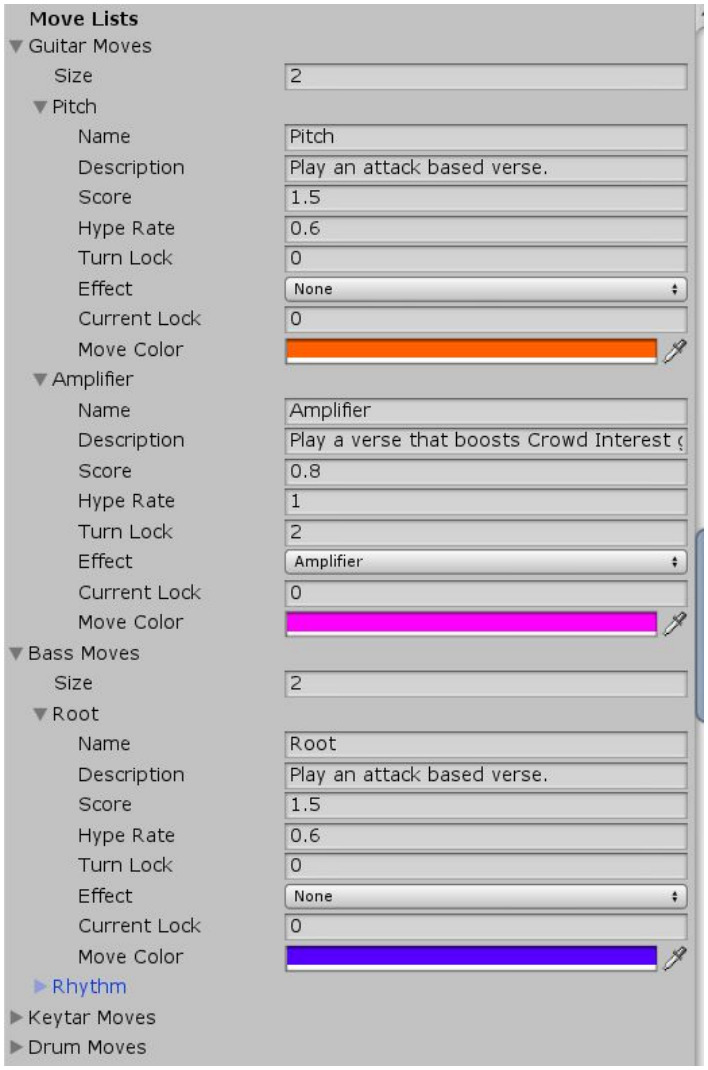
All band members have 2 attacks, 1 that generally provides a large crowd interest gain.



The other ones have special effects like:

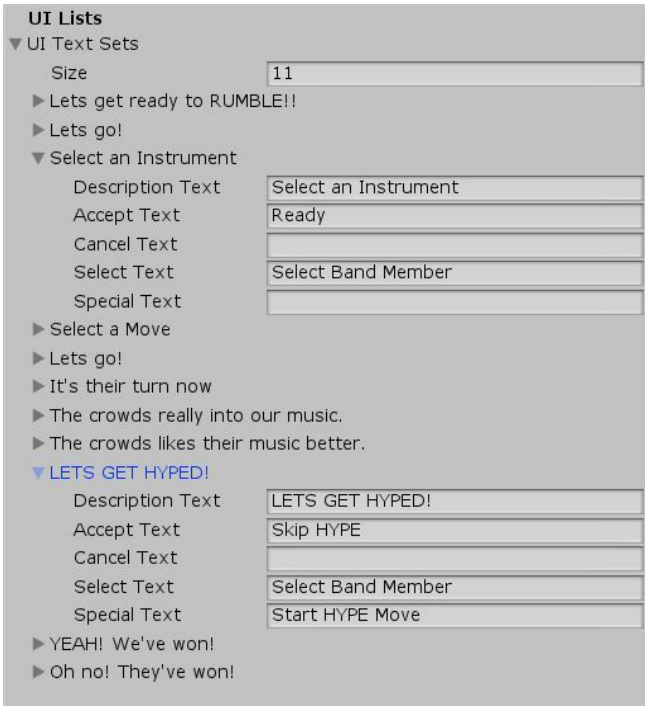
- Amplifier(Guitarist) : Play a verse that boosts Crowd Interest gain for the team, for 1 turn.
- Rhythm(Bassist) : Play a verse that reduces Crowd Interest gain for the opponent's team for 1 turn.
- Stomp(Drummer) : Play a verse that skips the next opponent's turn.
- Crazy Stand(Keytarist) : Play a devastating randomized verse of fast notes.

The moves have their own predefined colour settings (set by the designers) this allows for a lot of modification and balancing.
Usually, when an encounter starts, the band's regular and upgraded moveset will be looked up and EncounterConstants will be populated with a new move set. Displayed below are 2 attacks for the guitarist and 1 attack for the Bassist. This is the UI that was used by designers to tweak values and change different properties of an attack like, score, hype rate, locking, effects and colours



5. Main UI

The main UI class is used to show button prompts and descriptions of what needs to be done using TextMeshPro objects. This allows customization of the text and allows our entire dialogue system to be in EncounterConstants which can be modified by the main game as well. This can later be used to add customizable text from the exploration scene as well. For example if a particular character or player needs to be called out.

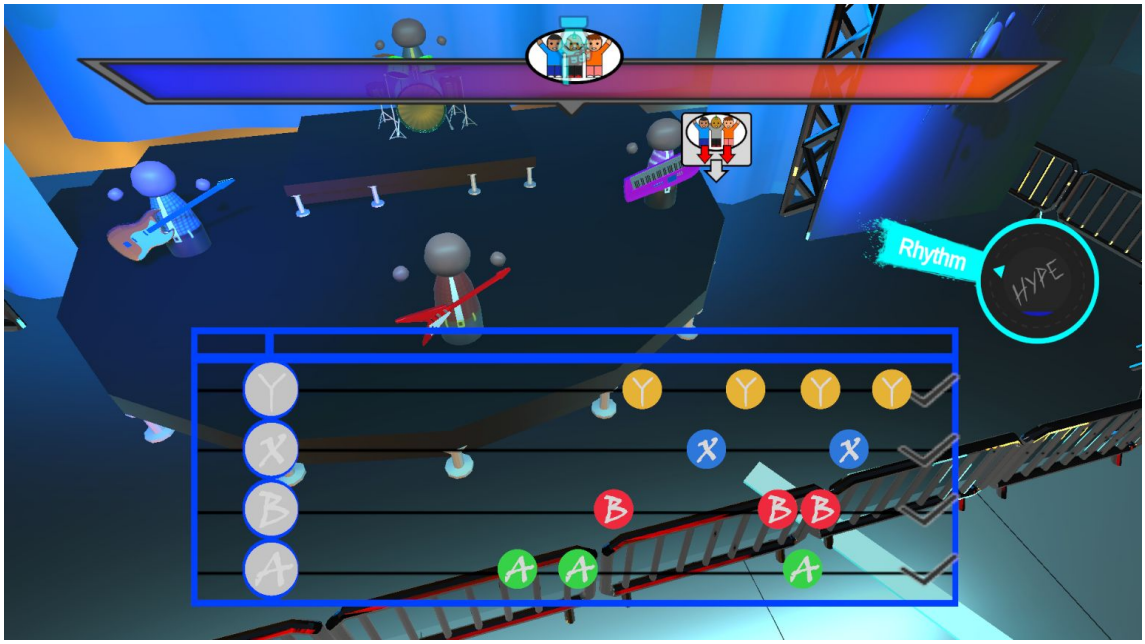


Different animations for the knob UI, selection and knobs fading away were also implemented as can be seen in the gameplay video. This was done through DoTween.

6. Notes gameplay

The mini game for actually collecting notes is played by pressing the correct button for each incoming music note. Pressing the note correctly at the right time will apply a grade to it that gives bonus score values. Pressing a note button unnecessarily will distort the music for the instrument

and reduce your crowd interest. There was a plan to also have different mini games and patterns for each instrument but that was left as future scope for design.



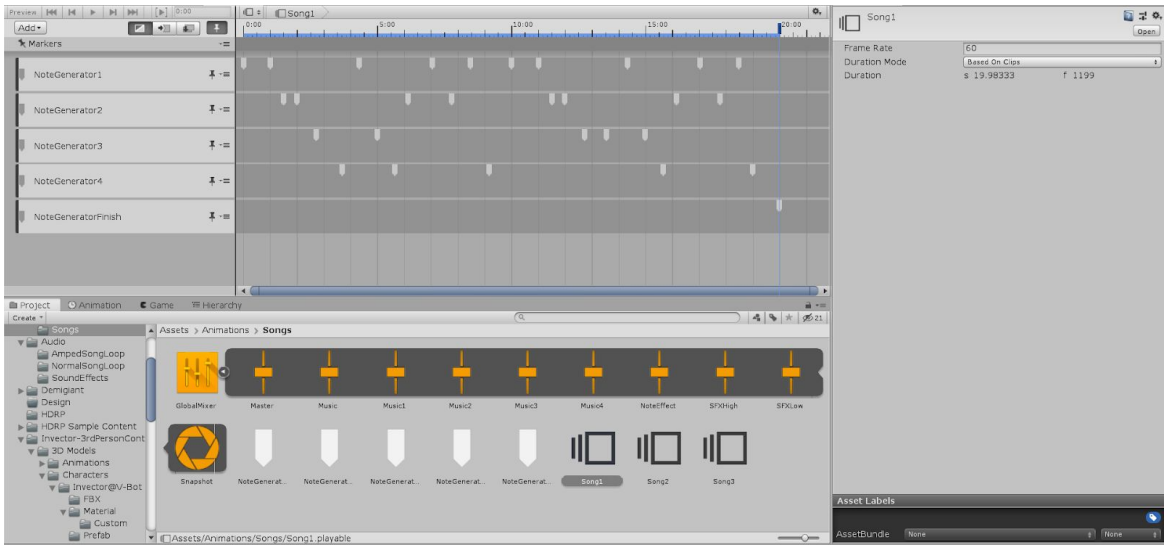
The game uses a Unity Playable Director to create notes from a timeline that is created by the designers. The timeline calls functions that create the notes in a synchronized order.

The gameplay involves “Notes” classes created in the “NoteGeneratorManager”. “Selectors” with colliders are used to detect if a collision is happening with any nearby notes when it is pressed. If nearby a grade is assigned to the note and it is collected with a corresponding increase in hype value as well.

When the notes game is being played, 4 different audio sources(for each instrument are activated) and all of them play a set of audio together(to allow for syncing). The audio is played on 4 different Sound Groups and will be discussed in the “Audio Mixer section”.

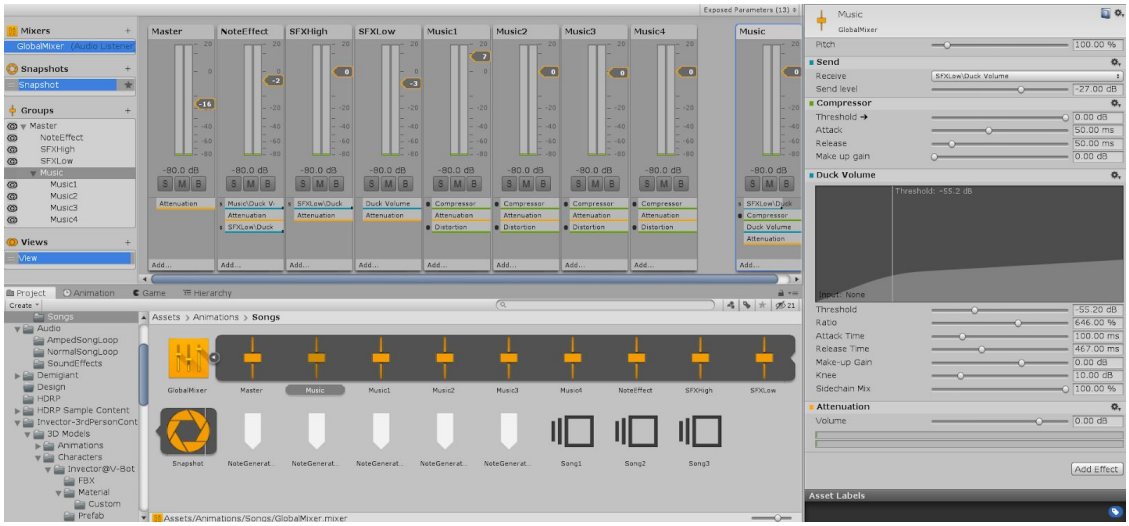
7. Unity Timeline

The unity timeline was used to set up song patterns by one of the designers (the same one who handled creating the song files). This allowed him to correctly time notes appearing and also made him able to test out different iterations quickly and easily. It involved using a Playable Director in the NoteManagerClass that will listen to signals and call a function to generate the notes at the correct position.



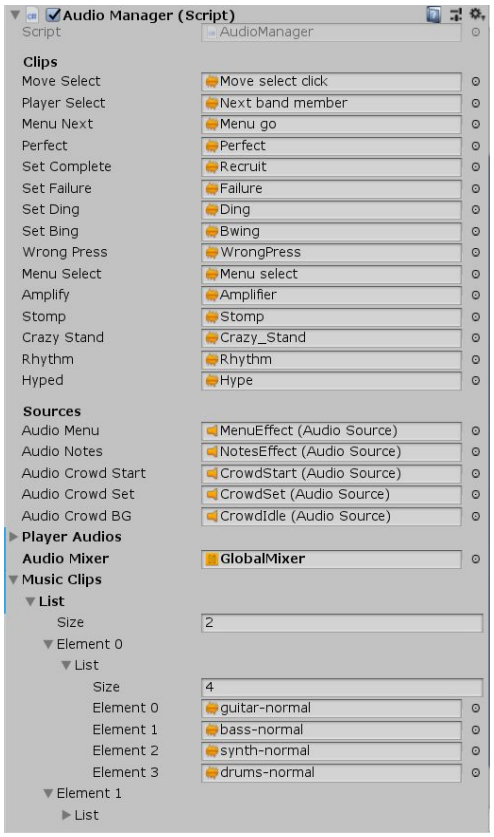
8. Audio Manager

Audio was handled using a Unity Audio Mixer which had different mixer groups. The mixer groups would also allow for ducking and compressing certain channels so that sound effects can be applied on individual tracks rather than the song as a whole.



Audio Mixer groups were also assigned for low(crowds talking)/high(crowds cheering) sound effects. There were ducking, compressor and distortion systems set up on each group to allow certain sounds to stand out.

A custom AudioManager class was used to decide when to play different crowd, song and sound effects and also which clips and listeners they were assigned to.



9. Enemy Players

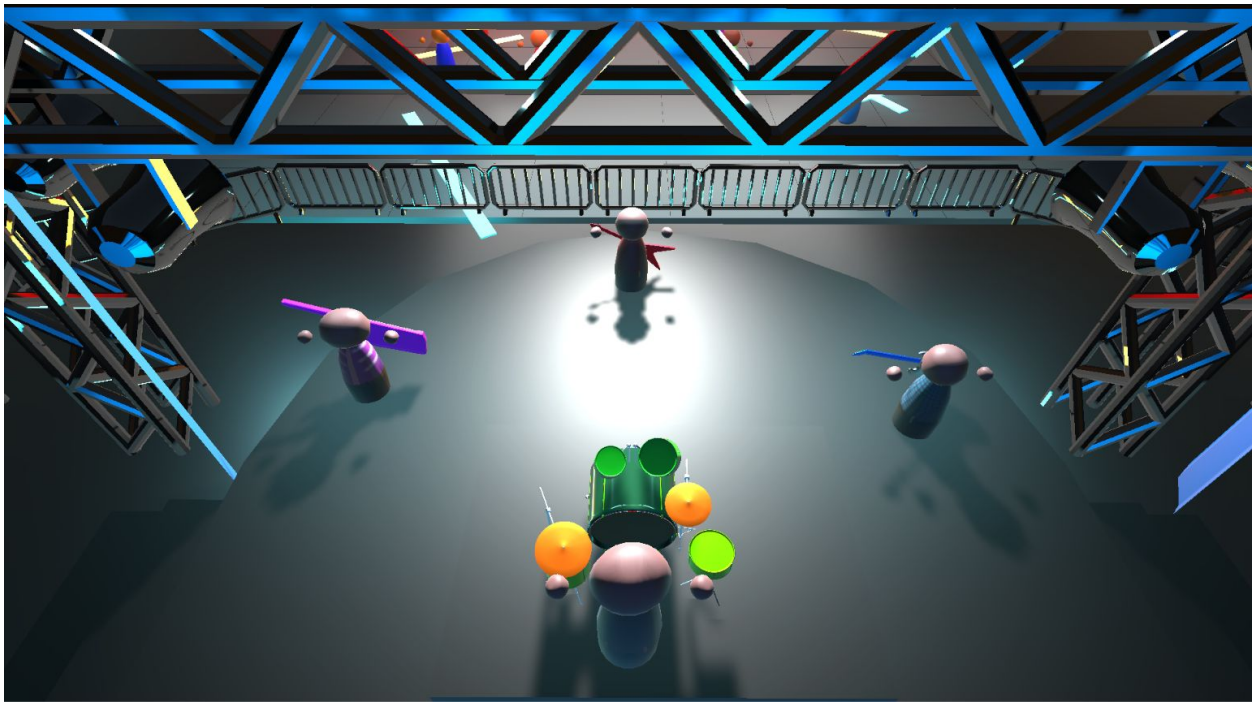
The enemies do not have their own moves as of now and will only play a predefined set of track sequences. But their difficulty can be modified to allow for more misses or more and precise attacks when they are playing depending on how difficulty should be scaled.

10. Object Pooling

Since notes being generated are a time sensitive function, object pooling needs to be implemented so that too much memory is not wasted instantiating and destroying objects. Instead 200 deactivated notes were maintained during run time so that they can be used when they are required.

11. Tool Used: HDRP

HDRP was used as we were planning to make the game for consoles and High Fidelity was important. Due to a few issues in the pipeline, we weren't able to get textured assets due to confusions in the materials for the 3D models being set to the default material name. This meant reworking the texture files provided by the artists during which bender was used to create material properties and metallic reflections/cloth dullness.

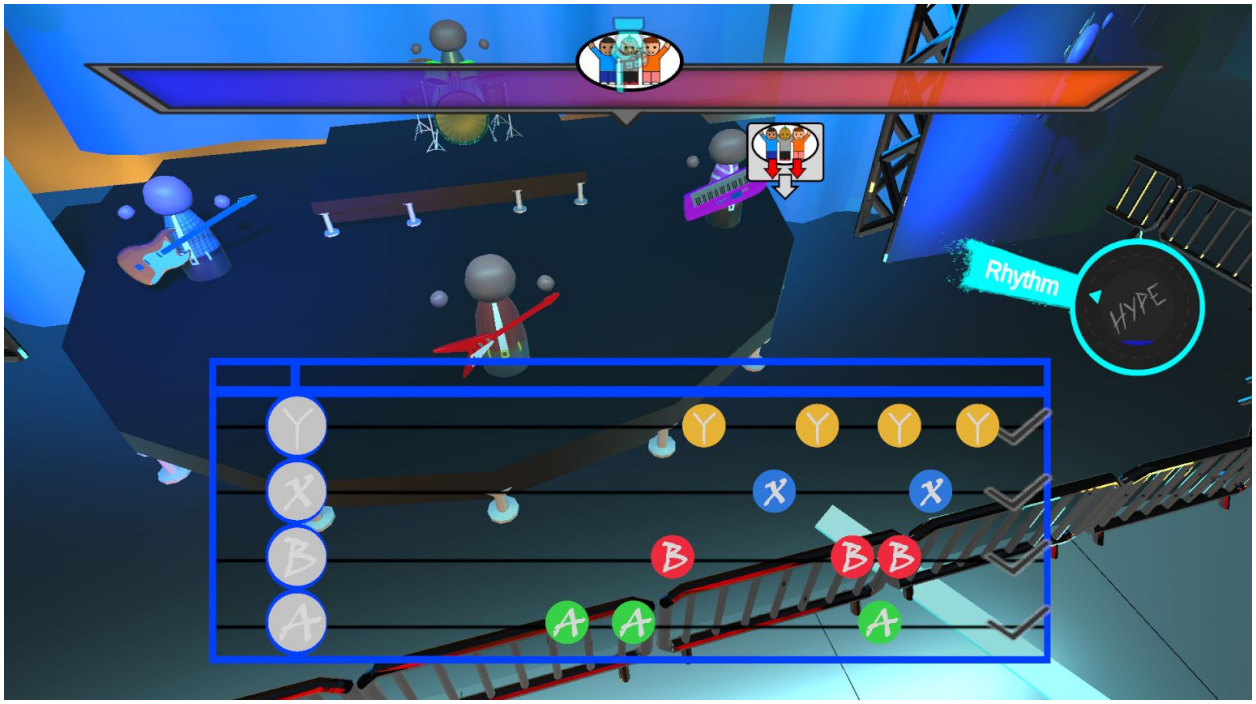


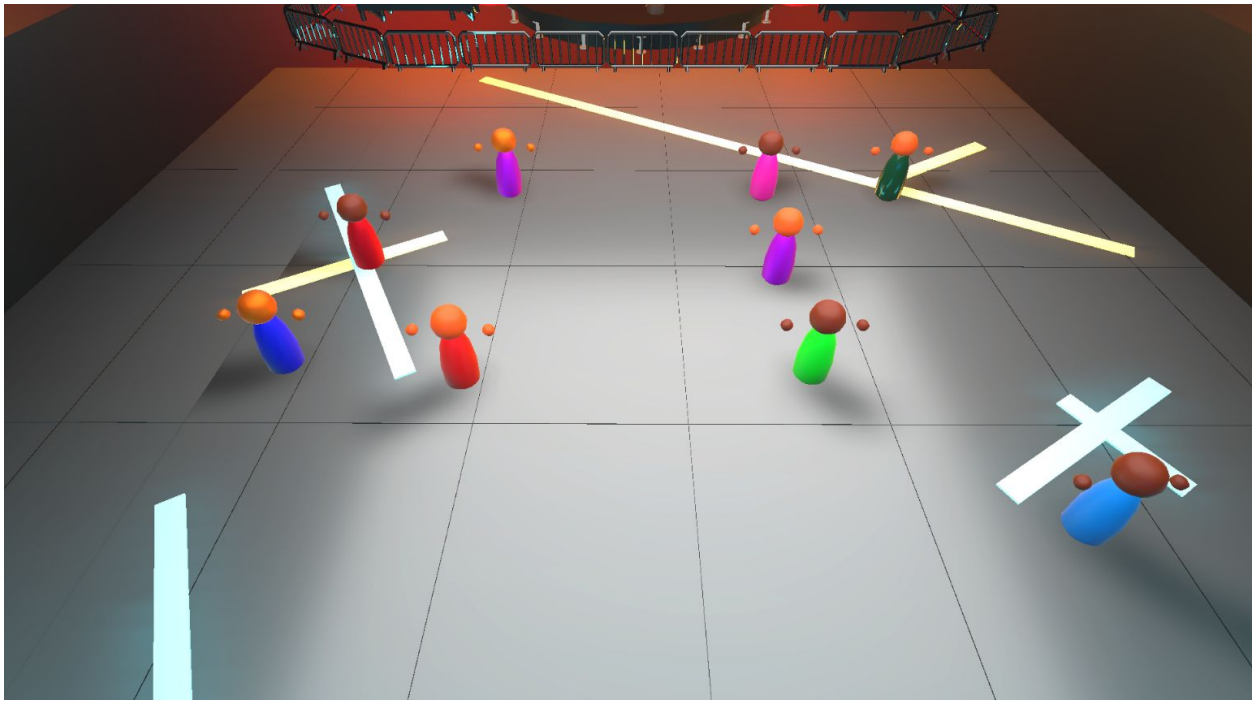
Reflection Probes were used to correctly reflect metallic objects along with volumetric and baked lighting to ensure performance as well as visual appeal.

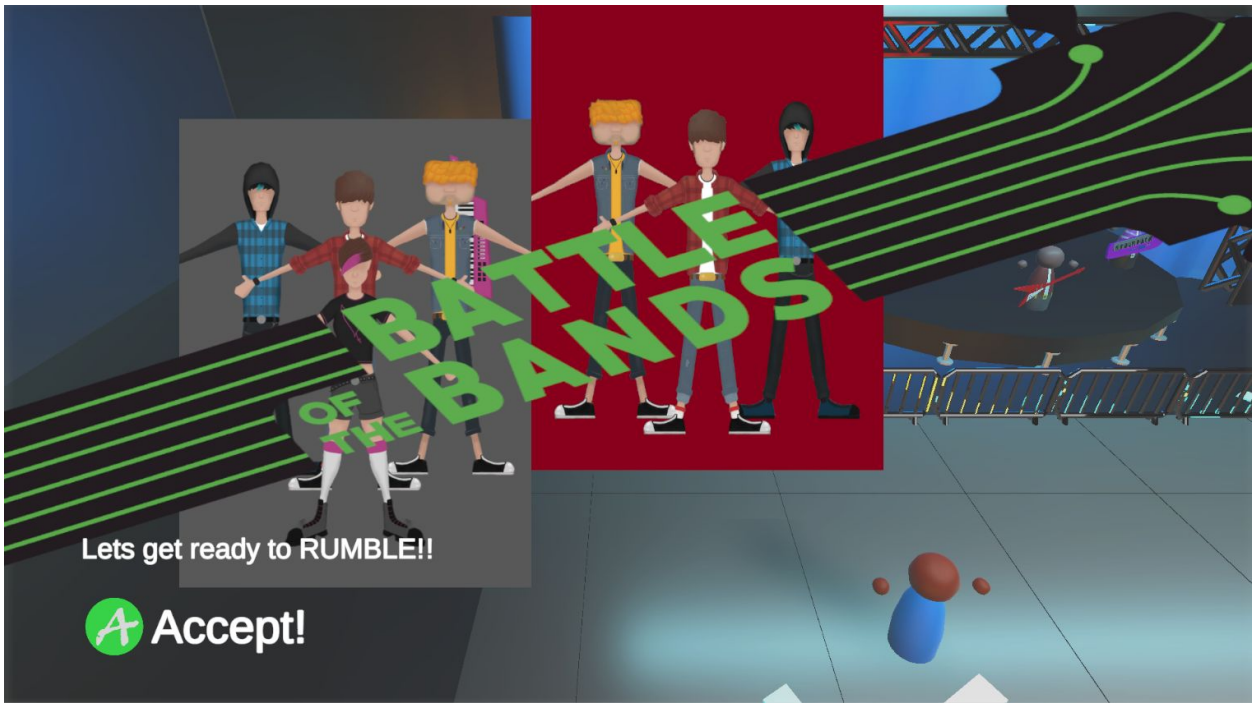
12. Tool used: DOTween

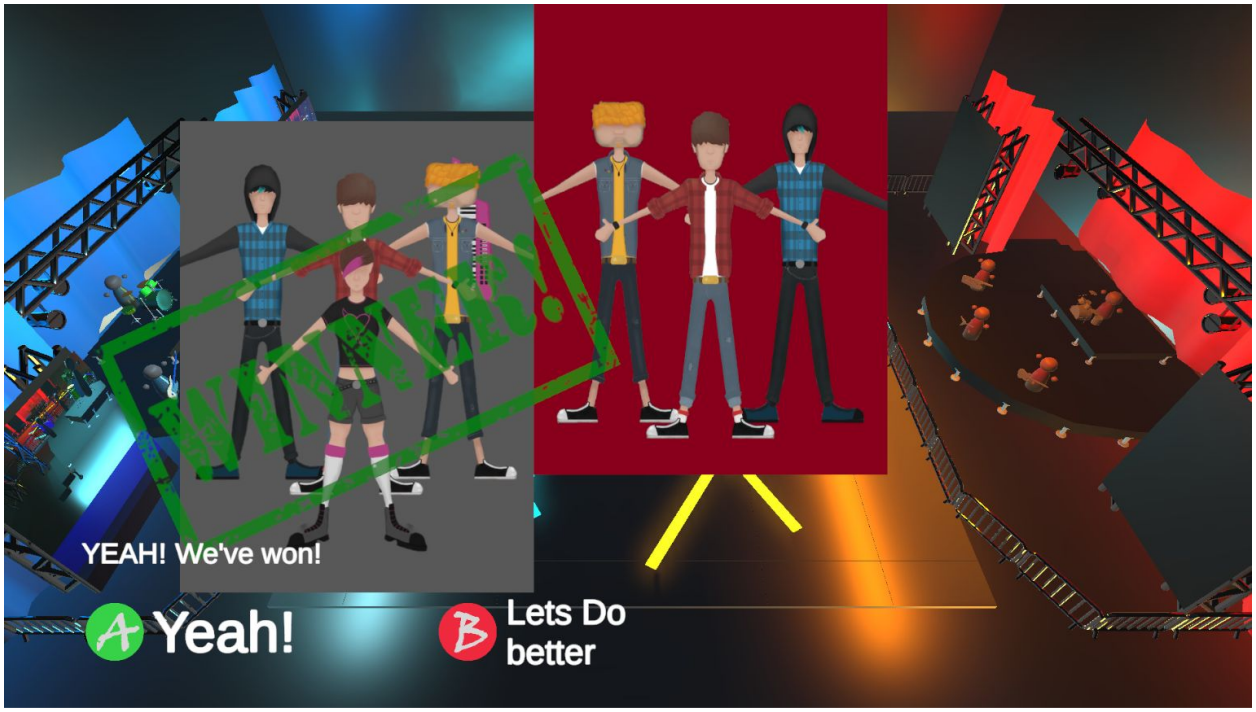
Do tween is a free unity tool that allows for quick animations using scripts and also synchronizing them with callbacks. This was very useful to have certain animations playing and syncing up timelines since it made sure that animation transitions would be completed before a new input is registered in the state machine. This was used for crowd movement, light animations, and also UI animations. It is a powerful tool that allows for easy sequencing of animations as well.

13. Screenshots:






BATTLE OF THE BANDS
Lets get ready to RUMBLE!!
Accept!


YEAH! We've won!
A Yeah!
B Lets Do better