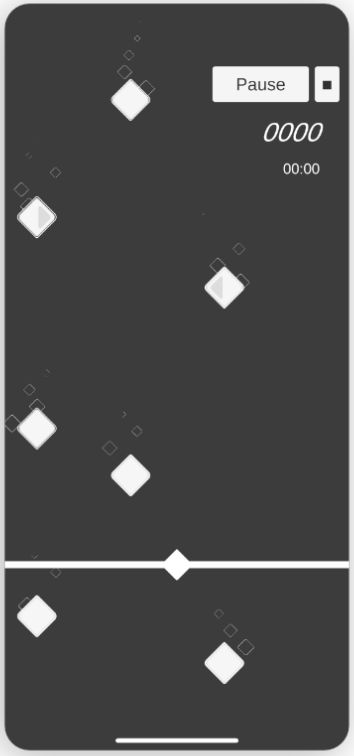
RhythmGameStarter Manual v 0.1.2

Supported Unity Version : 2018.4 LTS+
Suggested Unity Version : 2019.3+
Email : itechbenny@gmail.com
Website : bennykok.com



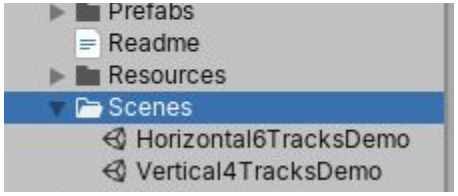
Features:

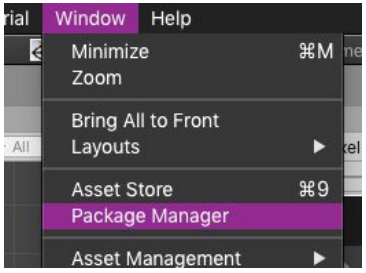
A starter rhythm game project with a complete workflow support importing midi file and mapping to different type of note type, in runtime the note transform will be synchronized with the music playing, also comes with a simple stats system.

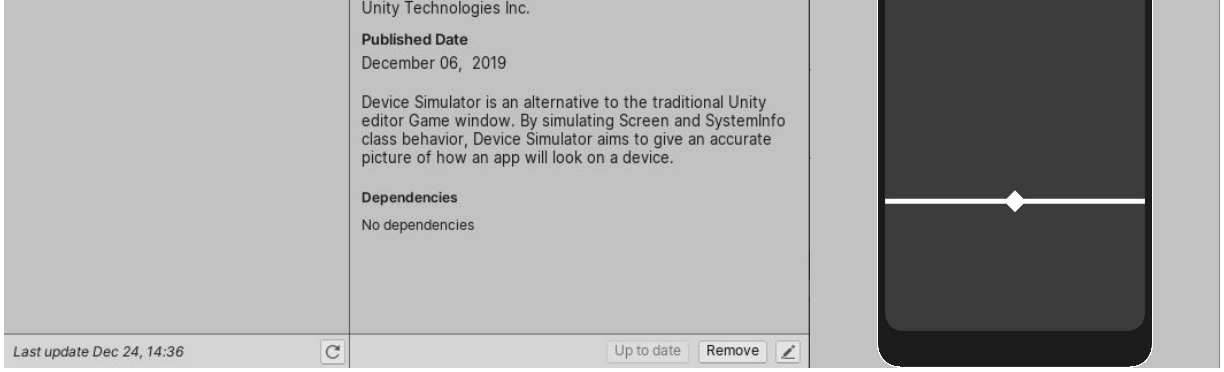
- ➜ A starter rhythm game project with full source included
- ➜ Touch input and keyboard input
- ➜ Importing midi file
- ➜ Object pooling note system and effect system
- ➜ Different type of note interaction (Tap, Long Press, Swipe)
- ➜ Synchronized music playback and note movement
- ➜ 2 Demo scene, Vertical 4 tracks & Horizontal 6 tracks
- ➜ Stats System (Combo, Score, Missed.. etc)

Setup/

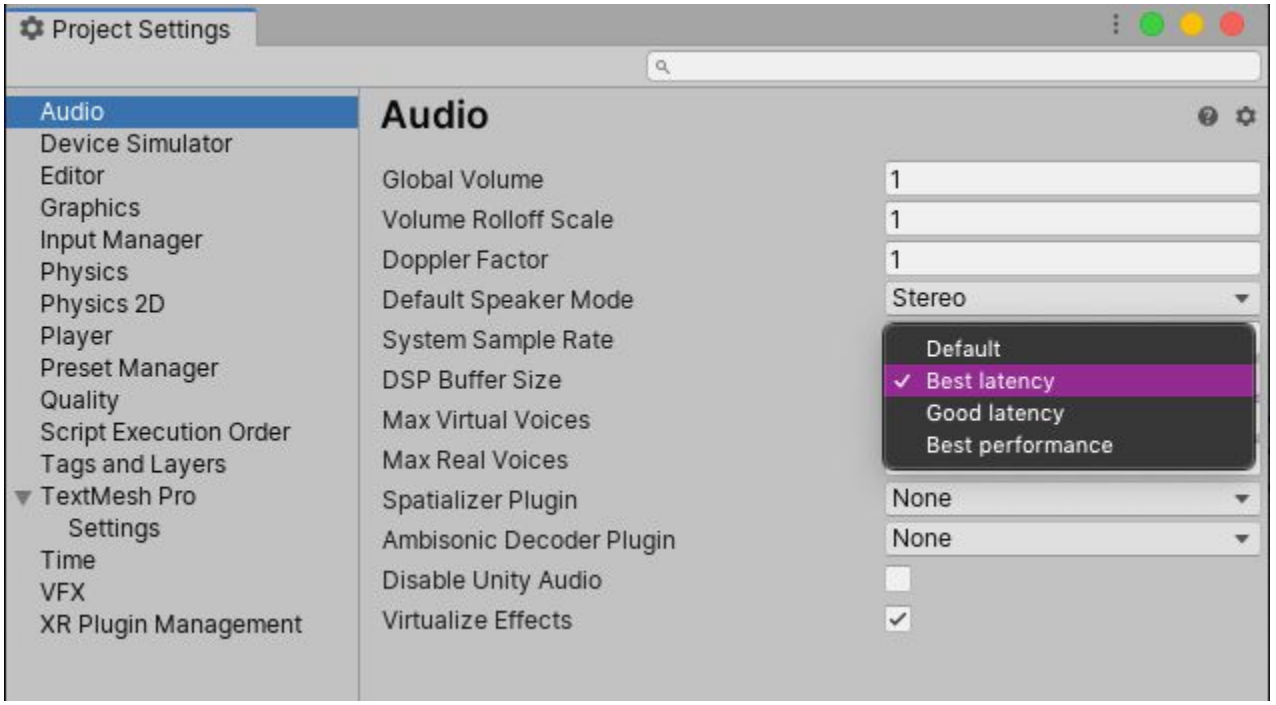
1. Import the unity package
   (Suggested to be using 2019.3+ with Universal Render Pipeline)
2. Locate and check out the 2 demo scenes
3. To test with mobile touch input, please install the device simulator from the package manager
   (Device Simulator only support 2019.3+)







4. Before testing the game, its suggested to set the dsp buffer size to ”Best latency” in the Audio Project Settings.

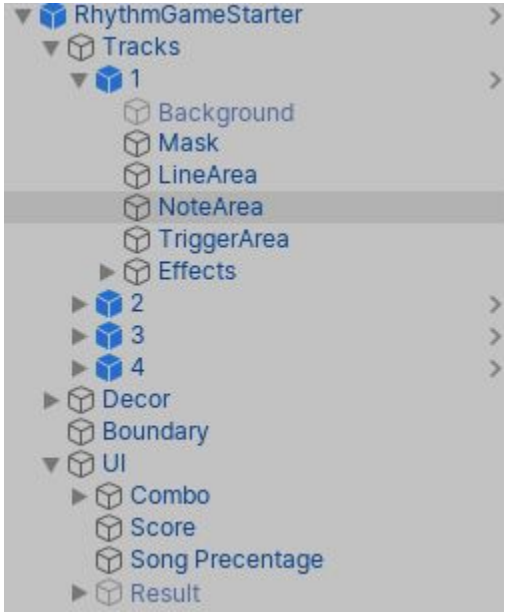


5. The demo scene uses TextMeshPro, so you will also need to set it up.
6. Press play in the demo scene and you should be good to go!

Overview/

You will find the “RhythmGameStarter” prefab in each of the demo scene, and the structure is explained here.

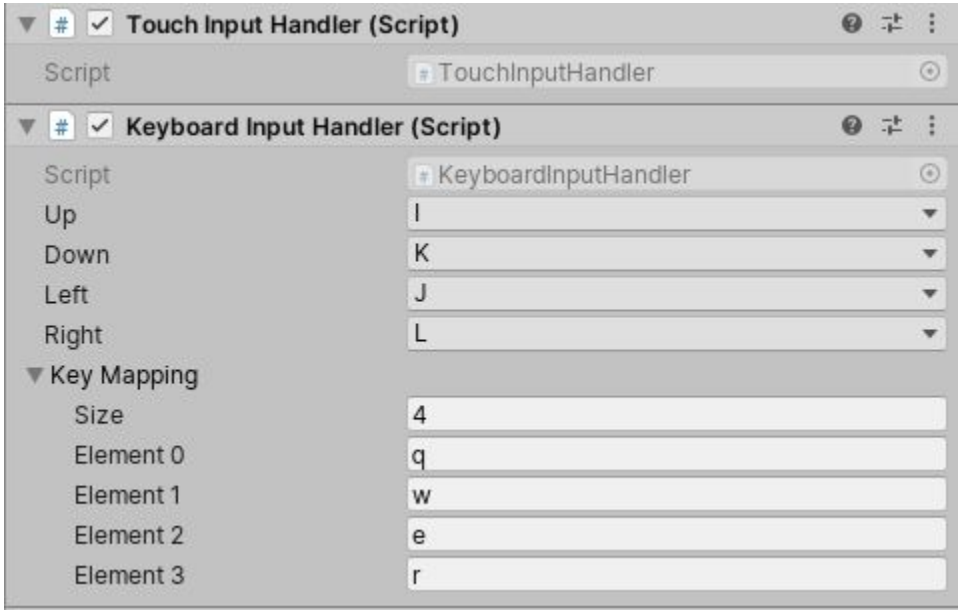
1. **RhythmGameStarter** holds main system’s script for audio playback and synchronisation.
2. Each track is a seperate **Track** prefab and can be positioned freely.
3. The **Boundary** is where the note get recycled or destroyed when the note passed through.
4. The **UI** is a template interface for showing the combo, score, song percentage and the stats information, and most UI is linked by the built-in **UnityEvent** from corresponding script.



Input/

In the “RhythmGameStarter” prefab, there’s two script which handles

5. **Touch input** : to test with touch input, please install the device simulator from the package manager
6. **Keyboard input** : you can specify different keys for each track, and also the direction key for “swipe” note, which you can trigger by pressing the corresponding track and direction key at the same time

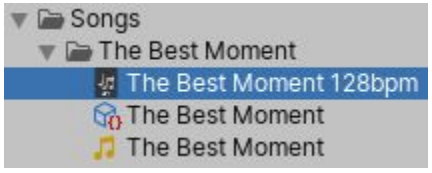


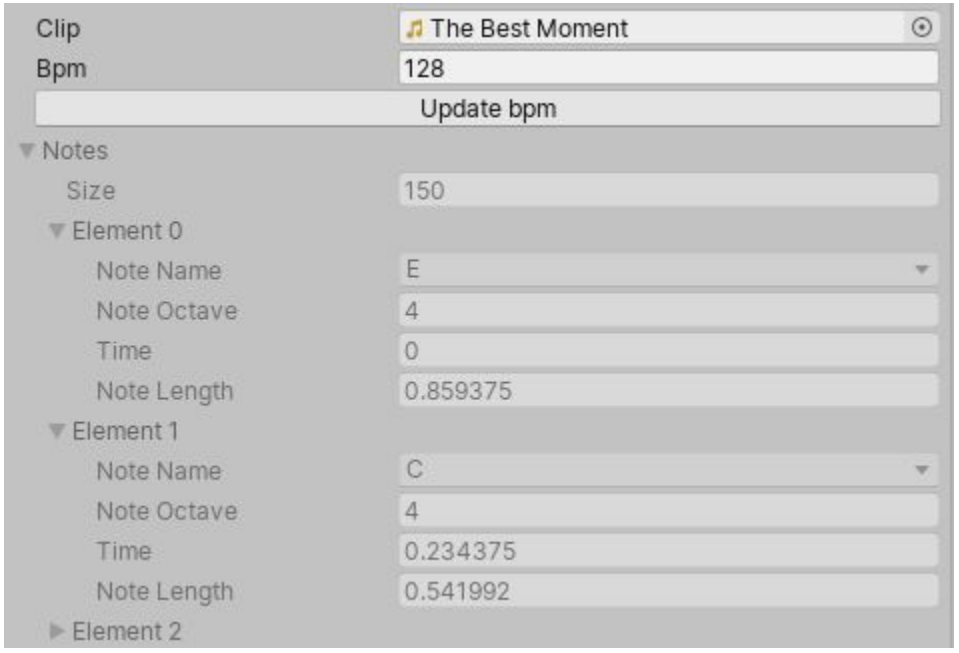
Midi/

Midi file is used to generate note object on the fly, so in order to create your own note sequence for your songs, you have to create a midi file using other software, possibly some DAW as FL Studio, Logic Pro, Ableton.

However, RhythmGameStarter will convert your midi file into a **SongItem** scriptable object and will parse it using “DryWetMidi” open source midi library.

Be sure to put a bpm tag in the midi file, thus the **SongItem** will automatically read it from the midi file name.

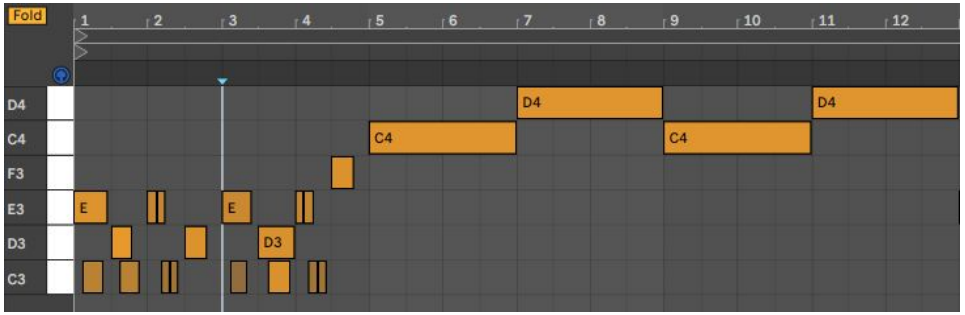


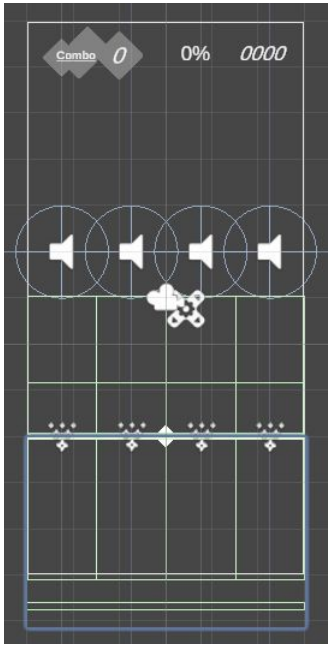


Midi Mapping/

The screen below is the midi editor in Ableton Live (DAW), if you have prior knowledge or experience with digital music production, then it will be easier for you to understand.

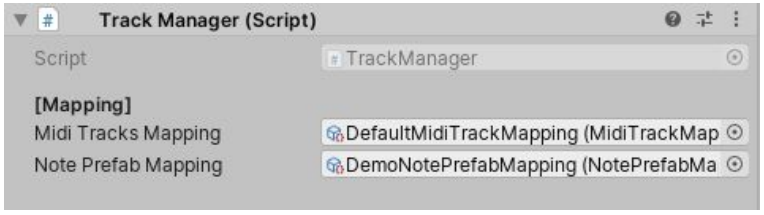
Simply put, we have to map the notes here to the tracks in game.





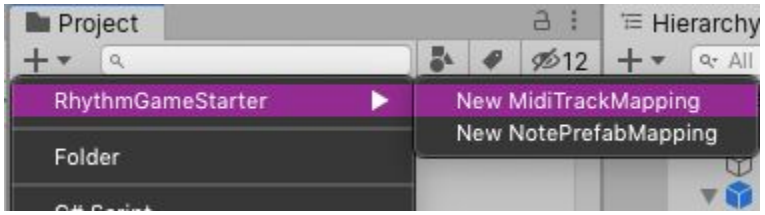
That's how the two mapping scriptable object come into place

1. **Midi Track Mapping**
   (Map the note in midi to the track in game)
2. **Note Prefab Mapping**
   (Map the note to different types of note object)



Midi Track Mapping/

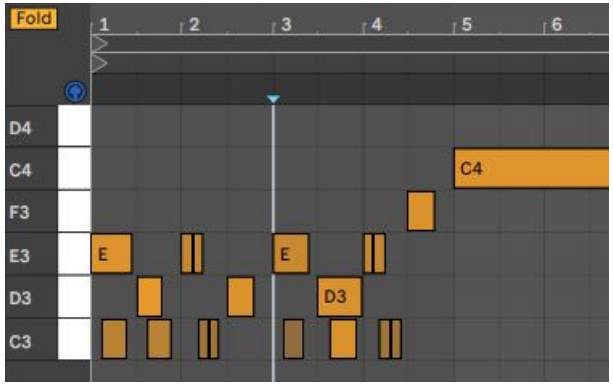
To create new track mapping

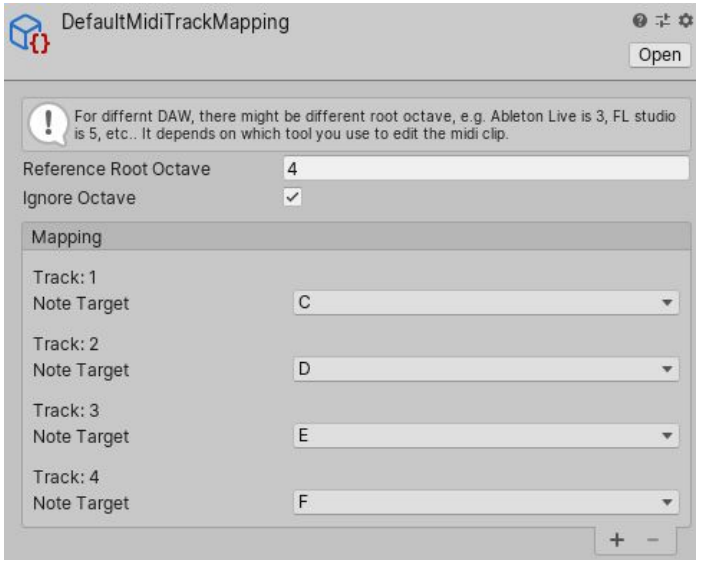


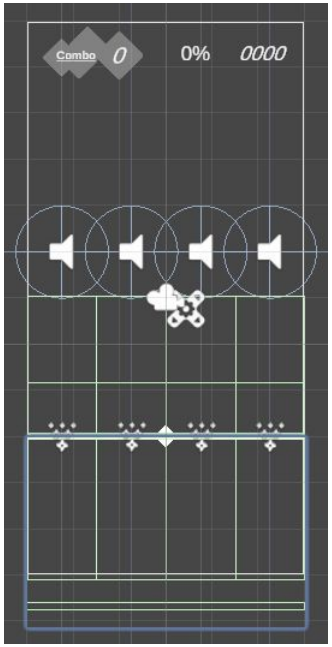
or

The default mapping

So here all the C note will be mapped to the first track, same principle for other track.







Midi Prefab Mapping/

For the game to know which note prefab to use (tap, long press, swipe), we have to define a mapping for them.

1. Note prefab

A list of prefabs definition and size of the pool need to pre-spawn for each of the note object, however you can disable object pooling in the **TrackManager**

2. Mapping

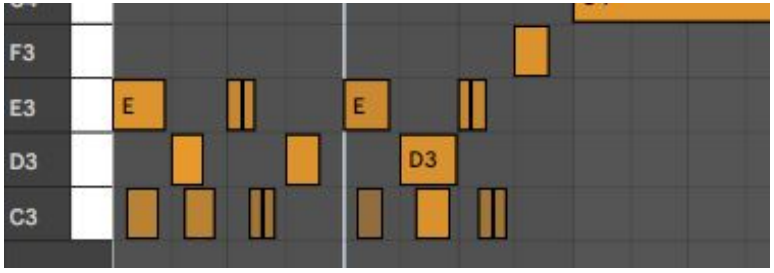
In this example, we distinguish different types of note by it's **octave**, so we can define different types of note in one single midi file.

Midi Prefab Mapping/

It's interesting that different DAW handles the root octave differently

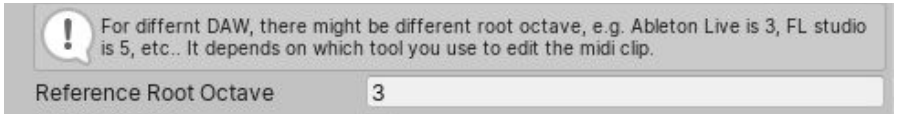
The same midi file

Ableton Live

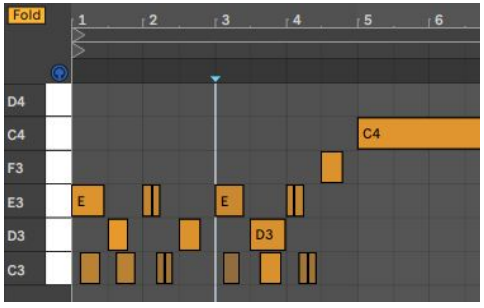


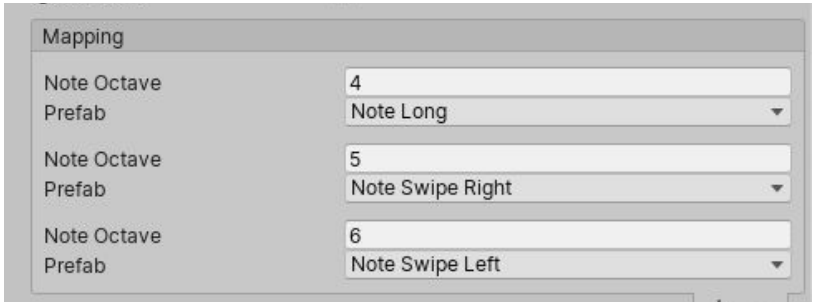
Fl Studio

So there's a options for you to set the reference root octave,



so that the mapping matches what you see in your midi editing software





Note Prefab/

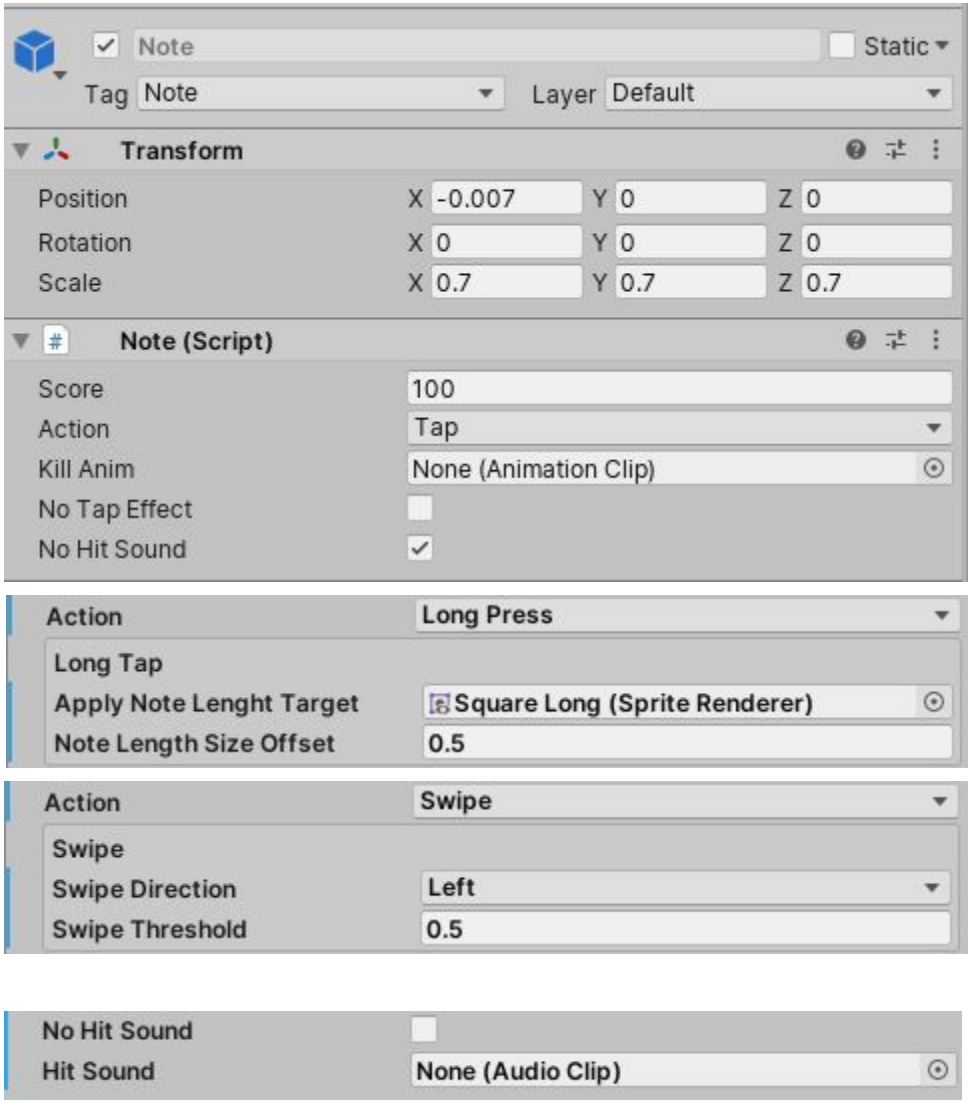
There's currently 3 note action type

1. Tap
2. Long Press
3. Swipe
   (Up, Down, Left, Right)

All support both touch and keyboard input.

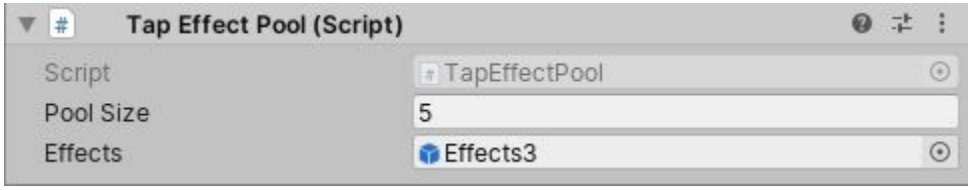
To make custom note, simply duplicate one of the prefab and swap out the sprites.

Specific audio can be assign to different note hit sound

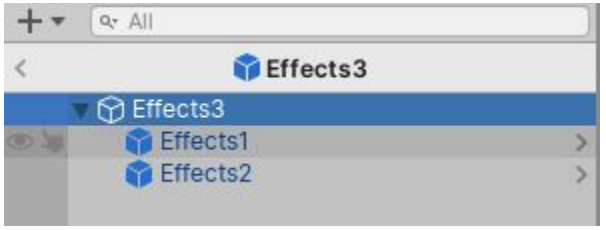


Note Effect/

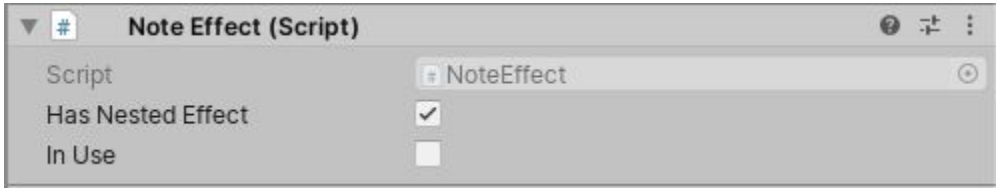
The **No Tap Effect** bool in **Note** actually disable the tap effect emitted through the **TapEffectPool** in the “RhythmGameStarter”

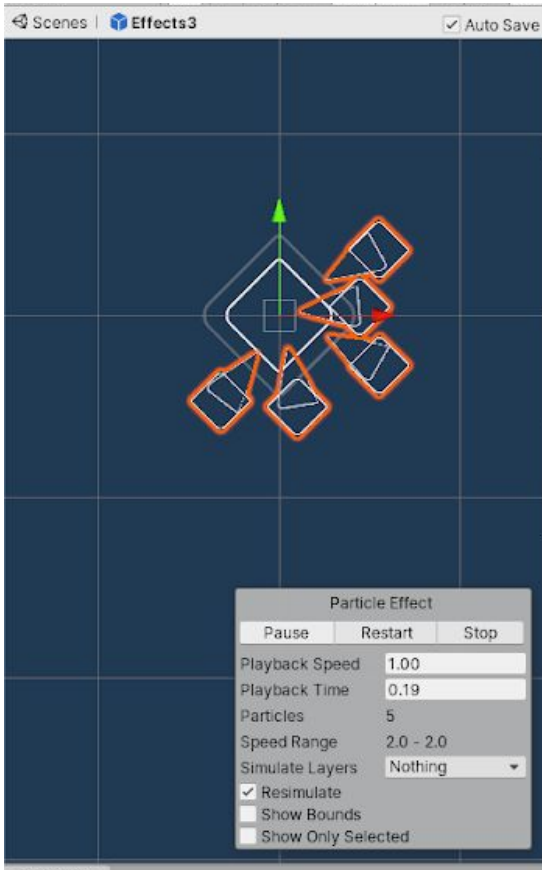


“Effects3” is actually a combination of Effects1 & Effects2



And both of them are just non-looping particle effects, that triggered at the same time by the **NoteEffect** script in “Effects3”

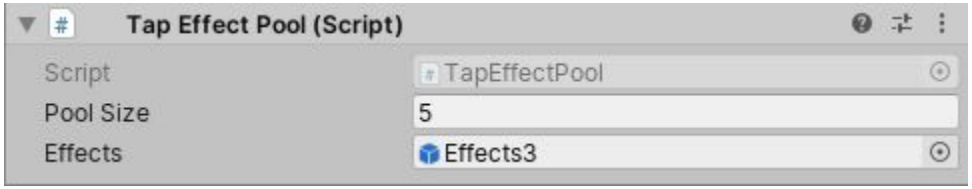




Note Effect/

To customise tap effect, duplicate the existing effects to tweak and combine them through nested effect.

Then assign the new effect to the **TapEffectPool** in the “RhythmGameStarter”

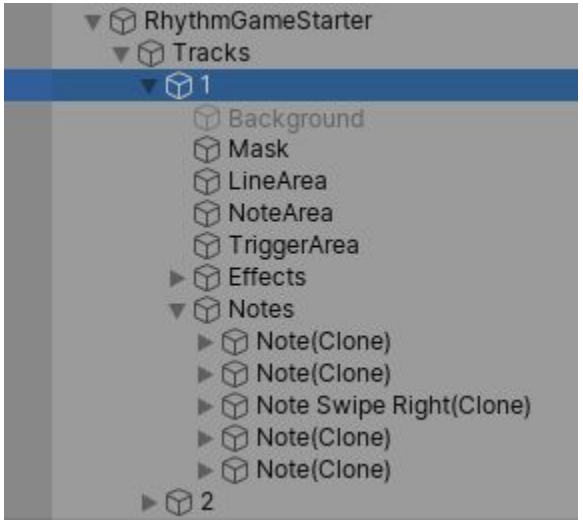


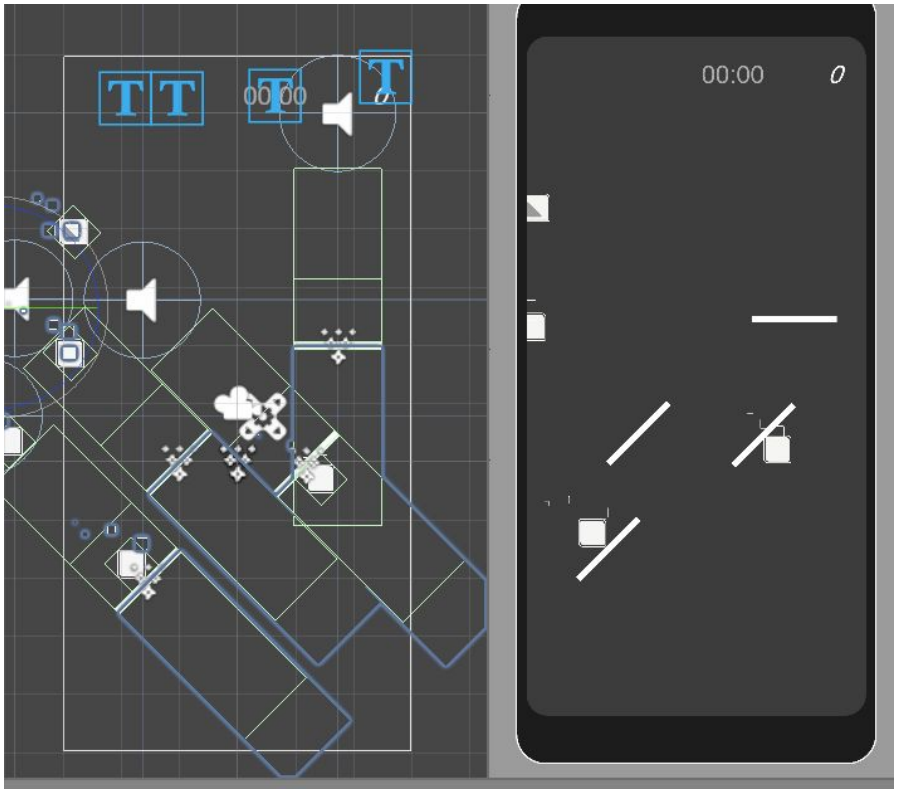
TapEffectPool will automatically emit the effect upon note trigger, and place the effect at the position of the note.

Tracks/

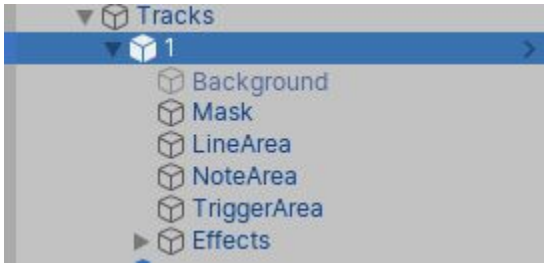
Each track is a **self-contained** object, which means can be move and rotate around freely,

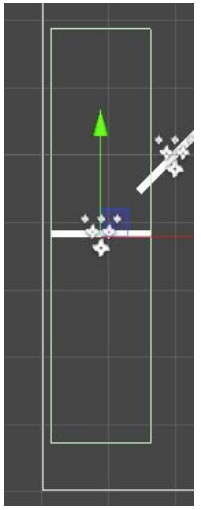
In runtime, the notes will be parented under each individual track, so it's also possible to move each track during runtime.

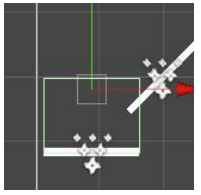




Tracks/







1. **Trigger Area**

Each track has its own TriggerArea, which has a BoxCollider that will only accept touch input if you hit inside the collider.

2. **Note Area**

The NoteArea is used to detect whether a note is in range for trigger, and it handles most of the note trigger logic.

3. **Line Area**

This marked the position for the note to hit at corresponding time, and also used to calculate how precise a note is being triggered.

4. **Mask**

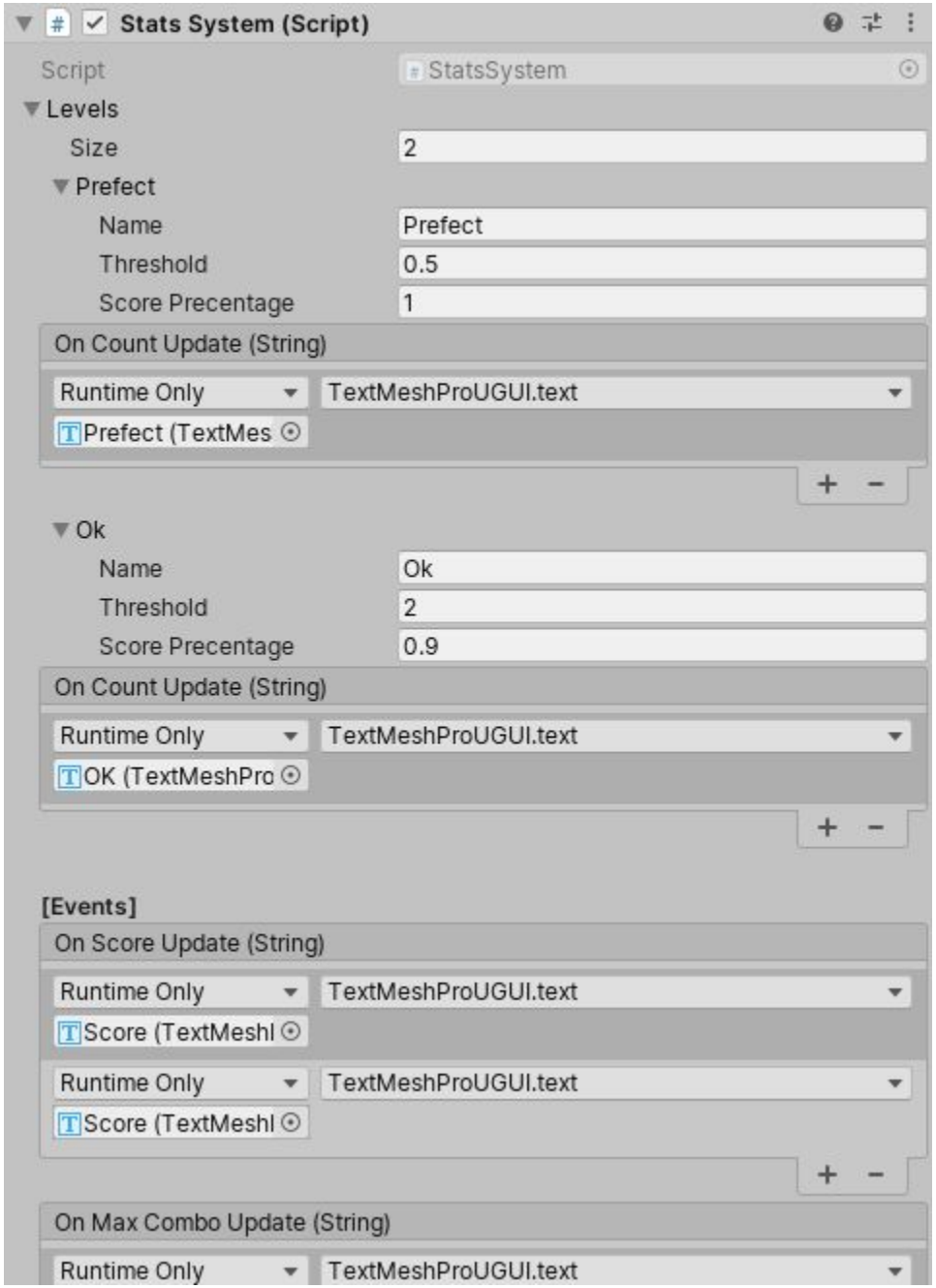
This is simple sprite mask, important for hiding the long note part

Stats System/

This handles score, combo, max combo, missed, and different level of hit.

Threshold is calculated in world space, compared to the difference between the Track's **LineArea** and the note triggered position.

It's also connected to each individual UI object via different **UnityEvent**.



Song Manager/

This handles the audio playback and the timing for synchronisation.

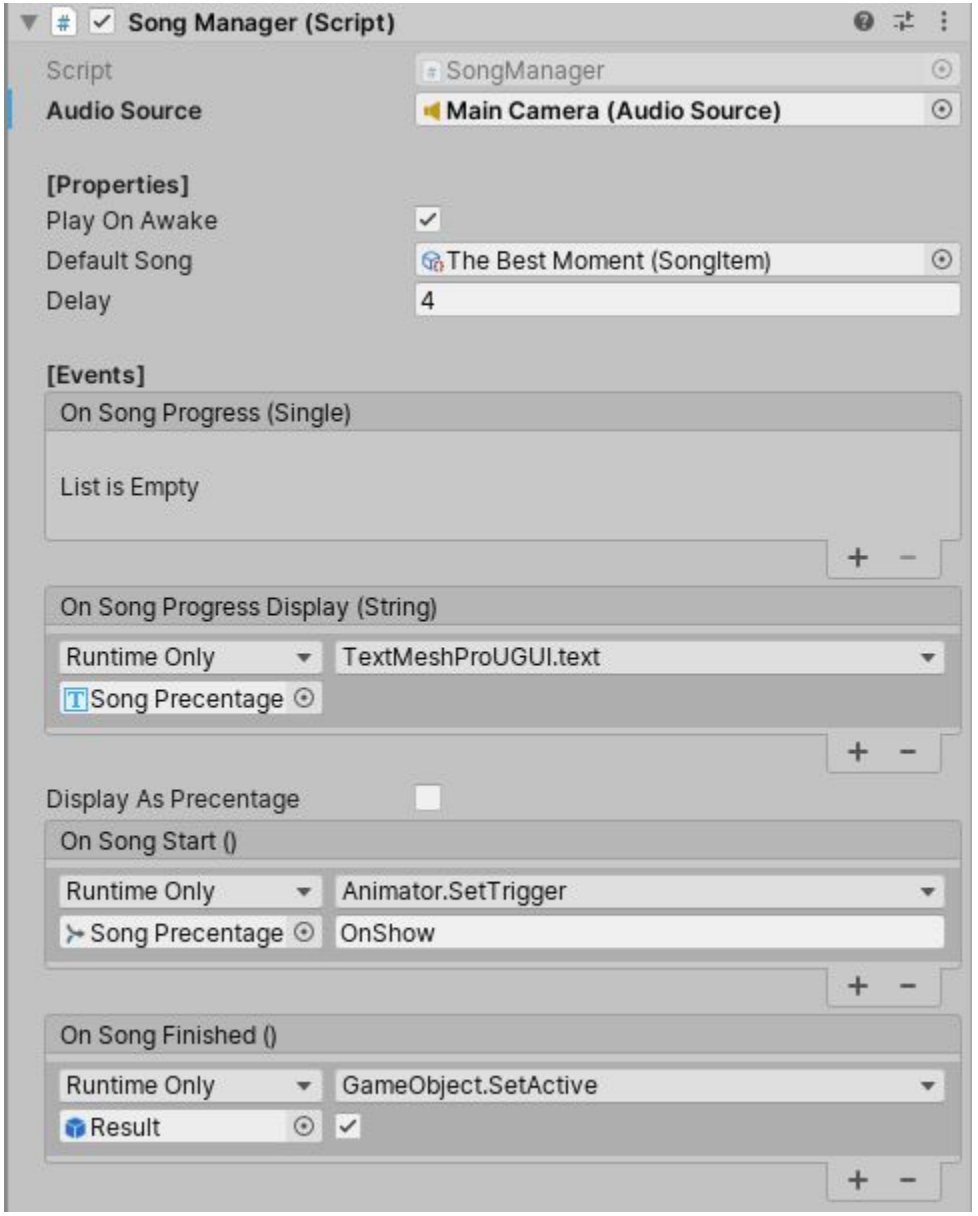
To play any song form code,

```
songManager.PlaySong(songItem);
```

Call this method anywhere and provide a **SongItem**

To pause, resume and stop

```
songManager.PauseSong();
songManager.ResumeSong();
songManager.StopSong();
```



Please contact directly for any problems, I will be there to help
Email : itechbenny@gmail.com