소개 Intro

유니티 리듬게임 Unity RhythmGame

It's a note type rhythm game.

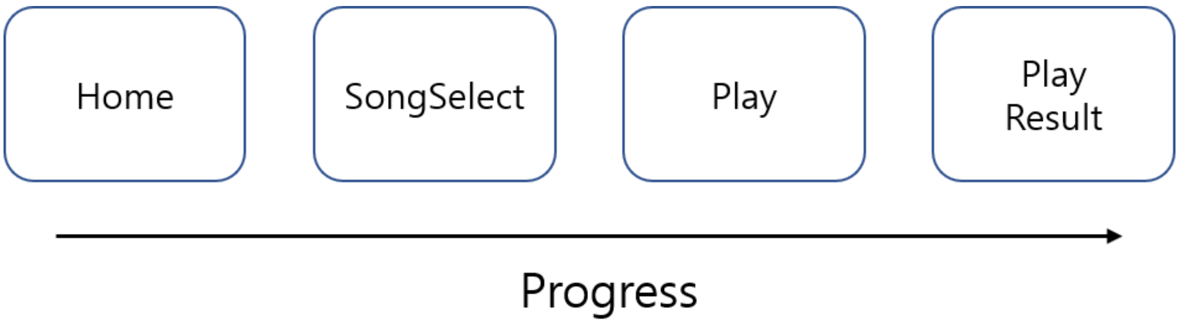
Github URL : https://github.com/LHEALP/UnityRhythmGame

※ readme

To learn the Unity rhythm game...

"The scenes except Play are not important because they are UI.
So I will briefly explain and move on.
If you don't understand, please visit the GitHub page and leave a message."

씬 진행도 Scene Progress



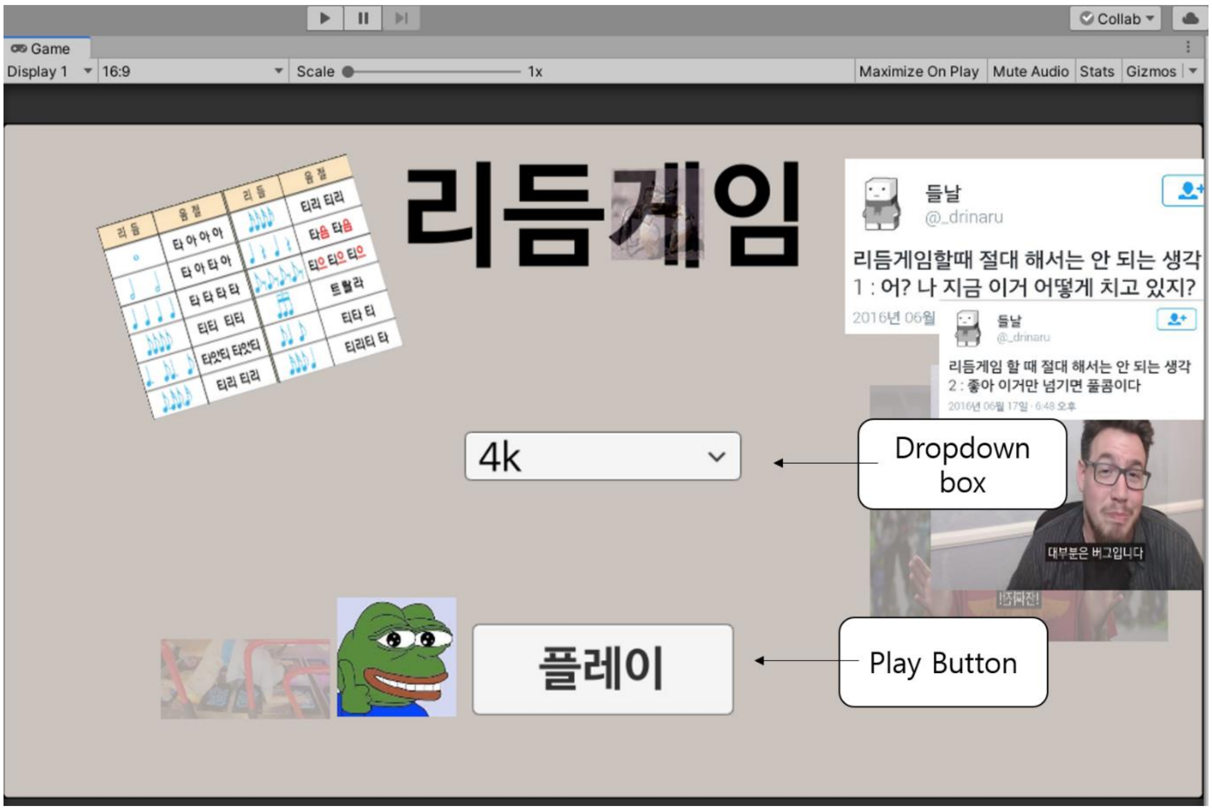
Home : Title Scenes Shown in First Execution

SongSelect : a scene of choosing music to play

Play : a scene in which a note is generated and processed (※ core)

PlayResult : a scene showing the play result (score)

Home Scene



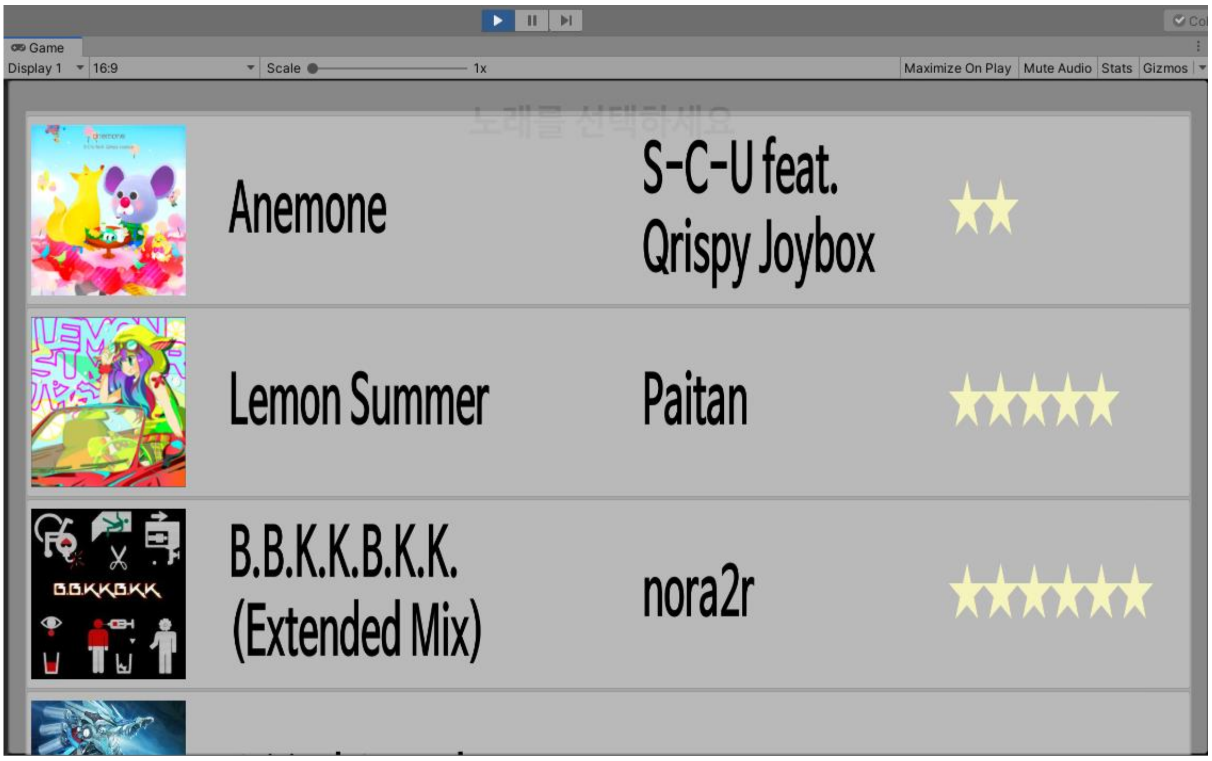
들날
@_drinaru
리듬게임할때 절대 해서는 안 되는 생각 1 : 어? 나 지금 이거 어떻게 치고 있지?
2016년 06월

들날
@_drinaru
리듬게임 할 때 절대 해서는 안 되는 생각 2 : 좋아 이거만 넘기면 풀콤이다
2016년 06월 17일 · 6:48 오후

대부분은 버그입니다

Select the number of keys to play and press the Play button to proceed.

SongSelect Scene



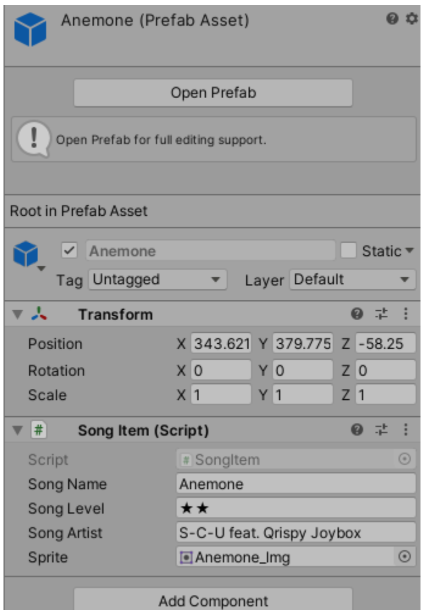
	Anemone	S-C-U feat. Qrispy Joybox	★★
	Lemon Summer	Paitan	★★★★★
	B.B.K.K.B.K.K. (Extended Mix)	nora2r	★★★★★★

choose the music you want.

check several music through mouse scroll.
preview the music by clicking the item once, and if you press it once more, move to the Play scene to play the game.

It shows the form of Listview and was created with the help of https://youtu.be/GpPWbM_6DVY.

How to add music to a list view


Anemone (Prefab Asset)

Open Prefab

Open Prefab for full editing support.

Root in Prefab Asset

Anemone Static

Tag Untagged Layer Default

Transform

Position X 343.621 Y 379.775 Z -58.25

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Song Item (Script)

Script SongItem

Song Name Anemone

Song Level ★★

Song Artist S-C-U feat. Qrispy Joybox

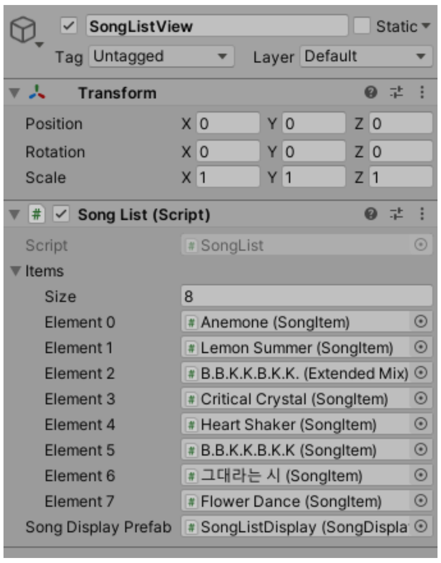
Sprite Anemone_Img

Add Component

Create an empty game object and add a SongItem script to write the music data and prefab it.

<u>At this time, since the **SongName** item is used when reading the **file**, it must be **written accurately**.</u>

Ex) Anemone -> Anemone.mp3, Anemone_data.txt


SongListView Static

Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Song List (Script)

Script SongList

Items

Size 8

Element 0 Anemone (SongItem)

Element 1 Lemon Summer (SongItem)

Element 2 B.B.K.K.B.K.K. (Extended Mix)

Element 3 Critical Crystal (SongItem)

Element 4 Heart Shaker (SongItem)

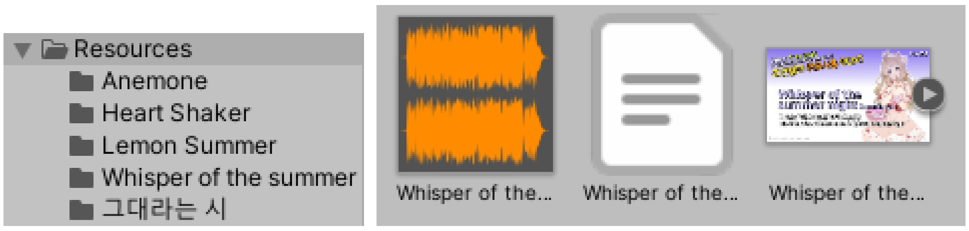
Element 5 B.B.K.K.B.K.K (SongItem)

Element 6 그대라는 시 (SongItem)

Element 7 Flower Dance (SongItem)

Song Display Prefab SongListDisplay (SongDispla

Add Prefab to the SongList script on the SongListView of Object.


Resources

Anemone

Heart Shaker

Lemon Summer

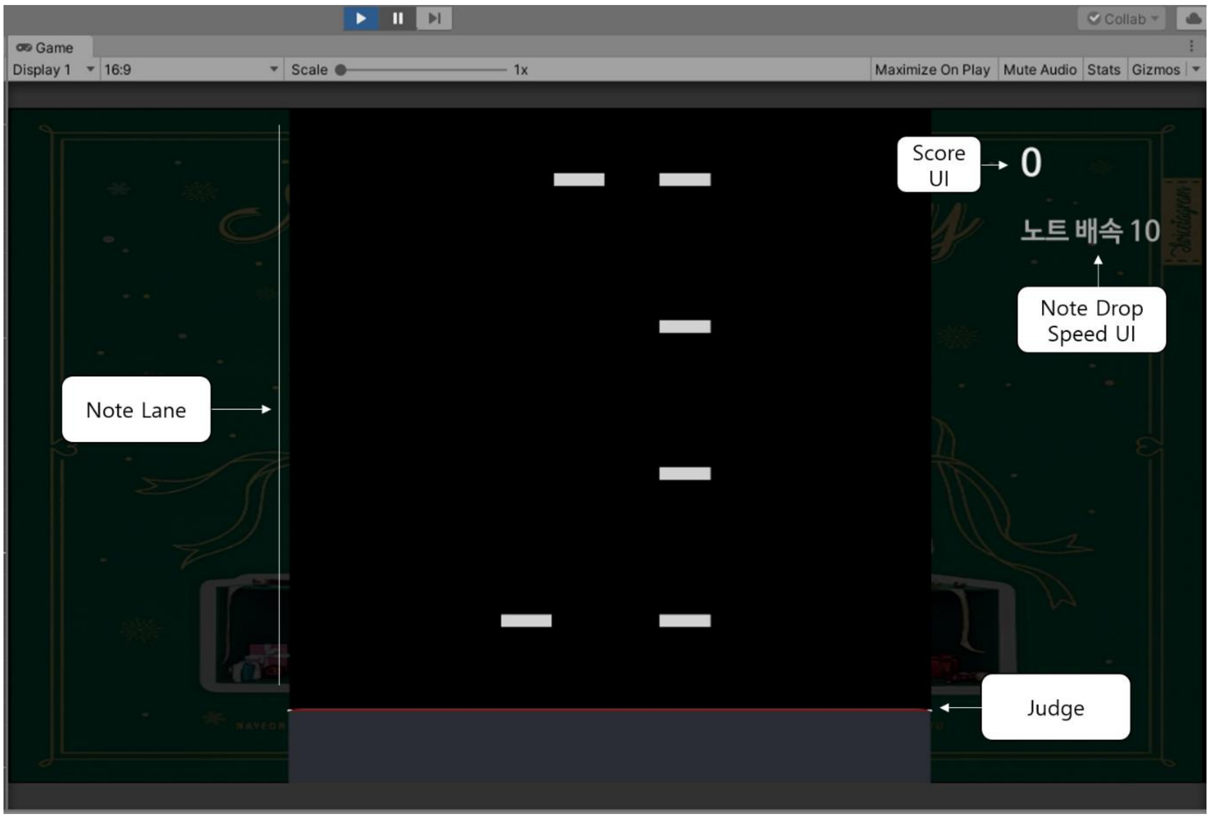
Whisper of the summer

그대라는 시

Finally, the same folder as the contents written in SongName in the Resources folder is created.
SongName.* / Sheet (Note) file is SongName_data.txt / image file is SongName_Img.*
The *mark of the extension refers to the extension supported by Unity (mp3, wav, jpg, png...)

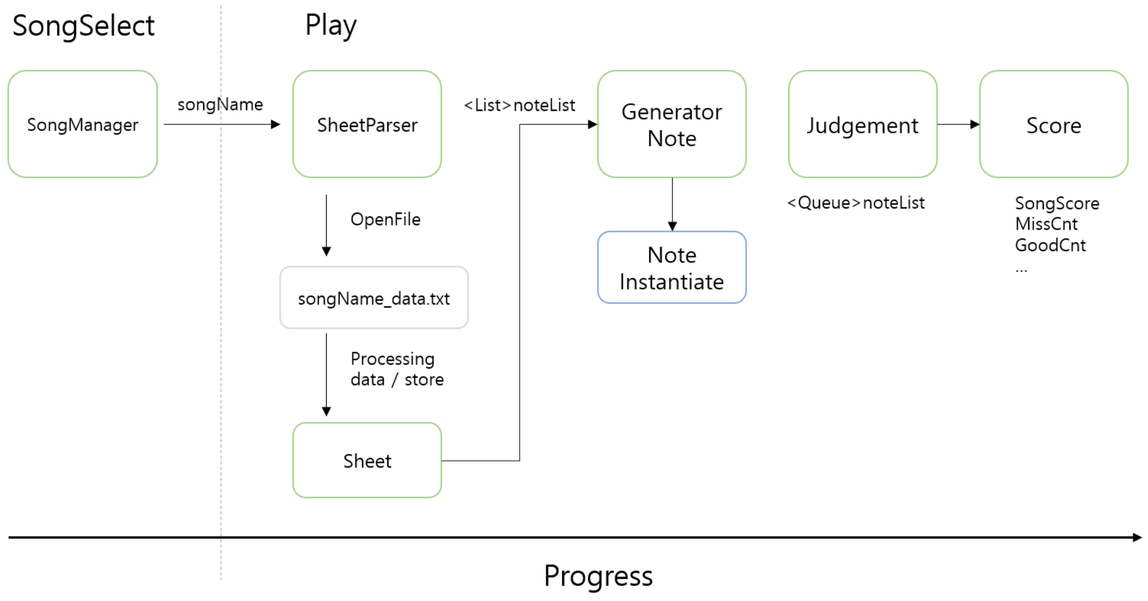
I recommend automating this system if you have the opportunity.

Play Scene



The screen that appears when the song is selected.

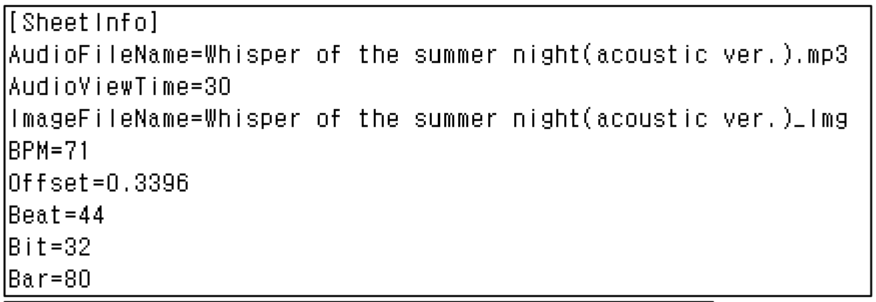
Below is a rough progression structure.



※ green: script / gray: text file / blue: prefab

I will explain it because I made it by referring to the data format of OSU! before parsing the note.

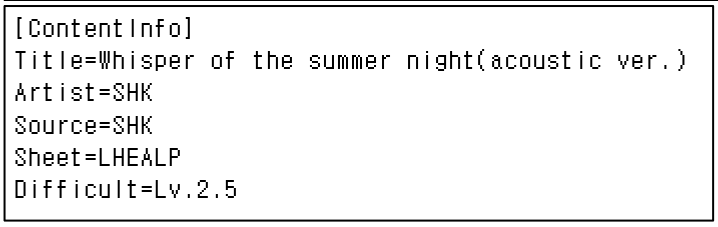
ex) Whisper of the summer night(acoustic ver.)_data.txt



```
[SheetInfo]
AudioFileName=Whisper of the summer night(acoustic ver.).mp3
AudioViewTime=30
ImageFileName=Whisper of the summer night(acoustic ver.)_Img
BPM=71
Offset=0.3396
Beat=44
Bit=32
Bar=80
```

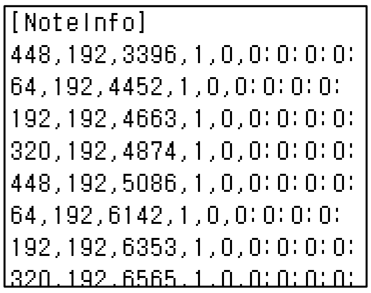
[SheetInfo]

It has the overall content of music.



```
[ContentInfo]
Title=Whisper of the summer night(acoustic ver.)
Artist=SHK
Source=SHK
Sheet=LHEALP
Difficult=Lv.2.5
```

[ContentInfo] It has the content to be displayed in the game.



```
[NoteInfo]
448,192,3396,1,0,0:0:0:0:
64,192,4452,1,0,0:0:0:0:
192,192,4663,1,0,0:0:0:0:
320,192,4874,1,0,0:0:0:0:
448,192,5086,1,0,0:0:0:0:
64,192,6142,1,0,0:0:0:0:
192,192,6353,1,0,0:0:0:0:
320,192,6565,1,0,0:0:0:0:
```

[NoteInfo] I have the location of the note.

※ How to read

x, y, time, notetype, ...

The data to be used here are x value and time. Other values are parsed, but are not used or are not parsed.

Time is expressed in milliseconds, and 3396 is 3.396 seconds.
This value is not only used to judge notes, but also determines where notes are first created and dropped. The way to record these time values can be understood without difficulty if you have ever created the concept of a BPM or a metronome.

(Based on 4/4) 60BPM is considered to play one beat per second. If four notes drop every second, you can record 1000, 2000, 3000, 4000. (1 second = 1000 milliseconds)

In the case of Whisper of the summer night (acoustic ver.), It is 71 BPM, so the unit of one night is 0.845 seconds. Converting it to milliseconds gives 845.

※ Since the current editor has not implemented, the note can be used by using the OSU! Mania editor to import and use only the necessary data.

노트파싱 NoteParsing

When SheetParser.cs receives the name of the selected music from SongManager.cs in the SongSelect scene, Resources.Load the file and save it in the TextAsset variable. Then, while reading the text file line by line, the necessary data is separated and stored in a variable in Sheet.cs.
There is nothing specially different from the principle of general parsing.

노트 Note

Note.cs (NotePrefab) only serves to drop down or change the speed of the note.
This script is not involved in the judgment of the notes at all. Just show it on the screen and the judgment is handled separately in the background.

노트생성 NoteGenerator

In GeneratorNote.cs, one data is brought from the noteList of Sheet.cs. (foreach)
※ noteList stores the time value of the note.

As I said earlier, the time value is also used as the location where the note will be generated.
Let's look at some of the codes.

GeneratorNote.cs ... `IEnumerator GenNote()`

```
...
foreach(float noteTime in sheet.noteList1)
            Instantiate(notePrefab, new Vector3(2.3f, noteStartPosY + sync.offset +  notePosY * (noteTime * noteCorrectRate)), Quaternion.identity);
...
```

Variables in Vector3's Y value are calibrated to be generated at the desired location.
Here, the reason for multiplying noteCorrectRate in noteTime is as follows.

noteStartPosY + sync.offset, notePosY * noteTime = too high Y Pos!!
330,1200,5000, ...

noteTime * noteCorrectTime = moderate Y Pos!!
330 * 0.001 = 0.33

싱크 Sync

Sync.cs is responsible for the sink of the note; the variables to note are as follows.
It is about frequency, offset, scrollSpeed; the rest is not for PlayTik() corrutins, or a large part of the

current project.

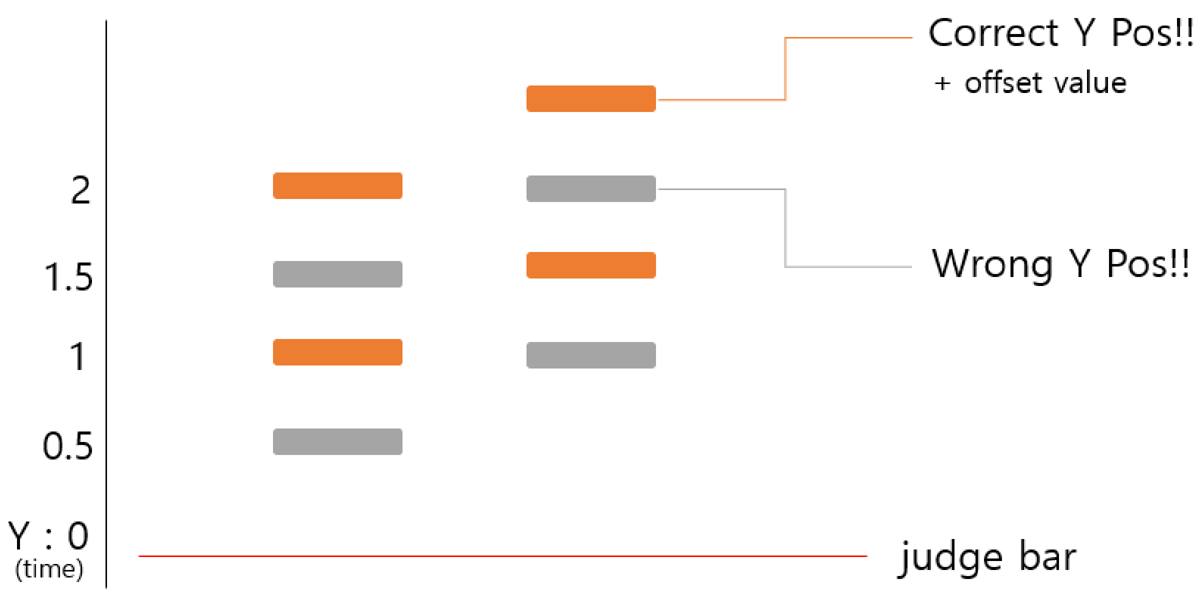
frequency : Get the PCM value of the currently selected music. This is important, so I'll explain it in the note judging part.

offset : Suppose you have multiple files of the same music. These music files exist because the point where the first bit of music starts is set differently in the process of the service provider receiving and processing the sound source. For example, someone does not want to play as soon as they run the music file, so let's say that they inserted a second of silent pile at the front of the audio. The time values of notes recorded in the text file are not considered (dummy), so we need an offset value.

```
[NoteInfo]
64,192,500,1,0,0:0:0:0:
320,192,1000,1,0,0:0:0:0:
320,192,1500,1,0,0:0:0:0:
64,192,2000,1,0,0:0:0:0:
```

Let's say you created a note by parsing the note data above.

If the offset of the music file is 0, no special action is required, but in the case of a file with a 0.5 second dummy inserted, adding a value to the Y value when creating a note to match the sync.



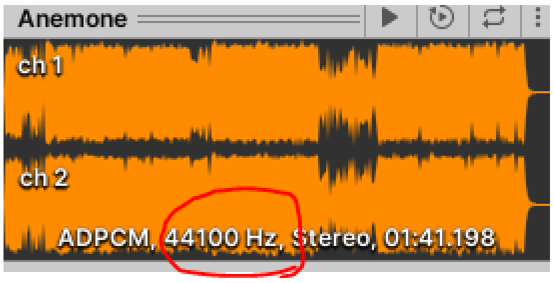
scrollSpeed : Specifies the interval of notes; this allows the player to press the key to change the scrolling speed to the desired speed.

키 입력 Input Key

InputKey.cs is responsible for keyboard inputs; this game receives four notes, so you can confirm in the code that you are receiving D, F, J, and K. Call the note-judgment method in the judgment class to determine the note each time you press the key; and you can see that you are using the F5 and F6 keys to change the scrolling speed. When you adjust the speed by pressing the key, you will find a game object with a Note tag, change the Y position, and change the speed at which the note comes down.

노트판정 Note Judgement

There is a topic to understand first before dealing with note judgment.
The frequency value mentioned in the sync part.



ADPCM, 44100 Hz, Stereo, 01:41.198

The picture refers to the circled value, which means that 44100 pieces of sound are played in a second. (The frequency value of the sound source can be obtained through the AudioClip.frequency variable.)

And we can get more detailed values to know how much music has progressed through the AudioSource.timeSamples variable, so let's check the example below to understand this.

If the 44100Hz music file has been played for two seconds, the timeSamples variable returns 88200Hz, which multiplies 44100Hz by 2.
※ Reference materials
https://docs.unity3d.com/kr/2019.3/ScriptReference/AudioSource-timeSamples.html

I processed the judgment using the PCM sample values of the sound source.
First, you need to specify the scope for accepting the judgment. This is also based on sample values.

Judgement.cs

```
... // 44100 = 1sec
float greatRate = 3025f; // 3025 ≒ 0.06sec
float goodRate = 7050f; // 7050 ≒ 0.15sec
float missRate = 16100f; // 16100 ≒ 0.36sec
...
```

These values are adjusted as much as you want.

Declared in Update () because the judgment requires the timeSample value of the currently playing music.

```
…
currentTime = songManager.music.timeSamples;
…
```

Assume that the time value of the note to be judged when analyzing the codes below is 1000 (1 second).

Miss Judgement (when Note went down the judging bar)

```
…
if (judgeLane1.Count > 0) // Do not run if there are no more data in the queue
{
    currentNoteTime1 = judgeLane1.Peek(); // Data is not pulled out and it looks at.
    currentNoteTime1 = currentNoteTime1 * 0.001f * songManager.music.clip.frequency;
    // How to convert the current note time value to PCM sample value
    // When the time value of the note is 1000 (1 second), the value of 44100 comes out when
the above formula is executed.

    if (currentNoteTime1 + missRate <= currentTime)
    // 44100 + 16100 <= currentTime(the location of the current music playback)
    // = 1sec + 0.36sec
    {
          judgeLane1.Dequeue();
          score.ProcessScore(0);
    }
}
…
```

나머지 판정들

```
…
// If the current note time is greater or less than the judgment range calculated from the
current time, the part is judged.

if (currentNoteTime1 > currentTime - missRate && currentNoteTime1 < currentTime + missRate)
// Note checking range: If you don't specify a range, you can dequeue all the notes in an
instant. The range is ± 0.36 seconds from the current music playback position.
{
     if (currentNoteTime1 > currentTime - goodRate && currentNoteTime1 < currentTime +
goodRate)
     {
        // Great Judgement ± 0.15초
        if (currentNoteTime1 > currentTime - greatRate && currentNoteTime1 < currentTime +
greatRate)
```

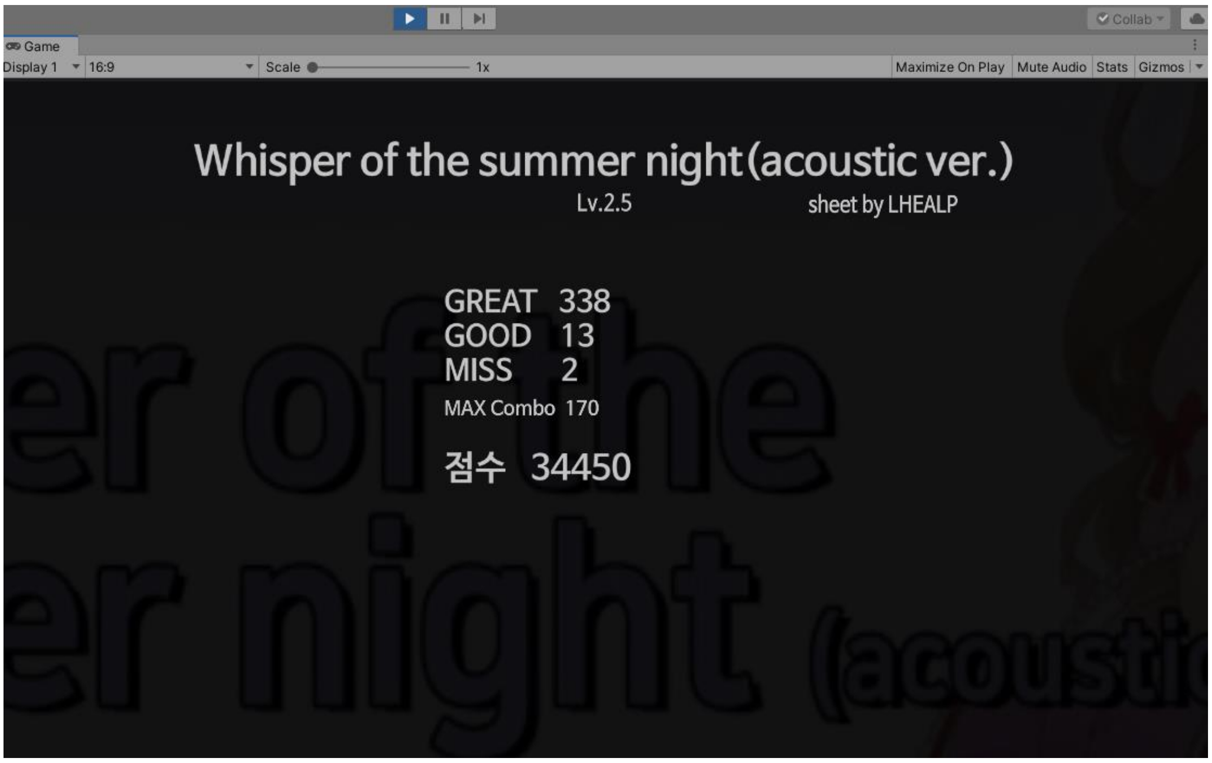
```
        {
              judgeLane1.Dequeue();
              score.ProcessScore(2);
        }
        // Good Judgement ± 0.06초
        else
            {
                  judgeLane1.Dequeue();
                  score.ProcessScore(1);
            }
        }
        // When you enter it too quickly, misprocess (prevents you from processing your
notes by just pressing the keyboard )
        else
        {
              judgeLane1.Dequeue();
              score.ProcessScore(0);
         }
    }
…
```

The reason why the time value is ± when judging is not only to hit the note quickly but also to make the same judge to hit late. This is information that you can find if you enjoy rhythm games, so you can skip the detailed description.

점수 Score

Perhaps if you have understood the contents and codes so far, you do not have to explain it.

PlayResult Scene



Whisper of the summer night(acoustic ver.)
Lv.2.5 sheet by LHEALP

```
GREAT   338
GOOD    13
MISS    2
MAX Combo  170
점수  34450
```

At the end of play, you switch to the result scene and you can see the score you have acquired.