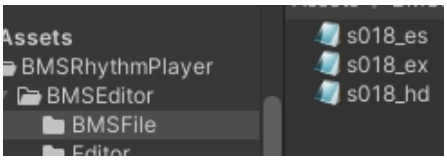
Summary:

Bmsrthrmplayer is an extension that allows you to play BMS files written on ibmsc on unity or save BMS file information to backup data.

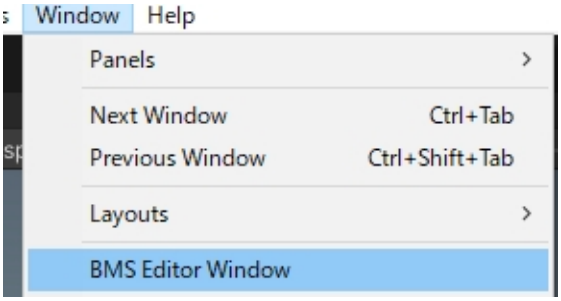
Use the editor extension to convert BMS files to asset data.

Place BMS files in bmsfile folder.



Place a sound source file in the notes folder. (Notes/018/018bgm.ogg)

Open the extension window from Window - > BMS Editor Window



BMS File List：BMS file name is displayed.

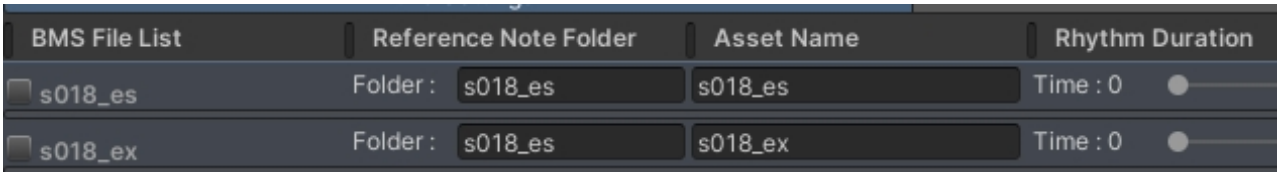
Reference Note Folder：Sound source file reference folder.。

Asset Name：Specifies the name used to save as asset data.

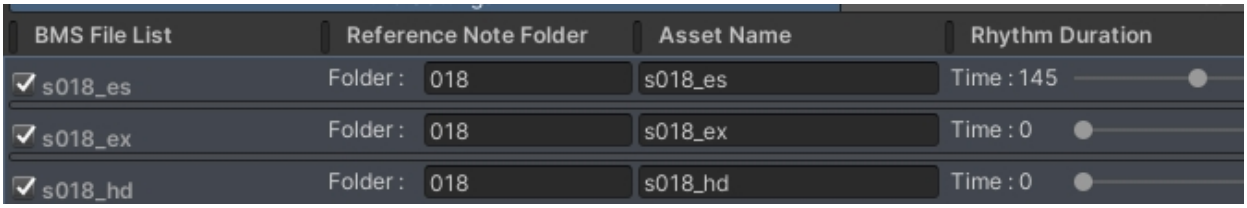
Rhythm Duration：Specifies the length of time. If 0, it automatically specifies the length of time optimized for the last kevotan.

(Please adjust manually if time length does not match.)

Reference：Before change



Reference：After the change



Track settings to select track panels.

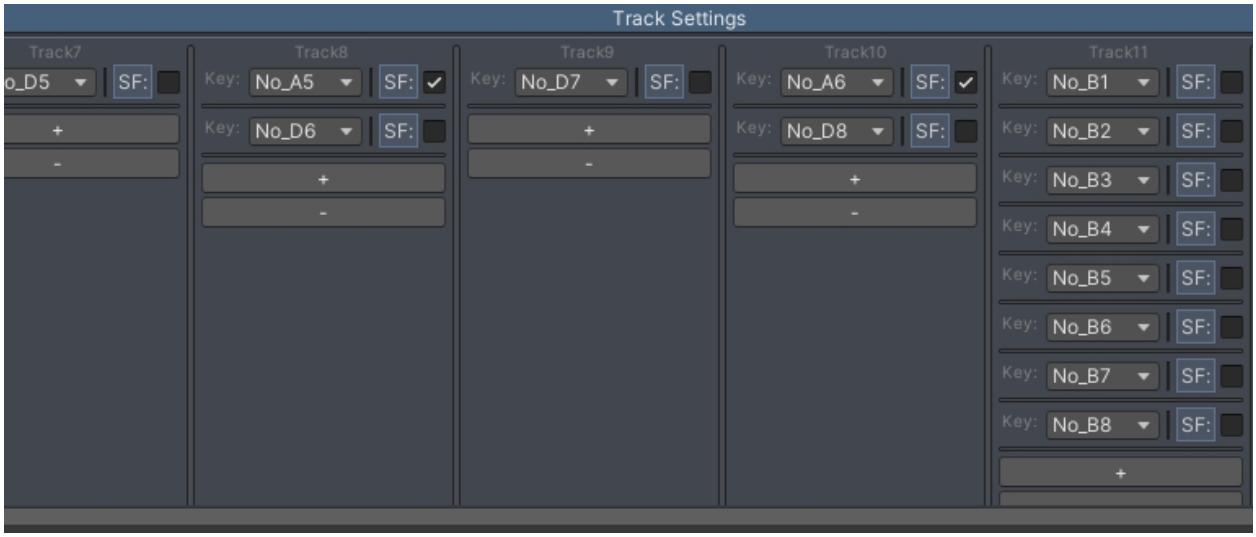
Add and remove tracks using [+ Add Track] and [- Delete Track].

Key: select the key number you want to move on the fixed track.

Key numbers are numbers from A1 to B15 of ibmsc.(B1 through B15 are BGM tracks.)

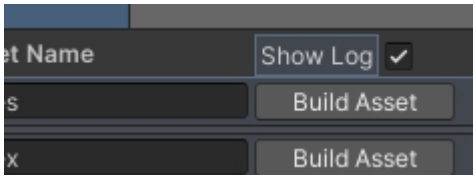
SF（SimultaneouslyFalling）：

We will check for keys that drop simultaneously only among the selected key numbers.

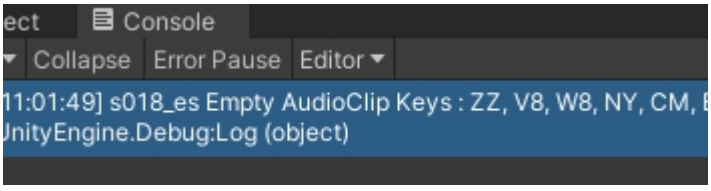


Show Log：After the asset build is complete, output the keys for any missing audio files.

if an audio file is not supported, please ignore it.



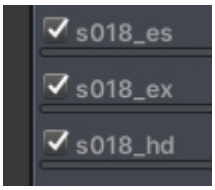
Sample diagram for Log display：



Save Settings：After completing the build and track settings, if you want to retain the current setttings, press the [Save Settings] button to save them. This will create BMSEditorPlayer/Resources/BMSEditorReference.json.

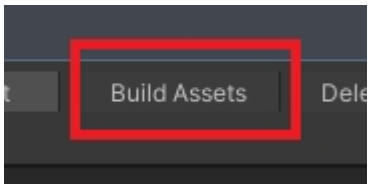
Build Asset Individually：Press the [Build Asset] button.

Build Multiple Assets Simultaneously：Check the BMS files you want to build.



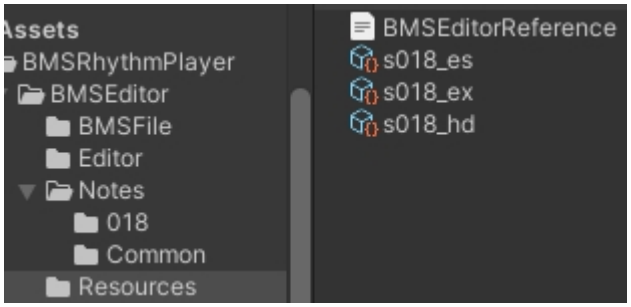
↓

Press the [Build Assets] button.



↓

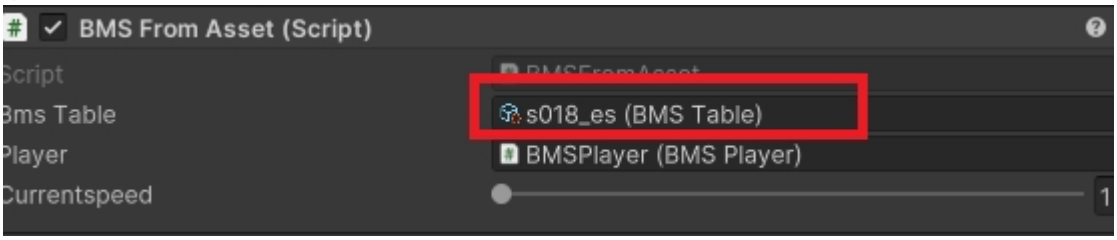
The asset data will be saved under 「BMSRhythmPlayer/BMSEditor/Resources」.



To reference And Play the built Assets

Use the Demo BMSFromAsset.

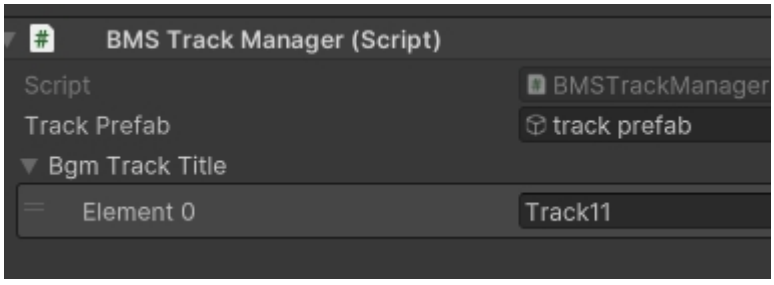
Specify the Asset in the Bms Table.



↓

Bgm Track Title：Specify the BGM track title.

Once done, you can play and verify it



Referencing And Operating Asset Data with Class BMSTable

Reference:

Total Number of Keys（Long keys are Counted twice）

public int GetNoteCount

Reference Speed

public float GetBasicNoteSpeed

Rhythm Duration

public float GetRhythmDuration

Returns a list of BPM Track information for use with tempo change Functions.

public List GetBpmTrackInfo()

Returns a Package list of All Track information including BGM Tracks
(Excluding BPM Tracks)

public List GetTrackPackage()

This Method can be used for tempo change functions but can also be used without tempo changes.

index：Index for Referencing BPM track list starting from 0; returns the next index.

time：Time per frame.

totalTime：Time elapsed since the start.

basicVariableSpeed：Current Reference speed.

public BPMVariableInfo GetBPMVariableSpeedInfo(int index, float time, float totalTime, float basicVariableSpeed)

Creating Tracks：Refer to the Demo's BMSTrackManager Class.

BGM tracks are processed separately as They do not have key button creation, display, or positioning.

Creating Key Buttons：Refer to the Demo's BMSTrack Class

Keys falling simultaneously are created then SimultaneouslyFall is True.

```
if(_track.noteInfo.keyType == KEY_TYPE.Short_Note)
{
    if (_track.noteInfo.simultaneouslyFall)
        noteOb = GameObject.Instantiate(shortRedPrefab);
```

ShortNote Class：Regular key buttons.

Specify generation coordinates:

```
var notePoint = (info.noteInfo.point[0] - totalDistance) * gearSpeed;

transform.localPosition = new Vector3(transform.localPosition.x, notePoint, 0f);
```

LongNote Class：Long key Buttons.

Specify generation coordinates：

```
var notePoint = (info.noteInfo.point[0] - totalDistance) * gearSpeed;

transform.localPosition = new Vector3(transform.localPosition.x, notePoint, 0f);
```

Specify length：

```
var width = info.noteInfo.noteLong * gearSpeed;

widthRect.sizeDelta = new Vector2(width, widthRect.rect.height);
```

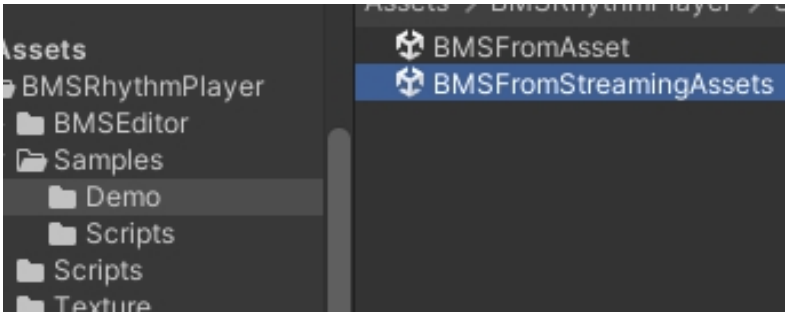
Key Movement Distance：Refer to the Demo's BMSPlayer Class

Use the GetBPMVariableSpeedInfo() Method of BMSTable to get movement distance and reference speed.

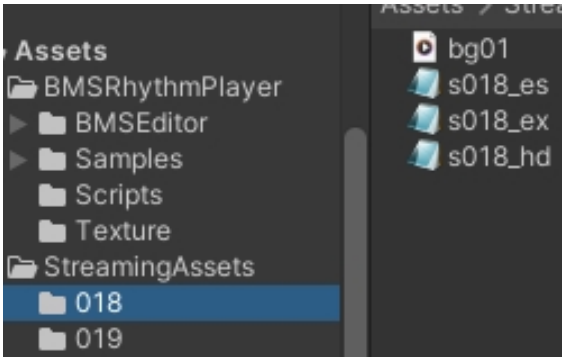
```
var f_time = Time.deltaTime;
totaltime += f_time;

var variableInfo = table.GetBPMVariableSpeedInfo(variableIndex, f_time, totaltime, basicVariableSpeed);
variableIndex = variableInfo.index;
basicVariableSpeed = variableInfo.basicVariableSpeed;
totalBasicDistance += variableInfo.distance;
var moveDistance = variableInfo.distance * gearSpeed;

trackmanager.Run(generateTimer, f_time, totaltime, gearSpeed, moveDistance, totalBasicDistance);
```

Referencing and Playing BMS Files from StreamingAssets

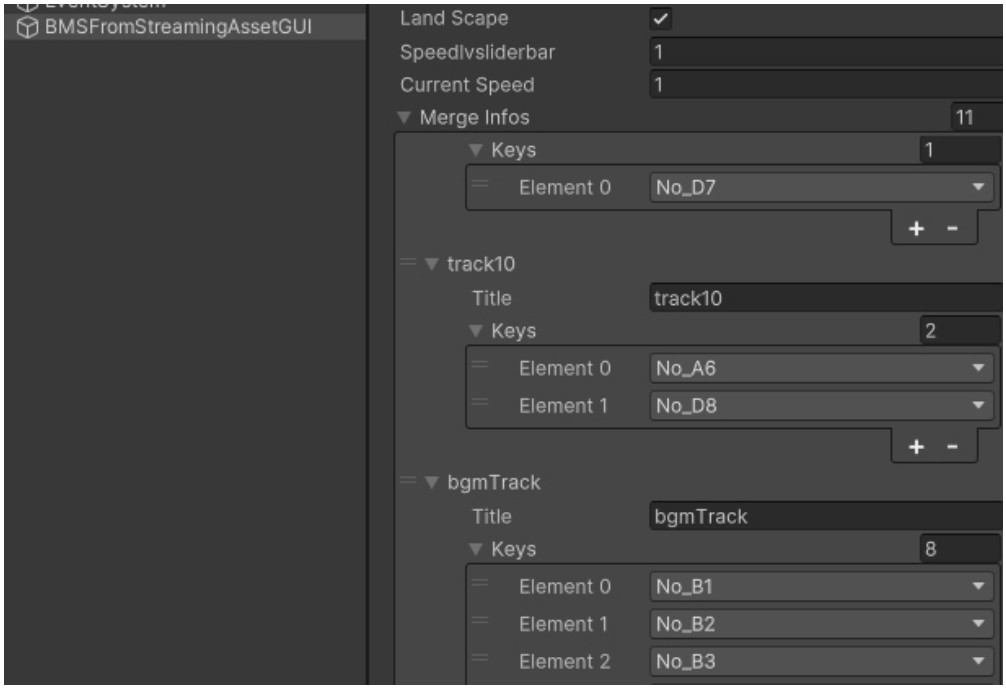
Demo: Use the Demo BMSFromStreamingAssets



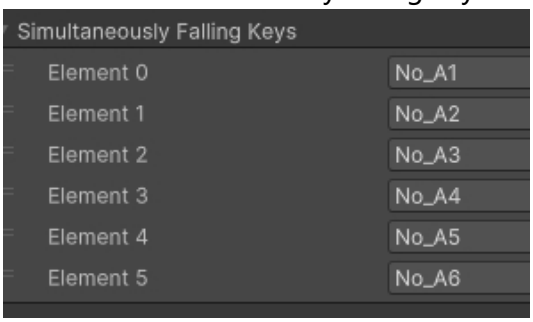
Place the BMS files under the StreamingAssets folder, as shown in the diagram.



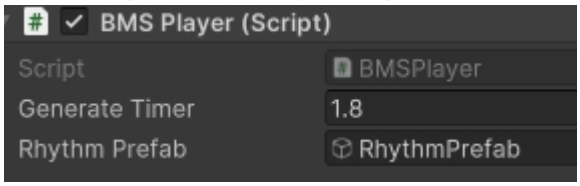
Edit MergeInfos from BMSFromStreamingAssetGUI to create tracks.
Enter the track title and the key numbers you want to move on the track.
Key Numbers: Use numbers from A1 to B15 in iBMSC (B1 to B15 are BGM tracks).



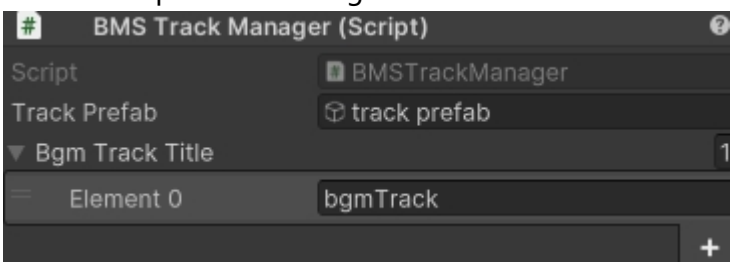
Simultaneously Falling Keys：Handles keys that fall simultaneously.
(For example, use this feature to make simultaneously falling keys the same color.)



GenerateTimer：：Specify the timing for when keys are generated and fall.



Bgm Track Title：Specify the title of the BGM track.
Ensure it matches the title specified in MergeInfos.



Select the BMS file saved in StreamingAssets, click the Play button to start playback.

Error：This message appears if the audio files for these keys are not specified or not found. (Ignore if audio files are not used.)

Empty AudioClip Keys：ZZ, UP...