

How to use geekdrums.MusicEngine

<https://github.com/geekdrums/MusicEngine>

Music Programmer

@geekdrums

MusicEngineとは

Unityで音楽同期演出やインタラクティブミュージックを実現できます。

<https://github.com/geekdrums/MusicEngine>

から無料でDLできます。

MITライセンスで公開してあります。

商用含め利用は自由ですが、不具合などがあってもサポートや修正をお約束はできませんのでご承知おきください。

利用に際してご連絡も不要ですが、お役に立てたようでしたら[Twitter:@geekdrums](https://twitter.com/geekdrums)までご一報ただけると嬉しいです。

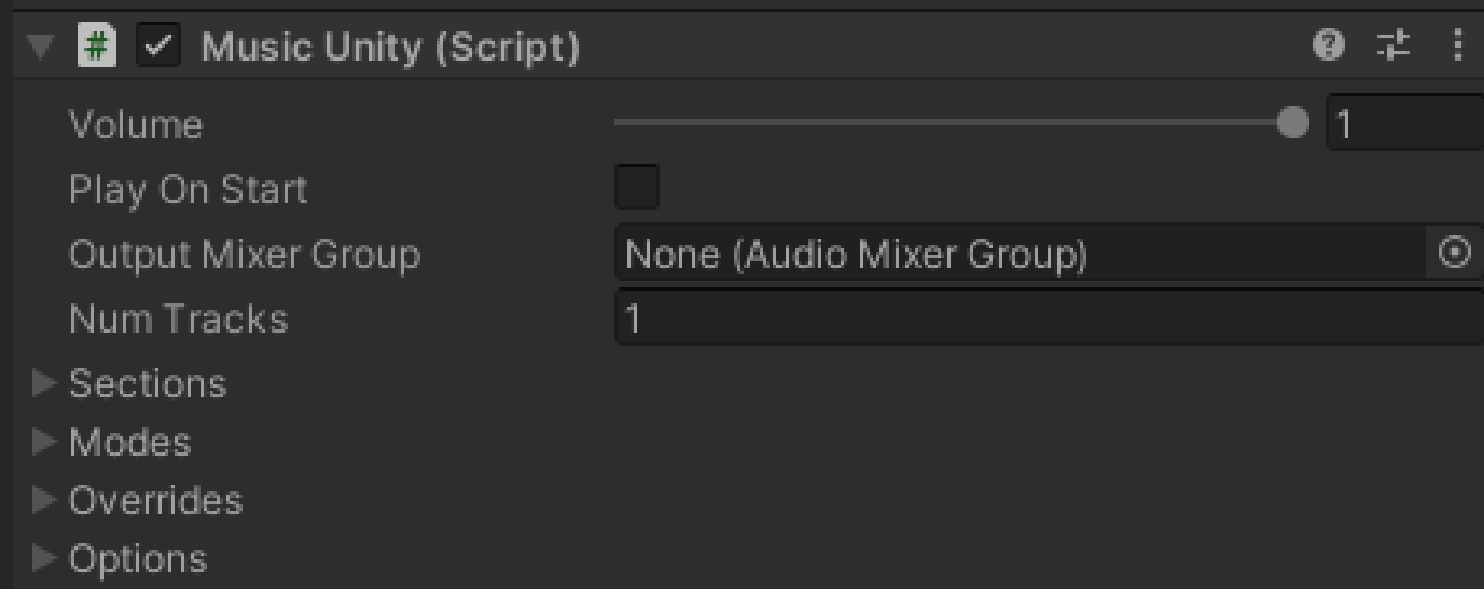
1,利用準備

1, Music Unityを追加

Play On Start : 最初から再生する場合はチェック

Output Mixer Group : 指定は任意です

Num Tracks : とりあえず1でOKです



2, Sectionを定義

最低一つ以上のSectionを定義します。

Track 0にセクションの波形（Audio Clip）を指定してください。

Transition Type :

- Loop : そのセクションをループ
- Transition : 自動的に別セクションに遷移
- End : そのセクションで終わり

から選べます。

ループする場合、Loop Start/Endのタイミングをそれぞれ指定してください（Loop StartはEntry Pointからの相対値で、Entry Pointより後にしか設定できません）。

The screenshot shows the 'MainLoop' configuration panel. It includes fields for Name, Transition Type, Track 0, Meters, Markers, and Transition. The 'Transition' section has sub-fields for Entry Point Timing, Exit Point Timing, Loop Start Timing, and Loop End Timing, each with a numeric input and a slider.

Property	Value
Name	MainLoop
Transition Type	Loop
Track 0	horizontal_main_loop
Meters (4/4, 148.00)	
Markers	1
Transition	
Entry Point Timing	1 0 0
Exit Point Timing	40 0 0
Loop Start Timing	0 0 0
Loop End Timing	40 0 0

3, Meterを定義

Metersに一つ以上の拍子情報を設定してください。

Start Bar : Start Bar=0のものが一つあればOK。
途中から拍子やテンポが切り替わる場合、これを指定してMetersを増やしてください。

Tempo : テンポを指定してください。

Meter : 拍子を登録してください。分母の方は/4,/8,/16から選べます。



The screenshot shows a software interface for defining musical meters and tempo. It features a dark theme with light gray text and input fields. The interface is organized into a list-like structure with expandable sections.

- ▼ Meters (4/4, 144.00)**: This section is expanded, showing a **Size** input field with the value **1**.
- ▼ Meter 0~ (4/4, 144.00)**: This section is also expanded, showing three input fields:
 - Start Bar**: Input field with the value **0**.
 - Tempo**: Input field with the value **144**.
 - Meter**: Input field with the value **4**, followed by a dropdown menu currently displaying **/4**.

(Optional) 複雑な拍子を扱う場合

デフォルトでは、1拍=16分音符4個分としています。例えばMeterで6/8を指定しても、1拍=2/8と解釈されます。

これを変更したい場合、Optionsの**Show Meters in UnitPerBeat**にチェックを入れると、Meterの表示が4/4など簡略的なものから詳細なものに切り替わります。

Unit Per Beat : 何unitを1拍と捉えるか

Unit Per Bar : 何unitを1小節と捉えるか

これを指定することで、任意の細かさ（3連付、6連付、5連付など）で拍を割ったり、任意の長さの小節を設定できます。

The screenshot shows a settings window with two main sections. The first section, titled '▼ Meters (4/4, 144.00)', contains a 'Size' field with the value '1'. Below it is another section titled '▼ Meter 0~ (4/4, 144.00)' which includes four fields: 'Start Bar' with value '0', 'Tempo' with value '144', 'Unit Per Beat' with value '4', and 'Unit Per Bar' with value '16'. The second section, titled '▼ Options', contains a single checkbox labeled 'Show Meters in UnitPerBeat' which is checked.

▼ Meters (4/4, 144.00)	
Size	1
▼ Meter 0~ (4/4, 144.00)	
Start Bar	0
Tempo	144
Unit Per Beat	4
Unit Per Bar	16
▼ Options	
Show Meters in UnitPerBeat	<input checked="" type="checkbox"/>

5, Music.Play(name)

Musicはstaticクラスで、**Music Unity**(or Music ADX2 / Music Wwise) インスタンスのうち現在再生中のもの一つだけを常にCurrentとして保持しています。

Music.Play(name)関数を利用することで、再生する音楽を指定できます。name引数には、Hierarchy上で指定したGameObject名を指定します。

あるいは、Play On Startを指定していれば自動的にMusic.Currentに反映されます。

音楽同期演出やインタラクティブミュージックは、基本的にこの**Music**というstaticクラスのAPIで実現できます。

```
void ResetGame()
{
    if( gameState_ != GameState.InGame )
    {
        if( Music.IsPlaying )
        {
            Music.Stop();
        }
        Music.Play("Music");
    }
}
```


2,音樂同期演出

Music.IsJustChanged○○

Update関数内でIsJustChanged○○系のAPIを使えば、音楽的なタイミングに合わせて演出をトリガーできます。

- IsJustChangedBar() : 小節ごと
- IsJustChangedBeat() : 拍ごと
- IsJustChangedAt(Timing) : 指定タイミング

```
if( (gameState_ == GameState.Result || IsReplaying) && Music.IsJustChangedBar() )  
{  
    ...  
    NewGameButtonAnim.Play();  
}
```

NEW GAME

NAME: **GEEKDRUMS**

Music.Just / Near

音楽時間を表すTimingクラスは

(Bar, Beat, Unit)

の情報から構成されます。それぞれ、

(小節, 拍, 16分音符※)

の解像度で音楽時間を表しています。

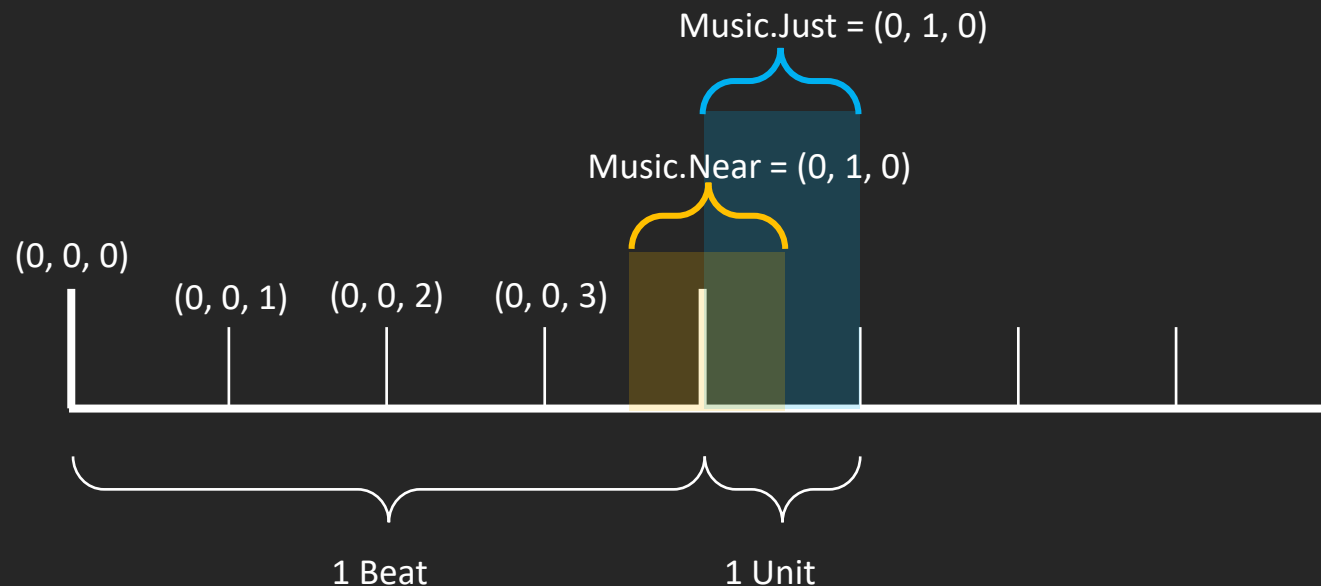
※UnitPerBeatの設定がデフォルトの4であればそのとおりだが、ここを変更すると違う単位にもできる

Timing型の

Music.Justは「経過した最後のタイミング」を、

Music.Nearは「最も近いタイミング」を

指し示しています。



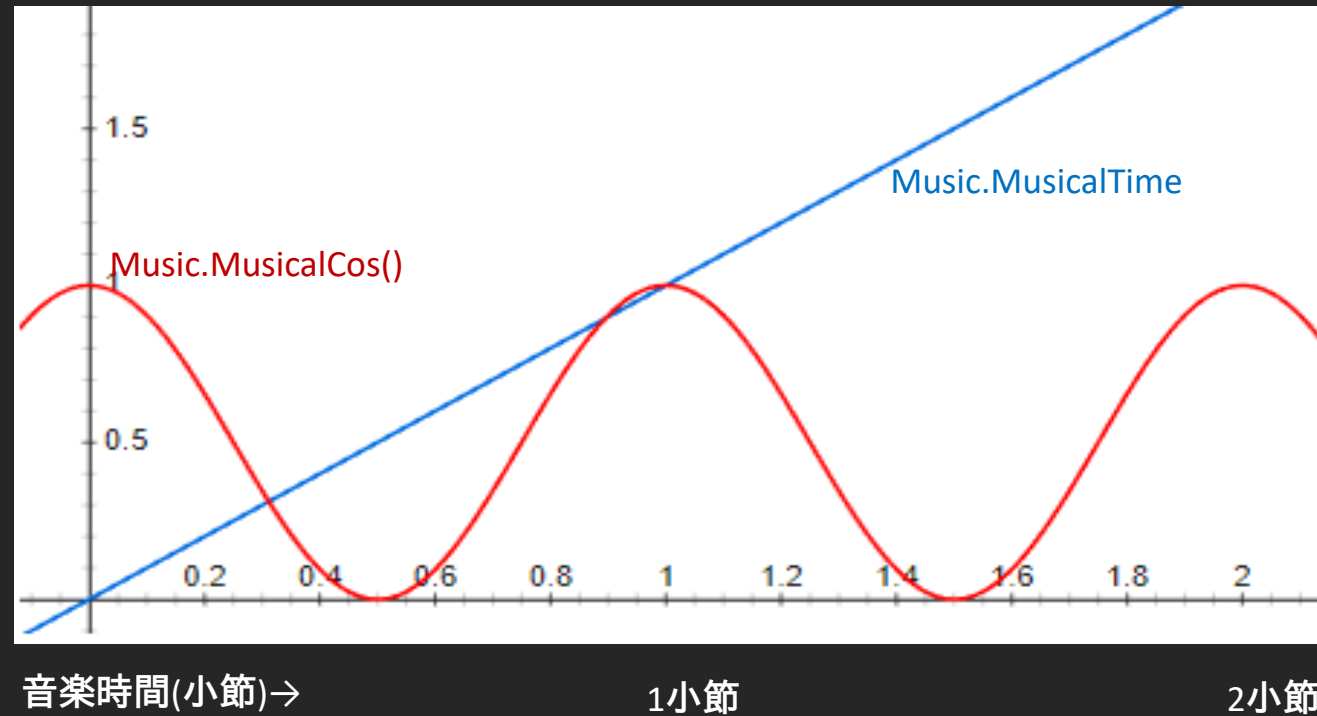
Music.MusicalTime

Music.MusicalTimeは、小節数※に合わせて連続的に変化するfloat値を取得できます。

Timing型のMusic.Justなどでは整数値しか取れないので、音楽に合わせた連続的な変化の演出にはこちらを活用してください。

※もし、途中で別のMeterが定義されてテンポや拍子に変更された場合は、それも加味したうえで小節数を正規化して返します。つまり、MusicalTimeの変化速度も音楽に合わせて動的に変化します。

このMusicalTimeをコサイン波に変換したものを個人的によく使うため、MusicalCosというAPIを用意しています。デフォルト引数では右図の赤線のような変化をします。



```
// 4小節で0から1になる変数  
float normalizedTime = (Music.MusicalTime / 4.0f);
```

3,インタラクティブミュージック

縦の遷移 / アレンジの変化

	Melody	Original
	Strings	Arrange A
	Bass	Arrange B
	Percussion	Arrange C

楽器やアレンジごとにレイヤーを分けて、音量変化で遷移

参考：

ゲーム音楽をイロイロに彩る「アレンジの変化」 | じーくどらむす | note

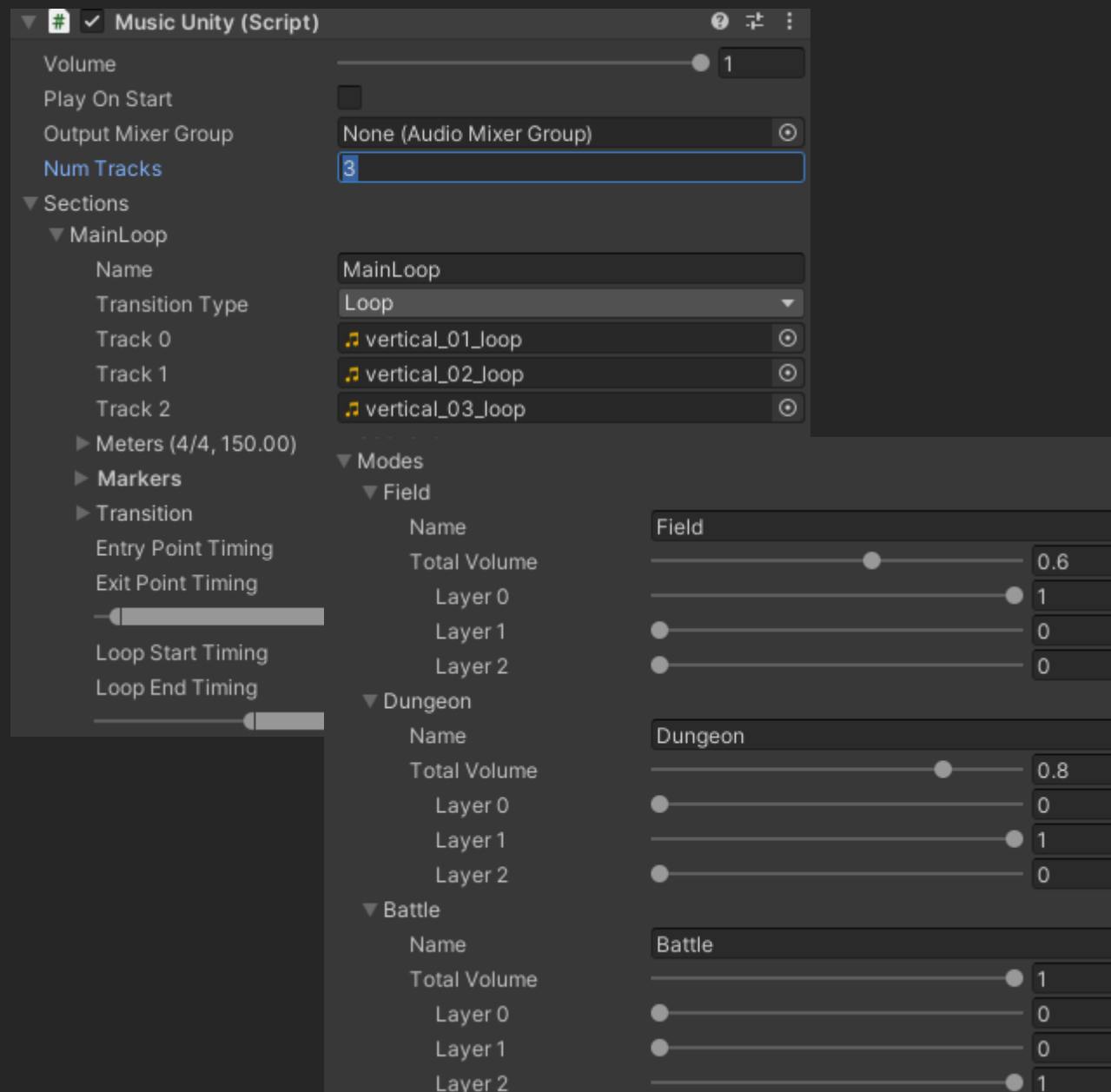
<https://note.com/geekdrums/n/need52e4a7344>

縦の遷移の準備

Num Tracksに同時再生する最大数を入力します。
各セクションのTrack 0 ～ にそれぞれ楽器ごとや
アレンジごとの波形を追加してください。

このままではすべてのレイヤーが同時再生されて
しまうため、指定したレイヤーだけを再生するた
めに、**Mode**というものを定義します。

各モードに対して、各レイヤーボリュームと全体
ボリュームをそれぞれ設定できます。このモード
を切り替えることで縦の遷移を実現します。



縦の遷移の実行と調整

Music.SetVerticalMix(name)に、モード設定でつけたモードの名前を指定することで遷移が実行されます。

このとき、クロスフェードのタイミングや秒数はModesの下に表示されるTransitionツリー内で指定できます。

SyncTypeをImmediate（即時）以外にすると、小節などに合わせてフェードを開始できます。

FadeOffsetをマイナスにすることで、アウフタクトからフェード開始して小節頭でフェード完了させることもできます。

さらに、特定のモード遷移でトランジションを別にしたい場合、Overrides以下のMode Transition Overridesに遷移元と遷移先を指定したうえでTransitionを定義することで上書きが可能です。

▼ Transition

Sync Type	Immediate
Time Unit Type	Sec
Fade Time	1

▼ Transition

Sync Type	1	Bar
Time Unit Type	Sec	
Fade Time	1	
Fade Offset	-1	

▼ Overrides

▶ Section Transition Overrides

▼ Mode Transition Overrides

Size	1
▼ from Battle to Field	
From	Battle
To	Field
▼ Transition	
Sync Type	Immediate
Time Unit Type	Sec
Fade Time	1

横の遷移 / 展開の変化



展開ごとにセクションを分けて、それぞれをループ

次の小節など、音楽的に繋がる遷移タイミングを待ってから遷移

参考：

ゲーム音楽が「ループ」をやめる時 | じーくどらむす | note

<https://note.com/geekdrums/n/n035a57e23a8a>

横の遷移を使うには

SectionsのAddボタンを押してセクションを増やします。

セクションごとにループする波形（あるいは終わり用や遷移用の波形）を登録し、それぞれに「1,利用準備」のところで説明したようにTransition TypeやLoop Start/Endを設定します。

右図では、MainLoopはループして、そこからPreEndに遷移した場合はPreEnd再生後に自動的にPreEndLoopに遷移し、最後にEndに遷移した場合は楽曲が終了するという作りになっています。

The screenshot displays the 'Sections' configuration panel with three sections: MainLoop, PreEnd, and End. Each section has a set of controls for Name, Transition Type, Track 0, Meters, Markers, and Transition timing.

Section	Name	Transition Type	Track 0	Meters (4/4, 148.00)	Markers	Transition Entry Point Timing	Transition Exit Point Timing
MainLoop	MainLoop	Loop	horizontal_main_loop		1	1 0 0	40 0 0
PreEnd	PreEnd	Transition	horizontal_pre_end		0	1 0 0	16 0 0
End	End	End	horizontal_end		0		

横の遷移の実行と調整

Music.SetHorizontalSequence(name)に、セクション設定でつけた名前を指定することで遷移が実行されます。

セクションごとにある「Transition」ツリーでは、このセクション「から」他のセクションへ遷移する際の設定を編集できます。

Sync Type：デフォルトでは波形をExit Pointになっていますが、小節ごと（Bar）、拍ごと（Beat）、その数（2小節ごととか）も指定できます。

Use Fade In/Out：遷移時にフェードインアウトを指定したい場合に利用してください。

Entry / Exit Point：遷移時にアウフタクトで重ねてほしい部分はEntry Pointより前に、遷移時にリリース部分として重ねてほしい部分はExit Pointより後になるように設定してください。

また、Overrides以下のSection Transition Overridesを追加することで、特定のセクション間の遷移設定を上書きできます。

The image displays two screenshots of a software interface for configuring transitions between music sections. Both panels are titled 'Transition' with a downward arrow icon.

Top Screenshot (Bar Sync Type):

- Sync Type:** 1 (dropdown menu)
- Use Fade Out:** ☒
- Fade Out Time:** 0.5 (input field)
- Fade Out Offset:** 0 (input field)
- Use Fade In:** ☐

Bottom Screenshot (Exit Point Sync Type):

- Sync Type:** Exit Point (dropdown menu)
- Use Fade Out:** ☒
- Fade Out Time:** 0.35 (input field)
- Fade Out Offset:** 0 (input field)
- Use Fade In:** ☐
- Entry Point Timing:** 1 (input field), 0 (input field), 0 (input field)
- Exit Point Timing:** 16 (input field), 0 (input field), 0 (input field)

Below the input fields in the bottom screenshot is a horizontal slider bar with a circular knob in the center.

geekdrums@gmail.com

Twitter: @geekdrums

おわり

ご質問やご感想は上記まで。
前述の通り、不具合などがあってもサポートや修正を
お約束はできませんのでご承知おきください。