

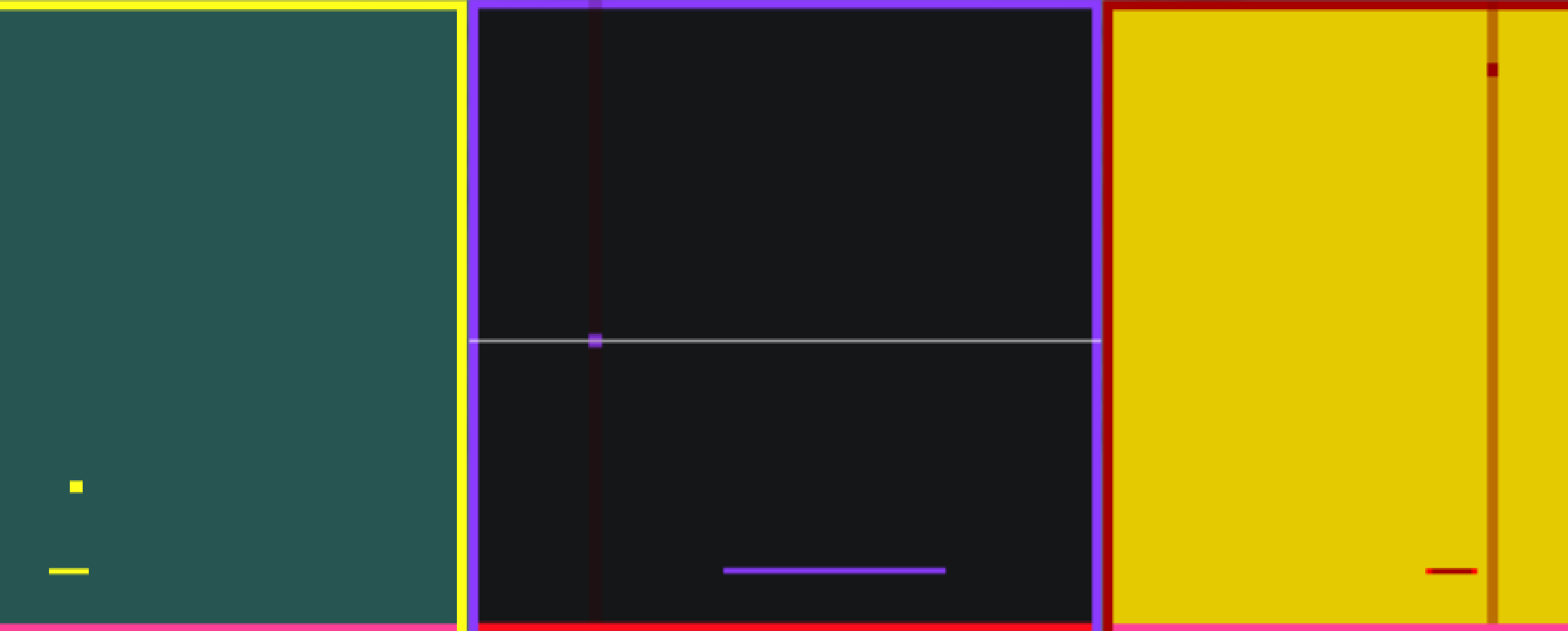
MusicEngine for Unity/ADX2LE

を使った簡単音楽同期ゲーム制作

スクウェア・エニックス サウンドプログラマー 岩本翔

MusicEngine採用事例

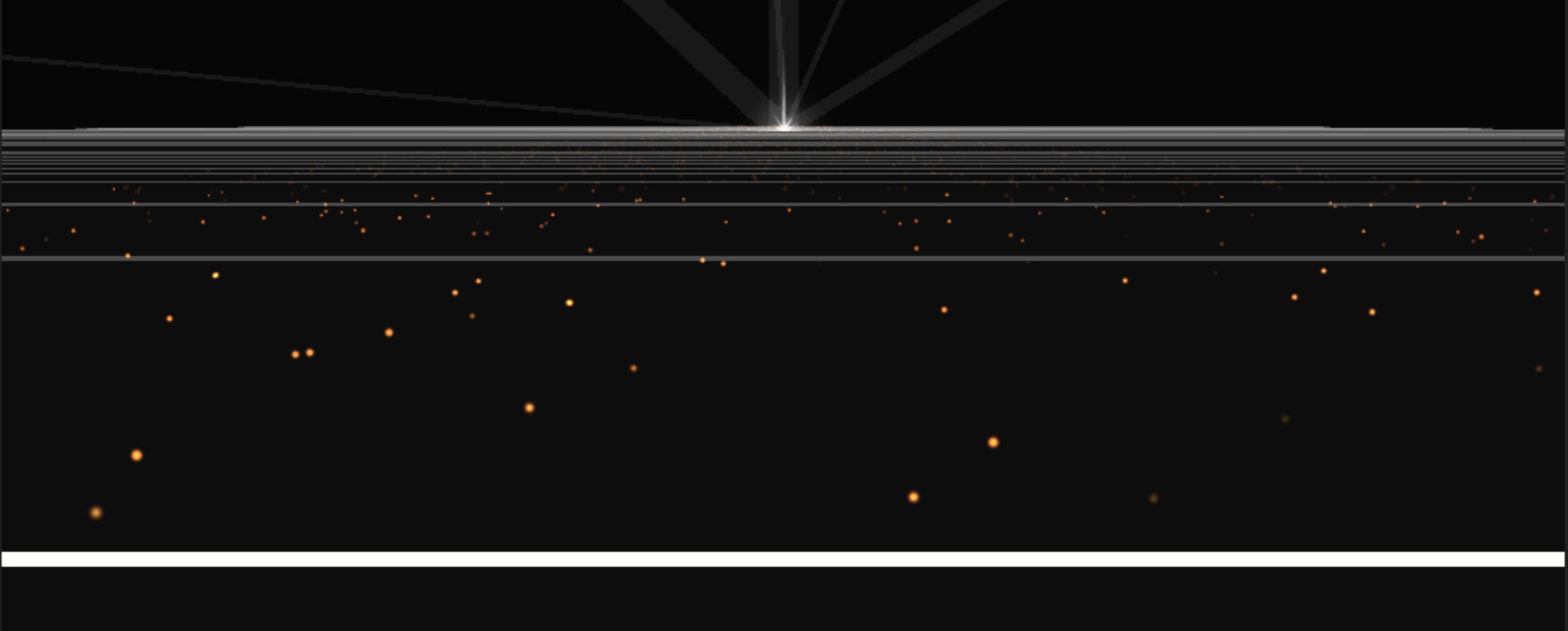
という名の宣伝



MusicPong

<http://unitygameuploader.jp/game/1233.html>

MusicEngineのサンプルとして付属

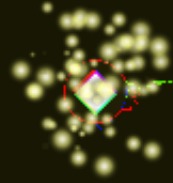


Space to go

<http://www.ludumdare.com/compo/ludum-dare-29/?action=preview&uid=25923>

LudumDare #29 個人戦オーディオ部門で1位獲得

Did you lost your memories?



What do you do now?

Who are you?

What's your name?

Where is this place

What is this game?

Enter to ask

<http://unitygameuploader.jp/game/3512.html>

GGJ2015にて

音楽X RPGの最先端。

2015
2015
DAMAGE

VOX
QUARTER

オクス^クォーター



VOXQUARTER

<http://voxquest.tumblr.com/>

Unity5+ADX2LEで鋭意開発中

「音楽時間」に アクセスできる ライブラリが必要。

“Intelligent Music System”をすべての開発者に。

MusicEngineとは

Unity用スクリプト「Music.cs」のこと。

できること

- 「今何小節目の何拍目のどこ？」が簡単に取得できる
- 「音楽に合わせてこう動かして！」が簡単にできる






できないこと

- 波形に反応してかっこよく動かして！→**GetSpectrumData**とか使ってください
- ゲームに合わせて音楽を変化させたい！→**ADX2LE**とか使ってください



早速ダウンロード

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

Update		
 geekdrums authored 11 hours ago		latest commit 6527dce184 
 Example	Update	11 hours ago
 Music.cs	Update	11 hours ago
 README.md	Update README.md	23 hours ago

Exampleの中にMusicPongがプロジェクトごと入ってます。Music.csだけでもOK。

または

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

前提

- 音楽は自分で用意(orテンポや拍子は自分で調べる)
 - 自動検出なんて、あるわけない
- 音楽はいつも1つ
 - **Music.〇〇**という感じで、コードのどこからでもstaticな音楽情報にアクセスできる
 - **クロスフェードは甘え**
- **1MusicalTime=16分音符(※)がすべての基準**
 - **Music.MusicalTime**は16分音符で1.0ふえる
 - ※6連符や8分音符など、任意の基準に(曲中でも)変更可能

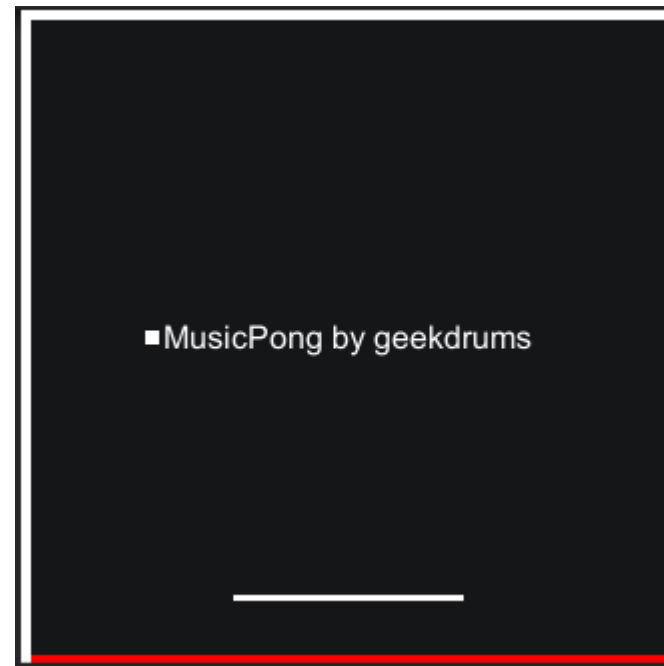
MusicPongの作り方

Music Pong | UnityGameUploader

<http://unitygameuploader.jp/game/1233.html>

Pongを作る

ぽんつと



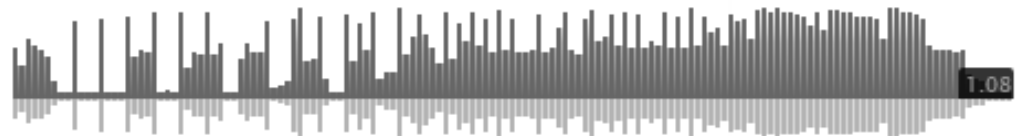
音楽を作る



geekdrums
MusicPong

8 hours

#game



さくっと(3日経過)

<https://soundcloud.com/geekdrums/musicpong>

あと5ステップくらいで完成

- Step1. Musicコンポーネントをつける
- Step2. クオンタイズで気持ちよく
- Step3. ビートに合わせてカッコよく
- Step4. 音楽に合わせてアニメーション
- Step5. 音楽に合わせてシーン遷移

Step1.Musicコンポーネントをつける

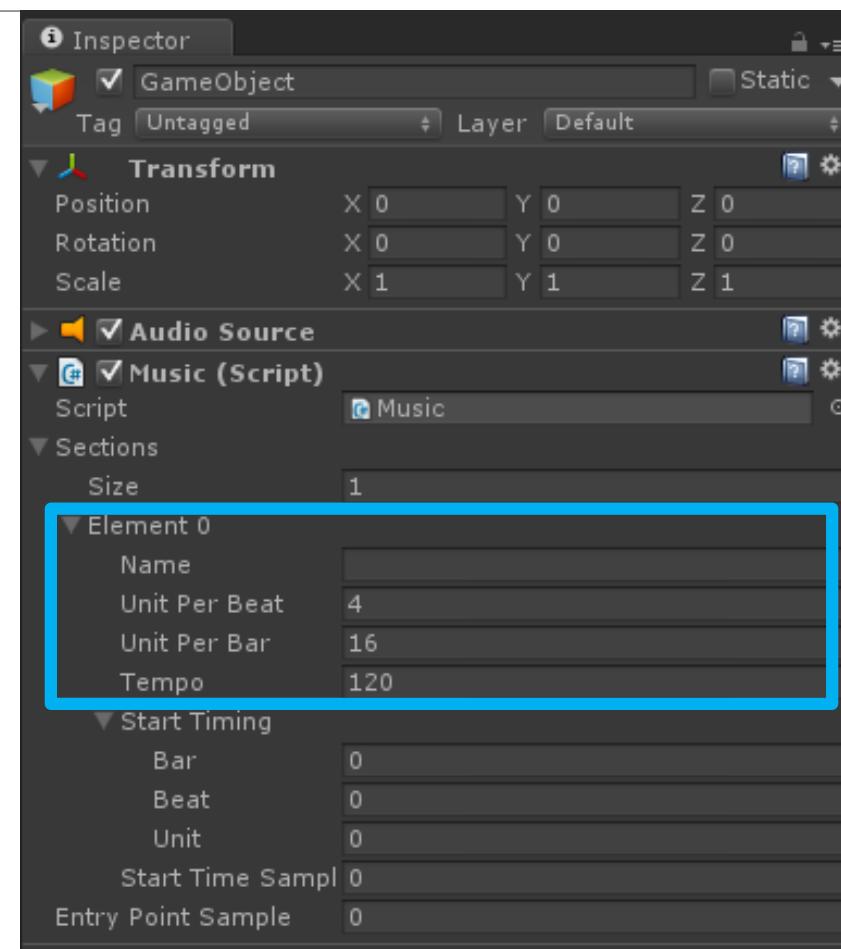
- GameObjectにMusic.csを追加。
 - 自動的にデフォルトのセクション情報
(16分音符で4拍子、テンポ120)が挿入されます。
- Music.Justでタイミング情報を取得可能。

Note:

複数の曲を使う場合は、Music.Play("name")で曲を変更。

1サンプル目から曲が始まらない場合のみEntryPointSampleを指定。

Sections内のStartTimeSamplesは自動計算されます。



Tips1: Timing & Section

➤ class Timing

- int Bar; //何小節目の
- int Beat; //何拍目の
- int Unit; //16分音符何個目

Note: (0,0,0)からスタート。4拍子で16分音符で4小節の曲の最後は (3,3,3) になる。

➤ class Music.Section

- int UnitPerBeat=4; //何unitで1拍とするか
- int UnitPerBar=16; //何unitで1小節とするか
- int Tempo=120; //テンポ(=拍数/1分)
- Timing StartTiming; //開始タイミング

Note: 例えば7/8拍子を使いたい場合は、UnitPerBeat=4ならUnitPerBar=14にすれば良い。

準備完了。

システムオールグリーン

Tips2: DebugText

3DText(TextMesh)を作ってDebugTextに→

Entry Point Sample

0

Debug Text

DebugText (Text Mesh)

現在のタイミング(0小節目の3拍目の3mt) 現在の音楽時間(1小節で16)

```
Just = 0 3 3, MusicalTime = 15.4764853880395  
section[0] = "Start" startTiming:0 0 0, Tempo:128
```

現在のセクション。0番目の"Start"セクションで、(0,0,0)から始まり、テンポは128

Note:音楽を途中で止めたりピッチ（再生速度）変えたりしても大丈夫。

Step2.クオンタイズで気持ちよく

- `Music.QuantizePlay(AudioSource source, int transpose);`
 - 自動的にMusicalTime(=16分音符)に合わせて再生される。
 - transposeは1で半音、12で1オクターブ。

例：Ball.csにて↓ 壁やパドルの反射音をクオンタイズ&音程変更

```
//side wall
velocity.x = Mathf.Abs( velocity.x ) * -Mathf.Sign( transform.position.x );
Music.QuantizePlay( GetComponent<AudioSource>() );
}
if( Field.FieldLength <= transform.position.y )
{
    //roof
    velocity.y = Mathf.Abs( velocity.y ) * -Mathf.Sign( transform.position.y );
    Music.QuantizePlay( GetComponent<AudioSource>(), 7 );
}
```

クオンタイズ+音程。
これだけで気持ちいい

コスパ高い

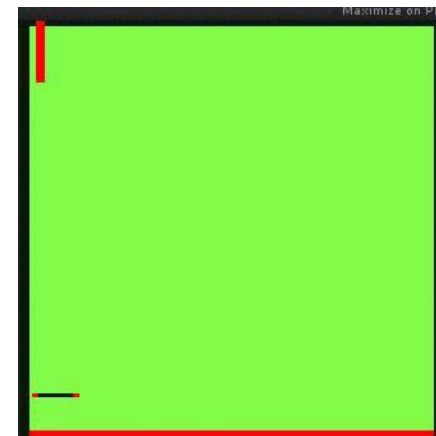
Step3.ビートに合わせてカッコよく

➤ `bool Music.IsJustChangedBar()/Beat()/At(Timing)`

➤ 小節ごと/拍ごと/任意のタイミングに来たフレームだけtrueになる

例：Field.csにて↓ 背景色の切り替え

```
if( Music.IsJustChangedBar() )  
{  
    MainCamera.backgroundColor = Music.Just.Bar % 2 == 0 ?  
        levels[currentLevel].BGColor : levels[currentLevel].BGChangeColor;  
}
```

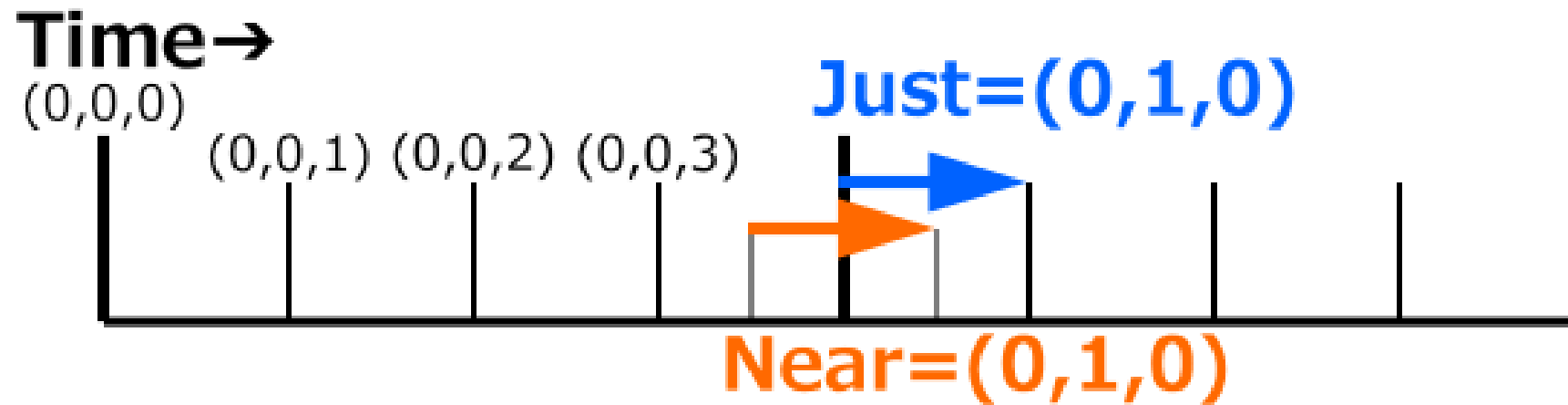


かっこいい演出完成。

背景色変えるのはオススメ

Tips3: Just & Near

- Timing Music.Just : 拍がちょうどに来てから切り替わる
- Timing Music.Near : 「最も近い拍」を示すように切り替わる



Note: テンポ120で60フレーム出てる場合は、1MusicalTime内に7.5フレーム存在する。

Note: 何か「この拍になる直前に処理しておきたい！」という時とかにIsNearChanged系を使うと便利。

Step4.音楽に合わせてアニメーション

クオリティをグッと上げる

Step4-1.点滅アニメーション

➤ bool Music.IsFormerHalf

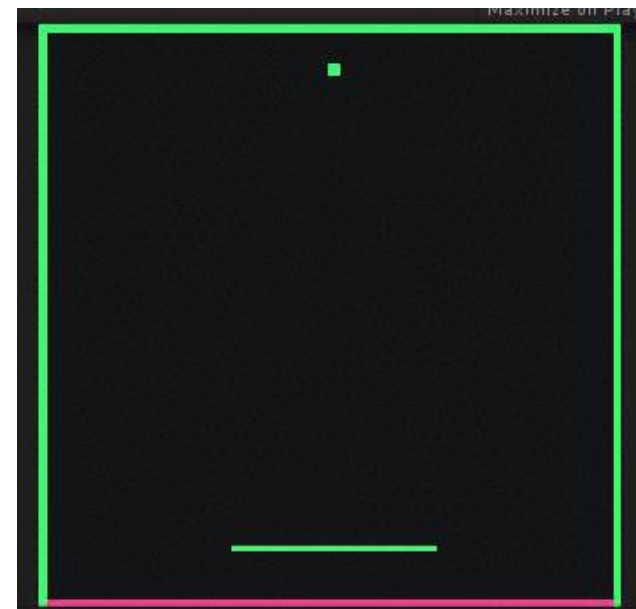
- 1MusicalTime内の前半と後半で切り替わる

➤ bool Music.IsJustChanged

- 16分音符ごとに1フレームずつtrueになる

例：Paddle.csにて↓ダメージの点滅や画面揺れに利用

```
//damage
if( damageMusicalTime > 0 )
{
    redBar.GetComponent<Renderer>().material.color = Music.IsFormerHalf ? Color.red : Color.white;
    if( Music.IsJustChanged )
    {
        --damageMusicalTime;
    }
}
```



Step4-2.出現アニメーション

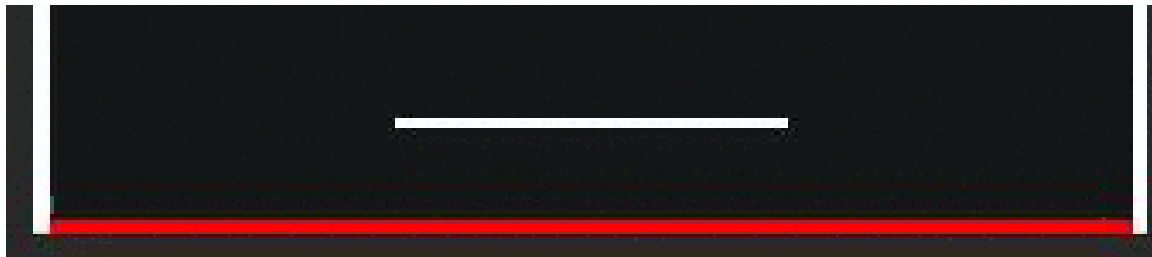
➤ float Music.MusicalTime

➤ float Music.MusicalTimeFrom(Timing)

➤ 音楽時間を浮動小数で取得

例：Padddle.csにて↓ 最初のバーが現れる演出

```
if( Music.CurrentSection.Name == "Start" )  
{  
    transform.localScale = new Vector3( initialScale.x * Mathf.Clamp01( (float)Music.MusicalTime / 16.0f ),  
}
```



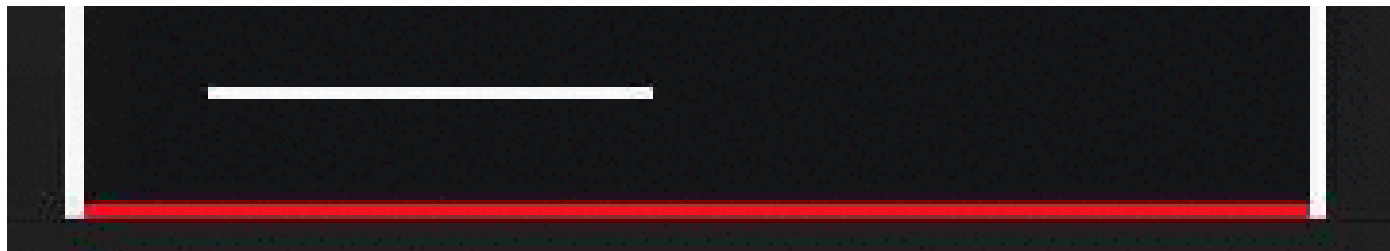
Step4-3.揺れるアニメーション

➤ float Music.MusicalCos(cycle, offset, min, max)

➤ 音楽に合わせてmax=1～min=0を動くコサインを返す

例：Field.csにて↓背景や落下地点の枠線の色をアニメーション

```
= Color.Lerp(EndBarColor, EndBarLerpColor, Music.MusicalCos(lerpMusicalTime));  
:Level].materialColor, levels[currentLevel].lerpMaterialColor, Music.MusicalCos(lerpMusicalTime));
```



Step5.音楽に合わせてシーン遷移

状態管理もラクラク

Step5-1.シークでシーン遷移

➤ Music.Seek(Timing)/SeekToSection(string name)

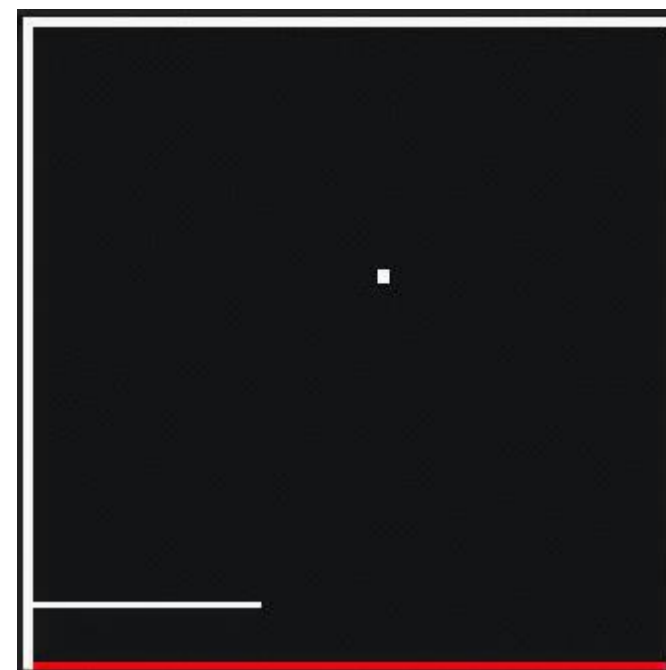
➤ 音楽の好きな箇所にシークすることができる。

例：Ball.csにて↓ゲームオーバー判定

```
else if( transform.position.y <= -Field.FieldLength )  
{  
    //floor  
    Music.SeekToSection( "GameOver" );  
}
```

例：Field.csにて↓リスタート処理

```
void Restart()  
{  
    Music.SeekToSection( "Start" );  
    Music.Play( "Music" );  
    ball.OnRestart();  
    paddle.OnRestart();  
}
```



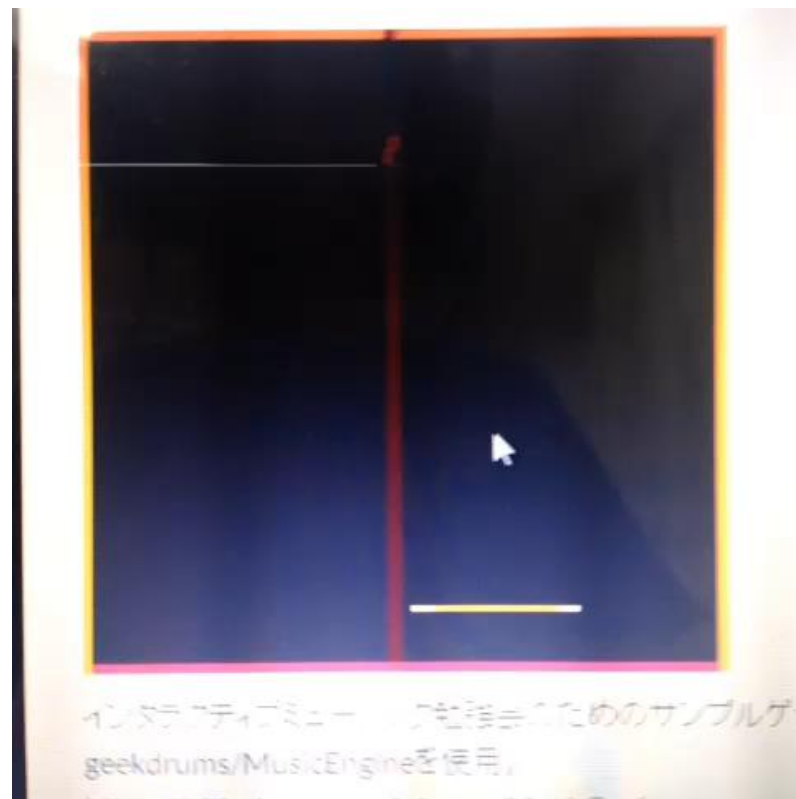
Step5-2.セクションでシーンを判定

➤ Music.CurrentSection

➤ 現在のセクション情報を取得

例：Field.csにて↓セクションを状態遷移判定に使う

```
switch( Music.CurrentSection.Name )
{
    case "Start":
        UpdateStart();
        break;
    case "Clear":
        UpdateClear();
        break;
    case "GameOver":
        UpdateGameOver();
        break;
}
```



できた。

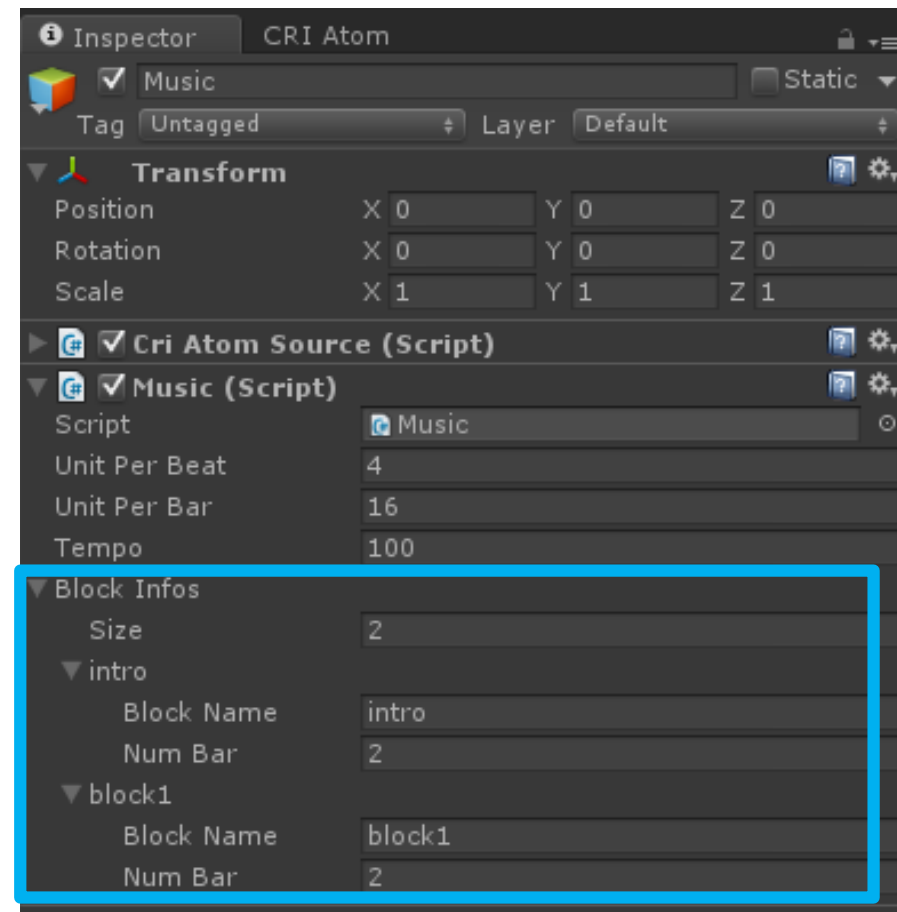
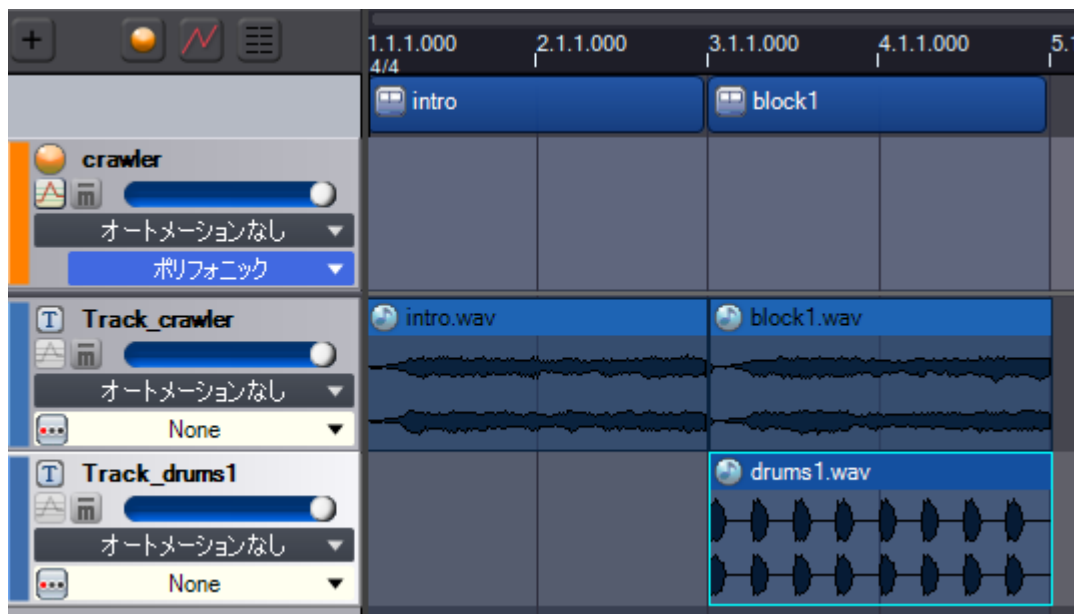
音楽がマスタークロックになってくれるから、
普通に作るより楽。

Unity+ADX2LE版と Unity StandAlone版の違い

(ほとんど同じように使えます)

SectionじゃなくてBlockを使う

- ADX2の機能「ブロック再生」と共存するため
 - 拍子やテンポは曲中では統一
 - Timing情報はブロックごとに(0,0,0)から始まる
 - 波形はブロックに対してぴったり合わせる



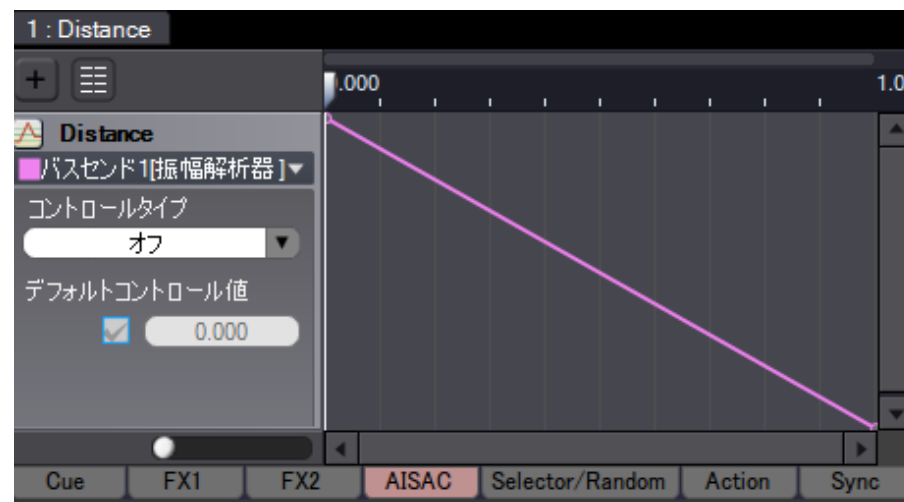
Block遷移、Aisac制御

➤ Music.Play(musicName, firstBlockName)

➤ Music.SetNextBlock(name/index)

➤ Music.SetAisac(name/index, value) →

などが利用可能。



おまじないの追加(必須)

Music.csの先頭のNoteにあるとおり

```
/* Note1: You must do this
Add this code in Plugins/CriWare/CriWare/CriAtomSource.cs
so that you can use SetFirstBlock function.
public CriAtomExPlayer Player
{
    get { return this.player; }
}
http://www53.atwiki.jp/soundtasukeai/pages/22.html#id\_6c095b2d
*/
```

これが無いとコンパイル通らないので注意。

詳しくは以下を参照

http://www53.atwiki.jp/soundtasukeai/pages/22.html#id_6c095b2d

おまじないの追加(必要であれば)

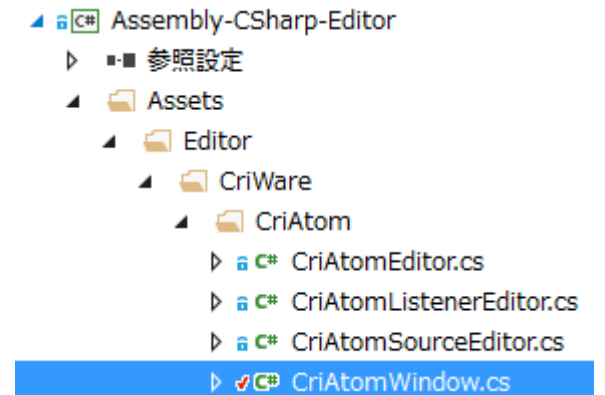
Music.csの先頭から2番めのNoteにある通り

```
/* Note2: You should do this
Auto update of block info.

Add this function in Editor/CriWare/CriAtom/CriAtomWindow.cs

////////////////////////////////////(geekdrums MusicEngine////////////////////////////////////
private void UpdateBlockInfo()
{
    string sourceDirName = atomCraftOutputAssetsRootPath.Replace( "/Assets", "" );
```

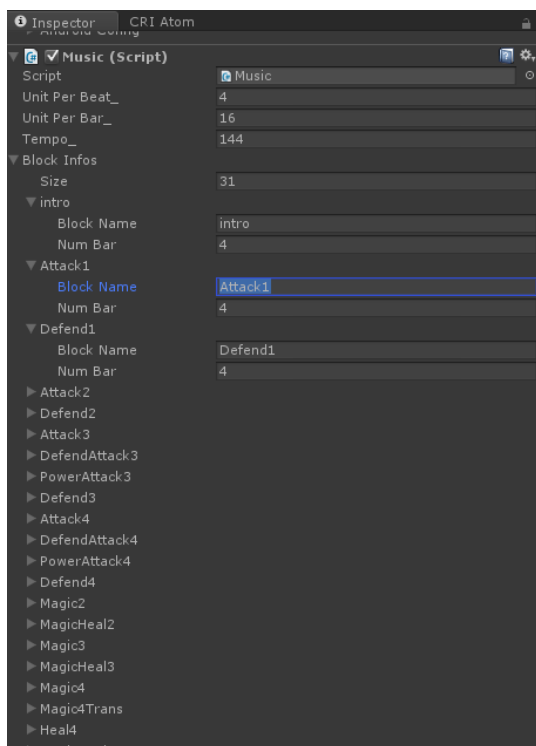
この関数をEditor/CriWare/CriAtom/CriAtomWindow.csに追加
すると……



Block情報の自動インポート

ブロックの数がこれくらいになってくると死ねる

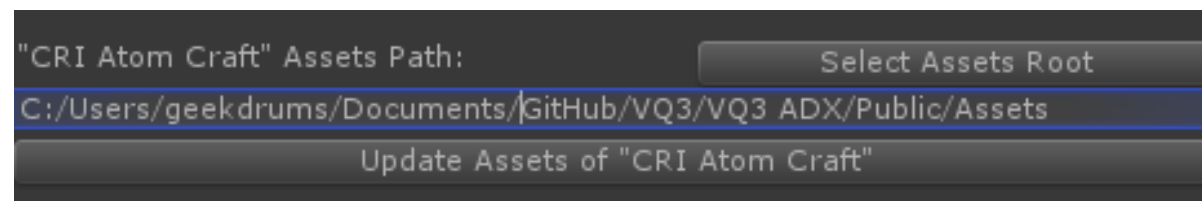
→acb_info.xmlから自動でUnityに反映



ADX2LEの「キューシートバイナリのビルド」オプション



Unityでの読み込み時に自動でブロック情報を反映



おしまい

[-] //Copyright (c) 2014 geekdrums

[//Feel free to use this for your lovely musical games :)