

Geekdrums/MusicEngine

を使った超速音楽同期ゲーム制作

採用事例(Unity+MusicEngine)

という名の宣伝

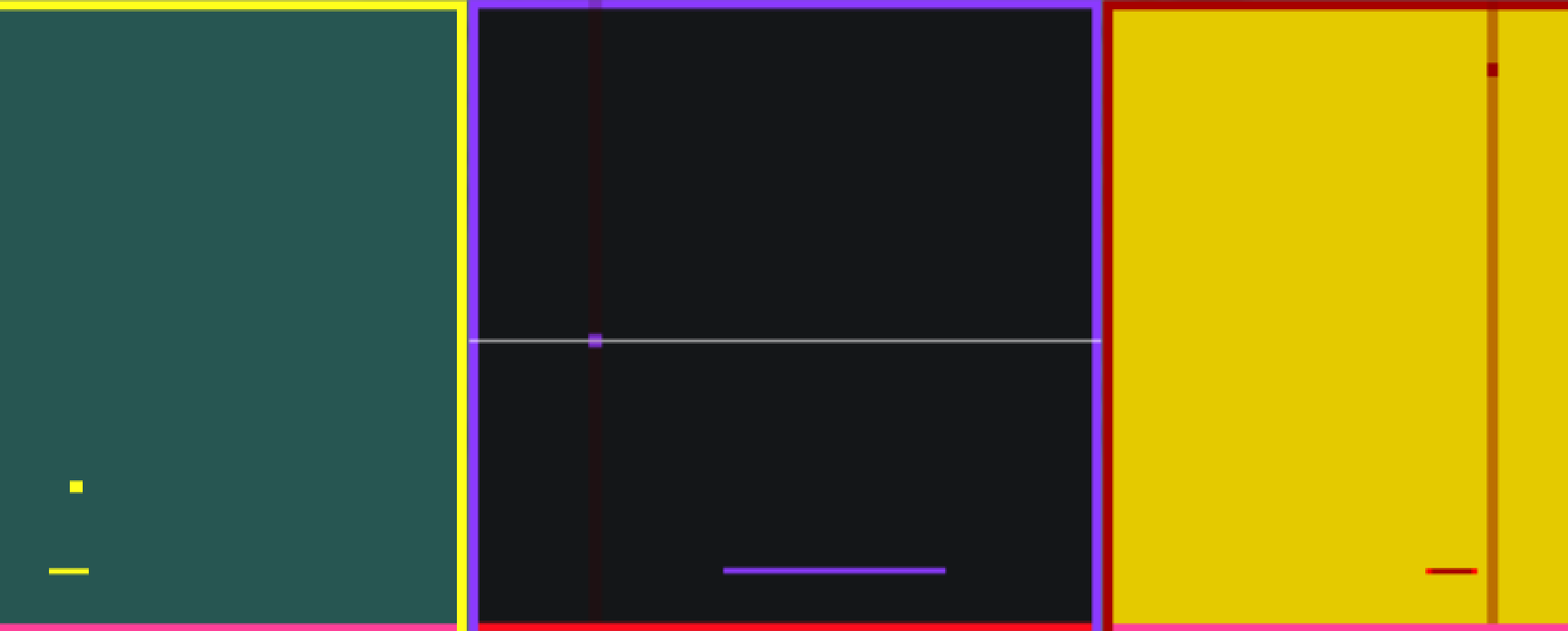


2014

VOXQUARTER(仮題)

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

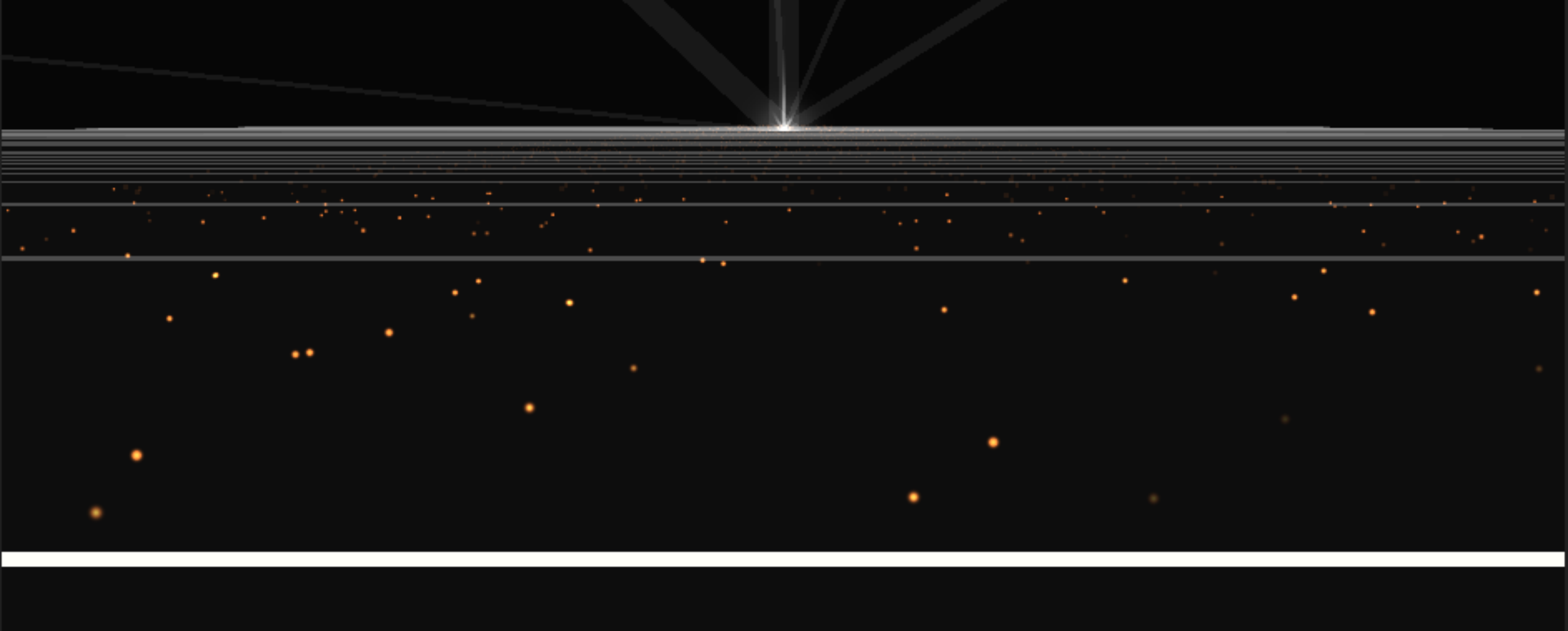
鋭意開発中



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 オーディオ部門で1位獲得

ゲーム制作は速さが命

神ゲーだと思い込んでいる内に作り上げろ

本題

ということで、ライブラリを使おう(OR作ろう)

MusicEngineとは

Unity用スクリプト「Music.cs」のこと。

これだけ→



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
- 「音楽のここ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！→**vjkit**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






「音楽に合わせたゲーム」
なら

めっちゃ速く作れる

サラマンダーより、ずっとはやい

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。

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(そのうち実装するかも)
- 16分音符(※)=1mt がすべての基準
 - **Music.MusicalTime**は16分音符の長さを1.0としている
 - ※3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

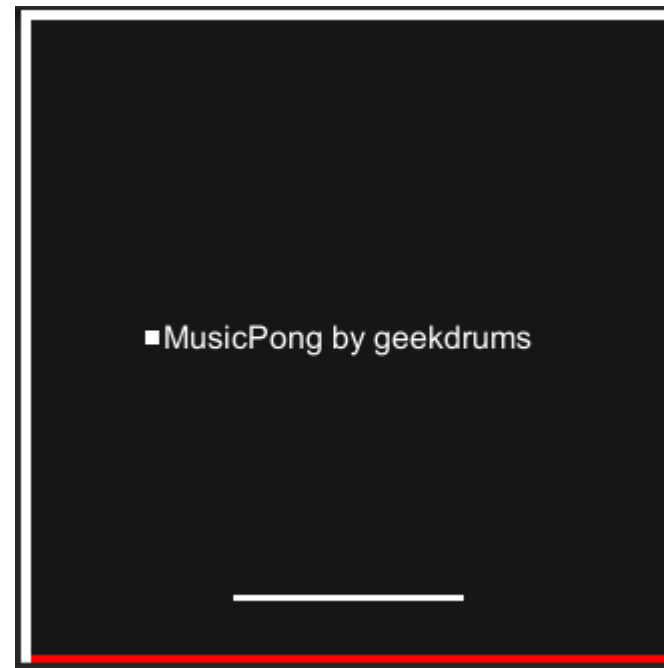
<http://unitygameuploader.jp/game/1233.html>

企画

「よし、Pongまだ作ってないから次はMusicPongだな」

Pongを作る

ぽんっと



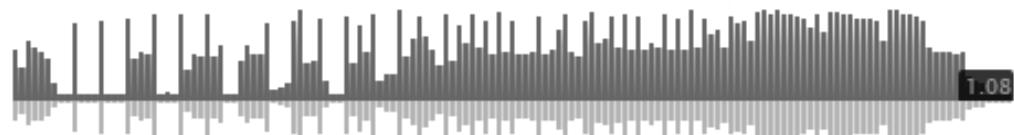
音楽を作る



geekdrums
MusicPong

8 hours

#game



1.08

さくっと(3日経過)

あと5ステップくらいで完成

はやい(確信)

Step1.Musicコンポーネントをつける

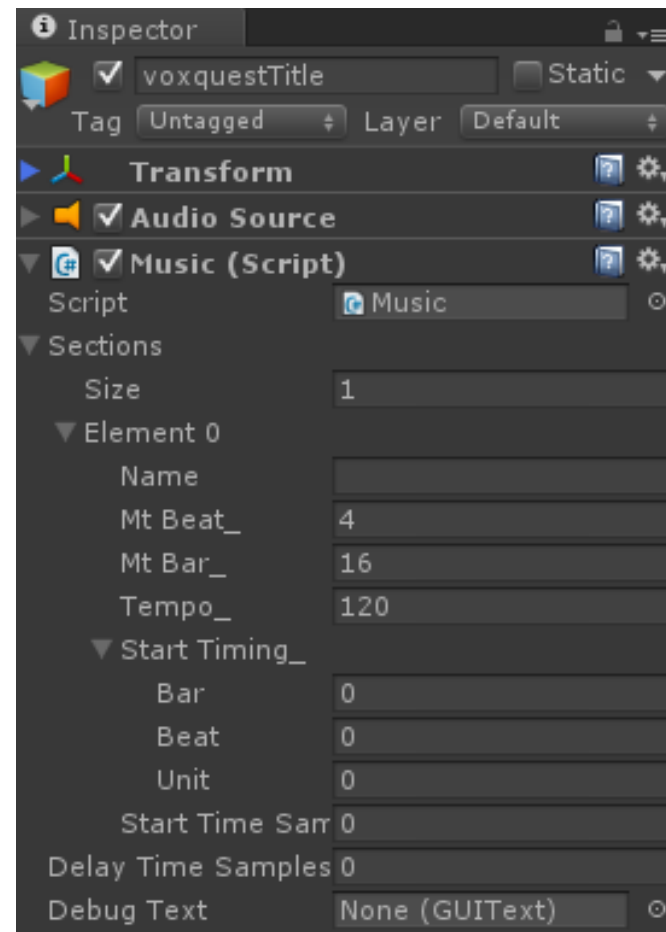
- AudioSourceをつけたオブジェクトにMusic.csを追加。
 - 自動的にデフォルトのセクション情報
(16分音符で4拍子、テンポ120)が挿入されます。

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DelayTimeSamples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 mtBeat_=4; //何mtで1拍とするか
- int mtBar_=16; //何mtで1小節とするか
- int Tempo_=120; //テンポ(=拍数/1分)

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

準備完了。

システムオールグリーン

Step2.とりあえずクオンタイズする

- `Music.QuantizePlay(AudioSource source, int transpose = 0);`
 - 自動的にmt(=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。

↓ 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( 1
Music.QuantizePlay( audio );
}
if( Field.instance.fieldLength <= transform.position.y )
{
    //roof
    velocity.y = Mathf.Abs( velocity.y ) * -Mathf.Sign( 1
Music.QuantizePlay( audio, 7 );
}
```

クオンタイズ+音程。
これだけで気持ちいい

めっちゃ楽

Step3.音楽に合わせて演出する

➤ 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

Field.csにて→
背景色を切り替え

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

```
if( shotMusicalTime > 0 )  
{  
    if( Music.isJustChanged )  
    {  
        --shotMusicalTime;  
    }  
}
```

➤ Music.isJustChanged

- 16分音符ごとに1フレームずつtrueになる

Ball.csにて→
ビーム時にボールを停止

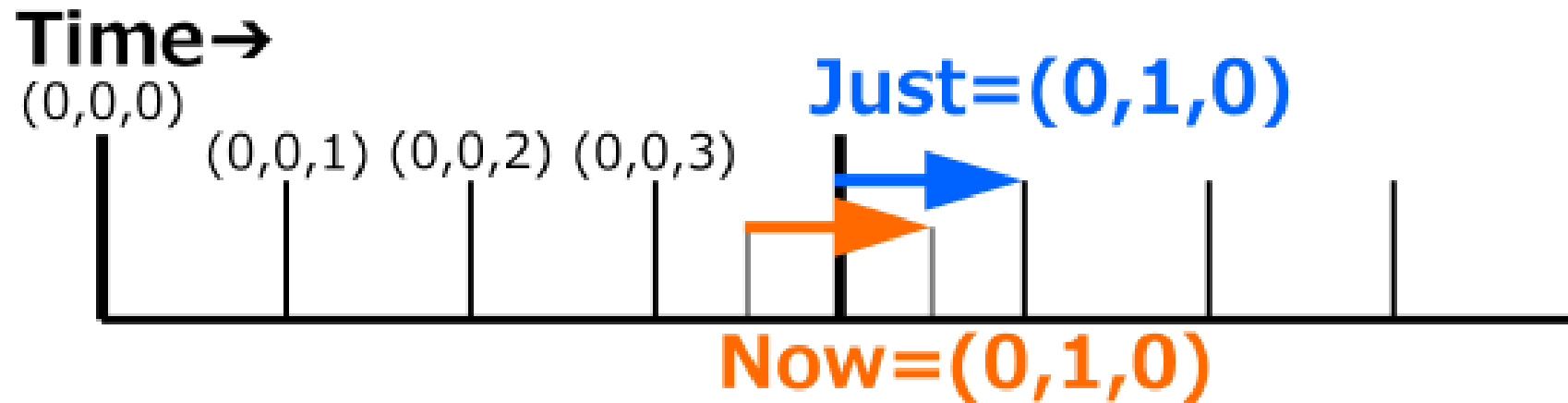
```
else  
{  
    UpdatePosition();  
}
```

かっこいい演出完成。

背景色変えるのはオススメ

Tips2: Just & Now

- Music.Just : 拍がちょうどに来てから切り替わる
- Music.Now : 「最も近い拍」を示すように(拍と拍の間で)切り替わる



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IsNowChanged系を使うと便利。

Step4.音楽に合わせてアニメーション

➤ Music.MusicalTime / MusicalTimeFrom(Timing)

➤ mt基準の時間を浮動小数で取得

↓Paddle.csにて。最初のバーが現れる演出

```
case "Start":  
    transform.localScale = new Vector3( initialScale.x * Mathf.Clamp01( (float)Music.MusicalTime / 16.0f ),  
    break;
```

↓Beam.csにて。shotTimingに合わせてアニメーション。

```
float d = Music.MusicalTimeFrom( shotTiming );  
if( Mathf.Abs( d ) <= 4.0f )  
{  
    //effect  
    float x = Mathf.Pow( beamPow, (-d - 0.5f) * beamScale );  
    transform.localScale = new Vector3( 1.0f + x, 1.0f, 0.03f + 1.0f / x );  
    renderer.material.color = ( d > 0 ? Color.Lerp( Color.red, Color.clear, (d - 1) / 2.0f ) : Color.white );  
}
```

Step5.音楽に合わせてシーン遷移する

➤ 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。

Ball.csにて→
ゲームオーバー判定

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

➤ Music.CurrentSection

- 現在のセクション情報を取得

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;
    default://Play
        UpdateColor();
}
```

できた。

パーフェクトプレイするのめっちゃむずい

No Damage!
Perfect Game!!!

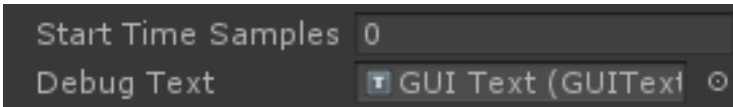
普通に作るより楽。

音楽がマスタータイマーになってくれる



Tips3: DebugText

インスペクタで設定→



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。

その他の機能とか

- `Music.isFormerHalf` `//1mtの前半後半で切り替わる。Blinkアニメなどにも使える`
- `Music.lagUnit` `//一番近いタイミングとの誤差を-0.5~0.5で返す。音ゲーなどに？`
- `Music.IsJustChangedWhen(Predicate)` `//デリゲートを使って好きなタイミングを抽出できる`
- `Timing operator <` `//if(Music.Just < timing)などで時間を比較できる(後の方が大きい)`
- `Timing operator -` `// Music.Just - timing など音楽時間を測れる。`

おしまい

[-] //Copyright (c) 2014 geekdrums

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