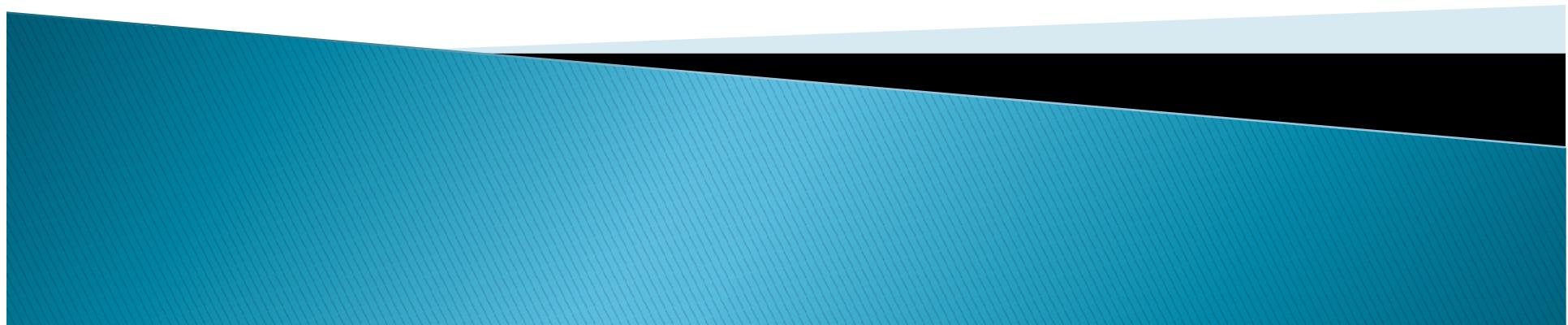


C#でゲームを作ろう2017

第3回 担当:ten

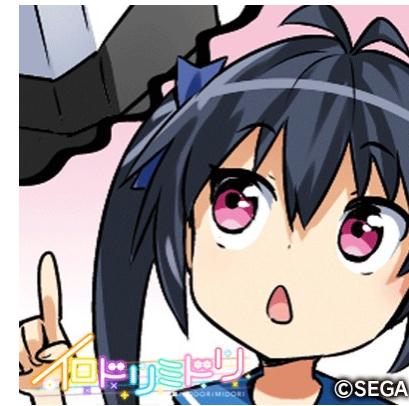


自己紹介

- ▶ ID:ten
- ▶ 京都大学工学部情報学科2回生
 - 計算機科学コース
- ▶ 第40代会長
- ▶ ゲーム制作して競プロしてる
- ▶ パズルと音ゲー



Twitter



Slackとか

なんと

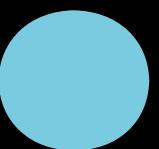
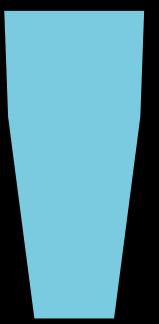
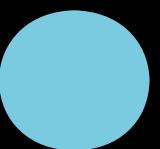
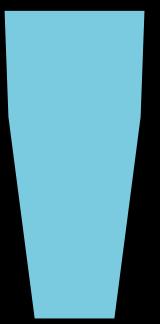
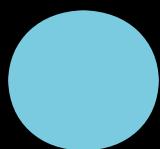
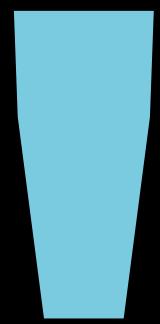
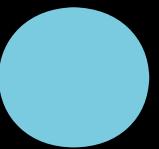
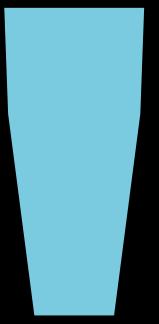
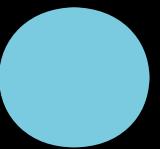
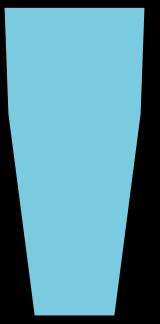
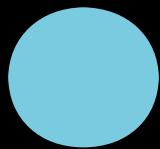
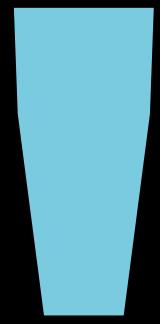
PLAYS

ラビット



FEAR





ラピッドコーディング祭り

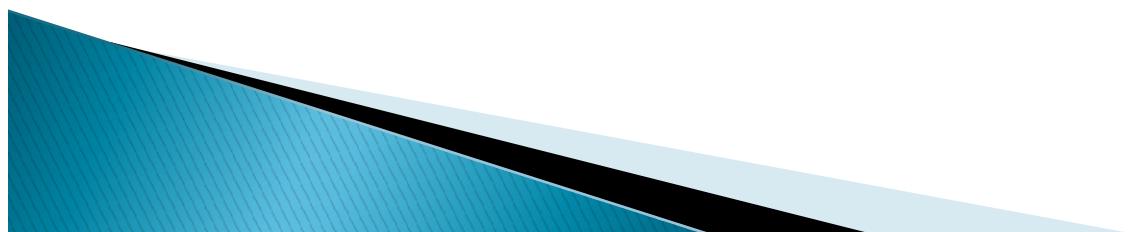
▶ **なんと明日はラピッドコーディング祭り！！！！！**

- ▶ 1日でゲームを作るすごいイベント
- ▶ 通称ラピゲ
- ▶ ぜひ参加してください



ラピッドコーディング祭り

▶ 昼も夜も奢



自己紹介をしましょう

- ▶ ID or 本名
- ▶ 所属（学部学科とか）
- ▶ 好きな or 印象的な講義
- ▶ なにか一言

※すべて任意です

※なにか一言が一番難しい気がする



今日の流れ

1. 復習
2. Monogameやる！！！



ゲームを考えてきてね

- ▶ 夏休み後あたりに発表会をする予定
- ▶ 夏休み前に内容を終えるので、そこから自分のゲームを作ってもらう予定
- ▶ というわけでゲームの内容を考えてきてください
- ▶ いわゆる「3分ゲー」のようなもので大丈夫です



Slack

- ▶ #csgame
- ▶ 質問があればここで or 口頭で
- ▶ この時間中の質問はいつでも受け付けています
- ▶ どんどん質問しましょう
- ▶ 忘れたらスライドとかどんどん見よう
- ▶ 調べよう



復習

- ▶ 関数とクラスってなんだっけというお話



関数

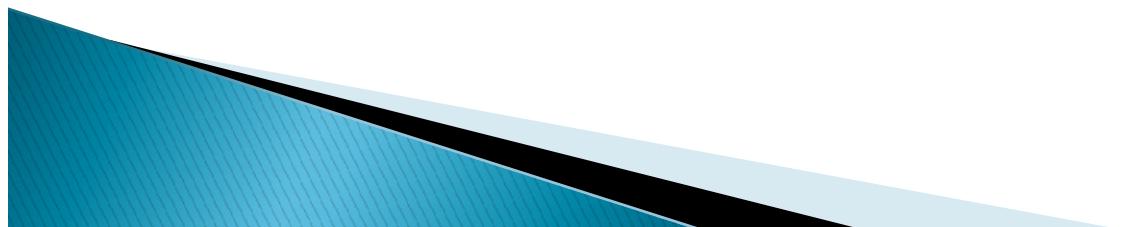
- ▶ ある引数を取って別の値を返す

```
static int Cube(int x)
{
    return x * x * x;
}
```



クラス

- ▶ クラスとは「設計図」
- ▶ クラスは以下のものを持っている
 - メンバ変数
 - メンバ関数
- ▶ 「設計図」を作ったら、
Main関数で「実体」を作る



クラス

▶ 敵の設計図

```
class Enemy {  
    private string name;  
    public string GetName()  
    {  
        return name;  
    }  
    public void SetName(string s)  
    {  
        name = s;  
    }  
}
```

クラス

- ▶ 実体を作る

```
class Program {  
    static void Main(string[] args) {  
        Enemy enemy = new Enemy();  
        enemy.SetName("こいつ");  
        System.Console.WriteLine  
        ("敵名：" + enemy.GetName());  
    }  
}
```



クラス

- ▶ この実体のことを「インスタンス」と呼ぶことがあります
- ▶ 実際のゲーム制作では、自分でクラスを作る以外にも、用意されてるクラスを使うことがある
 - その際は、インスタンスを作ること



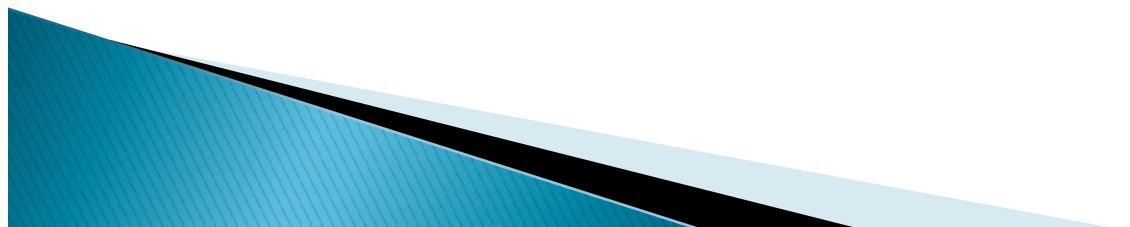
今日やること

- ▶ 黒い画面を脱出する



Monogameを使うにあたって

- ▶ ゲームを作っているとMonogameについての情報が欲しくなるときがある
- ▶ そこで検索ワードについて学びましょう
- ▶ プログラムが覚えられなくても、検索ができれば大丈夫!!!!



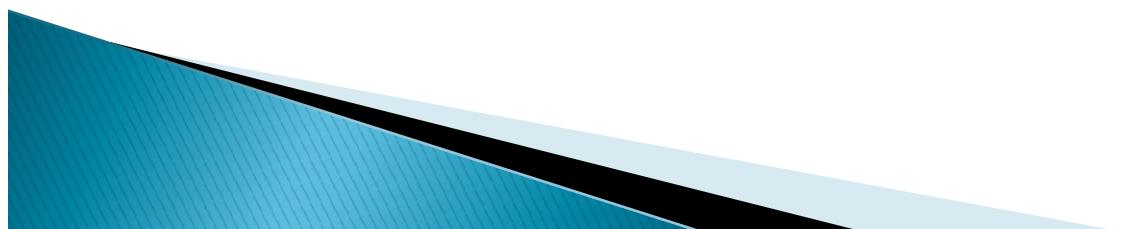
Monogameを使うにあたって

- ▶ もともとXNA Game Studioというゲームエンジンがあった
 - ▶ しかしXNAが死ぬ
 - ▶ そこで有志がXNAを再現したのがMonogame
-
- ▶ つまり
 - ▶ 「Monogame 画像表示」以外にも、「XNA 画像表示」と調べるとよいです



Monogameを使うにあたって

- ▶ 参考になるサイトの一例を紹介



Monogameを使うにあたって

- ▶ 公式サイト
 - <http://www.monogame.net/>

- ▶ 最強
- ▶ だが、初心者向けの情報はあまり載っていない
- ▶ リファレンス（機能の一覧）として見よう
- ▶ Documentationにリファレンスやイントロダクションが載ってる



Monogameを使うにあたって

- ▶ GARICCHI.COM
 - <http://garicchi.com/>
- ▶ ブログ（旧）のMonogameタグの記事
- ▶ 様々な事柄が分かりやすく解説



Monogameのプロジェクトを作る

新しいプロジェクト

.NET Framework 4.6.2 並べ替え: 既定

インストール済みテンプレートの検索 (Ctrl+E) 🔎

アイコン	名前	言語	説明
	MonoGame Windows Project	Visual C#	種類: Visual C# A MonoGame game project for the Windows desktop using DirectX.
	MonoGame Windows Store Project	Visual C#	
	MonoGame Windows Store (XAML) Project	Visual C#	
	MonoGame Windows Phone 8.1 Project	Visual C#	
	MonoGame Windows 8.1 Universal Project	Visual C#	
	MonoGame Windows 10 Universal Project	Visual C#	
	MonoGame Android Project	Visual C#	
	MonoGame Content Pipeline Extension Project	Visual C#	
	MonoGame Cross Platform Desktop Project	Visual C#	
	MonoGame iOS Project	Visual C#	

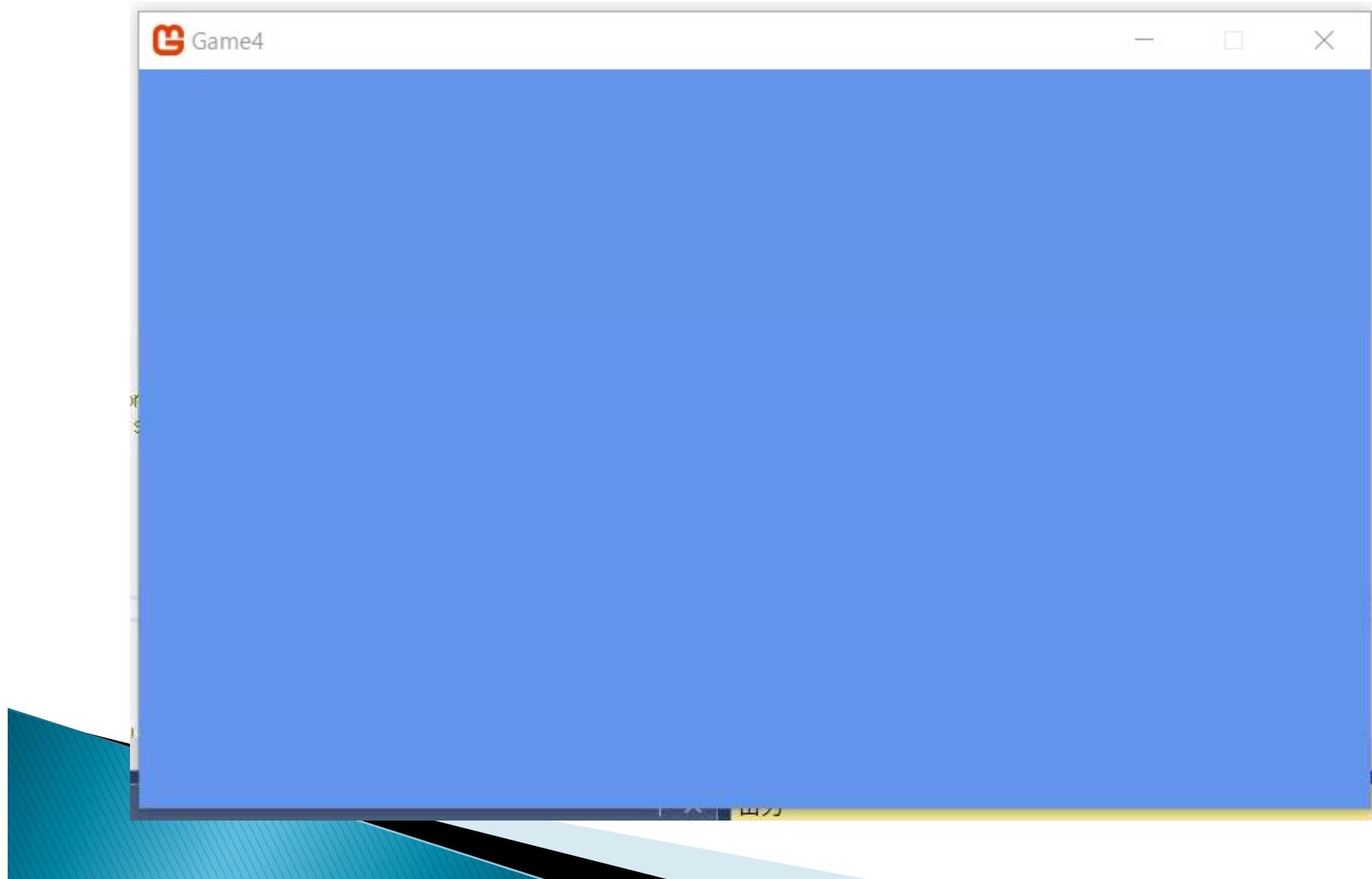
インストール済みテンプレート

- 最近使用したファイル
- インストール済み
 - テンプレート
 - Visual C#
 - Windows ユニバーサル
 - Windows クラシック デスクトップ
 - Web
 - Office/SharePoint
 - .NET Core
 - .NET Standard
 - Android
 - Cloud
 - Cross-Platform
 - Extensibility
 - iOS
 - MonoGame
 - tvOS
 - WCF
 - Workflow
 - テスト
 - Visual Basic
 - オンライン

Monogameのプロジェクトを作る



Monogameのプロジェクトを作る



Monogameのプロジェクトを作る

- ▶ 青い画面が出れば成功
- ▶ 成功してくれ！！！



Monogameを使い方

- ▶ Program.csとGame1.csができる
- ▶ 基本的にはProgram.csはいじりません
- ▶ 実際はProgram.csから始まって、その後Game1.csの関数が呼び出されてる



Game1.csの関数

- ▶ Game1()
- ▶ Initialize()
- ▶ LoadContent()
- ▶ UnloadContent()
- ▶ Update(GameTime gameTime)
- ▶ Draw(GameTime gameTime)



Monogameのメニュー

1. 画像表示
2. 音楽再生
3. テキスト表示



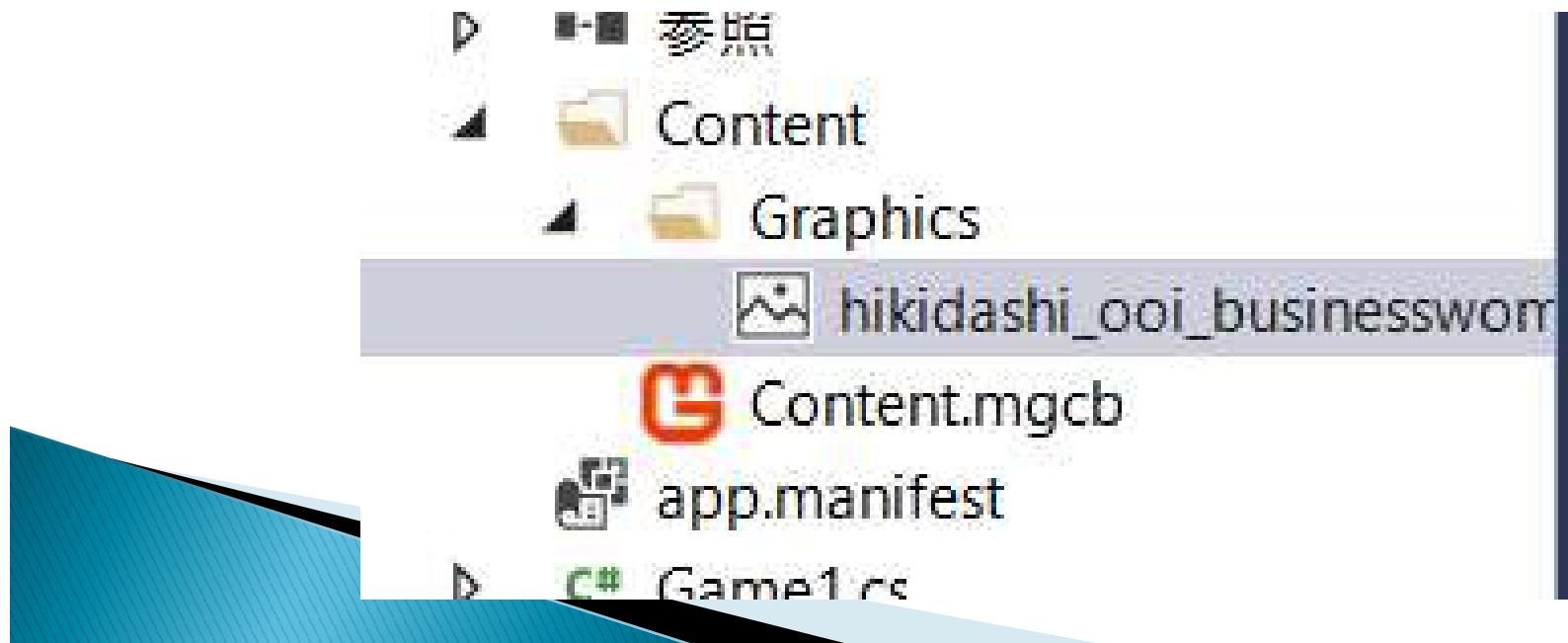
画像表示

- ▶ 画像を用意する
- ▶ いらすとや
 - <http://www.irasutoya.com/>



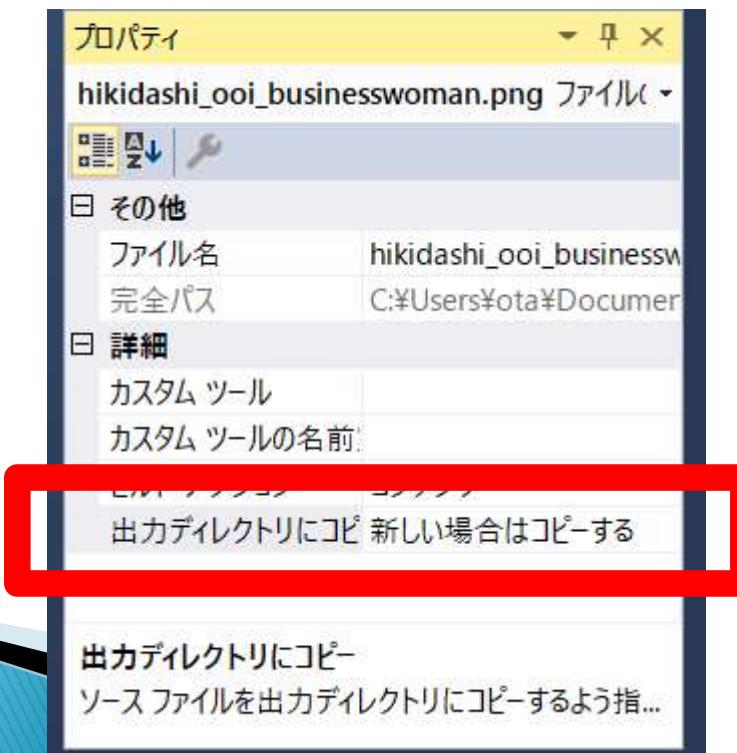
画像表示

- ▶ Content内にGraphicというディレクトリを作る
 - Content : ゲームの素材を入れる
 - 音楽素材などと分けるためにディレクトリを作る
- ▶ Graphics内にさっきの画像を入れる



画像表示

- ▶ プロパティを開く
- ▶ 詳細
- ▶ 「出力ディレクトリにコピー」のメニューを「新しい場合はコピーする」に変更



画像表示

- ▶ Game1.csをいじります



画像表示

- ▶ Texture2D（画像表示用のクラス）のインスタンスを宣言
 - 画像を入れるためのインスタンス

```
GraphicsDeviceManager graphics;  
SpriteBatch spriteBatch;
```

```
//追加  
Texture2D texture;
```



画像表示

- ▶ 画像を読み込む

```
protected override void LoadContent() {  
    spriteBatch = new SpriteBatch(GraphicsDevice);  
  
    //追加  
    texture =  
        Content.Load<Texture2D>("Graphics/hikidashi_ooi_business  
woman.png");  
  
}
```

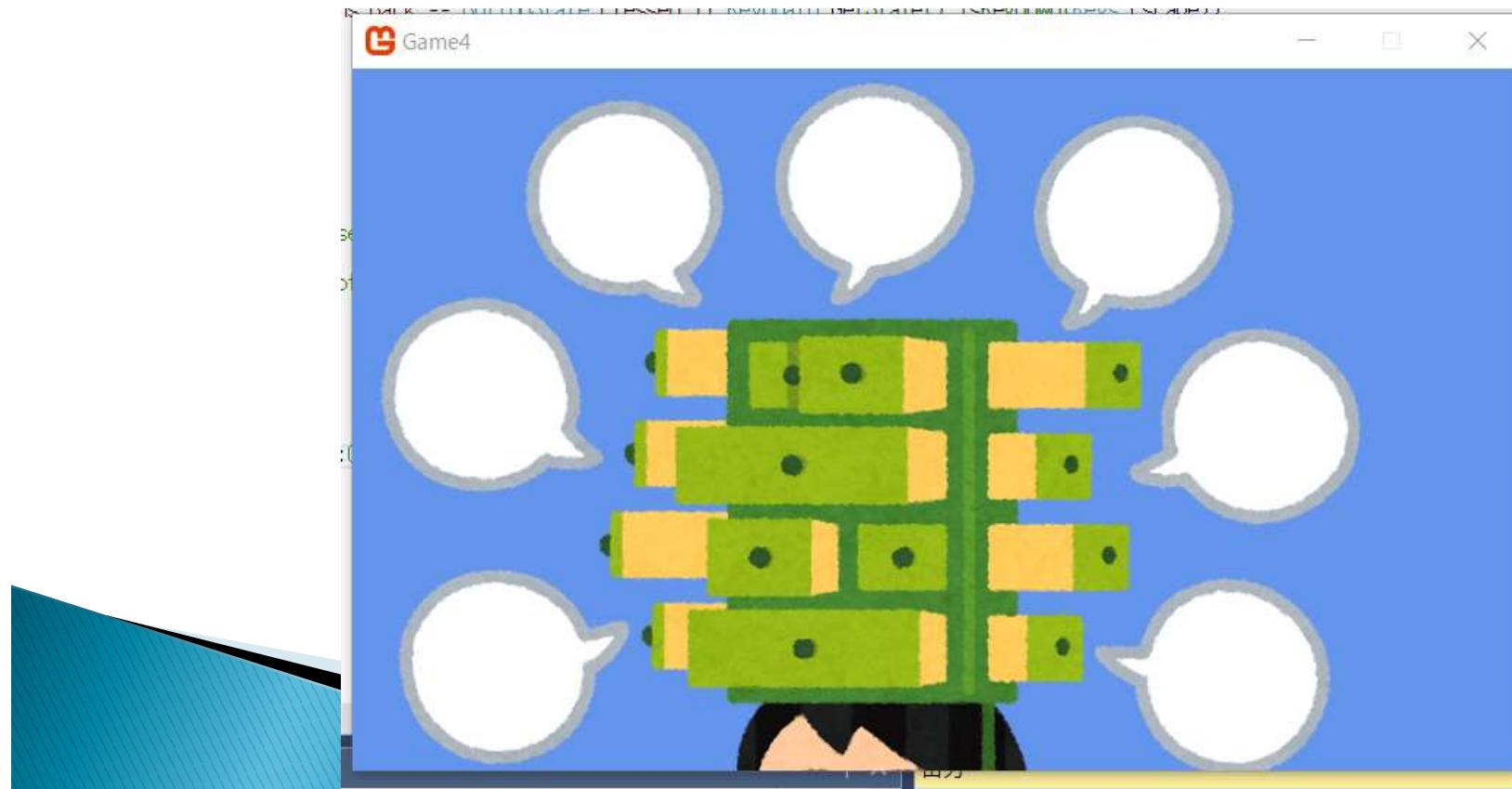
画像表示

- ▶ spriteBatch（実際に画面に表示するインスタンス）で画像を表示する

```
protected override void Draw(GameTime gameTime){  
    GraphicsDevice.Clear(Color.CornflowerBlue);  
  
    //追加  
    spriteBatch.Begin();  
    spriteBatch.Draw(texture, position: Vector2.Zero);  
    spriteBatch.End();  
  
    base.Draw(gameTime);  
}
```

画像表示

- ▶ 画像が出れば成功
- ▶ でけえ
 - 画像サイズの調整は時間があればやります



音楽再生

- ▶ 音楽再生
 - ▶ 音楽ファイルにはいろいろな形式がある
 - .wav .mp4 とか
 - ▶ Monogameでは「.xnb」 という特殊な形式を使う
-
- ▶ 世の中には.xnbなんてファイルはない
 - ▶ .wavとかのファイルを変換する必要がある



音楽再生

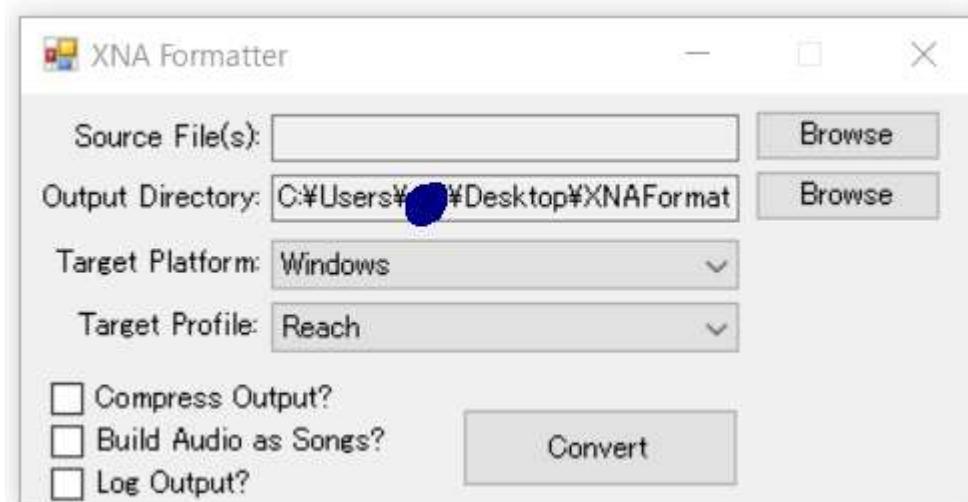
- ▶ 音楽を用意する
- ▶ BGMでも効果音でも構いません

- ▶ 魔王魂
 - <http://maoudamashii.jokersounds.com/>



音楽再生

- ▶ XNA Formatter
 - <https://sourceforge.net/projects/xnbbuilder/>
- ▶ このソフトで.xnbファイルに変換します



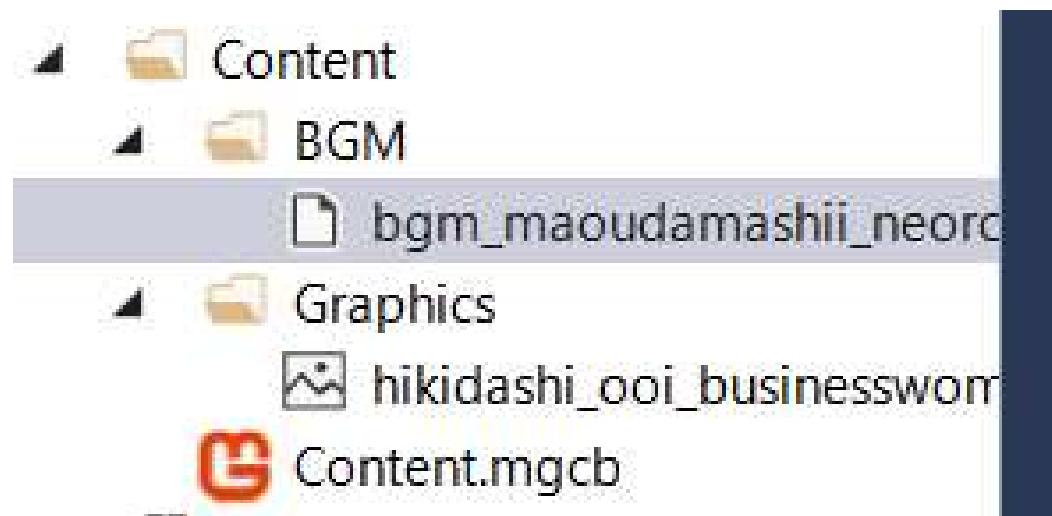
音楽再生

- ▶ さっき選んだ音楽ファイルを選ぶ
- ▶ Convertを押す
- ▶ 同梱のContentディレクトリに以下のような.xnbファイルができるはず



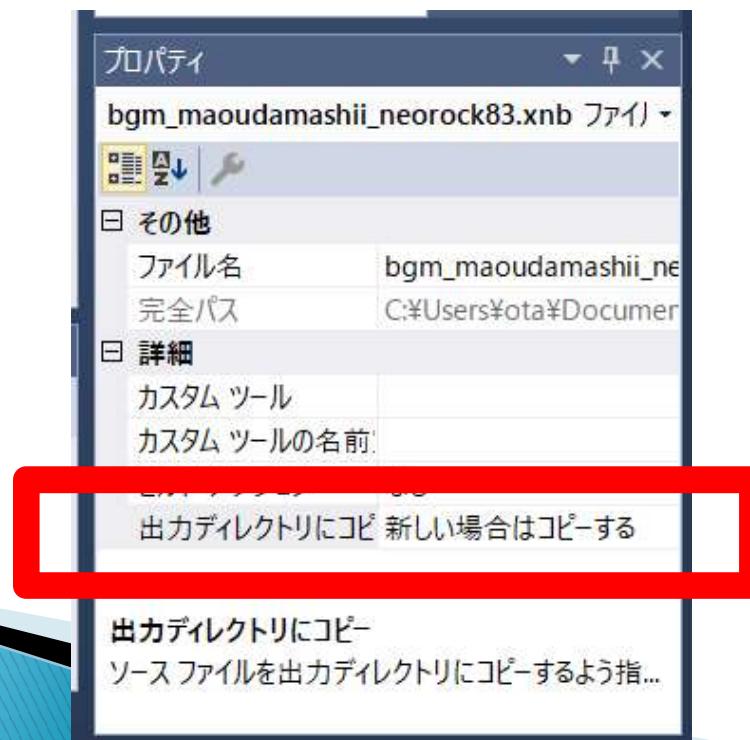
音楽再生

- ▶ Content内にBGMというディレクトリを作る
- ▶ BGM内にさっきの音楽ファイルを入れる



音楽再生

- ▶ プロパティを開く
- ▶ 詳細
- ▶ 「出力ディレクトリにコピー」のメニューを「新しい場合はコピーする」に変更



音楽再生

- ▶ 次はGame1.csをいじる



音楽再生

- ▶ 音楽関連の機能を使えるようにする

```
using Microsoft.Xna.Framework;  
using Microsoft.Xna.Framework.Graphics;  
using Microsoft.Xna.Framework.Input;  
  
//追加  
using Microsoft.Xna.Framework.Audio;
```



音楽再生

- ▶ SoundEffectクラス
 - 音楽を入れるインスタンスを宣言
- ▶ SoundEffectInstanceクラス
 - 音楽を実際に再生するためのインスタンスを宣言

```
GraphicsDeviceManager graphics;
```

```
SpriteBatch spriteBatch;
```

```
//追加
```

```
SoundEffect soundEffect;
```

```
SoundEffectInstance soundInstance;
```

音楽再生

- ▶ 音楽の読み込み
- ▶ 音楽を再生するためのインスタンスを作る

```
protected override void LoadContent(){  
    spriteBatch = new SpriteBatch(GraphicsDevice);  
  
    //追加  
    soundEffect =  
        Content.Load<SoundEffect>("BGM/bgm_maoudamashii_ne  
        orock83");  
    soundInstance = soundEffect.CreateInstance();  
    soundInstance.IsLooped = false; //ループする場合はtrue  
}
```

音楽再生

- ▶ soundInstanceが再生中でなければ再生する

```
protected override void Draw(GameTime gameTime)
{
    //追加
    if (soundInstance.State != SoundState.Playing)
    {
        soundInstance.Play();
    }

    GraphicsDevice.Clear(Color.CornflowerBlue);
    base.Draw(gameTime);
}
```

音楽再生

- ▶ 開始を押して音楽が再生されれば成功



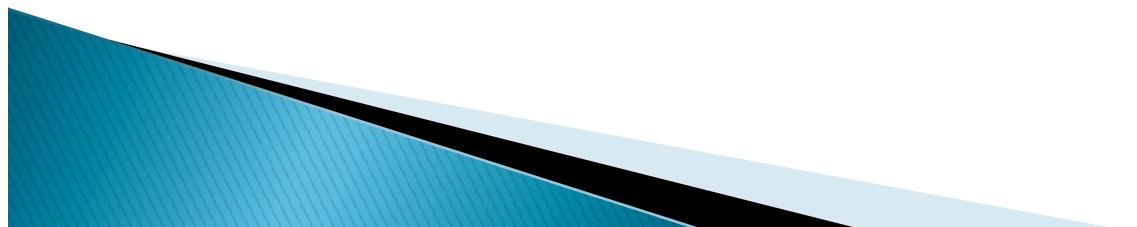
音楽再生

- ▶ Q. 音楽にノイズが入るんだけど？
- ▶ A. 音楽ファイルには8bit、16bit、…ってのがあるらしい
- ▶ 8bitだとうまく再生されない
- ▶ 「音楽ファイル 8bit 16bit」ってやつたら変換ツールとかがでてくるはず



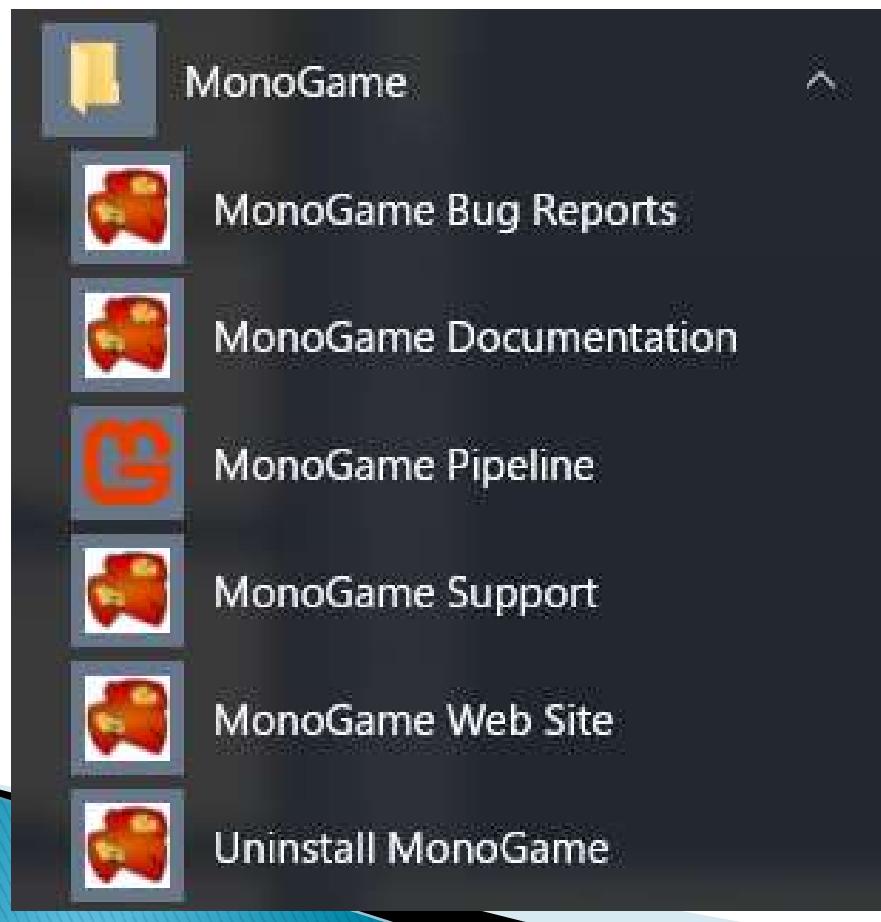
テキスト表示

- ▶ テキストを表示する
 - ▶ フォントファイルも「.xnb」に変換しなければならない
-
- ▶ フォントファイルも特殊な形式
 - ▶ 「.spritefont」という形式を「.xnb」に変換する



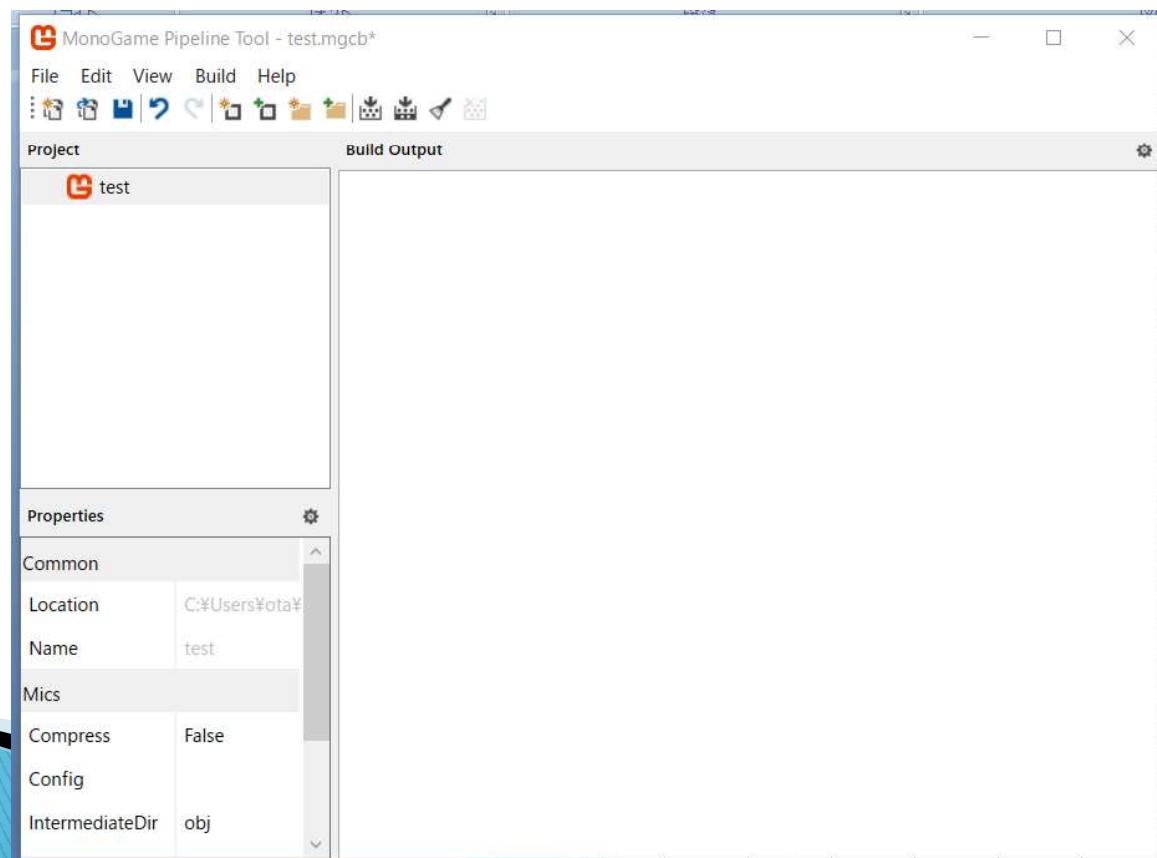
テキスト表示

- ▶ Monogame Pipeline入ってますか！！！



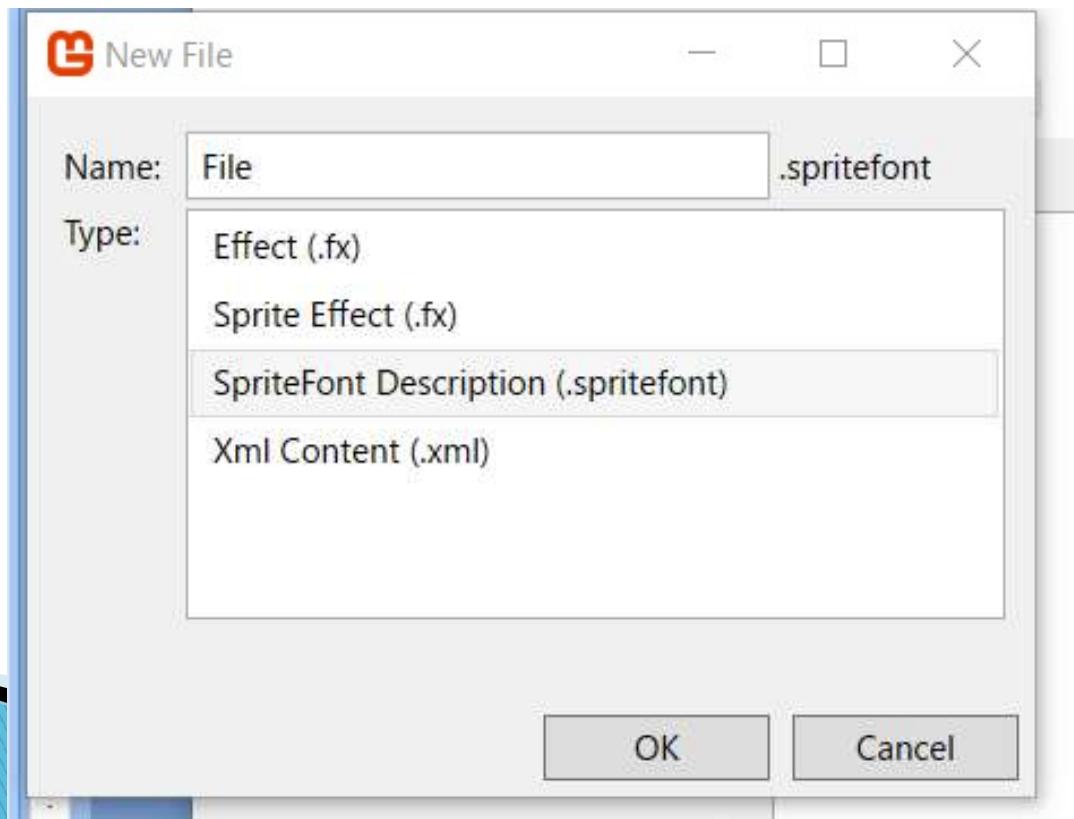
テキスト表示

- ▶ 「File」 → 「New」でプロジェクト作成
- ▶ Project内に作ったプロジェクトが表示される



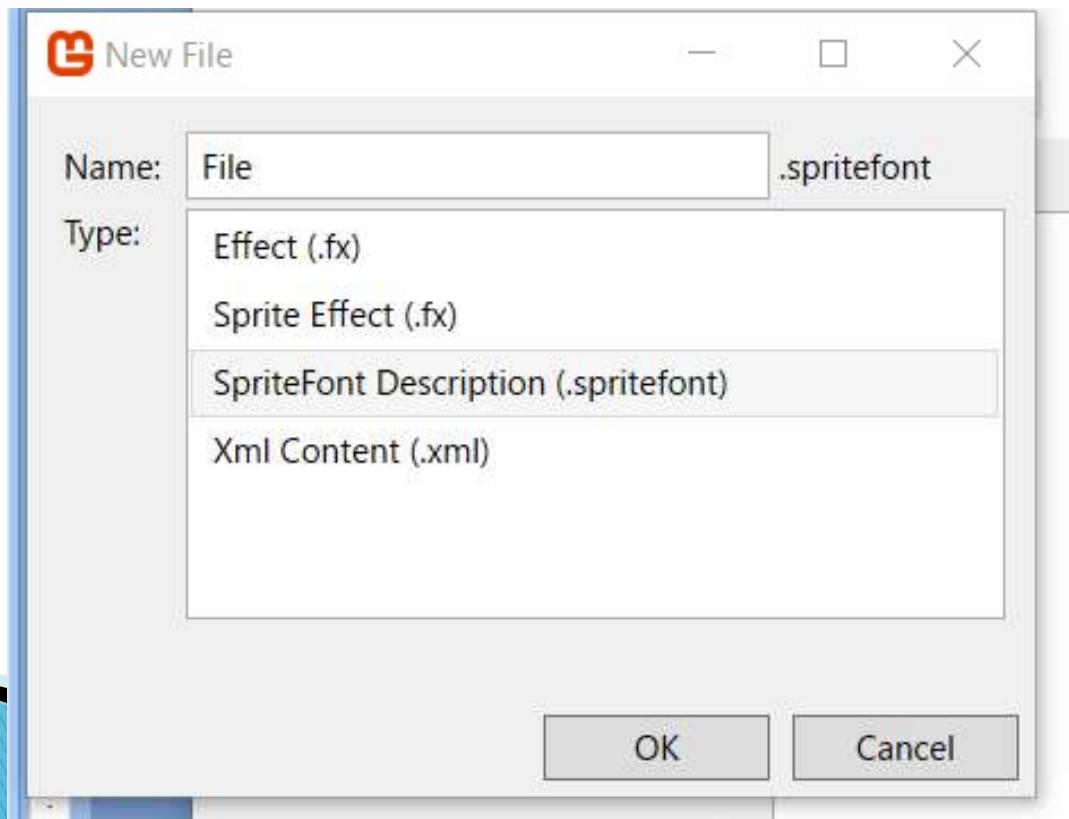
テキスト表示

- ▶ 「Add」 → 「New Item」
- ▶ SpriteFont Descriptionを選択
- ▶ 適当な名前で作成



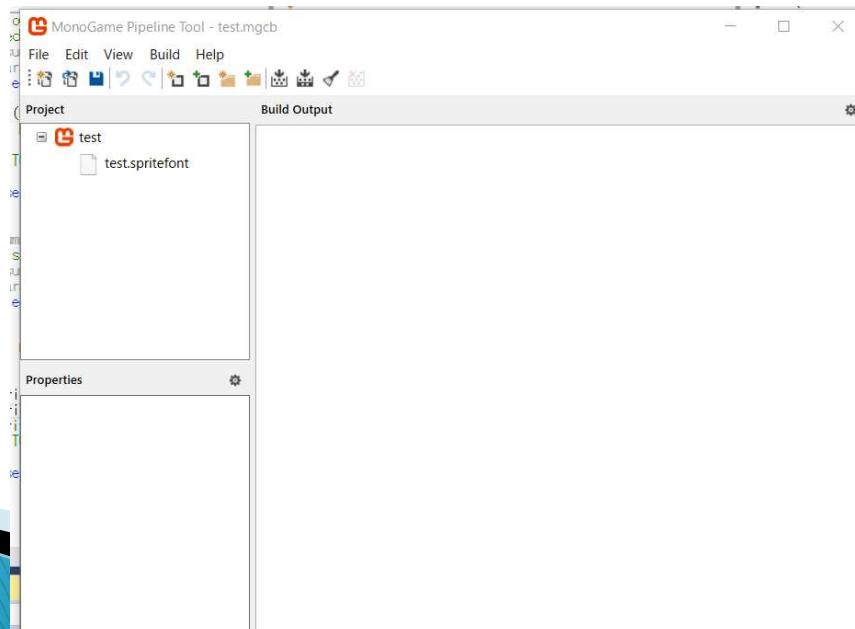
テキスト表示

- ▶ 「Add」 → 「New Item」
- ▶ SpriteFont Descriptionを選択
- ▶ 適当な名前で作成



テキスト表示

- ▶ spritefont1ができた
- ▶ 右クリックからRebuildをクリックするとこのプロジェクトのあるフォルダにbinというフォルダができる
- ▶ その中にxnbファイルがある



テキスト表示

- ▶ spritefontはxml形式で書かれている
- ▶ <FontName>タグを変えればフォントを変える

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This file contains an xml description of a font, and will be read by the XNA
Framework Content Pipeline. Follow the comments to customize the appearance
of the font in your game, and to change the characters which are available to draw
with.
-->
<XnaContent xmlns:Graphics="Microsoft.Xna.Framework.Content.Pipeline.Graphics">
<Asset Type="Graphics:FontDescription">

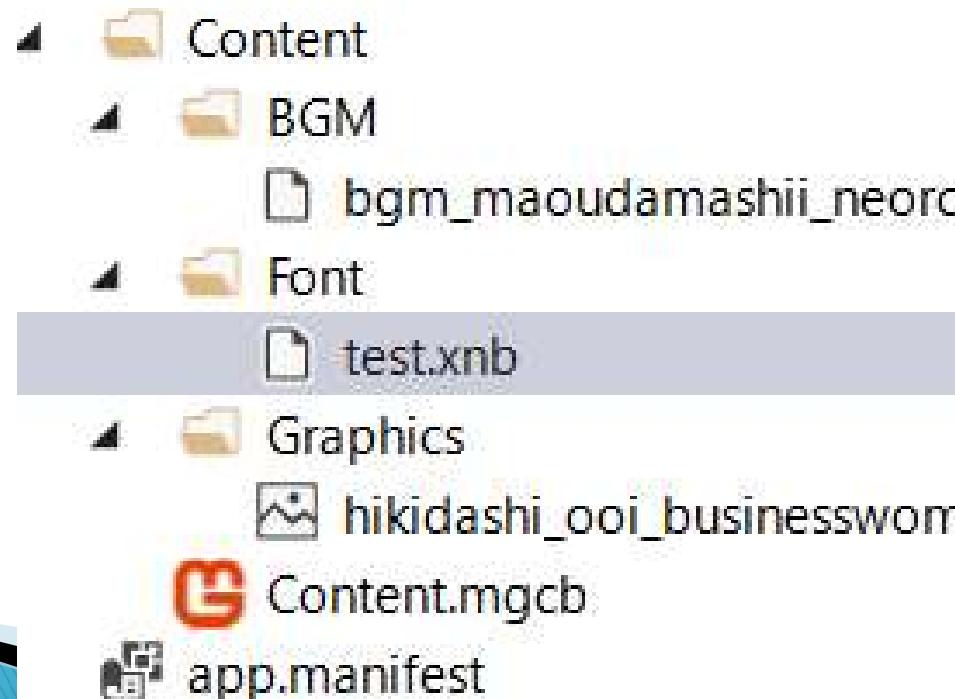
<!--
Modify this string to change the font that will be imported.
-->
<FontName>Arial</FontName>

<!--
Size is a float value, measured in points. Modify this value to change
the size of the font.
-->
<Size>12</Size>

<!--
Spacing is a float value, measured in pixels. Modify this value to change
the amount of spacing in between characters.
-->
<Spacing>0</Spacing>
```

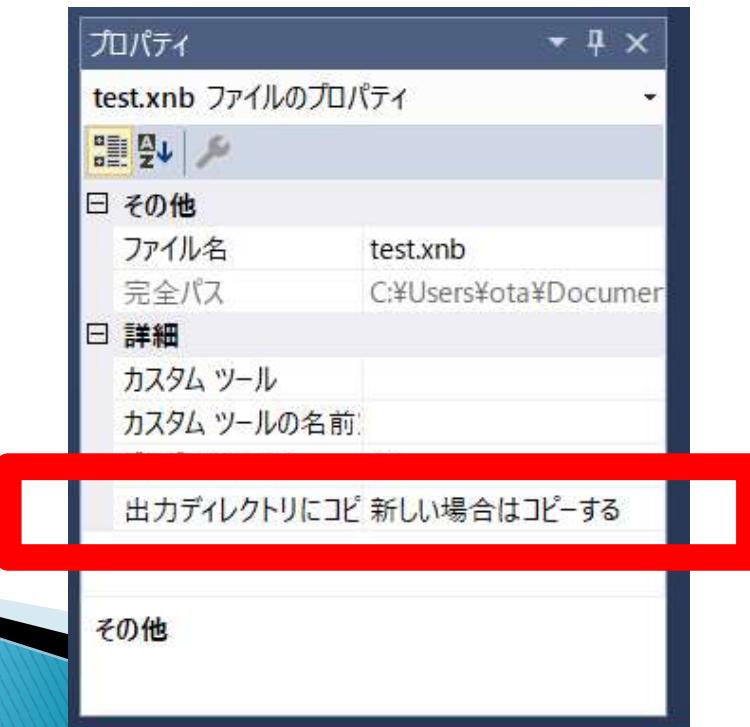
テキスト表示

- ▶ Content内にFontというディレクトリを作る
- ▶ Font内にさっきのフォントファイルを入れる



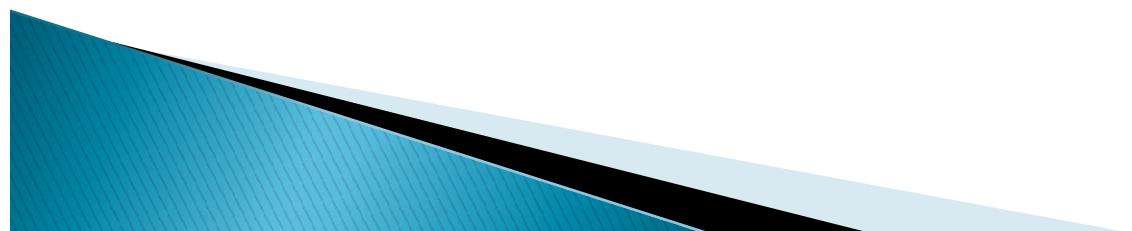
テキスト表示

- ▶ プロパティを開く
- ▶ 詳細
- ▶ 「出力ディレクトリにコピー」のメニューを「新しい場合はコピーする」に変更



テキスト表示

- ▶ 例によって次はGame1.csをいじる



テキスト表示

- ▶ SpriteFontクラス
 - フォントを入れるインスタンスを宣言

```
GraphicsDeviceManager graphics;  
SpriteBatch spriteBatch;
```

```
//追加  
SpriteFont spriteFont;
```



テキスト表示

- ▶ フォントを読み込む

```
protected override void LoadContent(){  
    spriteBatch = new SpriteBatch(GraphicsDevice);  
  
    //追加  
    spriteFont = Content.Load<SpriteFont>("Font/test");  
}
```



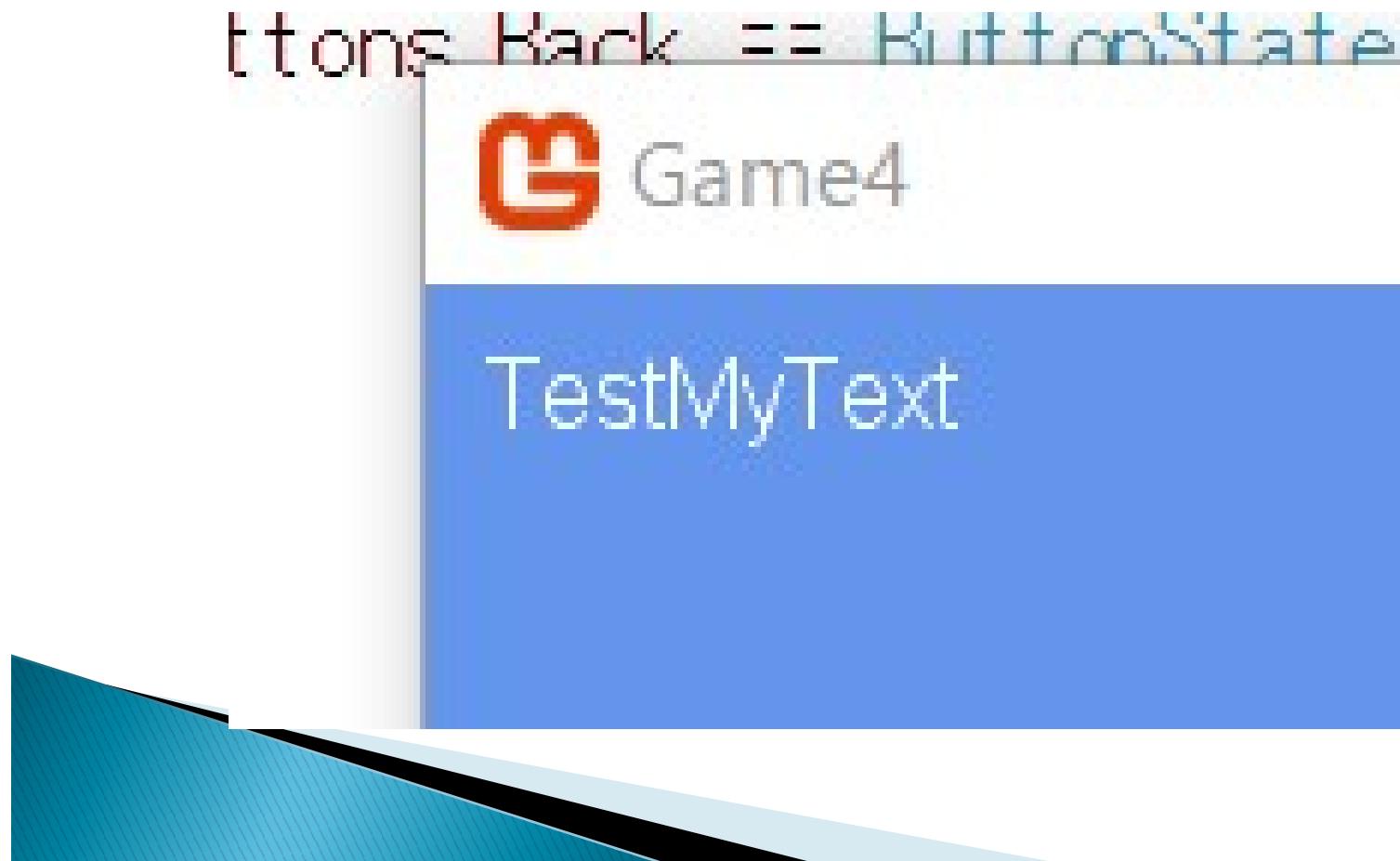
テキスト表示

- ▶ フォントを表示

```
protected override void Draw(GameTime gameTime){  
    GraphicsDevice.Clear(Color.CornflowerBlue);  
  
    spriteBatch.Begin();  
    spriteBatch.DrawString  
        (spriteFont, "TestMyText", new Vector2(10, 10), Color.White);  
    spriteBatch.End();  
  
    base.Draw(gameTime);  
}
```

テキスト表示

- ▶ 画面に“TestMyText”と表示されたら成功



おつかれさまでした

- ▶ 来週は休み！！！
 - ゴールデンウィーク！！！
- ▶ 次回は5/13(土)
 - 「雑な」シューティングの作り方をやると思う
 - ちゃんとしたゲームを作るための前準備

