XNA Game Studio
ゲームプログラミング

○　２Ｄ編 - 第３回　文字の描画　○

文字の描画

・XNAでは、文字を描画するには１文字ごとにテクスチャに変換しておく必要がある
・スプライトフォントという機能で上記をサポート。描画はスプライト扱いになる
・スプライトのため、「拡大縮小」「回転」「半透明」「Ｚバッファによる重なり」が使える
・そのままでは半角英数字しか描画できない
・描画はSpriteBatchクラスのDrawStringメソッドで行う
・スプライトフォントを生成し、文字の描画を行う方法は以下のとおり
　１．プロジェクトの"Content"にスプライトフォントを追加する
　２．ContentManagerクラスのLoadメソッドでスプライトフォントを読み込む
　３．DrawStringメソッドで描画(SpriteBatchクラスのBegin～Endメソッド内)

概要

XNAでは、文字の描画はスプライトフォントを用います。スプライトフォントは、コンテント・パイプラインをとおして１文字ごとにフォントデータをテクスチャに変換しておき、画面に描画するという仕組みです。スプライトなので、「スケーリング」「回転」「アルファブレンディング」「Ｚバッファ」を使うことができます。

しかし、システムにインストールされているフォントを指定し、テクスチャに変換せず文字を描画といったことはできません。XNAでは、フォントの著作権問題、描画速度、ゲームの雰囲気にあったフォントを選んで必要な文字だけデータ化できる、などの理由からスプライトフォントを作成しゲームに同梱するという方法を採用しています。

.spritefontファイル

コンテントにスプライトフォントを追加すると、拡張子が".spritefont"というファイルが作成されます。これは、フォントの情報を記述したファイルでXMLを拡張した形式で構成されています。フォントを変える場合、この内容を変更します。

ここでは、以下の属性が設定できます。

・フォント名	<FontName>	・文字スタイル	<Style>
・文字のサイズ	<Size>	・デフォルト文字(4.0以降)	<DefaultCharacter>
・文字間の幅	<Spacing>	・作成する文字コード	<CharacterRegions>
・カーニングの有無	<UseKerning>		

■フォント名　　<FontName>MS UI Gothic</FontName>
フォント名を指定します。フォントは開発環境にインストールされていれば、ビルド時にテクスチャとしてXNB形式に変換されるため、実行環境にインストールされていなくても問題なく使用できます。

■文字のサイズ　　<Size>14</Size>
文字のサイズです。小数も指定できます。

■文字間の幅　　<Spacing>0</Spacing>
文字と文字の間の幅を指定します。

■カーニングの有無　<UseKerning>true</UseKerning>
文字列を見やすいように間隔を自動調整(カーニング)するかどうかの指定です。
　true …カーニングを行います。自然な見栄えになります
　false…カーニングは行いません。文字列が詰まったようになります

■文字スタイル　　<Style>Bold, Italic</Style>
Regular(通常)、Bold(太字)、Italic(斜体)を組み合わせて文字のスタイルを指定します。組み合わせる場合は、カンマ","で区切ります。

■デフォルト文字　<DefaultCharacter>－</DefaultCharacter>
(4.0以降)スプライトフォントに登録されていない文字を表示しようとした場合に、代替として表示する文字の指定です。3.1以前は登録していない文字を表示すると例外が発生します。

■作成する文字コード
表示できる文字をUnicodeのコードで指定します。<Start>と<End>の間に含まれる文字がスプライトフォントとして登録されます。数値を広げれば日本語なども使用できるようになりますが、ビルドに時間がかかり、フォントデータのXNBファイルが大きくなってしまいます。

DrawStringメソッド(SpriteBatchクラス)

文字はSpriteBatchクラスのDrawStringメソッドで描画します。このメソッドはオーバーロードされており、引数の異なるものが複数定義されています。

試SpriteBatch.DrawString(*SpriteFont*, *String*, *Vector2*, *Color*);

SpriteFont	コンテントから読み込み済みのスプライトフォント
String	描画する文字列
Vector2	描画する画面上の座標
Color	文字列の色。透明度の指定も可能

－使用例－
(80, 124)に白で文字を描画
```
spriteBatch.DrawString(sprFont, "DrawString test",
                       new Vector2(80.0f, 124.0f), Color.White);
```

試SpriteBatch.DrawString(*SpriteFont*, *String*, *Vector2*, *Color*, *Single*, *Vector2*, *Single*, *SpriteEffects*, *Single*);

SpriteFont	コンテントから読み込み済みのスプライトフォント
String	描画する文字列
Vector2	描画する画面上の座標
Color	文字列の色。透明度の指定も可能
Single	原点を中心とした回転角(ラジアン単位) MathHelper.ToRadians(*角度*)で360度単位をラジアンに変換できます
Vector2	文字列の原点。文字列の左上隅が(0, 0)となります
Single	float型の拡大率。1.0fが基準
SpriteEffects	表示反転効果の指定。何もしない場合は"SpriteEffects.None"を指定
Single	深度。0.0f(最も前面)から1.0f(最も背面)の間で指定

－使用例－
(80, 124)に白で文字を描画。文字列の左上隅(10, 10)を軸に時計回りに45度回転。２倍に拡大
```
spriteBatch.DrawString(sprFont, "DrawString test",
                       new Vector2(80.0f, 124.0f), Color.White,
                       MathHelper.ToRadians(45.0f), new Vector2(10.0f,  10.0f),
                       2.0f, SpriteEffects.None, 0.0f);
```

試SpriteBatch.DrawString(*SpriteFont*, *String*, *Vector2*, *Color*, *Single*, *Vector2*, *Vector2*, *SpriteEffects*, *Single*);

SpriteFont	コンテントから読み込み済みのスプライトフォント
String	描画する文字列
Vector2	描画する画面上の座標
Color	文字列の色。透明度の指定も可能
Single	原点を中心とした回転角(ラジアン単位)
Vector2	文字列の原点。文字列の左上隅が(0, 0)となります
Vector2	拡大率。1.0fが基準。縦、横の拡大率をそれぞれ指定できます
SpriteEffects	表示反転効果の指定。何もしない場合は"SpriteEffects.None"を指定
Single	深度。0.0f(最も前面)から1.0f(最も背面)の間で指定

－使用例－
(80, 124)に半透明白で文字を描画。文字列の左上隅(10, 10)を軸に反時計回りに60度回転。
縦はそのまま、横に1.5倍拡大。深度は0.5
```
spriteBatch.DrawString(sprFont, "DrawString test",
                       new Vector2(80.0f, 124.0f), new Color(255, 255, 255, 128),
                       MathHelper.ToRadians(-60.0f), new Vector2(10.0f,  10.0f),
                       new Vector2(1.0f,  1.5f), SpriteEffects.None, 0.5f);
```

改行

XNAでは、改行を"Environment.NewLine"で表します。これを使用すると、動作時に各機種にあわせた改行コードに変換されるので、XNA対応ならどのような環境でも改行を処理できます。多くの言語で使用されている'\n'は使えませんが、WindowsとXbox 360(とWindows Phone)のみ対応であれば"\r\n"でも改行できます。

```
"Windows" + Environment.NewLine + "Xbox 360"
```

文字列の連結と変数の表示

XNAでは、文字列の連結は'+'で行うことができます。このとき、int型などの変数を+すると、数値が文字列(string)型へ自動的に変換され、連結されます。

コーディング例

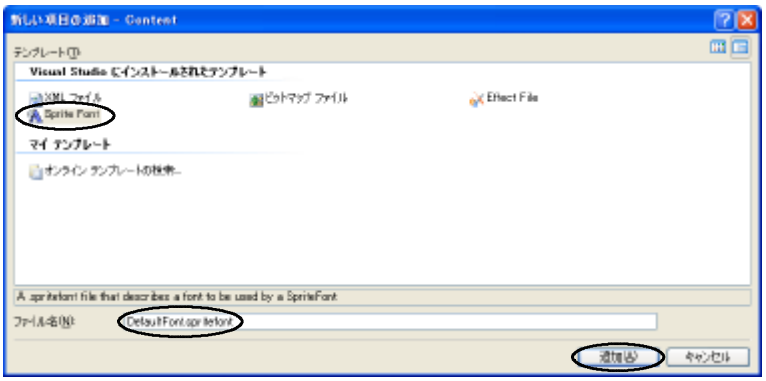
```
// フィールド変数
SpriteFont sprFont;          // スプライトフォントを管理
                    ⋮

protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
    // スプライトフォント読み込み
    sprFont = Content.Load<SpriteFont>("MyFont");
                    ⋮

protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

    // TODO: Add your drawing code here
    spriteBatch.Begin();

                    ⋮
    spriteBatch.DrawString(sprFont, "moji" + Environment.NewLine + "retsu",
                           new Vector2(300.0f, 10.0f), new Color(255, 255, 255, 255));
                    ⋮

    spriteBatch.End();

    base.Draw(gameTime);
}
```

課　　題

文字を描画しましょう。

(1)コンテントにスプライトフォントを追加します。

1. ソリューションエクスプローラの"Content"で右クリックし、「追加(D)→新しい項目(W)…」と選んでください。
2. 「新しい項目の追加」ダイアログが表示されます。"Sprite Font"を選択し、ファイル名(N):に"DefaultFont.spritefont"と入力してください。設定ができたら"追加(A)"ボタンをクリックします。



3. "DefaultFont.spritefont"の編集画面になります。フォント名を設定します。
   <FontName>DefaultFont</FontName>
   の部分を
   <FontName>MS UI Gothic</FontName>
   に変更してください。
4. ソリューションをリビルドしてください。

(2)画面に文字を描画しましょう。「コーディング例」を参考に、プログラムを追加してください。

(3)ソリューションをビルドし、実行してみましょう。エラーや例外が発生しなければ成功です。