ESライブラリ＆＆
ゲームプログラミング

○　２Ｄ編 - 第３回　文字列の描画　○

文字列の描画

・DirectX9は、基本的にモデルや頂点データといった3Dの描画しか行えない
・「文字列の描画」は、2Dグラフィクスの扱いになる
・文字をなんらかの形で3Dに変換する必要がある
　↓
・DirectX9には、文字の描画データをテクスチャに変換し、3D扱いで描画できる機能がある
・ESライブラリでは、上記の機能をもとに簡単に文字列を描画できる「SpriteBatch.DrawString関数」がある
・3D扱いなので、「BeginScene関数」と「EndScene関数」に囲まれた中でないと描画できない

概要

DirectX9(DirectX8以降)では、高速に2Dグラフィクスを描画できたDirectDrawが廃止され、すべての描画はDirect3DかWindowsのグラフィクス機能(GDI)を用いて描画します。

文字列には頂点データなどの3D情報はないため、Direct3Dで描画するには、文字列をなんらかの形で3Dで描画できる形式に変更しなければなりません。
DirectX9には、文字をテクスチャに変換し、スプライトをとおして描画できる機能があります。これを用いれば、文字列の描画が簡単に行えます。

ESライブラリでも、上記の機能をもとに作成した文字列描画関数「SpriteBatch.DrawString関数」があります。プログラムの起動時に、デフォルトフォントをDirect3Dで描画できるようにテクスチャへ変換しているので、この関数を実行するだけで簡単に文字列を描画できます。

```
// 文字列の描画
SpriteBatch.DrawString(DefaultFont, TEXT("文字列"),
                       Vector2(0.0f, 10.0f), Color(1.0f, 0.0f, 0.0f));
```

SpriteBatch.DrawString関数は、Direct3Dを使っているので、BeginScene関数とEndScene関数に囲まれた中でしか実行できません。また、内部的にスプライトを使っていますが、SpriteBatch.Begin関数とSpriteBatch.End関数の外でも描画できるようになっています。ただし、囲まれた中で描画した方が高速です。

課　　題

文字列を描画してみましょう。

(１)以下のプログラムを適切な場所に追加し、文字列を描画しましょう。

```
// 文字列の描画
SpriteBatch.DrawString(DefaultFont, TEXT("文字列"),
                       Vector2(0.0f, 0.0f), Color(1.0f, 1.0f, 1.0f));
```

(２)文字列の内容や座標、色を変更してみましょう
　ヒント：DefaultFont……デフォルトフォントの使用指定。フォントは自由に生成できます
　　TEXT("文字列") ……描画する文字列の指定。近年のスタイルでは、少々面倒ですが
　　文字列は「TEXT("")」で囲みます。
　　Vector2(x座標, y座標)…文字列の描画座標
　　Color(赤, 緑, 青)……描画色を左から赤、緑、青の順に指定。３つの成分が合成され、
　　1.0fが最も明るくなります(または整数0～255)。４番目は透明度。