ESライブラリ＆＆
ゲームプログラミング

○　３Ｄ編 - 第１３回　テクスチャ　○

テクスチャ

・テクスチャは、物体表面の質感を表現するための画像のこと
・ポリゴンの凹凸やマテリアルだけでは表現できない複雑な模様も簡単に表現できる
・テクスチャマップを応用したバンプマップ、ディスプレースメントマップなど、さまざまなマッピング技術が開発されている

概要

テクスチャとは、物体表面の質感を表現するために、ポリゴンに貼り付ける画像のことをいいます。同じ立方体のモデルでも、金属のテクスチャを貼り付ければ金属片に見え、木目のテクスチャを貼り付ければ木片に見えます。

テクスチャを用いると、ポリゴンの組み合わせやマテリアルでは表現できない複雑な模様を表現することができます。また、テクスチャを貼ることにより、少ないポリゴンでもリアリティあるオブジェクトを作ることができます。

テクスチャの生成

テクスチャには、さまざまな制限があります。もっとも特徴的なのは、テクスチャの大きさです。テクスチャの大きさは、ほとんどの環境で128ピクセルや256ピクセルといったように、２"に制限されます。最大サイズは、環境により異なりますが、DirectX Graphicsの仕様の上限は4,096×4,096ピクセルです。また、サポートするフォーマットも環境により異なります。テクスチャを生成する場合は、これらを考慮しなければなりません。

テクスチャの生成はIDirect3DDevice9::CreateTextureメソッド、D3DXCreateTexture関数、D3DXCreateTextureFromFile関数などで行います。Direct3DXの関数を用いれば、環境の差異を吸収してテクスチャを生成することができるので、コードを大幅に削減することができます。

```
// テクスチャ生成(pD3DDeviceは初期化済みのDirect3DDevice9オブジェクト)
IDirect3DTexture9*   pTexture;
::D3DXCreateTextureFromFile(pD3DDevice, "Texture.png", &pTexture)
```

生成に成功すると、テクスチャを制御するためのIDirect3DTexture9インタフェースが得られます。

ESライブラリでも、内部で上記の関数を呼び出し、テクスチャを生成しています。スプライトとしても扱えるよう、テクスチャはISpriteクラスで管理するようになっています。

テクスチャの設定

プリミティブやモデルにテクスチャを適用するには、テクスチャを生成したときに得られたインタフェースをIDirect3DDevice9::SetTexutreメソッドに渡すことで行います。以後、すべてのレンダリング時に、設定したテクスチャが適用されます。

```
// テクスチャ設定
pD3DDevice->SetTexture(0, pTexture);
```

表面の色は、デフォルトではマテリアルとテクスチャの色が乗算されて適用されます。マテリアルが黒[RGB(0.0, 0.0, 0.0)]の場合は、乗算の結果が０になるため、テクスチャが適用されていないように見えます。

ESライブラリでは、頂点バッファやモデルにテクスチャのファイル名を指定すれば、描画時に自動でテクスチャを適用するようになっています。

テクスチャ座標(ＵＶ座標)

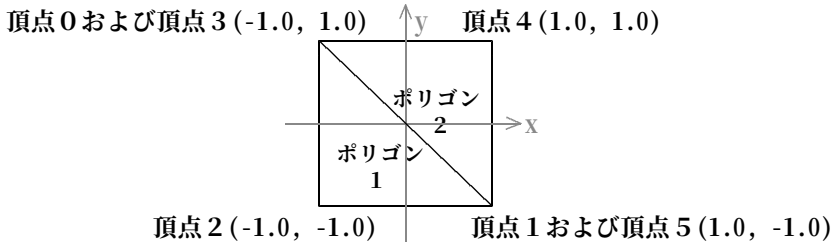
テクスチャ座標とは、ＵＶ座標とも呼ばれ、テクスチャに付けられた座標のことです。テクスチャの大きさにかかわらず、左上が(0.0，0.0)、右下が(1.0，1.0)となります。

ポリゴン(頂点)に単純にテクスチャを貼ってしまうと、ポリゴンの移動、変形に対応できないという問題が生じてしまいます。そこで、DirectX Graphicsでは「この頂点には、テクスチャのこの部分を貼る」という方法をとっています。このようにすれば、ポリゴンが動いても変形しても、テクスチャが正しく追従できるというわけです。これが、テクスチャ座標の基本的な考え方です。

課　題

ワールド空間にプリミティブを配置し、テクスチャを適用しましょう。

(１)プリミティブを使い、ワールド空間に以下のポリゴンを２つあわせた四角形を描画しましょう。



１．頂点の情報を設定します。以下のプログラムをCGameMain::Draw関数に追加しましょう。

```
// 頂点設定(未トランスフォーム・ライティング済み頂点)
VertexPositionColorTexture   v[6];

// 頂点０
v[0].x     = -1.0f;          // ｘ座標
v[0].y     =  1.0f;          // ｙ座標
v[0].color = Color_Cyan;     // 色
```

２．上記を参考に、頂点１～頂点５を配列v[1]からv[5]に設定しましょう。

３．頂点が設定できたらプリミティブを描画してみましょう。以下のプログラムを適切な場所に追加しましょう。

```
// ライティング済み頂点は、必ずDirect3Dによる照明演算を無効にする
GraphicsDevice.SetRenderState(LightEnable, FALSE);

// プリミティブ描画
GraphicsDevice.DrawUserPrimitives(PrimitiveType_TriangleList, v, 2, v[0].FVF());
```

(２)ＵＶ座標を設定し、テクスチャを読み込んで貼り付けてみましょう。

１．テクスチャとなる画像を準備します。以下のような画像をContentフォルダに準備しましょう。

２．テクスチャを格納する変数を宣言します。以下のプログラムを適切な場所に追加しましょう。

```
// 変数宣言
SPRITE    texture;
```

ESライブラリでは、テクスチャはスプライトとしても扱えるよう、ISprite*型(SPRITE型)変数で管理します。

３．画像を読み込みます。以下のプログラムを適切な場所に追加しましょう。

```
// テクスチャ読み込み
texture = GraphicsDevice.CreateSpriteFromFile( TEXT("chara.bmp") );
```

４．プリミティブ描画前に、テクスチャを適用するようにデバイスに通知します。以下のプログラムを適切な場所に追加しましょう。

```
// テクスチャ設定
GraphicsDevice.SetTexture(0, texture->GetTexture());
```

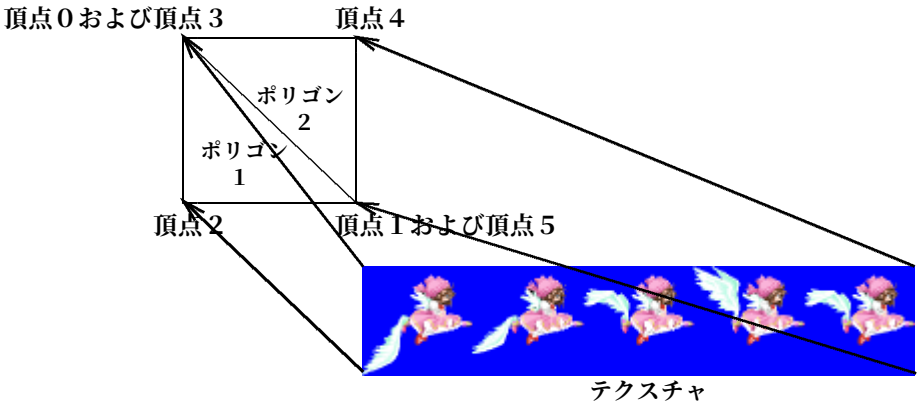
５．ＵＶ座標を頂点データに追加します。以下のプログラムを適切な場所に追加しましょう。

```
v[0].tu     = 0.0f;           // テクスチャU座標
v[0].tv     = 0.0f;           // テクスチャV座標

v[1].tu     = 1.0f;
v[1].tv     = 1.0f;

v[2].tu     = 0.0f;
v[2].tv     = 1.0f;

v[4].tu     = 1.0f;
v[4].tv     = 0.0f;
```

６．プログラムを実行し、ポリゴンにテクスチャが貼られていることを確認しましょう。

(３)ＵＶ座標を細かく設定し、キャラクター１コマだけ適用されるようにしましょう。

(２)の設定は、以下のよう画像全体を貼り付けるようになっています。



ＵＶ座標を設定にすることにより、任意の部分を適用できます。上記テクスチャは、横５コマ並んでいます。
Ｕ座標は、全体を１としたとき、１．０÷５＝０．２の間隔で指定すれば、任意のコマを取り出せる、ということになります。つまり、左は０．０，０．２，０．４，…、右は左＋０．２という計算で指定できると言うことです。
Ｖ座標は、縦には１コマしか並んでいないので、上が０．０，下が１．０になります。

(２)の５．で追加したプログラムを以下のように変更し、実行結果を確認しましょう。

```
v[0].tu = 0.0f;          // テクスチャU座標
v[0].tv = 0.0f;          // テクスチャV座標

v[1].tu = 0.2f;
v[1].tv = 1.0f;

v[2].tu = 0.0f;
v[2].tv = 1.0f;

v[4].tu = 0.2f;
v[4].tv = 0.0f;
```

(４)環境によっては、テクスチャが正しく貼り付けられない場合があります。ＵＶ座標を正しい値に訂正し、１コマ分の画像が貼り付けられるようにしましょう。

ヒント１：テクスチャの読み込み時、画像枠が２の乗数に変更される場合があります
　　幅６００ピクセル→幅１０２４ピクセル
ヒント２：画像枠が拡大された場合、それに従ってＵＶ座標も変更が必要です
　　本　来：幅　６００ピクセル、１コマ１２０ピクセル→Ｕ座標は０．２単位
　　拡大時：幅１０２４ピクセル、１コマ１２０ピクセル→Ｕ座標は０．１７１８７５…単位
ヒント３：上記のように割り切れない数になる場合があるので、テクスチャは２の乗数単位で作成した方が無難です

(５)Ｕ座標を毎フレーム変更し、アニメーションするようにしてみましょう。