

XNA Game Studio ゲームプログラミング

3D編 - 第7回 レンダリングステート

レンダリングステート

- ・レンダリングステートは、レンダリング時の動作を制御するさまざまな値のこと
- ・レンダリングステートは、GraphicsDevice.RenderStateクラスが管理している
- ・RenderStateクラスの対応するメンバの値を設定するだけで、レンダリング時の動作を変更できる

概要

レンダリングステートとは、レンダリング時の動作を制御するさまざまな設定値のことです。GraphicsDevice.RenderStateクラスの各メンバに値を設定することにより、レンダリングの動作を変更することができます。代表的なステートに、深度バッファ、カリング、アルファブレンド、プリミティブの描画モード、フォグなどがあります。

XNAのレンダリングステートは、一度設定すると変更されない限り維持され続けますが、SpriteBatch.Beginメソッドのように、内部でレンダリングステートを変更するメソッドがあります。このため、レンダリングステートがどのようになっているのかをしっかりと管理し、2D/3D/半透明など、それぞれにあったステートを設定してから描画を行う必要があります。

深度バッファ(Zバッファ)

深度バッファは、画面の各ピクセルの深度を保持する領域です。深度バッファを用いると、描画する順番にかかわらず、奥のものは描画されず、手前のものが描画されるようになります。

3Dグラフィックでは「奥にあるもの」より「手前にあるもの」が前に描画されなければなりません。奥にあるものは手前のものに隠される必要があり、2Dでは奥から描画するなど工夫して再現していました。3Dでも同様に、なんらかの方法でこの処理を行う必要があります。

XNAでは、デフォルトで自動的に奥のものが隠されて描画されるようになっています。これは、画面の色データ以外に、深度バッファという奥行きデータも保持しているため、より手前のピクセルしか描画されないようになっているからです。

ポリゴンのラスタライズ時に深度バッファが参照され、描画しようとしているピクセルよりも「遠い」ピクセルの場合は上書きされないようになっています。「近い」ピクセルの場合は上書きされ、深度バッファも更新されます。

スプライトを使うと、SpriteBatch.Beginメソッドが深度バッファを無効にしてしまいます。スプライトとモデルを混在して描画する場合は、モデルの描画前に、下記のように深度バッファを有効にする必要があります。

```
// 深度バッファを有効にする
GraphicsDevice.RenderState.DepthBufferEnable = true;
// 深度バッファへの書き込みを有効にする
GraphicsDevice.RenderState.DepthBufferWriteEnable = true;
```

カリング

カリングとは、ポリゴンの向きによって描画するかしないかを指定するものです。デフォルトでは、視点から見て「頂点が反時計回り」になっているポリゴンは、「裏を向いている」という扱いになり、描画されません。これは、裏を向いているポリゴンはモデルの背面 = 死角にあることが多いため、描画しない方がパフォーマンス向上につながるためです。

しかし、半透明モデルや一枚板などように、あえて裏面を描画しなければならないものもあります。このような場合は、"GraphicsDevice.RenderState.CullMode"の設定を行うことにより、カリングの動作を変更することができます。

```
// カリングモードの設定
GraphicsDevice.RenderState.CullMode = CullMode.None; // カリングなし
```

GraphicsDevice.RenderState.CullModeには、以下の値を設定できます。

CullMode.None	カリングしません。すべての向きが描画されます
CullMode.CullCounterClockwiseFace	反時計回りのポリゴンをカリングします
CullMode.CullClockwiseFace	時計回りのポリゴンをカリングします

なお、SpriteBatch.Beginメソッドは、カリングモードを"CullMode.CullCounterClockwiseFace"に設定します。また、デフォルトもこのモードです。

アルファブレンド

ブレンドとは、2つの色を混ぜて、それぞれの色を反映した新たな色を作成する処理です。このとき、ピクセルや頂点のアルファ値(透明度の値)をもとにブレンドを行うものをアルファブレンドと呼びます。アルファブレンドを行うと、奥のものが透けて見えるような効果を得ることができます。

アルファブレンドでは、「転送元の色」と「転送先の色」が混合されて最終的な色が計算されますが、どのように混ぜあわせるかを指定することができます。基本式は以下のようになっています。

(転送元の色 × 転送元ブレンド割合) (ブレンド演算) (転送先の色 × 転送先ブレンド割合)

XNAでは、「転送元ブレンド割合」をGraphicsDevice.RenderStateクラスのSourceBlendプロパティ、「転送先ブレンド割合」をDestinationBlendプロパティ、「ブレンド演算」をBlendFunctionプロパティで設定します。

一般的な半透明処理は、「転送元ブレンド割合」が転送元のアルファ値をそのまま使用、「ブレンド演算」が足し算、「転送先ブレンド割合」が1 - アルファ値(転送元と先のアルファ値を足すと1になるようになっています)を使用します。よって、

(転送元の色 ×) + (転送先の色 × (1 -))

という式になります。これは、以下のコードになります。

```
// アルファブレンドを有効にする
GraphicsDevice.RenderState.AlphaBlendEnable = true;

// アルファブレンディングの設定([転送元* ]+[転送先*(1- )])
GraphicsDevice.RenderState.BlendFunction = BlendFunction.Add; // 下の2つを足す
GraphicsDevice.RenderState.SourceBlend = Blend.SourceAlpha; // 値そのまま使用
GraphicsDevice.RenderState.DestinationBlend = Blend.InvSourceAlpha; // 1 - ( の反転)
...
(半透明モデルを描画する)
...
GraphicsDevice.RenderState.AlphaBlendEnable = false; // アルファブレンド無効
```

アルファブレンドは、デフォルトで無効になっているので、AlphaBlendEnableプロパティを有効にする必要があります。

レンダリングステートの設定を工夫することにより、「加算ブレンディング」「乗算ブレンディング」「減算ブレンディング」など、様々な効果を用いたブレンドが行えます。

```
// 加算ブレンディング
GraphicsDevice.RenderState.SourceBlend = Blend.One;
GraphicsDevice.RenderState.DestinationBlend = Blend.One;

// 乗算ブレンディング
GraphicsDevice.RenderState.SourceBlend = Blend.Zero;
GraphicsDevice.RenderState.DestinationBlend = Blend.SouceColor;
```

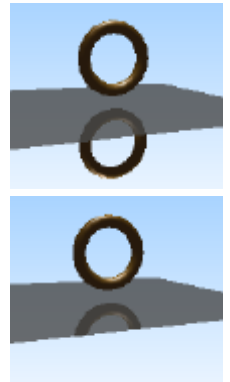
ステンシルバッファ・アルファテスト

ステンシルバッファは、モデルの影や映り込みを再現するときに有効な機能で、描画する領域を制限することができます。たとえばモデルの影(陰はマテリアルとライトで表現)は、影で暗くなる所をステンシルバッファによって描画できるようにしておけば、あとは半透明の黒を描画するだけで表現することができます。

アルファテストは、画面ピクセルのアルファ値によって、描画するかしないかを決めることができます。深度バッファの"深度"が"透明度"になったもので、深度ではなく透明度で描画するかどうかを決定したいときに使用します。多くの場合、深度バッファで不都合が出る場合にテクスチャと組み合わせで使用します。

たとえば、「細かい枝の隙間から後ろのオブジェクトが見える」というシーンの場合、細かい枝はポリゴン数がかかってしまうので、四角形ポリゴン+テクスチャで簡易的に再現します。このとき、後ろにあるオブジェクトは、深度バッファによって手前にある四角形ポリゴンに隠されるため、描画されません。このまま「細かい枝」のテクスチャを貼っても、奥のオブジェクトは描画自体されていないため、枝の隙間から見ることはできません。

このような場合、テクスチャに透明度情報(アルファ値)を設定しておき、深度ではなくアルファ値によってピクセルを描画するかどうかを判定させます。こうすれば、奥のオブジェクトも枝の隙間から見えるように描画することができます。



上ステンシルなし
下ステンシルあり



深度バッファだけでは正しく描画できない



アルファテスト+テクスチャで描画

// アルファテスト設定の例

```
GraphicsDevice.RenderState.AlphaTestEnable = true; // アルファテスト有効
GraphicsDevice.RenderState.AlphaFunction = CompareFunction.Greater; // 比較関数
GraphicsDevice.RenderState.ReferenceAlpha = 127; // とペアで使用。 値128以上で描画
GraphicsDevice.RenderState.DepthBufferEnable = false; // 深度バッファ無効
```

(モデルを描画する)

// 深度バッファを有効にする

```
GraphicsDevice.RenderState.AlphaTestEnable = false; // アルファテスト無効
GraphicsDevice.RenderState.DepthBufferEnable = true; // 深度バッファ有効
```

アルファテストはそのほかに、「複雑な形に画像を切り取る」という効果にも使えます。

アルファブレンドと名称は似ていますが、まったく別の動作です。アルファブレンドは設定された透明度によって、プリミティブ自体を半透明にブレンドするもので、アルファテストはラスタライズ時に画面に書き込まれるピクセルの透明度によって、そのピクセルを描画するかどうか判定するものです。

アルファテストを使うと円形に描画することもできる



課題

スプライトやモデルが正しく描画されるように、レンダリングステートを設定するプログラムを適切な場所に追加しましょう。