XNA Game Studio
ゲームプログラミング

○　３Ｄ編 - 第１回　プリミティブ　○

プリミティブ

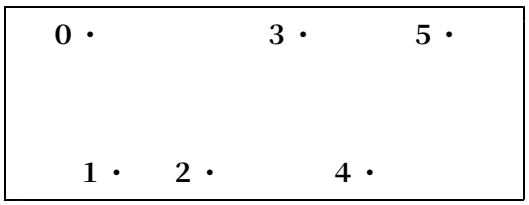
・プリミティブは、3Dオブジェクトを構成するための基本的な図形のこと
・XNAではプリミティブに「点」「線」「三角形」がある
・複雑なモデルもプリミティブが組み合わされているが、主に三角形が使われている

概要

プリミティブとは、3Dオブジェクトを構成するための基本的な図形のことです。複数の頂点をなんらかの方法でつなげることにより、プリミティブが構成され、さらにプリミティブを組み合わせることにより、3Dオブジェクトが形成されます。

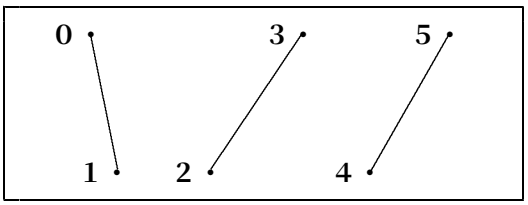
XNAでは、プリミティブとして「点」「線」「三角形」があり、以下の６つの方法で描画することができます(4.0以降は点リストと三角形ファンが削除されています)。よく使われているのがポリゴンと呼ばれるトライアングルリストです。

・ポイントリスト



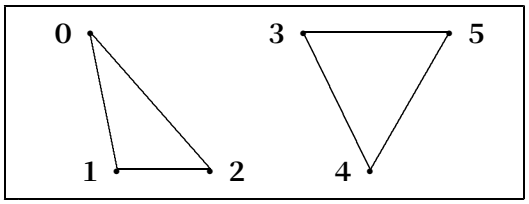
頂点を点で描画します

・ラインリスト



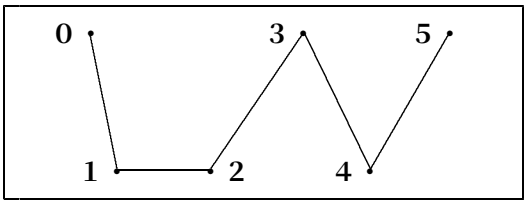
２つの頂点を別個の線分のリストとして描画します

・トライアングルリスト



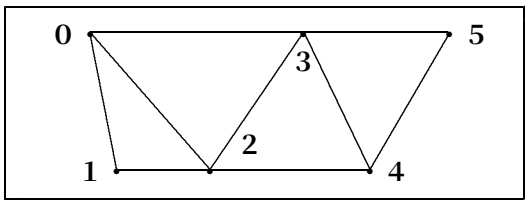
３つの頂点を別個の三角形のリストとして描画します

・ラインストリップ



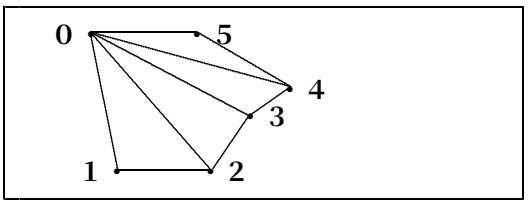
頂点を連続した線分のリストとして描画します

・トライアングルストリップ



頂点を連続した三角形として描画します

・トライアングルファン



頂点を三角形の扇型として描画します

頂点データの定義

・プリミティブを扱うには頂点データが必要
・頂点データは頂点バッファまたはメモリに作成する
・頂点データには「座標」「色」「テクスチャ座標」「法線データ」がある
・必要な属性だけを選択し、構造体にまとめる必要がある
・よく使う属性の組み合わせは、あらかじめ構造体が定義されている
・頂点は複数必要なので、構造体は配列によって宣言される
・グラフィックデバイスは、構造体にどのような属性が含まれているかわからないため、描画の前にVertexDeclarationクラスを用いて通知する必要がある

概要

プリミティブを表示するには頂点の情報が必要です。これを頂点データと呼びます。XNAでは、頂点データに含むことのできる属性として、頂点の「座標」「色」「テクスチャ座標」「法線データ」があります。これらの属性のうちどれが必要かをまとめた構造体を定義し、頂点データを設定します。

XNAには、よく使われる組み合わせがあらかじめ定義されています。

VertexPositionColor構造体	頂点の「座標」「色」を定義
VertexPositionColorTexture構造体	頂点の「座標」「色」「テクスチャ座標」を定義
VertexPositionNormalTexture構造体	頂点の「座標」「法線データ」「テクスチャ座標」を定義
VertexPositionTexture構造体	頂点の「座標」「テクスチャ座標」を定義

ポリゴンを扱うためには、最低でも３つの頂点が必要となるので、これらの構造体は配列で宣言されます。また、必要な属性だけをまとめたカスタム構造体を扱うこともできます。

グラフィックデバイスは、構造体にどのような属性が定義されているかわらないので、プリミティブの描画前に、VertexDeclarationクラスを用いて通知する必要があります。このクラスは複数定義してもかまわないので、3Dモデルやゲームのシーンによって、その都度切り替えることができます。

プリミティブの描画

・プリミティブの描画はGraphicsDeviceクラスのDrawPrimitives系のメソッドで行う
・データが頂点バッファにある場合はDrawPrimitives、メモリならDrawUserPrimitivesメソッド
・配列のインデックスを用いて重複する頂点を効率よく扱うDrawIndexedPrimitivesメソッドもある
・描画前に、エフェクトとパスの開始を宣言し、描画後にそれらの終了を宣言する必要がある
・エフェクト、パスの開始はBeginメソッド、終了はEndメソッド。この間に何度でもDrawPrimitives系のメソッドを呼びすことができる

概要

プリミティブの描画はGraphicsDeviceクラスのDrawPrimitives系のメソッドで行います。頂点データがどこに格納されているかで呼び出すメソッドが異なり、データが頂点バッファ(グラフィックデバイス上の頂点を保持するメモリ)にある場合はDrawPrimitivesメソッド、メモリにある場合はDrawUserPrimitivesメソッドです。

これらのメソッドを呼び出す前に、EffectクラスのBeginメソッドを呼び出し、エフェクトの開始を宣言する必要があります。その後、Effect.Techniques.PassesのBeginメソッドを呼び出し、パスの開始を宣言します。プリミティブの描画が終わったら、これらのEndメソッドを逆順に呼び出して、パスとエフェクトの終了を宣言します(4.0以降はApplyメソッドの呼び出しのみ必要)。

エフェクトとは、プリミティブをどのように画面に描画するのかを定義したものです。シェーダ(Shader)と呼ばれるプログラムが必要となります。シェーダとは、ライティング(光源)、シェーディング(陰影)、レンダリング(ピクセル化)をどのように行うかプログラムに記述し、グラフィックデバイスに知らせるものです。頂点をどのように座標変換し、画面に描画するのかをプログラムで自由に設計することができるようになっています。GPUに直接命令を出すので、高速・高度にハードウェアの性能を引き出し、プログラムを実行できます。NINTENDO64やPS3でも同じ考え方が導入されています。
XNAには、BasicEffectという代表的な機能をまとめたクラスがあります。基本的な描画に関してはシェーダプログラムを書かなくてもいいようなっています。

BasicEffectには以下の機能があります。

座標変換行列(World, View, Projection)
頂点カラー
ライティング
マテリアル(DiffuseColor, SpecularColor, EmissiveColor)
鏡面光(Specular)つき平行光源(Directional Light)
テクスチャ(シングルレイヤーのみ)
フォグ(視点からの距離によるもの)

これらの機能に関しては、シェーダを作成する必要なく、値の設定のみで扱うことができます。

さらに、エフェクトにはパスというものがあります。パスは、具体的にどのシェーダを使って描画するのかを指定するものです。１つのエフェクトに複数登録できるので、複数のシェーダを組み合わせて描画することもできます。BasicEffectクラスには基本的な描画を行うパスが１つ登録されています。

DrawPrimitivesメソッド(GraphicsDeviceクラス)

プリミティブの描画は、頂点データが頂点バッファにある場合はGraphicsDeviceクラスのDrawPrimitivesメソッド、メモリにある場合はDrawUserPrimitivesメソッドで行います。

試GraphicsDevice.DrawPrimitives(*PrimitiveType*, *int*, *int*);

PrimitiveType	プリミティブのタイプ
int	何番目の頂点から描画するかの指定
int	描画するプリミティブの数(頂点の数ではありません)

－説明－
頂点バッファの頂点を"PrimitiveType"で指定された形状で描画します。

－使用例－
頂点バッファの「先頭から100番目」の頂点を１つ、点として描画

```
GraphicsDevice.DrawPrimitives(PrimitiveType.PointList, 100, 1);
```

試GraphicsDevice.DrawUserPrimitives<T>(*PrimitiveType*, *T[]*, *int*, *int*);

<T>	頂点のデータ型
PrimitiveType	プリミティブのタイプ
T[]	頂点のデータが格納された配列
int	何番目の頂点から描画するかの指定
int	描画するプリミティブの数(頂点の数ではありません)

－説明－
メインメモリなどに格納された頂点を"PrimitiveType"で指定された形状で描画します。

－使用例－
頂点バッファの「先頭から100番目」の頂点を線として描画します(頂点は2つ描画されます)
※vertexは頂点データが格納された配列とします

```
GraphicsDevice.DrawUserPrimitives<VertexPositionColor>
                                    (PrimitiveType.LineList, vertex, 100, 1);
```

上記以外にも、共有されている頂点を効率よく扱うためのDrawIndexedPrimitives, DrawUsrIndexedPrimitivesメソッドがあります。

コーディング例

```
// フィールド変数
VertexDeclaration vrtxDec;
BasicEffect basicEffect;
VertexPositionColor[] vertex;
                    ⋮

protected override void Initialize()
{
    // TODO: Add your initialization logic here
```

```
        // BasicEffectの設定
        basicEffect = new BasicEffect(GraphicsDevice, null);
        basicEffect.VertexColorEnabled = true;
        basicEffect.View = Matrix.CreateLookAt(new Vector3(0.0f, 0.0f, 1.0f),
                                               Vector3.Zero, Vector3.Up);
        basicEffect.Projection =
            Matrix.CreateOrthographicOffCenter(0,
                                               (float)GraphicsDevice.Viewport.Width,
                                               (float)GraphicsDevice.Viewport.Height,
                                               0, 0.0f, 1.0f);

        // 頂点データの属性を宣言
        vrtxDec = new VertexDeclaration(GraphicsDevice, VertexPositionColor.VertexElements);

        // 頂点データを設定
        vertex = new VertexPositionColor[2];
        vertex[0] = new VertexPositionColor(new Vector3(300.0f, 100.0f, 0.0f),
                                            new Color(255, 255, 255, 255));
        vertex[1] = new VertexPositionColor(new Vector3(300.0f, 250.0f, 0.0f),
                                            new Color(255, 255, 255, 255));
                            ⋮

    protected override void Draw(GameTime gameTime)
    {
        graphics.GraphicsDevice.Clear(Color.CornflowerBlue);
                            ⋮
        // 描画する頂点データの属性をデバイスに知らせる
        GraphicsDevice.VertexDeclaration = vrtxDec;

        // 3D描画開始
        basicEffect.Begin();

        // パスの数だけ繰り替えし描画(BasicEffectは通常１回)
        foreach (EffectPass pass in basicEffect.CurrentTechnique.Passes)
        {
            pass.Begin();   // パス開始

            GraphicsDevice.DrawUserPrimitives<VertexPositionColor>
                                (PrimitiveType.LineList, vertex, 0, 1);

            pass.End(); // パス終了
        }

        basicEffect.End();
                            ⋮
```

頂点バッファ

- 頂点バッファはグラフィックデバイス上のメモリに作成されるため、非常に高速
- VertexBufferクラスが頂点バッファを管理
- VertexBufferクラスのSetDataメソッドで頂点データを頂点バッファに設定
- GraphicsDeviceクラスのSetSourceメソッドで頂点バッファをグラフィックデバイスに関連づける
  (3.1まで。4.0以降はSetVertexBufferメソッド)

概要

頂点バッファ(Vertex Buffer)は、グラフィックデバイスが直接扱うことのできる、頂点データを格納する領域です。ビデオメモリ(VRAM)やAGPメモリに作成されるので、メインメモリを用いる配列に比べ、レンダリング速度の向上が望めます。

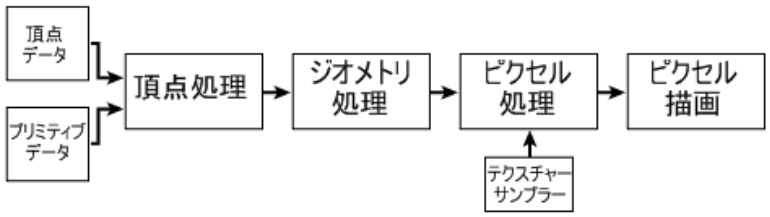
XNAには、頂点バッファを扱うためのVertexBufferクラスがあります。このクラスに頂点データを設定すれば、ただちにグラフィックデバイス上のメモリに反映されますが、直接メンバにアクセスすることはできません。よって、一度配列に頂点データを作成し、それをVertexBufferクラスSetDataメソッドで転送します。転送したデータはGetDataメソッドで取得することができます。

ただし、このままでは「グラフィックデバイス上のメモリに頂点データがある」だけの状態です。描画の前に、GraphicsDevice.VerticesのSetSourceメソッドでデバイスと頂点バッファを関連づける必要があります。

頂点データのレンダリング

頂点データは、物体や図形に関する情報を数値にして配列に並べているだけです。これを、計算によって画像にする必要があります。3Dグラフィクスでは、視点、光源の種類・位置・数、モデルの形状(ポリゴンの向き)を考慮して面の消去や陰影付けなどを行って画像を計算します。

この一連の流れをレンダリングと呼びます。XNAで使われているDirectX9(DirectX Graphics9)では、頂点データを以下の流れで画像にしています。



頂点データは、まず頂点処理が行われます。頂点処理では、Effectクラスのワールド行列、ビュー行列、射影行列といった座標変換行列により、視点から見た座標に変換されます。次にジオメトリ処理により、クリッピング(画面外ポリゴンの削除)、背面カリング(背面ポリゴンの削除)、ラスター化(ポリゴンにピクセルを割り当てる処理)が行われます。
ラスター化されたピクセルは、色がついていません。ピクセル処理では、Effectクラスをもとに、頂点カラー、テクスチャ、ライティングといったカラー処理が施され、ポリゴンに割り当てられたピクセルの色が計算されます。最後にピクセル描画が行われ、透明度やｚバッファを適用し、最終的なピクセルカラー値を設定します。

コーディング例

```
// フィールド変数
VertexDeclaration vrtxDec;
BasicEffect basicEffect;
VertexBuffer vertexBuf;
                        ⋮

protected override void Initialize()
{
    // TODO: Add your initialization logic here
    // BasicEffectの設定
    basicEffect = new BasicEffect(GraphicsDevice, null);
    basicEffect.VertexColorEnabled = true;
    basicEffect.View = Matrix.CreateLookAt(new Vector3(0.0f, 0.0f, 1.0f),
                                           Vector3.Zero, Vector3.Up);
    basicEffect.Projection =
        Matrix.CreateOrthographicOffCenter(0,
                                           (float)GraphicsDevice.Viewport.Width,
                                           (float)GraphicsDevice.Viewport.Height,
                                           0, 0.0f, 1.0f);

    // 頂点データの属性を宣言
    vrtxDec = new VertexDeclaration(GraphicsDevice, VertexPositionColor.VertexElements);
```

```
        // 頂点データの設定
        VertexPositionColor[] vrtx = new VertexPositionColor[3];
        vrtx[0] = new VertexPositionColor(new Vector3(640.0f, 240.0f, 0.5f),
                                          new Color(255, 0, 0, 255));
        vrtx[1] = new VertexPositionColor(new Vector3(800.0f, 480.0f, 0.5f),
                                          new Color(0, 255, 0, 128));
        vrtx[2] = new VertexPositionColor(new Vector3(480.0f, 480.0f, 0.5f),
                                          new Color(0, 0, 255, 64));

        // 頂点バッファを作成し、頂点データを転送
        vertexBuf = new VertexBuffer(GraphicsDevice,
                                     typeof(VertexPositionColor), vrtx.Length,
                                     BufferUsage.None);
        vertexBuf.SetData<VertexPositionColor>(vrtx);
                              ⋮

    protected override void Draw(GameTime gameTime)
    {
        graphics.GraphicsDevice.Clear(Color.CornflowerBlue);
                              ⋮
        // 描画する頂点データの属性をデバイスに知らせる
        GraphicsDevice.VertexDeclaration = vrtxDec;

        // 描画する頂点バッファをグラフィックデバイスに設定
        GraphicsDevice.Vertices[0].SetSource(vertexBuf, 0, VertexPositionColor.SizeInBytes);

        // 3D描画
        basicEffect.Begin();                                    // エフェクト開始
        basicEffect.CurrentTechnique.Passes[0].Begin();         // パス開始

        GraphicsDevice.DrawPrimitives(PrimitiveType.TriangleList, 0, 1);

        basicEffect.CurrentTechnique.Passes[0].Begin();         // パス終了
        basicEffect.End();                                      // エフェクト終了
                              ⋮
```

課　題

頂点バッファを使って頂点データを定義し、三角形を２つ組み合わせて画面に正方形を描画しましょう。
※時計回りが表ポリゴンとなります。反時計回りは背面ポリゴンとなり、そのままでは描画されません