ESライブラリ＆＆
ゲームプログラミング

○　３Ｄ編 - 第１２回　プリミティブ　○

プリミティブ

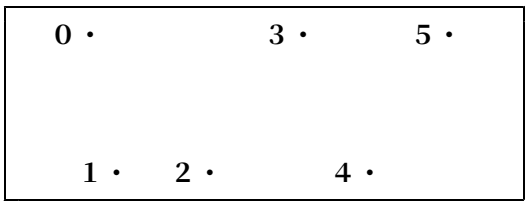
・プリミティブは、3Dオブジェクトを構成するための基本的な図形のこと
・プリミティブには、「点」「線」「三角形」がある
・複雑なモデルもプリミティブが組み合わされているが、主に三角形が使われている

概要

プリミティブとは、3Dの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のオブジェクトが形成されます。

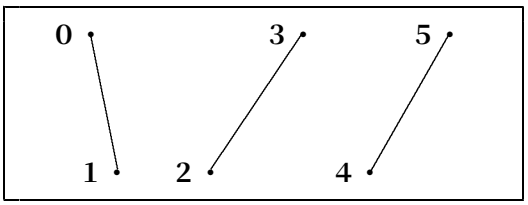
Direct3Dでは、プリミティブとして「点」「線」「三角形」があり、以下の６つの方法で描画することができます。よく使われているのがポリゴンと呼ばれるトライアングル～です。

・ポイントリスト



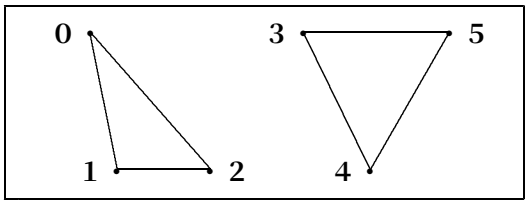
頂点を点で描画します

・ラインリスト



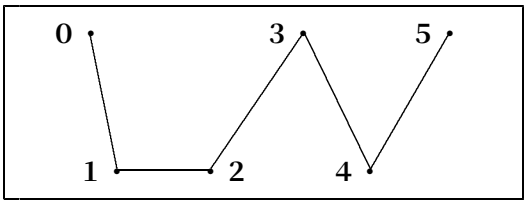
２つの頂点を別個の線分のリストとして描画します

・トライアングルリスト



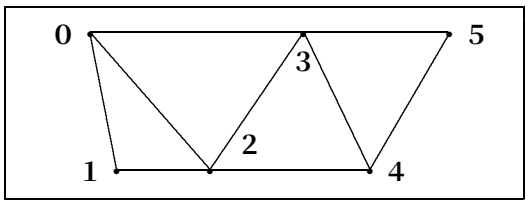
３つの頂点を別個の三角形のリストとして描画します

・ラインストリップ



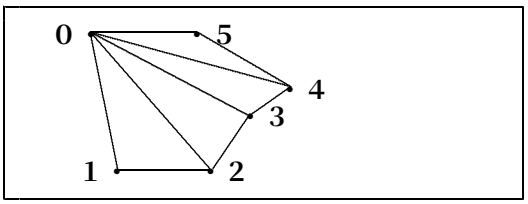
頂点を連続した線分のリストとして描画します

・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
・よく使う頂点データには「座標」「色」「テクスチャ座標」「法線」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
・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

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。Direct3D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」などがあります。これら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。

柔軟な頂点フォーマット(FVF)

頂点を描画するには、頂点の座標や色といった情報(頂点データ)が必要になります。DirectX Graphicsでは、プログラマが頂点データのフォーマットを定義することになっています。

任意のフォーマットを定義できることから、柔軟な頂点フォーマット(FVF:Flexible Vertex Format)と呼びます。柔軟な頂点フォーマットに含めることのできるおもな情報は、以下のものです。

x，y，z座標	法線ベクトル	ディフューズ色	スペキュラ色	テクスチャのＵＶ値
RHW	頂点ブレンディングの重み		ポイントサイズ	

これらを組み合わせて頂点データのフォーマットを定義します。ただし、RHWと法線ベクトルのように、同時に使用できないデータもあります。また、格納する順序は、以下のように決められています。

座標(float×３)
RHW(float)
頂点ブレンディングの 重み(float×１～３)
頂点の法線ベクトル (float×３)
頂点のポイントサイズ (float×３)
ディフューズ色(D3DCOLOR)
スペキュラ色(D3DCOLOR)
テクスチャのＵＶ値１ (float×１～４) ⋮ テクスチャのＵＶ値８ (float×１～４)

頂点データは構造体で定義します。頂点データ構造体の配列を後述のIDirect3DDevice9::DrawPrimitive系のメソッドに渡せば、プリミティブが描画されます。なお、このままでは、どのようなフォーマットなのかDirectX Graphicsは認識できないので、事前にIDirect3DDevice9::SetFVFメソッドで知らせておきます。

SetFVFメソッドに渡す値は、以下のフラグの組み合わせになります。

フラグ	内　　容
D3DFVF_XYZ	未座標変換の頂点座標(モデル座標系の頂点)
D3DFVF_XYZRHW	座標変換済みの頂点座標(スクリーン座標系の頂点)
D3DFVF_XYZB1 ～D3DFVF_XYZB5	頂点ブレンディングの重み
D3DFVF_NORMAL	頂点の法線ベクトル
D3DFVF_PSIZE	頂点のポイントサイズ
D3DFVF_DIFFUSE	頂点のディフューズ色
D3DFVF_SPECULAR	頂点のスペキュラ色
D3DFVF_TEX0 ～D3DFVF_TEX8	テクスチャのＵＶ値(D3DFVF_TEX0は指定無しと同じ) マルチテクスチャのため、１～８まで用意されている

よく使われるのが、「トランスフォーム済みライティング済みの頂点」「未トランスフォームライティング済みの頂点」「未トランスフォーム未ライティングの頂点」です。それぞれ、以下のように構造体とフラグを定義します。

・トランスフォーム済みライティング済みの頂点(色付きのスクリーン座標の頂点)
スクリーン座標と色を指定して頂点を描画することができます。画面上の４点を結んで四角形ポリゴンを作り、テクスチャを貼ることによって2Dキャラクタを描画することができます。

```
struct TLVERTEX {
    float      x, y, z, rhw;    // 座標変換済みの頂点座標
    D3DCOLOR   color;           // 頂点の色
    float      tu, tv;          // テクスチャのＵＶ値
};

// FVF設定(pD3DDeviceは、初期化済みのDirect3DDevice9オブジェクトのインタフェース)
pD3DDevice->SetFVF(D3DFVF_XYZRHW | D3DFVF_DIFFUSE | D3DFVF_TEX1);
```

・未トランスフォームライティング済みの頂点(色付きのモデル座標の頂点)
3Dオブジェクトの原点がもとになる座標系の頂点です。座標変換はDirectX Graphicsが行い、頂点の色は自ら計算するときに使用します。

```
struct LVERTEX {
    float      x, y, z;         // 頂点座標
    D3DCOLOR   color;           // 頂点の色
    float      tu, tv;          // テクスチャのＵＶ値
};

// FVF設定
pD3DDevice->SetFVF(D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_TEX1);
```

・未トランスフォーム未ライティングの頂点(法線方向ベクトルを持つモデル座標)
3Dオブジェクトの原点がもとになる座標系の頂点です。座標変換も照明演算もDirectX Graphicsに行わせるときに使用します。頂点の色は、ライト、マテリアル、法線の向きによって計算されます。

```
struct VERTEX {
    float   x, y, z;            // 頂点座標
    float   nx, ny, nz;         // 法線ベクトル
    float   tu, tv;             // テクスチャのＵＶ値
};

// FVF設定
pD3DDevice->SetFVF(D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_TEX1);
```

頂点バッファ

頂点バッファ(Vertex Buffer)は、DirectX Graphics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。頂点バッファに格納された頂点は、DirectX Graphicsのすべての機能を使用してレンダリングすることができます。また、頂点バッファはビデオメモリ(VRAM)やAGPメモリに作成することができるので、システムメモリを用いる頂点データの配列に比べ、レンダリング速度の向上が望めます。

ただし、CPUとGPUの転送は時間がかかり、また頂点バッファの書き込みにはロックが必要となります。ロック中はジオメトリパイプラインが停止してしまうので、頂点バッファに作成したデータの書き換えを頻繁に行う場合は、書き込み専用の頂点バッファにするか、システムメモリにデータを置き、DrawPrimitiveUPメソッドでレンダリングした方が高速となる場合があります。

ポリゴン

ポリゴン(polygon)とは、3Dグラフィクスで立体を表現するときに使用する多角形のことです。計算のしやすさから、たいてい三角形が使われます。

コンピュータで立体図形を扱う場合、物体表面を三角形のポリゴンに分割してデータ化します。ポリゴンの数を増やせば増やすほど表現が精細になりますが、計算量が増えるため、描画に時間がかかるようになります。

１秒間に処理できるポリゴンの数がビデオカードやゲーム機の性能の指標として使われることがあります。PlayStation2は7,500万、PlayStation3やXbox 360では数億以上のポリゴンを１秒間に描画することができます。

ESライブラリの対応

ESライブラリでは、頂点バッファクラス(CVertexBuffer)によって、頂点バッファを扱いやすくしています。システムメモリの頂点データも描画できるように、GraphicsDevice.DrawUserPrimitives関数を用意しています。さらに、ワールド座標へ変換できるように、GraphicsDevice.SetWorldMatrix関数、SetMaterial関数、SetTexture関数もあります。

頂点データの格納領域として、前述のTLVERTEX, LVERTEX, VERTEX, ScreenVertex構造体が定義済みとなっています。

課　　題

プリミティブを画面に描画してみましょう。

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
ScreenVertex v[6];

// 頂点０
v[0].x     = 310.0f;                    // ｘ座標
v[0].y     =   3.0f;                    // ｙ座標
v[0].color = Color(1.0f, 0.0f, 0.0f);   // 色(左から赤、緑、青。0.0から1.0の範囲)
```

２．プリミティブを描画する準備ができたので、描画してみましょう。

以下のプログラムを１．の下に追加しましょう。

```
// プリミティブ描画
GraphicsDevice.DrawUserPrimitives(PrimitiveType_PointList, v, 1, v[0].FVF());
```

３．点の大きさを変えることもできます。以下のプログラムを２．の前に追加してみましょう。

```
GraphicsDevice.SetFloatRenderState(PointSize, 3.0f);
```

４．頂点を１つ増やしてみましょう。以下のプログラムを１．のあとに追加しましょう。

```
// 頂点１
v[1].x     = 87.0f;                      // ｘ座標
v[1].y     = 67.0f;                      // ｙ座標
v[1].color = Color(0.0f, 1.0f, 0.0f);    // 色
```

５．描画をするときに、２つ描画するようにします。２．のプログラムを以下のように変更しましょう。

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

６．２つの点を結び、線として描画することができます。２．のプログラムを以下のように変更しましょう。

```
// プリミティブ描画(線を１つ描画)
GraphicsDevice.DrawUserPrimitives(PrimitiveType_LineList, v, 1, v[0].FVF());
```

７．三角形を描画してみましょう。

頂点を３つ指定すると、それらが結ばれて三角形を形成できます。頂点は２つ設定されているので、あと１つ追加すると三角形にできます。頂点２として、ｘ座標5.5、ｙ座標0.5に設定してみます。
以下のプログラムを完成させ、適切な場所に追加しましょう。

```
// 頂点２
v[2].x     = ※ここを考えてください※;  // ｘ座標
v[2].y     = ※ここを考えてください※;  // ｙ座標
v[2].color = Color(0.0f, 0.0f, 1.0f);   // 色
```

８．３点を結んで三角形を描画してみましょう。

三角形では、線が３本必要になります。以下のようにプログラムを変更し、線が３本レンダリングされるようにしましょう。

```
// プリミティブ描画
GraphicsDevice.DrawUserPrimitives(PrimitiveType_LineList, v, 3, v[0].FVF());
```

９．描画方法を以下のように変更しましょう。

線を３本レンダリングするとき、ラインリストの場合は、２つの頂点を開始点と終了点のペアで扱うので、頂点は６つ必要になります。以下のようにレンダリングの方法を変更し、ラインストリップにしてみましょう。

```
// プリミティブ描画
GraphicsDevice.DrawUserPrimitives(PrimitiveType_LineStrip, v, 3, v[0].FVF());
```

10. 描画方法を以下のように変更しましょう。

線をレンダリングするとき、ラインストリップの場合は「レンダリングする線の数＋１」の頂点が必要になります。以下のプログラムを適切な場所に追加し、頂点を１つ増やしましょう。

```
// 頂点３
v[3] = v[0];
```

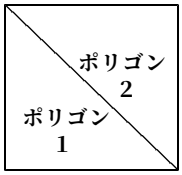
11. レンダリング方法をラインストリップから三角形リストに変更してみます。以下のようにレンダリングの方法を変更しましょう。

```
GraphicsDevice.DrawUserPrimitives(PrimitiveType_TriangleList, v, 1, v[0].FVF());
```

三角形リストの場合は、３つの頂点が結ばれて１つの三角形になります。

12. 以下のように頂点を定義し、三角形を２つ合わせて四角形を描画しましょう。

頂点０および頂点３(260.0，180.0)　　頂点４(380.0，180.0)



頂点２(260.0，300.0)　　頂点１および頂点５(380.0，300.0)