オブジェクト指向と
ゲームプログラミング

▶ DirectX Graphics編 － 第１回　DirectX Graphics ◀

DirectX Graphics

DirectX Graphicsは、3Dグラフィックカードを直接的に制御し、高速で高品質なグラフィックスを描画するためのインタフェースを提供します。主な機能として以下のものがあります。

フルスクリーンモードとウィンドウモードのサポート
ディスプレイを占有する「フルスクリーンモード」と通常のアプリケーションのようにウィンドウ内で動作する「ウィンドウモード」をサポートします。２つのモードは動作の異なる部分がありますが、その違いの多くをDirectX Graphicsが吸収します。また、２つのモードを動的に切り替えることがDirectDrawに比べ、簡潔に実現できます。

ダブルバッファリングによる描画
ちらつきのない描画をするために必要となるダブルバッファリングおよびトリプルバッファリング(現在の画像を表示するための領域と、次の画像を準備するための作業領域を使用する方法)による描画サポートします。また、これらの切り替えをディスプレイに同期して行うことができます。

画像の高速転送とデバイスコンテキストのサポート
GDIのBitBlt関数やStretchBlt関数よりも高速に画像を転送できます。このとき、拡大縮小を行うことができます。また、デバイスコンテキストもサポートしているので、従来のGDIを用いた描画も行えます。

画像領域へのポインタの取得
画像を保持している領域のアドレス(ポインタ)を取得することができます。画像を直接書き換えることにより、高度な画像効果を作成することができます。

ガンマコントロール
ディスプレイの色を構成する赤、緑、青それぞれについて明るさの設定を行えます。この機能を使えば、赤のみを強くしたり、画面全体を暗くしたりすることが簡単に行えます(フルスクリーンのみ)。

高速なポリゴン描画
ハードウェアの違いをあまり意識することなく、高速なポリゴン(プリミティブ)描画を行えます。環境による機能の違いをDirectX Graphicsが吸収するので、それぞれのハードウェアが持つ機能を最大限に生かすことができます。

ジオメトリと照明
3D空間における座標の移動、回転、射影といったジオメトリ(幾何学)演算および照明の演算をハードウェアで処理できます(ハードウェアによるTransformation[座標変換] & Lighting[照明])。

マルチテクスチャ
複数のテクスチャを１つのオブジェクトに対して使用することができます。

ミップマップ
ミップマップとは、同一イメージが徐々に低解像度になっていく一連のテクスチャのことで、1 つのイメージに何種類ものフィルタをかけることにより作成されます。視点が近いときには高解像度のイメージが使用され、遠ざかっていくと(画面での表示が小さくなるに従って)低解像度のイメージが使用されます。

バンプマップ
ポリゴンの組み合わせでは表現できない微妙な凹凸を表現することができます。深さの変化がテクスチャ内に格納されており、テクスチャブレンディング技術を使って適用することができます。

バイリニアフィルタ
テクスチャが拡大縮小されるときに、周囲のピクセルを組み合わせてなめらかに表示します。

アンチエイリアシング
ポリゴンのエッジ部分をなめらかに表示することができます。

ｚバッファ、ｗバッファ
　ｚおよびｗバッファは、ピクセルの前後判断に使用されるバッファです。画面のピクセルと１対１で対応しており、深度値の小さいピクセルは深度値の大きいピクセルを上書きして描画されます。

ステンシルバッファ
　ステンシルバッファは、ピクセル単位で描画を有効または無効にします。このバッファを使うと、レンダリングした画像の一部をマスクで覆うことが可能になり、その画像が表示されなくなります。オブジェクトの影などが簡単に表現できます。

アルファブレンド
　アルファは透明度のことで、ブレンディングはピクセルの合成方法のことです。加算、減算、乗算、反転などの演算を用いた合成を行い、多様なエフェクトを再現することができます。

テクスチャステージ
　複数のテクスチャの組み合わせ方を設定し、簡単に切り替えることができます。

バーテックスシェーダ
　入力された頂点を射影座標に変換する処理をアセンブリ言語やＣ言語風のシェーダ言語によりプログラムし、動作を自由に設定することができます。

ピクセルシェーダ
　レンダリング画像のピクセル描画処理をアセンブリ言語やＣ言語風のシェーダ言語によりプログラムし、動作を自由に設定することができます。

マルチエレメントテクスチャ
　ピクセルシェーダからテクスチャの複数の要素を同時に書き出すことができます。

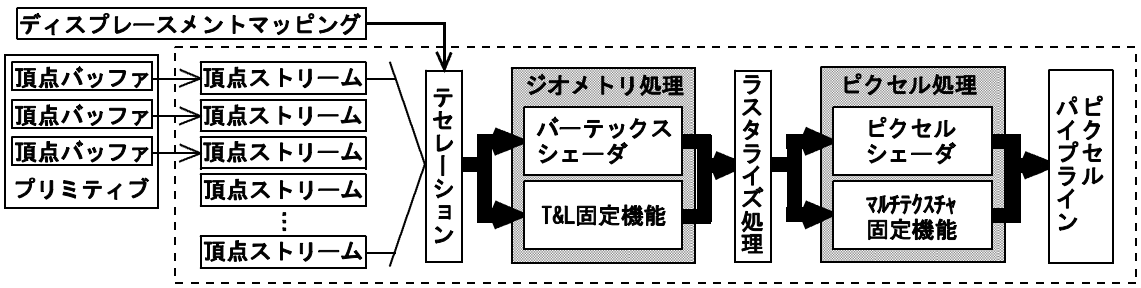
浮動小数点サーフェイス
　ピクセルのデータを浮動小数点で保持する浮動小数点サーフェスがサポートされます。

複数のレンダリングターゲット
　複数のレンダリングターゲット(3Dオブジェクトの描画先)を設定することができます。

※DirectX Graphicsを使ったプログラムをビルドする場合は、ヘッダファイル<d3dx9.h>とライブラリ"d3d9.lib", "d3dx9.lib"が必要になります。

DirectX Graphicsのアーキテクチャ

DirectX Graphicsの描画処理は、以下のような描画パイプラインで行われます。



▲描画パイプライン

①プリミティブ
　プリミティブとは、3Dオブジェクトの表面の形(頂点の集まり)のことです。3Dグラフィックスを描画するには、まず、描画パイプラインに3Dオブジェクトのプリミティブを渡します。プリミティブには、頂点間が線形補完される基本的なプリミティブと、高次補間されて曲面を表現できる「高次プリミティブ(パッチ)」があります。
　DirectX Graphicsの高次プリミティブには、「Nパッチ」と「Rect/Tri(RT)パッチ」があり、RTパッチでは矩形プリミティブがサポートされます。

②頂点バッファ
　プリミティブを描画するには、頂点データを描画パイプラインに渡します。頂点データは、「頂点データの配列」または「頂点バッファ」に格納します。頂点の成分には、プリミティブの表現に必要な「座標」「色」「法線」「テクスチャ座標」などが含まれます。

③頂点ストリーム
　頂点バッファは、頂点ストリームに結合することで描画に使えるようになります。頂点ストリームは最大１６個用意できます。１つの頂点は、複数の頂点バッファを組み合わせて作成できます。
　たとえば、成分が「座標のみ」「色のみ」「テクスチャ座標のみ」の３つの頂点バッファを用意します。ここで、テクスチャが不要なときは「座標のみ」と「色のみ」の頂点バッファをストリームに設定して描画を行います。テクスチャが必要で色が不要なときは「座標のみ」と「テクスチャ座標のみ」をストリームに設定して描画します。こうすると余分なデータの処理や転送を省くことができます。
　また、複数のモーションをモーフィングするときにも使用できます。複数のストリームにそれぞれ異なるモーションを登録しておき、それらを合成した中間モーションを出力することができます。

④テセレーション
　テセレーションは、曲面を三角形に分割する処理です。分割の細かさを「テセレーションレベル」と言います。頂点の深度に応じてレベルが変化する処理を「適応型テセレーション」と言います。高次プリミティブは、このテセレーション処理によって、適当な三角形に分割されます。

⑤ディスプレースメントマッピング
　DirectX Graphicsでは、テセレーションでNパッチを分割する際に、ディスプレースメントマッピングを行うことができます。ディスプレースメントマッピングとは、頂点の座標を変化させてプリミティブ表面の凹凸を再現することです。バンプマッピングと似ていますが、ディスプレースメントマッピングでは、テセレーションで分割された三角形の頂点座標を変化させることにより、プリミティブの表面を実際に凹凸させます。これに対しバンプマッピングでは、法線ベクトルを変化させることでプリミティブ上に擬似的な凹凸をテクスチャとして描画します。

⑥ジオメトリ処理
　ジオメトリ処理を行うジオメトリパイプラインでは、これまでの処理を通過してきた頂点データに対して、座標変換や照明などの処理を行い、ラスタライズ可能な頂点データ(実際に描画できるデータ)を作ります。
　DirectX Graphicsのジオメトリパイプラインには、「T&L固定機能パイプライン」と「プログラマブルなバーテックスシェーダ」があり、どちらかを選んで使います。固定機能パイプラインは、動作をステートの設定によって制御します。これに対し、プログラマブルなバーテックスシェーダでは、必要な処理を簡単なコードで直接的に記述します。

⑦ラスタライズ処理
　ラスタライズ処理は、三角形をピクセル単位に分割する処理です。このピクセル単位に分割されたものを「フラグメント」といいます。ここではラスタライズの前に「背面カリング」と「クリッピング」処理も行われます。背面カリングは、後ろ向きのポリゴンは見えないものとして排除する処理です。クリッピングは、領域内に収まるようにポリゴンを切り取る処理です。

⑧ピクセル処理
　ラスタライズ処理から出力されたフラグメントに対して処理を行い、色などを計算します。ジオメトリパイプラインと同様に、「マルチテクスチャ固定機能パイプライン」と「プログラマブルなピクセルシェーダ」があり、どちらかを選んで使います。固定機能パイプラインは、動作をステートの設定によって制御します。これに対し、プログラマブルなピクセルシェーダでは、必要な処理を簡単なコードで直接的に記述します。

⑨ピクセルパイプライン
　ピクセル処理が終わったら、ピクセルパイプラインが実行されます。ここでは、以下のような処理が順番に行われます。

(１)シザーテスト
　ビューポートのシザー枠内だけを描画します。デフォルトのシザー枠は、ビューポート全体です。

(２)ｚテスト、ステンシルテスト
　深度値(ｚ値)、ステンシル値の順に比較を行います。

(３)スペキュラ追加
　スペキュラを追加します。

(４)フォグ
　フォグの処理を行います。ピクセルシェーダVer3.0では、フォグの最終処理をピクセルシェーダで行うので、この処理は無効になります。

(５)アルファブレンド
　アルファブレンド処理を行います。

(６)フレームバッファに出力
　これまでの結果をフレームバッファ(バックバッファ)に書き込みます。

課　　題

DirectX Graphicsをカプセル化したクラスCDXGraphics9を作成しましょう。

(１)CDXGraphics9クラスの定義を行います。クラスのヘッダファイル(DXGraphics9.hpp)を以下のように作成しましょう。

－DXGraphics9.hpp－

```
/*
===========================================================================================
                                オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
===========================================================================================

  【対象ＯＳ】
    Microsoft Windows2000/XP

  【コンパイラ】
    Microsoft Visual C++ 2005

  【プログラム】
    DXGraphics9.hpp
              DirectX Graphics9クラスヘッダ

  【履歴】
    * Version    1.00       2005/03/dd  hh:mm:ss

===========================================================================================
*/

#pragma once

/*****************************************************************************************/
/*                                    インクルードファイル                                 */
/*****************************************************************************************/

/*****************************************************************************************/
/*                                 DirectX Graphics9クラス定義                             */
/*****************************************************************************************/
class CDXGraphics9 {
public:

private:

};
```

(２)「インクルードファイル」にDirectX Graphicsで必要となるファイルを追加しましょう。

(３)CDXGraphics9クラスのソースファイル(DXGraphics9.cpp)を以下のように作成しましょう。

－DXGraphics9.cpp－

```
/*
===========================================================================================
                                オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
===========================================================================================
```

```
【対象ＯＳ】
   Microsoft Windows2000/XP

【コンパイラ】
   Microsoft Visual C++ 2005

【プログラム】
   DXGraphics9.cpp
            DirectX Graphics9クラス

【履歴】
   * Version    1.00       2005/03/dd  hh:mm:ss

================================================================================================
*/

/*********************************************************************************************/
/*                                      インクルードファイル                                 */
/*********************************************************************************************/

/*********************************************************************************************/
/*                                      静的リンクライブラリ                                 */
/*********************************************************************************************/
```

(４)「インクルードファイル」に、このソースファイルをコンパイルするのに必要となるファイルを追加しましょう。

(５)「静的リンクライブラリ」に、このソースファイルをリンクするときに必要となるファイルを２つ追加しましょう。

(６)以下のプログラムをprivateに追加しましょう。これにより、オブジェクトのコピーができなくなります。

```
    // コピーできないようにする
    CDXGraphics9(const CDXGraphics9&);              // コピーコンストラクタ
    CDXGraphics9& operator=(const CDXGraphics9&);   // 代入演算子
```

コピーコンストラクタと代入演算子の関数本体が定義されていないので、CDXGraphics9クラスのオブジェクトを=演算子などでコピーしようとすると、コンパイル時にエラーになります。

(７)コンストラクタとデストラクタを作成します。以下のプロトタイプをpublicに追加しましょう。

```
    CDXGraphics9();         // コンストラクタ
    ~CDXGraphics9();        // デストラクタ
```

(８)コンストラクタとデストラクタを実装します。以下のプログラムをソースファイルに追加しましょう。

```
/*********************************************************************************************/
/*                                      コンストラクタ                                       */
/*********************************************************************************************/
CDXGraphics9::CDXGraphics9()
{
}

/*********************************************************************************************/
/*                                      デストラクタ                                         */
/*********************************************************************************************/
CDXGraphics9::~CDXGraphics9()
{
}
```

(９)CDXGraphics9クラスが継承されることも考えられるので、デストラクタを仮想デストラクタに変更しましょう。

(10)CDXGraphics9クラスをシングルトン化しましょう。

CDXGraphics9クラスのインスタンスは、１つのアプリケーションにつき１つあればよく、複数のインスタンスが存在すると資源の奪い合いなどで不具合が起こると考えられます。
以下の手順に従ってシングルトン化し、インスタンスの生成を制限しましょう。

①コンストラクタをpublicからprivate(継承する場合はprotected)に変更します。

②唯一のインスタンスを取得するGetInstance関数を以下のように作成し、publicに追加します。

```
// 唯一のインスタンスの取得
static CDXGraphics9& GetInstance()
{
    static CDXGraphics9 theDXGraphics;
    return theDXGraphics;
}
```

③短い名前でインスタンスを取得できるように、以下のインライン関数をヘッダファイルに追加します。

```
/*****************************************************************************************************/
/*                                          インライン関数                                           */
/*****************************************************************************************************/
inline CDXGraphics9& DXGraphics() { return CDXGraphics9::GetInstance(); }
```

インライン関数DXGraphics()を呼び出すと、CDXGraphics9クラスの外部でも、各メンバにアクセスすることができます。

(11)DirectXをカプセル化したクラスのヘッダファイルをまとめてインクルードする"DirectX.h"を以下のように作成しましょう。

－DirectX.h－

```
/*
=====================================================================================================
                                オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
=====================================================================================================

【対象ＯＳ】
   Microsoft Windows2000/XP

【コンパイラ】
   Microsoft Visual C++ 2005

【プログラム】
   DirectX.h
            DirectXヘッダ

【履歴】
   * Version    1.00      2005/03/dd  hh:mm:ss

=====================================================================================================
*/

#pragma once

/*****************************************************************************************************/
/*                                          インクルードファイル                                     */
/*****************************************************************************************************/
#include "DXGraphics9.hpp"
```