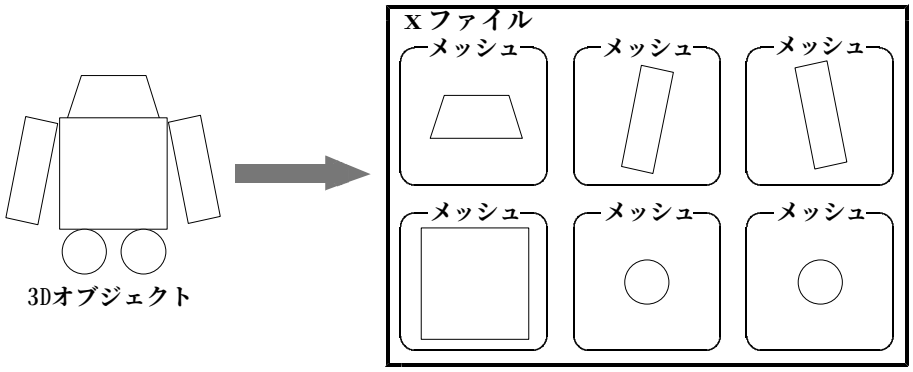
オブジェクト指向と ゲームプログラミング

▶ DirectX Graphics 3D編 － 第１２回　ｘファイルの読み込み ◀

ｘファイル

ｘファイルは、DirectX Graphics(Direct3DX)が標準でサポートする3Dオブジェクトの格納形式です。ｘファイルには、3Dオブジェクトの形状データがメッシュ単位で格納されています。１つのｘファイルの中に、複数のメッシュを格納することができ、メッシュごとにマテリアルなどの情報を設定することができます。



ｘファイルの読み込み

ｘファイルは、D3DXLoadMeshFromX関数で読み込むことができます。この関数でｘファイルを読み込むと、メッシュを制御するためのID3DXMeshインタフェース、隣接面データ、テクスチャのファイル名を含むマテリアル情報の配列、エフェクト、サブセット数が返されます。

D3DXLoadMeshFromX関数

－説明－

ｘファイルからメッシュを読み込みます。

－書式－

```
HRESULT D3DXLoadMeshFromX(LPCTSTR pFilename, DWORD Options, LPDIRECT3DDEVICE9 pDevice,
                          ID3DXBuffer** ppAdjacency, ID3DXBuffer** ppMaterials,
                          ID3DXBuffer** ppEffectInstances, LPDWORD pNumMaterials,
                          ID3DXMesh* ppMesh);
```

－パラメータ－

１つ目の引数(pFilename)は、読み込むファイル名です。

２つ目の引数(Options)は、メッシュの作成オプションの指定です。

３つ目の引数(pDevice)は、メッシュに関連づけるIDirect3DDevice9インタフェースのポインタを指定します。

４つ目の引数(ppAdjacency)は、隣接面データを受け取る場合に使用します。必要ない場合、NULLを指定します。

５つ目の引数(ppMaterials)は、マテリアルデータを受け取る場合に使用します。必要ない場合、NULLを指定します。

６つ目の引数(ppEffectInstances)は、エフェクトインスタンスを受け取る場合に使用します。必要ない場合、NULLを指定します。

７つ目の引数(pNumMaterials)は、サブセット数を受け取るDWORD型変数のアドレスを指定します。

８つ目の引数(ppMesh)は、生成されるメッシュオブジェクトのインタフェースを受け取る変数のアドレスです。ID3DMesh*型またはLPD3DXMESH型の変数のアドレスを指定します。

－戻り値－
成功した場合はD3D_OK、それ以外はエラーコードを返します。

```
// Ｘファイル読み込み(pD3DDeviceは、初期化済みのDirect3DDevice9オブジェクト)
ID3DXMesh*      pMesh;          // メッシュ
ID3DXBuffer*    pMtrlBuf;       // マテリアル配列のバッファ
DWORD           subset;         // サブセット数
::D3DXLoadMeshFromX("3DObject.x", D3DXMESH_MANAGED, pD3DDevice,
                    NULL, &pMtrlBuf, NULL, &subset, &pMesh);
```

D3DXLoadMeshFromX関数で生成したメッシュは、頂点データのフォーマットが読み込んだＸファイルによって左右されます。そのため、必要なデータが含まれてなかったり、逆に必要ないデータが含まれている場合があります。
メッシュの頂点データを任意の形式に変更するには、ID3DXMesh::CloneMeshFVFメソッドを使います。このメソッドは、現在のメッシュをもとに、指定したFVFを持つ新しいメッシュを生成します。
たとえば、頂点データが「座標」「法線」「テクスチャ座標」のみの新しいメッシュを生成するには、以下のようになります。

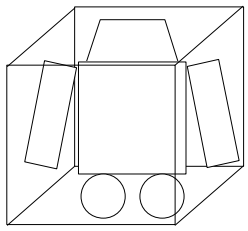
```
// 「座標」「法線」「テクスチャ座標」のみのメッシュを生成する
ID3DXMesh*    pNewMesh;
pMesh->CloneMeshFVF(pMesh->GetOptions(), D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_TEX1,
                    pD3DDevice, &pNewMesh);
```

CloneMeshFVFメソッドを使うことにより、指定したフォーマットを持つ新しいメッシュを生成することができますが、もとのメッシュにない情報は、そのデータを格納する場所しか生成することができません。
Ｘファイルによっては、法線(頂点の向きを示すベクトル)がない場合があります。法線がない場合、光を反射することができないので、レンダリングしても色は付きません。
このような場合は、D3DXComputeNormals関数を使います。この関数は、頂点に隣接する面の向きの平均をとり、頂点に法線を設定します。

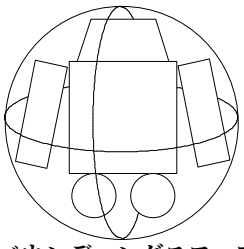
```
// 法線を計算によって求める
D3DXComputeNormals(pNewMesh, NULL);
```

バウンディングボリューム

バウンディングボリューム(境界ボリューム)とは、3Dオブジェクトに設定された３次元の空間のことをいいます。バウンディングボリュームには、形状によってバウンディングボックス(境界ボックス)とバウンディングスフィア(境界球)があります。



▲バウンディングボックス



▲バウンディングスフィア

バウンディングボリュームは、3Dオブジェクトの大きさを調べるときや、3Dオブジェクト同士の衝突検出などに用いることができます。
Direct3DXは、3Dオブジェクトを包括するバウンディングボックスを求めるD3DXComputeBoundingBox関数と、3Dオブジェクトを包括するバウンディングスフィアを求めるD3DXComputeBoundingSphere関数を提供しています。

課　　題

CDXGraphics9クラスに、ｘファイルからメッシュを生成する機能を追加しましょう。

(１)CDXGraphics9クラスに、ｘファイルからメッシュを生成するCreateMeshFromX関数を作成します。
以下のプログラムを完成させて適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                          メッシュ生成                                             */
/*****************************************************************************************************/
IMesh* CDXGraphics9::CreateMeshFromX(LPCTSTR inFileName)
{
#ifdef _DEBUG
    if(m_pD3DDevice == NULL) {
        ::OutputDebugString("*** Error - Direct3DDevice9未初期化(CDXGraphics9_CreateMeshFromX)\n");
        return NULL;
    }
#endif

    // ｘファイル読み込み
    IMesh*          pMesh;
    ID3DXMesh*      pDXMesh    = NULL;
    ID3DXBuffer*    pDXMtrlBuf = NULL;
    DWORD           subset;
    try {
        if(::????????????????????(inFileName, D3DXMESH_MANAGED, m_pD3DDevice,
                                   NULL, &pDXMtrlBuf, NULL, &subset, &pDXMesh) != D3D_OK)
            throw "*** Error - ｘファイル読み込み失敗(CDXGraphics9_CreateMeshFromX)\n";

        // 頂点データを「座標」「法線」「テクスチャ座標」のみにする
        const DWORD   FORMAT     = pDXMesh->GetFVF();
        const DWORD   NEW_FORMAT = D3DFVF_XYZ | D3DFVF_NORMAL | (D3DFVF_TEX1 & FORMAT);
        if(NEW_FORMAT != FORMAT) {
            // 必要なデータのみのメッシュに再構築する
            ID3DXMesh*   pDXNewMesh;
            if(pDXMesh->CloneMeshFVF(pDXMesh->GetOptions(), NEW_FORMAT, m_pD3DDevice, &pDXNewMesh)
               != D3D_OK)
                throw "*** Error - メッシュ再構築失敗(CDXGraphics9_CreateMeshFromX)\n";

            // 法線がない場合、色が付かないので法線を計算する
            if((FORMAT & D3DFVF_NORMAL) == 0)
                ::????????????????????(pDXNewMesh, NULL);

            pDXMesh->Release();         // 古いメッシュの解放
            pDXMesh = pDXNewMesh;
        }
    } catch(LPCSTR ErrorString) {
        // 失敗した場合はNULLメッシュを返す
        ::OutputDebugString(ErrorString);

        SafeRelease(pDXMesh);
        SafeRelease(pDXMtrlBuf);

        pMesh = new CNullMesh();
        m_Mesh.push_back(pMesh);

        return pMesh;
    }

    // メッシュオブジェクトを生成し、リストへ追加
    pMesh = new CMesh(m_pD3DDevice, pDXMesh, subset);
    pDXMesh->Release();
    m_Mesh.push_back(pMesh);

    // ファイルパス取得
    TCHAR   Drive[_MAX_DRIVE + 1];
    TCHAR   Dir  [_MAX_DIR   + 1];
    ::_splitpath(inFileName, Drive, Dir, NULL, NULL);

    // マテリアル設定
    LPD3DXMATERIAL   pMaterial = (LPD3DXMATERIAL)pDXMtrlBuf->GetBufferPointer();
    for(DWORD i = 0; i < subset; i++) {
        // マテリアル設定
```

```
        pMesh->SetMaterial(i, pMaterial[i].MatD3D);

        // テクスチャファイル名設定(ｘファイルのパス＋ファイル名にする)
        TCHAR   file_name[MAX_PATH + 1];
        ::wsprintf(file_name, "%s%s%s", Drive, Dir, pMaterial[i].pTextureFilename);

        // テクスチャ設定
        pMesh->SetTexture(i, FileName);
    }
    pDXMtrlBuf->Release();

    return pMesh;
}
```

(２)メッシュクラスにバウンディングボリュームを返す機能を追加します。以下のバウンディングボックス構造体とバウンディングボリューム構造体を適切な場所に追加しましょう。

```
    // バウンディングボックス構造体
    struct DXGBOUNDINGBOX {
        D3DXVECTOR3   Min;          // 境界ボックスの左下隅の座標
        D3DXVECTOR3   Max;          // 境界ボックスの右上隅の座標
    };

    // バウンディングスフィア構造体
    struct DXGBOUNDINGSPHERE {
        D3DXVECTOR3   Center;       // 境界球の中心の座標
        float         Radius;       // 境界球の半径
    };
```

(３)メッシュクラスにバウンディングボリュームを返す関数を追加します。以下のプロトタイプをIVertexBufferクラス定義の適切な場所に追加しましょう。

```
    virtual DXGBOUNDINGBOX    GetBoundingBox()    = 0;
    virtual DXGBOUNDINGSPHERE GetBoundingSphere() = 0;
```

(４)CMeshクラスにバウンディングボリュームを返す関数を追加します。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。

```
/*****************************************************************************************************/
/*                                    バウンディングボックス取得                                     */
/*****************************************************************************************************/
DXGBOUNDINGBOX CMesh::GetBoundingBox()
{
    // メッシュの頂点バッファを得る
    IDirect3DVertexBuffer9*   pVB;
    m_pMesh->GetVertexBuffer(&pVB);

    // バッファへのポインタを得る
    LPVOID   pVertices;
    pVB->Lock(0, 0, &pVertices, D3DLOCK_READONLY);

    // バウンディングボックスを計算する
    DXGBOUNDINGBOX   box;
    ::??????????????????????((D3DXVECTOR3*)pVertices, m_pMesh->GetNumVertices(),
                             ::D3DXGetFVFVertexSize(m_pMesh->GetFVF()), &box.Min, &box.Max);
    pVB->Unlock ();
    pVB->Release();

    return box;
}

/*****************************************************************************************************/
/*                                    バウンディングスフィア取得                                     */
/*****************************************************************************************************/
DXGBOUNDINGSPHERE CMesh::GetBoundingSphere()
{
    // メッシュの頂点バッファを得る
    IDirect3DVertexBuffer9*   pVB;
    m_pMesh->GetVertexBuffer(&pVB);

    // バッファへのポインタを得る
```

```
    LPVOID   pVertices;
    pVB->Lock(0, 0, &pVertices, D3DLOCK_READONLY);

    // バウンディングスフィアを計算する
    DXGBOUNDINGSPHERE   sphere;
    ::????????????????????????????((D3DXVECTOR3*)pVertices, m_pMesh->GetNumVertices(),
                                ::D3DXGetFVFVertexSize(m_pMesh->GetFVF()), &sphere.Center, &sphere.Radius);

    pVB->Unlock ();
    pVB->Release();

    return sphere;
}
```

(５)CNullMeshクラスにバウンディングボリュームを返す関数を追加します。以下のプログラムを適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                        バウンディングボックス取得                                   */
/*****************************************************************************************************/
DXGBOUNDINGBOX CNullMesh::GetBoundingBox()
{
    DXGBOUNDINGBOX   box;
    ::ZeroMemory(&Box, sizeof(box));

    return box;
}

/*****************************************************************************************************/
/*                                        バウンディングスフィア取得                                   */
/*****************************************************************************************************/
DXGBOUNDINGSPHERE CNullMesh::GetBoundingSphere()
{
    DXGBOUNDINGSPHERE   sphere;
    ::ZeroMemory(&Sphere, sizeof(sphere));

    return sphere;
}
```

(６)メッシュクラスにメッシュを最適化する機能を追加します。以下のプログラムを適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                          メッシュ最適化                                             */
/*****************************************************************************************************/
bool CMesh::Optimize(const DWORD inFlags)
{
    ID3DXMesh*   pOptMesh;
    if(m_pMesh->Optimize(inFlags, NULL, NULL, NULL, NULL, &pOptMesh) != D3D_OK) {
        ::OutputDebugString("*** Error - メッシュ最適化失敗(CMesh_Optimize)\n");
        return false;
    }

    m_pMesh->Release();
    m_pMesh = pOptMesh;

    return true;
}
```

CMesh::Optimize関数の仕様は、以下のとおりです。

Optimize　　その他

メッシュの面および頂点の順番を変更し、パフォーマンスを最適化します。

試bool Optimize(const DWORD inFlags);

Return	成功：true　それ以外：false
inFlags	最適化の種類を指定。おもに、次のフラグの組み合わせを指定
	D3DXMESHOPT_COMPACT
	面の順番を変更し、使用されていない頂点と面を削除する

D3DXMESHOPT_ATTRSORT
　IID3DXMesh::DrawSubsetのパフォーマンスを上げるため、面の順番を変更する
D3DXMESHOPT_VERTEXCACHE
　面の順番を変更し、頂点キャッシュのキャッシュ ヒット率を向上させる
D3DXMESHOPT_STRIPREORDER
　面の順番を変更し、隣接する三角形の長さを最大にする
D3DXMESHOPT_IGNOREVERTS
　面のみを最適化し、頂点は最適化しない
D3DXMESHOPT_DONOTSPLIT
　属性のソート中に、属性グループ間で共有している頂点を分割しない

(７)IMeshクラスにOptimize関数の純粋仮想関数、CNullMeshクラスに「trueを返す」Optimize関数を作成しましょう。

(８)ｘファイルを読み込んで描画してみます。CTestScene::CTestScene関数を以下のように変更しましょう。

```
CTestScene::CTestScene()
{
    // ここに、機能テストの初期化処理を記述します
    // カメラ視野設定
    DXGCamera()->SetProjection(30.0f, 640.0f, 480.0f, 0.1f, 1000.0f);

    // カメラ位置、角度設定
    DXGCamera()->SetView(D3DXVECTOR3(0.0f, 0.0f, -5.0f), D3DXVECTOR3(0.0f, 0.0f, 0.0f));

    // ｘファイル読み込み
    m_pMesh = DXGraphics().CreateMeshFromX("ｘファイルの名前.x");

    // ライト設定
    D3DLIGHT9   light;
    light.Type      = D3DLIGHT_DIRECTIONAL;                 // ディレクショナルライト
    light.Diffuse   = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);    // ディフューズ色
    light.Ambient   = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);    // アンビエント色
    light.Specular  = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);    // スペキュラ色
    light.Direction = D3DXVECTOR3(0.0f, -1.0f, 0.0f);       // ライトの向き(下)
    DXGraphics().SetLight(0, light, TRUE);                  // ライトを設定

    FPSTimer().Reset();     // タイマリセット
}
```

(９)CTestScene::ActiveProc関数を以下のように変更しましょう。

```
int CTestScene::ActiveProc()
{
    // ここに、機能テストのメイン処理を記述します

    // ここに、機能テストの描画処理を記述します
    if(FPSTimer().IsSkip())
        return 0;

    DXGraphics().Clear(D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, D3DXCOLOR(0.0f, 0.0f, 0.5f, 0.0f));

    DXGraphics().BeginScene();

    m_pMesh->Render();              // メッシュのレンダリング

    DXGraphics().EndScene();

    DXGraphics().UpdateFrame();

    return 0;
}
```