15　デスクトップ 3D マスコット

佐藤　一美

指導教員　小笠原　祐治

1.　はじめに

私は、3D プログラムに興味があり 3D グラフィックアプリケーションを作成したいと思いこのテーマに決めた．また，3D グラフィックの API は、OpenGL や DirectX があるが，Windows 環境での動作，開発を考えたうえで安定，高速処理ができる DirectX を使用することにした．

2.　研究概要

2.1　目的

DirectX11 を使用し，Windows デスクトップマスコットを作成する．マスコットは文字での会話をできるようにする．この研究を通して今までの復習，新しい知識の取得や技術の向上を目指す．

3.　開発環境

環境	名称
OS	Windows7
言語	C++
開発ツール	DirectX11SDK
統合開発環境	VisualStudio C++2010

4.　3D マスコットについて

4.1　システム概要

パソコンのデスクトップ画面に 3D キャラクタを表示させ、アニメーションをする．
アプリケーションの詳細設定(モデルの切り替え、ウィンドウサイズの調整等)は、別に用意したモデルデータ設定ファイルに記述する．

4.2 アプリの処理内容

- レイヤード(透過)ウィンドウの生成
- 設定ファイルの読み取り
- 3D モデルファイル読み取り
- 3D モデルファイル表示、アニメーション

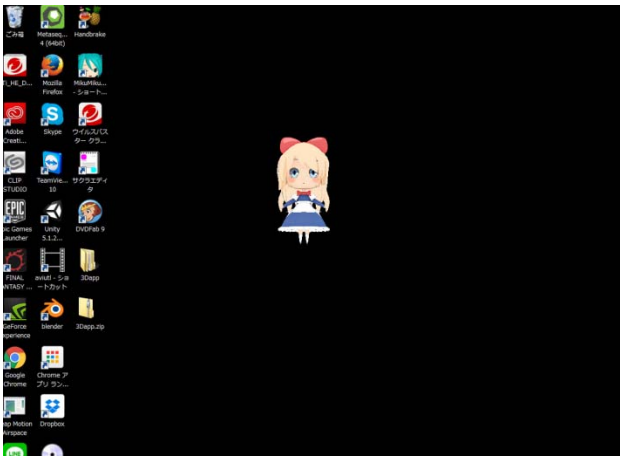


図 1　完成図

5.　研究内容

5.1　DirectX について

DirectX とは、マイクロソフトが開発したゲーム・マルチメディア処理用の API の集合である。オーバーヘッドを少なくしたデバイスの仮想化・抽象化を提供する。Windows・Xbox・Xbox 360・Xbox One 向けのようにマイクロソフト製のプラットフォームおよびデバイスにおいて広く利用されている。グラフィックスに関しては、DirectX (Direct3D) 互換のビデオカードを利用することにより、高品質の 2 次元・3 次元コンピュータグラフィックスを高速にレンダリングできる。

5.2　DirerctX ライブラリについて

DirectX11 のプログラム作成の際に以下の５つのクラスに分け制作した。

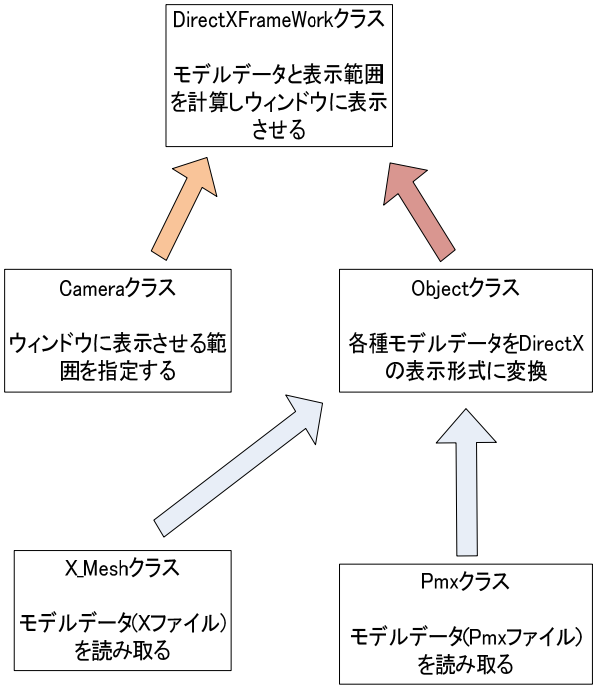


図 2. DirectX ライブラリのデータの流れ

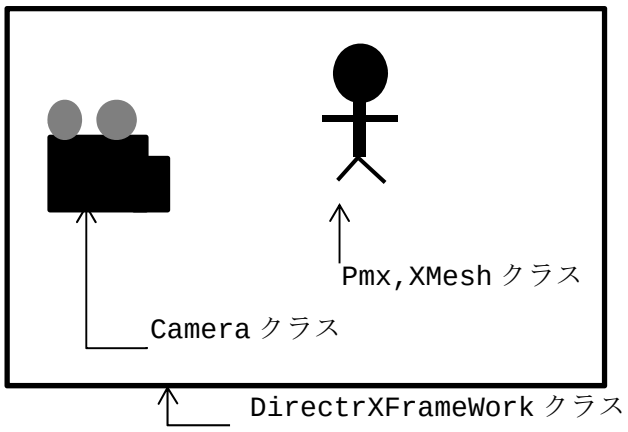


図 3. 各クラスの役割

これにより DirectX の初期化やデバイスの作成、オブジェクトの表示などアプリケーション側の記述を大幅に減らすことができた.
また、アプリケーション内のコードを見やすくすることができた.

クラスごとの主な実装関数(返り値、引数省略)を以下に示す.

DirecrtXFrtameWork クラス

- コンストラクタ()…アプリケーション側から Divice,DiviceContext を取得し、ウィンドウやグラフィックボードへの接続の設定をする.
- ReturnDevice()…コンストラクタで作成した設定用クラス Device を返す
- ReturnDeviceContext()…コンストラクタで作成したグラフィックボードへの設定適用クラス DeviceContext を返す

Camera クラス

- Camera_Pos()…カメラの設置位置、注視設定用関数

Object クラス

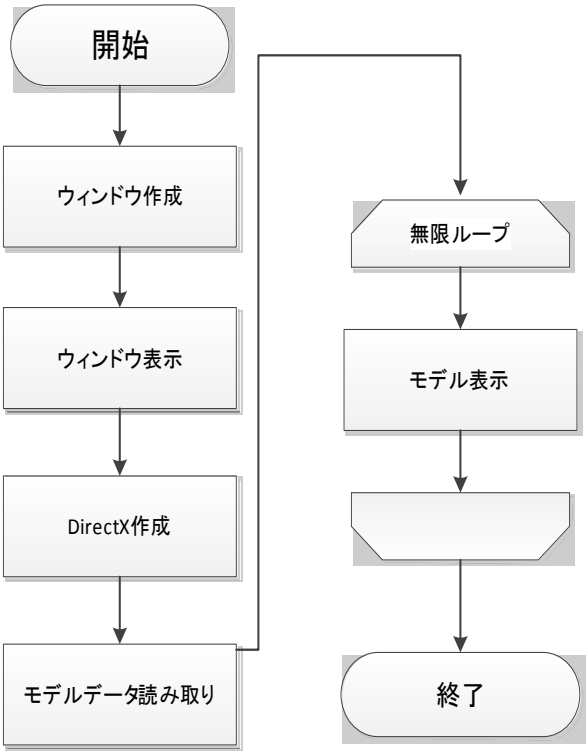
- InoutLayout_Creat()…描画するポリゴンデータを DirectX の描画用処理へ渡す.
- VertexShader_Creat()…モデルデータを二次元の座標系に変換する。
- PixelShader_Creat()…光源データやカラーを計算してピクセル単位でモデルに着色していく.
- Rasterizer_Create()…作成されたポリゴンをピクセルデータへ変換する.
- BlendState_Creat()…テクスチャの透過色の処理を行う
- Texture_Creat()…テクスチャを作成する

XMesh クラス

- XMeshLoad()…指定された X ファイル(モデルデータ)のデータを読み込む
- XMeshCreat()…読み込まれた、X ファイルのデータを DirectX の表示形式用に変換する.
- XMeshDraw()…X ファイル(モデルデータ)内のモデルを表示する

Pmx クラス

- PmxLoad()…指定された Pmx ファイル(モデルデータ)内のデータを読み込む.
- PmxCreat()…読み込まれた、Pmx ファイルのデータを DirectX の表示形式用に変換する.
- PmxDraw()…Pmx ファイル(モデルデータ)内のモデルを表示する.



6. モデル表示について

モデル表示方法について、現在はモデルデータ設定ファイルからモデルファイルが存在するファイルパスなどを記述し、そのファイルを読み込むことでモデルを表示するという動作にしている.

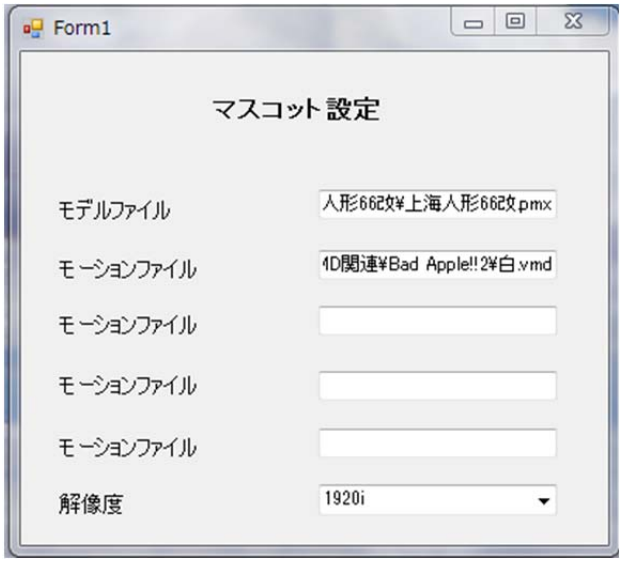


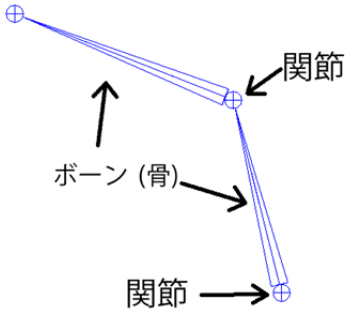
図 4 モデルデータ設定ファイル内容

6.1 アニメーションについて

3D モデルアニメーション制御では、頂点のみで制御をするスキンメッシュアニメーションとボーン(骨)を使用するボーンアニメーションの 2 種類がある.

今回のアプリケーションでは、ボーンアニメーションを使用して 3D モデルのアニメーション制御を行っている.

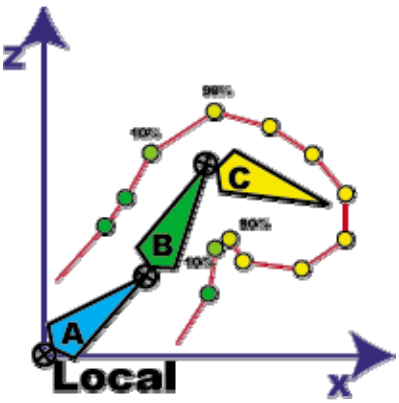
ボーンアニメーションの制御方法



ボーンアニメーションは、ボーンを移動回転することにより３D モデルが歩く、座るなどの動作をする.

基本的に、三次元空間の移動、回転は、座標変換行列を使用しますが。この研究ではでは処理速度の面や、メモリの節約を考え、クォータニオンという数式使用した.

アニメーション制御は、ボーン操作→操作したボーンに関連する頂点の座標変換の順番に制御する.



6.2 ボーン操作(クォータニオン回転用　公式)

V= (vx,vy,vz)　…回転軸ベクトル

th…回転する角度

Q = (cos(th / 2); x * sin(th / 2), y * sin(th / 2), z * sin(th / 2))

R …Q の共役四元数

R = (cos(th / 2); -x * sin(th / 2), -y * sin(th / 2), -z * sin(th / 2))

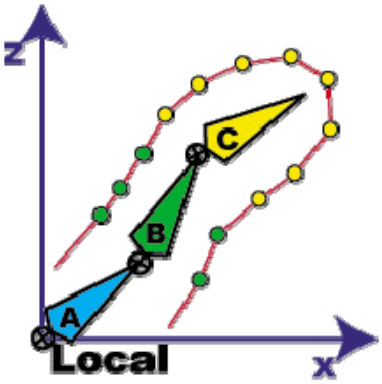
回転前の座標

P_0= (0; x_0, y_0, z_0)

回転後の座標値

R * P_0 * Q = (0; x_1,y_1,z_1);

6.3 頂点操作



ボーンの周りにはスキンのポリゴンが存在する.

各スキンのポリゴンは、それぞれのボーンにどのくらい影響するかという影響率(ウェイト値)を持っている

ボーンを操作するとスキンのポリゴンも影響率の分、移動するようになっている.

このように、ボーンからの影響率を持つ、頂点の座標変換には以下の公式を用いる.

$$P_1 = P_0 + \sum_{n=1}^{i} (Bone_n * Weight_n)$$

P_1…変換後の座標点

P_0…変換前の座標点

i…頂点に影響するボーンの総数

Bone…ボーンの座標点

Weight…影響値

7.　最後に

テーマ決定の際に比較的簡単だと予想していた作業が DirectX などと組み合わせることで、複雑化し、予想以上にてこずってしまった.

DirectX11 の参考サイトや書籍などが少なく、仕様を完璧に把握することが困難だった.

3D のアプリケーションにおいて座標変換行列や四元数など専門的な数学の知識が必要であり、それらの理解、学習に多く時間を取られてしまった.

文字での簡単な会話の実装については，手が回らなかったため実装することができなかった.

画像引用元 URL：http://urx.blue/pUtL