ESライブラリ＆＆
ゲームプログラミング

○　導入編 - 第２回　ESライブラリの仕様　○

ESライブラリ主な機能

・２Ｄ：スプライト[デバイスコンテキスト取得可能(GDI)]、文字列描画
・３Ｄ：カメラ、バーテックスバッファ、モデル、アニメーションモデル、シェーダー、
　　レンダリングターゲット、アルファブレンド、平面投影シャドウ
・入力：キーボード、マウス、ゲームパッド、振動
・サウンド：wave, midi, ogg対応

スプライト概要

DirectX Graphicsの機能(ポリゴン＋テクスチャ)を使い、2D画像を描画します。画像として読み込める形式は.bmp, .dds, .dib, .jpg, .png, .tgaで、アルファチャンネル対応、読み込み時にカラーキーとスプライトのフォーマットを指定できます。描画時に、ｚ値、スケーリング、ｚ軸回転、回転軸、透明度および乗算値が使用できます。

関連：GraphicsDevice.CreateSpriteFromFile関数、ISpriteインタフェース、SpriteBatchクラス

カメラ概要

3D空間を捉えるカメラの基本的な制御を行えます。カメラの位置、向き、視野角、前方および後方クリップ面の制御ができます。

関連：Cameraクラス

バーテックスバッファ概要

頂点データを格納する領域の生成と描画を行えます。各種変換行列を用い、T&L固定機能またはシェーダーを使って描画できます。

関連：GraphicsDevice.CreateVertexBuffer関数、IVertexBufferインタフェース

モデル

シンプルシェイプまたはｘファイルを読み込み、描画することができます。各種変換行列を用い、T&L固定機能またはシェーダーを使って描画できます。スキンメッシュ対応にも対応できます。

関連：GraphicsDevice.CreateModelFromSimpleShape関数、GraphicsDevice.CreateModelFromX関数、
　　GraphicsDevice.CreateAnimationModelFromFile関数、
　　IModelインタフェース、IAnimationModelインタフェース

シェーダー

バーテックスシェーダーおよびピクセルシェーダーが記述されたエフェクトファイルを読み込み、バーテックスバッファとモデルの描画を行えます。
プロジェクトフォルダの"Shader"フォルダにシェーダーを記述した.fxファイルをおいておけば、ビルド時に自動的にコンパイルされ、"FX"フォルダに.fxcファイルとして出力します。コンパイル済みでもコンパイルしてしまいますので、コンパイルの必要がない場合は拡張子を.fx以外に変更するなどしてください。

関連：GraphicsDevice.CreateEffectFromFile関数、IEffectインタフェース

キーボード、マウス、ゲームパッド概要

入力デバイスの状態(押されている／離されている)を取得できます。また、各デバイスのバッファを読み取り、押された瞬間や離された瞬間の検知もできます。デバイスが対応していれば、振動エフェクトも行えます。

関連：Keyboardクラス、Mouseクラス、GamePadクラス

サウンド概要

大きなサウンドファイルを読み込みながら再生するIMusicインタフェースと再生までのタイムラグが少ないISoundインタフェースがあります。IMusicインタフェースは、.wav, .mid, .oggに対応しています。ISoundインタフェースは.wav形式しか対応していませんが、再生までのタイムラグが少なく、エフェクトと3Dサウンドが使用できます。なお、システムにデコーダーが登録されていれば、イメージが圧縮されているwav形式も読み込むことができます。

関連：SoundDevice.CreateMusicFromFile関数、IMusicインタフェース、
SoundDevice.CreateSoundFromFile関数、ISoundインタフェース

カスタマイズ

・ウィンドウタイトル
Framework\GameFrameWindow.cpp－CGameFrameWindow::Create関数

```
// ウィンドウ生成
m_hWnd = ::CreateWindowEx(0, WindowClassName, TEXT("ES Game Library"), 0,
                          0, 0, 0, 0, NULL, NULL, hInstance, NULL);
```

下線部でウィンドウタイトルを指定しています。

・解像度(ウィンドウサイズ)
Framework\GameApp.cpp－CGameApp::Initialize関数

```
// フレームウィンドウ生成
m_Windowed = true;
if(m_GameFrameWindow.Create(hInstance, CGameApp::WndProc, 1280, 720, m_Windowed)
   == false)
    return false;
```

下線部が画面(ウィンドウ)の幅、高さの指定です。変数'm_Windowed'に'false'を指定するとフルスクリーンモードで初期化されます。

・マウス、ゲームパッド対応
Framework\GameApp.cpp－CGameApp::Initialize関数

```
// マウス生成
if(DInput().CreateMouse() == false)
    return false;

// ゲームパッド生成
if(DInput().CreateGamePad(4) == 0)
    return false;   // ゲームパッドが１つもない場合
```

といった感じでそれぞれ初期化できます。CreateGamePad関数の引数は、初期化したいゲームパッドの個数です。マウス、ゲームパッドがなくてもかまわない場合は記述しないか、if文を省略します。

・アイコン
リソースのアイコンを編集してください。