ESライブラリ＆＆
ゲームプログラミング

○　２Ｄ編 - 第８回　ゲームパッドとマウス　○

ゲームパッド

・ゲームパッドの状態は「GamePad(*x*)->GetState関数」で取得
・「GamePad(*x*)->GetState関数」は、GamePadState型の変数にゲームパッドの状態を格納する
・方向ボタンは「軸」という扱いになるが、同時にボタンとして認識できるものもある
・「軸」は、ｘ軸とｙ軸があり、０より大きいか小さいかでどの方向に倒されているかを判別
・ボタンが「押されているか」は定数'ButtonState_Pressed'(または非０)との比較
・ボタンが「離されているか」は定数'ButtonState_Released'(または０)との比較
・ゲームパッドが対応していれば、振動エフェクトも行える

概要

ゲームパッドを利用するには、ゲームパッドにアクセスするまでに、以下のようなプログラムを適切な場所で実行し、ゲームパッドの初期化を行う必要があります。

```
// ゲームパッド生成(２つ)
DInput().CreateGamePad(2);
```

上記処理の引数は、初期化を行うゲームパッドの最大数です。戻り値は、初期化に成功したゲームパッドの数が返されます。この処理は、キーボードの初期化のように、アプリケーション初期化時(GameApp.cpp)に行うのが一般的ですが、GameMain.cppで行っても問題ありません(ただし、ライブラリ側では、ゲームパッドの動的検出は行っていないので、ゲーム中にゲームパッドを指しても新たな追加検出はしません。また、複数接続されている場合、任意の１つを指定することもできません)。

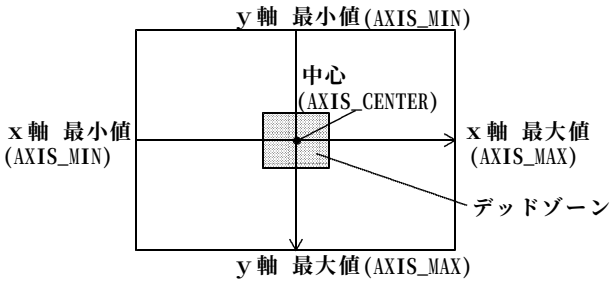
ゲームパッドがないとプレイできないゲームの場合は、以下のように関数の成否を判定し、失敗した場合は以降の処理を行わないようにします。

```
// ゲームパッド生成(４つ接続しないと不可の場合)
if(DInput().CreateGamePad(4) < 4)
    return false;

// ゲームパッド生成(１つでもあれば可の場合)
if(DInput().CreateGamePad(4) == 0)
    return false;
```

ゲームパッドの状態は、GamePad(*x*)->GetState関数で取得できます。カッコ内の*x*は、ゲームパッドを識別するためのインデックスです。最初(プレイヤー１)のゲームパッドが０または定数PlayerIndex_One、次(プレイヤー２)のゲームパッドが１または定数PlayerIndex_Two、といったように指定します。どのゲームパッドがプレイヤー１になるかは、ライブラリ側では指定できず、OSが決定します。ただし、同じポートに接続されていれば、毎回同じ結果になります。

この関数を実行すると、軸とボタンの状態がGamePadState型の変数に格納されます。
軸は、アナログ入力に対応しているため、倒している方向への数値が返されます。範囲は、以下の図の用になっており、最大値または最小値との割合で、どの程度倒されているかの判定が行えます。



範囲の数値は、定数としてInputDevice.hpp(とXNA.h)で定義されています。

ボタンは、単純に押されているか、離されているかが格納されます。定数ButtonState_Pressedと比較すると押されているかどうかの判定が行えます。離されているかどうかは定数ButtonState_Releasedとの比較になります(それぞれ非０、０との比較でも判定できます)

```
// プレイヤー１のゲームパッド状態の取得
GamePadState Pad = GamePad(0)->GetState();

// キャラクター移動
if(Pad.X < AXIS_CENTER)
    // 軸が左に倒されているときの処理
if(Pad.Y > ASIX_CENTER)
    // 軸が下に倒されているときの処理

// ボタン処理
if(Pad.Buttons[0] == ButtonState_Pressed)
    // ボタン０が押されているときの処理
```

マウス

・マウスの状態は「Mouse->GetState関数」で取得
・Mouse->GetState関数は、MouseState型の変数にカーソルの座標やボタンの状態を返す
・カーソル座標はMouseState型変数のPointerPosition関数で取得する
・ボタンが「押されているか」はMouseState構造体のメンバとButtonState_Pressed を比較する
・ボタンが「離されているか」はMouseState構造体のメンバとButtonState_Releasedを比較する
・ホイールはMouseState構造体のScrollWheelValueメンバに変化量が設定される
・マウスの移動量(変化量)もMouseState構造体のXメンバとYメンバに設定される

概要

マウスを利用するには、マウスにアクセスするまでに、以下のようなプログラムを適切な場所で実行し、マウスの初期化を行う必要があります。

```
// マウス生成
DInput().CreateMouse();
```

マウスの状態は、Mouse->GetState関数で取得できます。この関数を呼び出すと、MouseState構造体にボタンの状態と前回の呼び出し時からの変化量が格納されます。

```
// マウス状態の取得
MouseState mouse = Mouse->GetState();
```

マウスカーソルの座標は、MouseState型変数のPointerPosition関数で取得できます。

```
POINT  cursol = mouse.PointerPosition();
```

ボタンの状態は、ButtonState列挙体で定義される定数との比較で行います。ボタンが押されているかは"ButtonState_Pressed"、離されているかは"ButtonState_Released"と比較します。
ボタンを表すメンバは"LeftButton", "RightButton", "MiddleButton", "XButton1"から"XButton5"があります。XButtonは、ボタンが多数あるマウスで、たいていブラウザの「戻る」や「進む」に割り当てられているボタンです。

```
// マウスボタン判定
if (mouse.LeftButton  == ButtonState_Pressed )
    // マウスの左ボタンが押されているときの処理
if (mouse.RightButton == ButtonState_Released)
    // マウスの右ボタンが放されているときの処理
```

ホイールの移動量は、MouseState型変数のScrollWheelValueメンバで取得できます。同じように、マウスの移動量もXメンバとYメンバに格納されています。移動量とは、前回のMouse->GetState呼び出しからどのぐらい変化があったか、という変化量です。