FPS制御

FPS(フレームレート)とは？

FPS(Frames Per Second)とは、
1秒間に画面を何回描き換えるか を示す指標です。

例：60FPS → 1秒間に60回描画（1フレーム ≒ 16.67ms）

例：30FPS → 1秒間に30回描画（1フレーム ≒ 33.33ms）

FPSが安定していないと、
ゲームの動きが速くなったり遅くなったりして、プレイ感覚が変化してしまいます。

DxLibにおけるFPS制御

DxLibのScreenFlip()関数は、ディスプレイのリフレッシュタイミングに
同期して画面を切り替える仕組みになっています。
これを、" 垂直同期(Vertical Synchronization, 略して Vsync) " といいます。

そのため、DxLibのScreenFlip()の実行間隔は、
ディスプレイのリフレッシュレート(Hz)に依存します。

ディスプレイ	リフレッシュレート	実質的なFPS上限	動作
学校	ほぼ、60Hz	約60FPS	普通
自宅	60Hz / 120Hz / 144Hz	60Hz / 120Hz / 144FPS	早い

学校と自宅で、ゲーム動作が異なるのはこのため。

●技術ポイント

DxLibのScreenFlip()内では、ダブルバッファリングだけではなく、
ディスプレイの垂直同期も行われている。

DxLibの垂直同期を解除しない限り、
ディスプレイのリフレッシュレートを超えるFPSは出ない。

FpsControllerで簡単制御

既にFPS制御されている方は、そのまま使用して頂いて大丈夫ですが、
BaseProjectのCommon配下にFpsControllerクラスを配布しておりますので、
オリジナルゲームにも、FPS制御を導入するようにしてください。

導入方法

Application.h

```
#pragma once
#include <string>
class FpsController;

class Application
{

public:

    // スクリーンサイズ
    static constexpr int SCREEN_SIZE_X = 1280;
    static constexpr int SCREEN_SIZE_Y = 720;

    // 固定FPS
    static constexpr int FRAME_RATE = 60;

    ～ 省略 ～

private:

    // 静的インスタンス
    static Application* instance_;

    // FPS制御
    FpsController* fpsController_;
```

Application.cpp

```
#include <DxLib.h>
#include <EffekseerForDXLib.h>
#include "Manager/InputManager.h"
#include "Manager/ResourceManager.h"
#include "Manager/SceneManager.h"
#include "Common/FpsController.h"
#include "Application.h"

void Application::Init(void)
{

    // アプリケーションの初期設定
    SetWindowText("3DAction");

    // ウィンドウサイズ
    SetGraphMode(SCREEN_SIZE_X, SCREEN_SIZE_`
    ChangeWindowMode(true);

    // FPS制御初期化
    fpsController_ = new FpsController(FRAME_RATE);

    // DxLibの初期化
    SetUseDirect3DVersion(DX_DIRECT3D_11);
    isInitFail_ = false;
    if (DxLib_Init() == -1)
    {
        isInitFail_ = true;
        return;
    }

    ～ 省略 ～

}

void Application::Run(void)
{

    InputManager& inputManager = InputManager::GetInstance();
```

```
    SceneManager& sceneManager = SceneManager::GetInstance();

    // ゲームループ
    while (ProcessMessage() == 0 && CheckHitKey(KEY_INPUT_ESCAPE) == 0)
    {

        inputManager.Update();
        sceneManager.Update();

        sceneManager.Draw();

#ifdef _DEBUG

        // 平均FPS描画
        fpsController_->Draw();

#endif // _DEBUG

        ScreenFlip();

        // 理想FPS経過待ち
        fpsController_->Wait();

    }

}

void Application::Destroy(void)
{

    // FPS制御メモリ解放
    delete fpsController_;

    ～ 省略 ～

}
```

解説

① FpsControllerのインスタンス生成はDxLibの初期化前に行うこと

DxLibの垂直同期待ちを無効化にする、
SetWaitVSyncFlag(false)関数は、DxLib_Init関数の前に呼び出す
必要があります。

② SleepやDxLibのWaitTimer関数は1msよりも、精度が低い

最短指定時間は、1ms。更に、1msピッタリ停止するわけではなく誤差あり。
0.016秒の戦いにおいて、慎重に取り扱いたい。

③ FPSは、指定回数分の平均値を表示

④ FPSを指定したい場合は、コンストラクタの引数で値を設定