キャラクターの移動とアニメーション

カメラの実装と解説が終わりましたので、

演習①　　カメラの回転情報を使用して、
キャラクターの移動処理を実装してください

WASDキーによる、4方向移動で大丈夫です。

Player.h

```
public:

    // スピード
    static constexpr float SPEED_MOVE = 5.0f;
    static constexpr float SPEED_RUN = 10.0f;

private:

    ～ 省略 ～

    // 移動スピード
    float speed_;

    // 移動方向
    VECTOR moveDir_;

    // 移動量
    VECTOR movePow_;

    // 移動後の座標
    VECTOR movedPos_;          →    ※ transform_.pos に移動後座標を
                                       直接代入するのではなく、
    ～ 省略 ～                          一度この変数に代入してください
                                       // 移動
    // 操作                            transform_.pos = movedPos_;
    void ProcessMove(void);
```

次にアニメーションの切り替え処理を実装していきます。
アニメーションを簡単に制御できるように、AnimationController クラスを
作成しています。

DxLibで３Ｄモデルのアニメーション再生を行う際に、
アニメーションのアタッチやデタッチを行い、フレーム再生を行う必要が
ありますが、そのあたりを自動的に処理するように作ったラッパークラスです。

【AnimationControllerの簡単な使い方】

○ 初期化

コンストラクタの引数に、
アニメーションを行いたいモデルのハンドルIDを渡します。

Add関数の第１引数にて、再生したいアニメーションが入っている、
mvl ファイルのパスを指定して、ロードを行います。
第２引数には、再生するスピードを指定します。

Player の InitAnimation関数

```
std::string path = Application::PATH_MODEL + "Player/";

animationController_ = std::make_unique<AnimationController>(
    transform_.modelId);

animationController_->Add(
    (int)ANIM_TYPE::IDLE, path + "Idle.mvl", 20.0f);
animationController_->Add(
    (int)ANIM_TYPE::RUN, path + "Run.mvl", 20.0f);
animationController_->Add(
    (int)ANIM_TYPE::FAST_RUN, path + "FastRun.mvl", 20.0f);

・・・
・・・
```

○ 更新

```
// アニメーション再生
animationController_->Update();
```

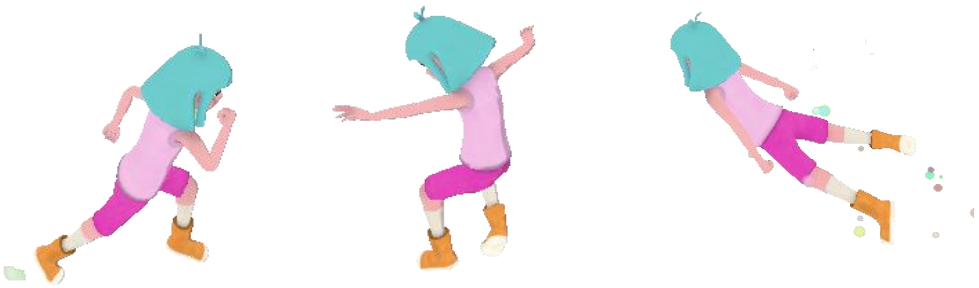
Update関数を呼び出すことで、自動的にアニメーションを進行させます。

○ アニメーションの切り替え

```
animationController_->Play((int)ANIM_TYPE::RUN);
```

Play関数を呼び出せば、指定アニメーションが再生されます。

FAST_RUN　　JUMP　　FLY



【AnimationControllerの内部処理補足】

○ AnimationController::Play
アニメーションを再生する

int type	→	再生したいアニメーションタイプ
bool isLoop	→	再生をループするか否か
float startStep	→	再生開始フレーム
float endStep	→	再生終了フレーム
bool isStop	→	アニメーション再生しない
bool isForce	→	再生中と同じアニメーションタイプを 指定した場合、Play関数を何度呼び出しても、 再生処理には影響しません。 強制的に、始めから再生したい場合などは、 trueを指定します。

○ AnimationController::SetEndLoop
アニメーション再生終了後、特定のフレームを往復再生する

今回は、ジャンプ中(※)アニメーションを表現する際に使用。
※ 対空時間は、ジャンプスピードや、地形によって大きく変わります
一定スピードでアニメーション進行してしまうと、
不自然なアニメーションになりますので、
プログラムでごまかせるような機能を実装しています

○ 全体

アニメーションブレンドには全く対応しておらず、
アニメーションタイプが変更となった場合、

```
// モデルからアニメーションを外す
playAnim_.attachNo = MVIDetachAnim(modelId_, playAnim_.attachNo);
```

アニメーションのデタッチを必ず行うようにしています。

待機アニメーションから、歩行アニメーションに遷移する際に、
自然なアニメーションの移り変わりを表現したい場合は、
アニメーションのブレンド機能を試してみると良いでしょう。

MVISetAttachAnimBlendRate(int MHandle, int AttachIndex, float Rate);
アタッチしているアニメーションのブレンド率を設定する

演習②　キャラクターに３種類のアニメーションを実装してください

・ IDLEアニメーション
移動や操作を行っていない時は、常にこのアニメーションを取るように
```
animationController_->Play((int)ANIM_TYPE::IDLE);
```
上記をフレーム毎に実装していれば、常にIDLEとなる。

・ RUNアニメーション
WASDで移動する際には、RUNアニメーションが実行されること。

・ FAST_RUNアニメーション
右Shiftキーを押しながら移動すると、FAST_RUNが実行されること。