３Ｄアニメーション

敵キャラの３Ｄモデルの表示と移動ができたかと思いますが、
アニメーションを行っておりませんので、非常に違和感に感じるかと思います。

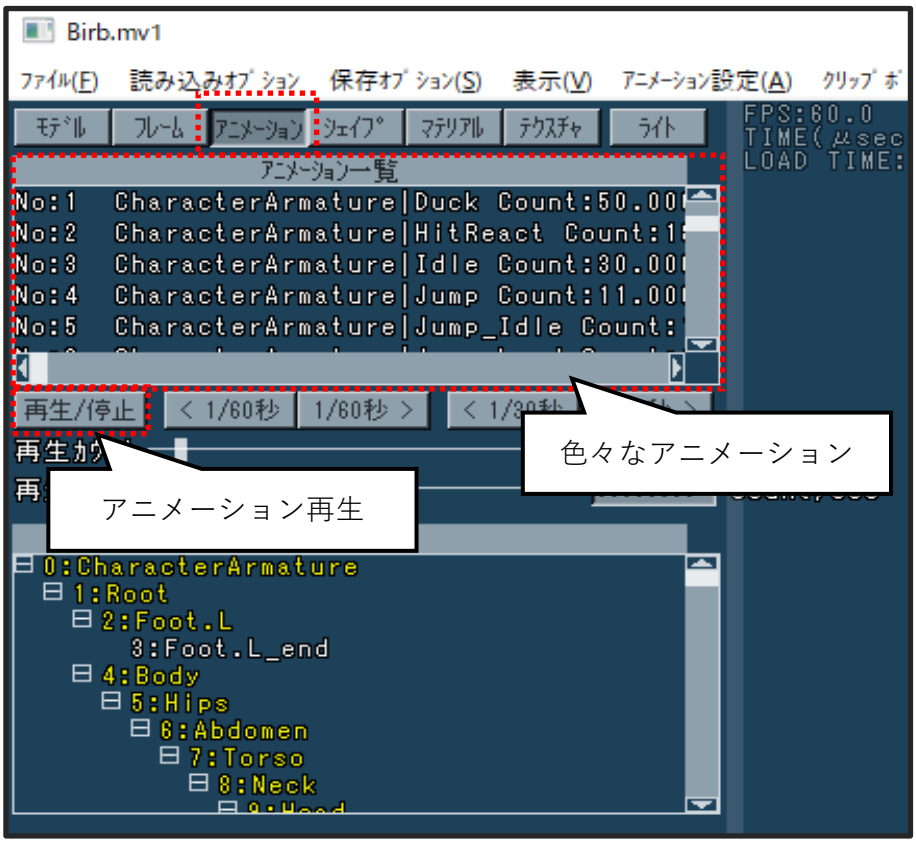
３Ｄアニメーションは、基本的にはデザイナーさんのお仕事になりますので、
ここでは詳細は解説しませんが、３Ｄソフトを使って作られます。
以前紹介したFBXファイルは、とても汎用的なファイル形式で、
モデルデータ、画像データ、アニメーションデータも一緒にまとめることが
できます。もちろん、別にすることもできます。

ＦＢＸ
モデル アニメーション 画像

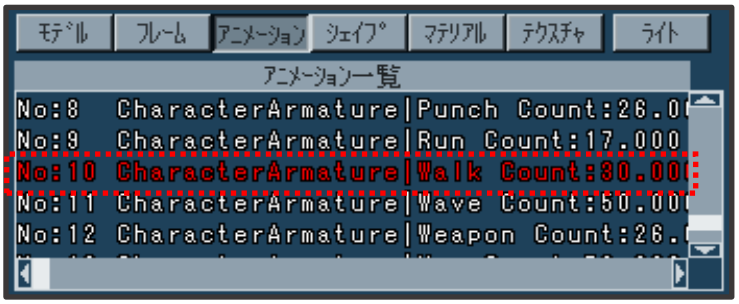
ＦＢＸ
モデル

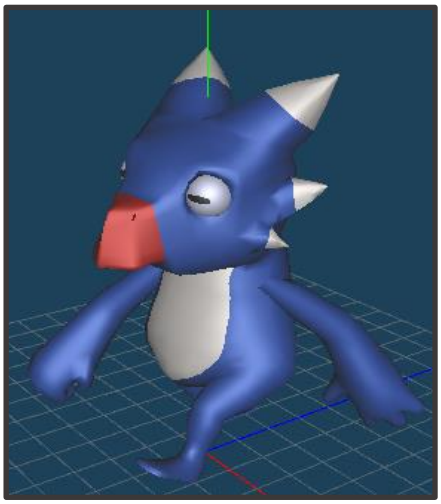
ＦＢＸ
アニメーション

今回の敵キャラモデルデータは、アニメーションが含まれているものを
用意しましたので、DxLibModelViewerで確認してみましょう。



たくさんのアニメーションが含まれていることが確認できると思います。
今回は、No 10の Walk アニメーションを敵キャラの移動に使用してきましょう。





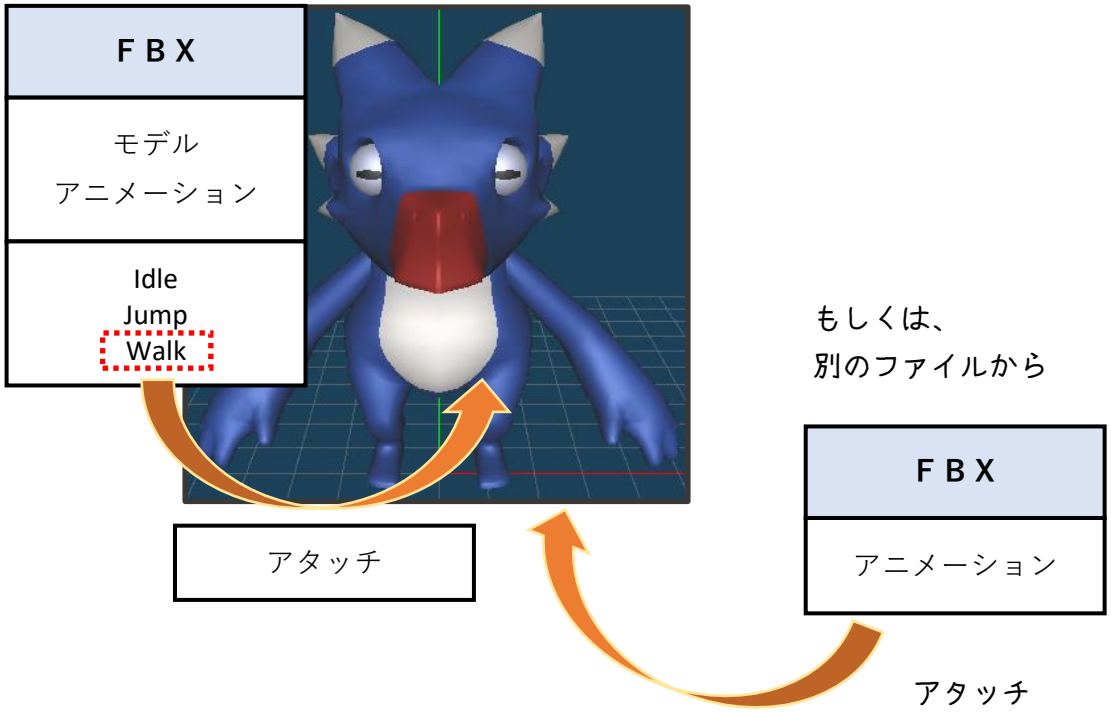
それでは、No 10 のアニメーションを再生する手順を解説していきます。

① モデルにアニメーションをアタッチする

```
int MV1AttachAnim(
    int MHandle, int AnimIndex, int AnimSrcMHandle, int NameCheck ) ;
```

第１引数　　：　アニメーションをアタッチするモデルのハンドルＩＤ
　　（ アニメーション再生させたいモデル ）
第２引数　　：　アタッチするアニメーション番号
　　DxLibModelViewerで確認したアニメーション番号。
　　今回は、No 10 を使用するので 10。
第３引数　　：　アタッチするアニメーションを持っているモデルのハンドルＩＤ
第４引数　　：　AnimSrcMHandleを指定した時に、モデルに合ったアニメーション
　　かを確認して、アタッチしない場合は、true。
　　アタッチする場合は、false。
　　falseを指定しても、多くの場合は、アニメーションが
　　ぐちゃぐちゃになります。
返り値　　：　アニメーションアタッチ番号(-1はエラー)

今回は、同じファイルの中にアニメーションデータが含まれていますので、
第３引数と第４引数は省略して大丈夫です。

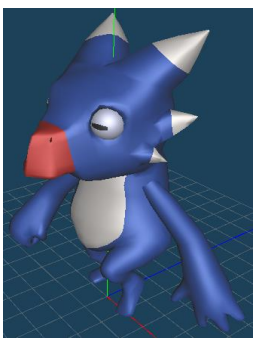


② アニメーションを再生する

```
int MVISetAttachAnimTime( int MHandle, int AttachIndex, float Time ) ;
```

第１引数　　：　アニメーション再生するモデルのハンドルＩＤ
第２引数　　：　①で取得したアニメーションアタッチ番号
第３引数　　：　再生時間

この関数を実行したら勝手に再生されるわけではなく、
第３引数で、増加する時間を指定して、画面に表示させたいアニメーションを
自分で指定する必要があります。



再生時間がおよそ10の時のモデル表示。
（ Walkアニメーション ）

移動であったり、待機のアニメーションは、基本的にはループ再生になります。
Walkアニメーションは、30カウント(再生時間)が最大値になっていますが、

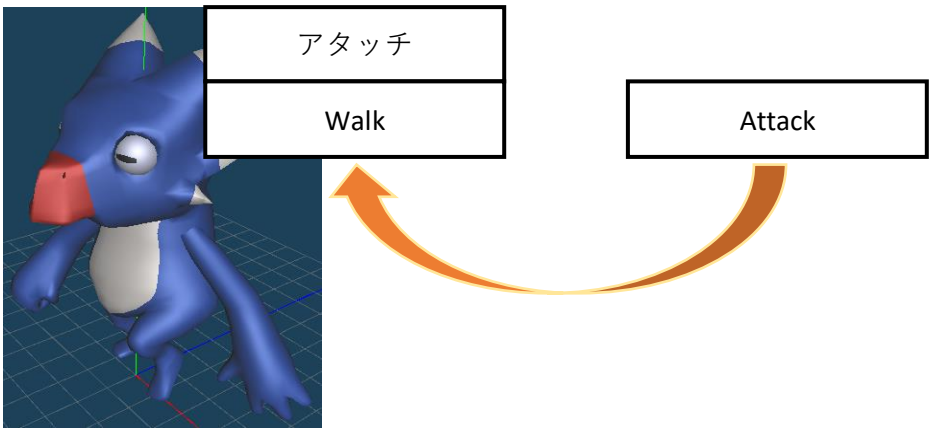
ループ再生を行うためには、再生時間が30を超えた場合、またゼロに
戻して上げる必要がありますので、
アニメーションの総再生時間、という情報が必要になります。

DxLibには、アタッチしているアニメーションの総時間を取得するという
関数があります。

float MV1GetAttachAnimTotalTime(int MHandle, int AttachIndex) ;

第１引数　　：　アタッチされているモデルのハンドルＩＤ
第２引数　　：　総時間取得するアニメーションのアタッチ番号
　　（①で取得したアニメーションアタッチ番号 ）
返り値　　：　アニメーションの総時間

この手順だけで、アニメーションを再生することができます。
今回は使用しませんが１つ、注意点があります。
Walk以外のアニメーションを再生したい時なのですが、

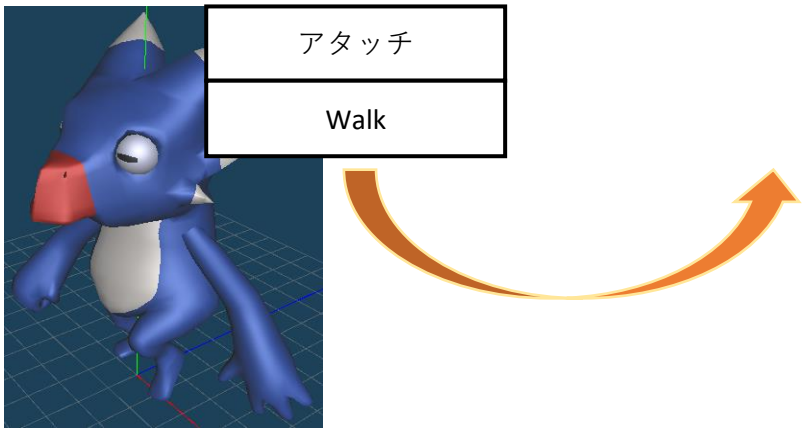


Attackのアタッチから手順を始めていくと思いますが、
これをそのままやってしまうと、WalkとAttackが混じったアニメーションになり、
アニメーションが崩れてしまいます。

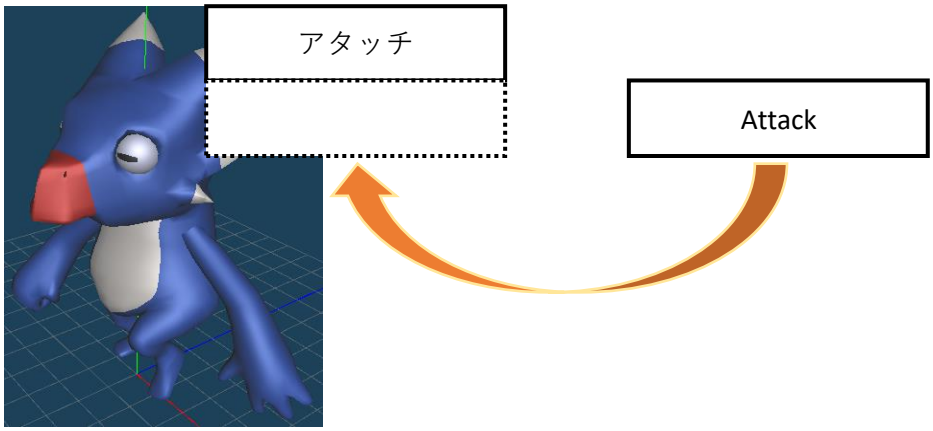
アニメーションのクロスフェードを行う場合に、敢えてこの特性を
使用することもありますが、初めのうちは純粋に該当のアニメーションを
再生させたいだけになるかと思いますので、

③ 別アニメーションをアタッチする前にデタッチする

デタッチ



アタッチ




```
int MVIDetachAnim( int MHandle, int AttachIndex ) ;
```

第１引数　　：　アニメーションをデタッチするモデルのハンドルＩＤ
第２引数　　：　デタッチするアニメーションのアタッチ番号
　　（①で取得したアニメーションアタッチ番号）

それでは実装してみましょう。

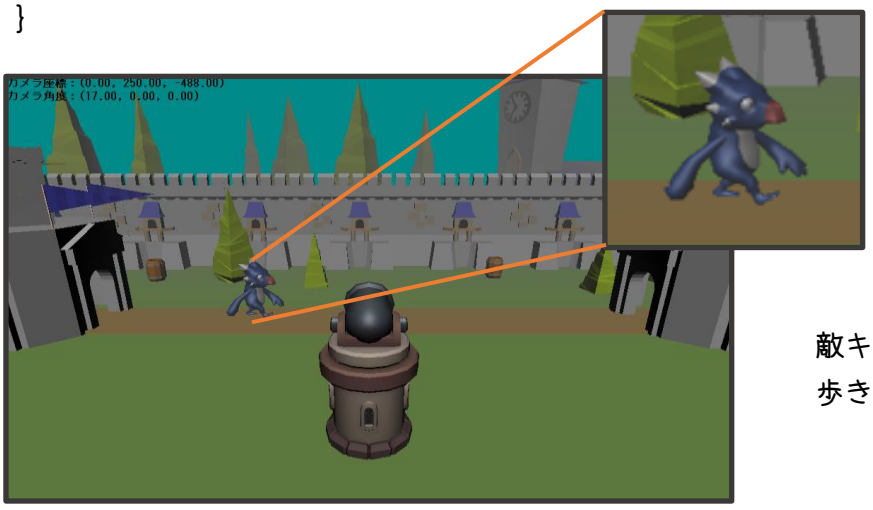
EnemyBase.h

```
    ～ 省略 ～

protected:

    ～ 省略 ～

    // アニメーションをアタッチ番号
    int animAttachNo_;

    // アニメーションの総再生時間
    float animTotalTime_;

    // 再生中のアニメーション時間
    float stepAnim_;

    // アニメーション速度
    float speedAnim_;

}
```

EnemyBase.cpp

```
void EnemyBase::SetParam(void)
{

    ～ 省略 ～

    // アニメーションをアタッチする
    animAttachNo_ = MV1AttachAnim(modelId_, 10);

    // アタッチしているアニメーションの総再生時間を取得する
    animTotalTime_ = MV1GetAttachAnimTotalTime(modelId_, animAttachNo_);

    // 再生中のアニメーション時間
    stepAnim_ = 0.0f;
```

```
    // アニメーション速度
    speedAnim_ = 30.0f;

}

void EnemyBase::Update(void)
{

    ～ 省略 ～

    // アニメーション再生

    // 経過時間の取得
    float deltaTime = 1.0f / SceneManager::DEFAULT_FPS;

    // アニメーション時間の進行
    stepAnim_ += (speedAnim_ * deltaTime);
    if (stepAnim_ > animTotalTime_)
    {
        // ループ再生
        stepAnim_ = 0.0f;
    }

    // 再生するアニメーション時間の設定
    MV1SetAttachAnimTime(modelId_, animAttachNo_, stepAnim_);

    ～ 省略 ～

}
```



敵キャラが、
歩き出したら成功です