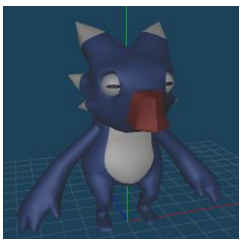
敵の作成

弾の発射や衝突判定、爆発アニメーションが揃いましたので、
的となる敵を作成していきます。

複数の種類の敵を画面に表示させる前提でクラスや仕組みを作っていきます。




EnemyBase.h

```
#pragma once
#include <string>
#include <DxLib.h>

class EnemyBase
{

public:

    // コンストラクタ
    EnemyBase(int baseModelId);

    // デストラクタ
    virtual ~EnemyBase(void);

    // 初期処理(最初の１回のみ実行)
    virtual void Init(void);

    // パラメータ設定(純粋仮想関数)
    virtual void SetParam(void);

    // 更新処理(毎フレーム実行)
    virtual void Update(void);

    // 描画処理(毎フレーム実行)
```

このクラスを継承して、
複数種類の敵を作成していきますので、
基底クラスとして設計していきます。

基底クラスのデストラクタは、
自己防止のため、必ずvirtualを

後々、派生クラスで上書きする関数はvirtualを付けて、オーバーライド可能にしておきます。

```
    virtual void Draw(void);

    // 解放処理(最後の１回のみ実行)
    virtual void Release(void);

    // 座標の取得
    VECTOR GetPos(void);

    // 座標の設定
    void SetPos(VECTOR pos);

    // 生存判定
    bool IsAlive(void);

    // 生存判定
    void SetAlive(bool alive);

    // ダメージを与える
    void Damage(int damage);

protected:

    // 元となる弾のモデルID
    int baseModelId_;

    // 弾のモデルID
    int modelId_;

    // 大きさ
    VECTOR scl_;

    // 角度
    VECTOR rot_;

    // 表示座標
    VECTOR pos_;

    // 移動速度
    float speed_;

    // 移動方向
```

```
    VECTOR dir_;

    // 体力
    int hp_;

    // 体力最大値
    int hpMax_;

    // 生存判定
    bool isAlive_;

};
```

EnemyBase.cpp

```
#include "EnemyBase.h"

EnemyBase::EnemyBase(int baseModelId)
{
    baseModelId_ = baseModelId;
}

EnemyBase::~EnemyBase(void)
{
}

void EnemyBase::Init(void)
{
    SetParam();
    Update();
}

void EnemyBase::SetParam(void)
{

    // 使用メモリ容量と読み込み時間の削減のため
    // モデルデータをいくつもメモリ上に存在させない
    modelId_ = MV1DuplicateModel(baseModelId_);

    // 大きさの設定
    scl_ = { 0.3f, 0.3f, 0.3f };
```

```
    // 角度の設定
    rot_ = { 0.0f, -90.0f * DX_PI_F / 180.0f, 0.0f };
    // 位置の設定
    pos_ = { -350.0f, 30.0f, 75.0f };

    // 右方向に移動する
    dir_ = { 1.0f, 0.0f, 0.0f };

    // 移動スピード
    speed_ = 1.5f;

    // 初期は生存状態
    isAlive_ = true;

}

void EnemyBase::Update(void)
{

    ※生存していけなければ、処理しない
    ※移動処理
    ※大きさ、角度、座標の３Ｄ制御

}
```

あとは、自分で考えながら実装してみてください。

GameScene.h

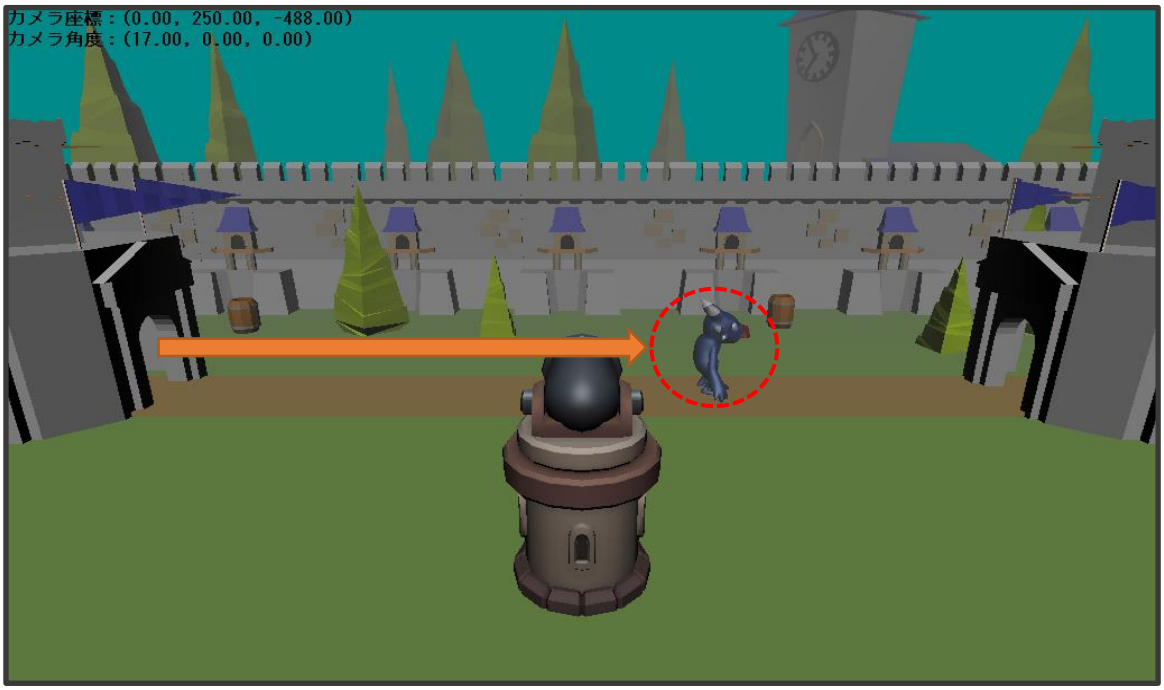
```
～ 省略 ～

private:

    ～ 省略 ～

    // 敵の画像
    int enemyModelId_;
    std::vector<EnemyBase*> enemys_;

};
```

GameScene.cpp

```
void GameScene::Init(void)
{

    ～ 省略 ～

    // 敵のモデル
    enemyModelId_ = MVlLoadModel(
              (Application::PATH_MODEL + "Enemy/Birb.mvl").c_str());
    auto enemy = new EnemyBase(enemyModelId_);
    enemy->Init();
    enemys_.push_back(enemy);

}
```

仮で１体画面に出現させる。

敵の更新、描画、解放処理を忘れずに。



敵キャラの３Ｄモデルが画面に表示されて、
直立状態ではありますが、画面右に移動するようになったらＯＫです。

敵や弾は複数種類作ることが多いかと思いますので、
継承機能を使って実装できるように基底クラスを設計できるように
なっておきましょう。