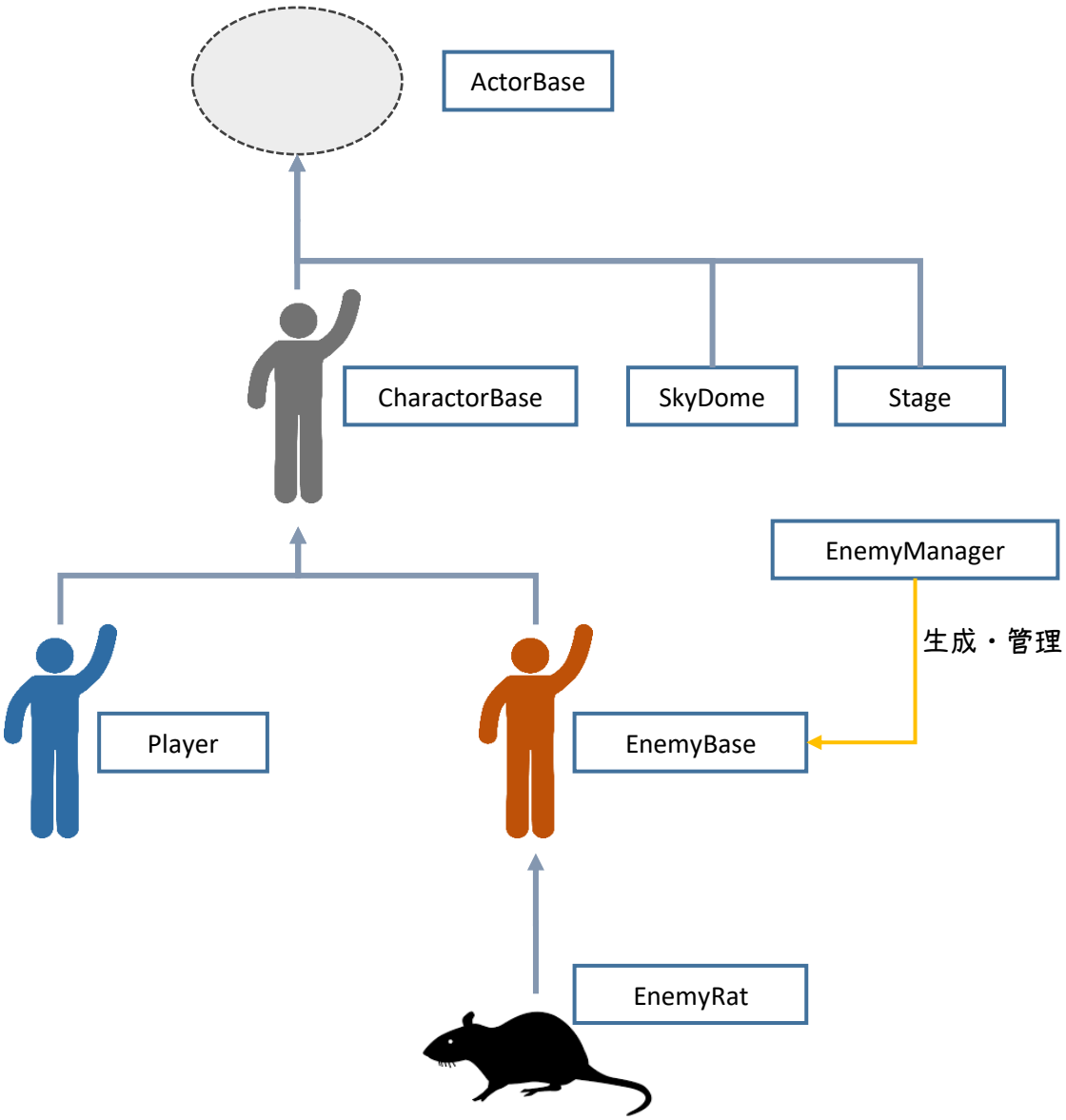
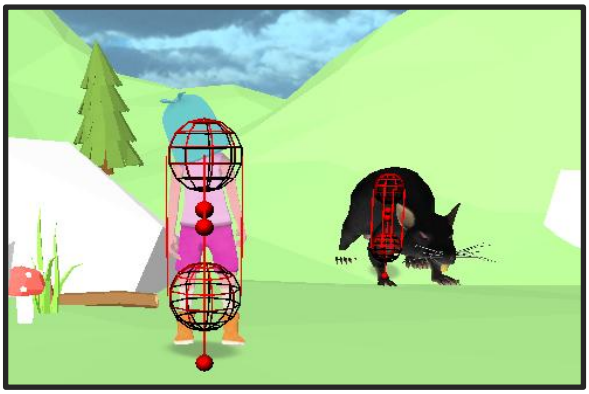
エネミーの追加

ネズミの3Dモデルを使用した、エネミーを追加していきます。
改めて、キャラクター関連の設計を確認しながら進めていきます。





■ EnemyBase

これから複数種のエネミーを作成する際に基底となるクラス。

■ EnemyRat

ネズミ型エネミー。エネミー作りの練習台。

■ EnemyManager

エネミーを生成したり、配置情報を決めたり、管理するクラス。
複数のオブジェクトをゲーム内に登場させる場合は、
そのオブジェクトの管理クラス(Managerクラス)を作ることをオススメ。

EnemyBase.h

```
#pragma once
#include "../CharactorBase.h"

class EnemyBase : public CharactorBase
{

public:

    // コンストラクタ
    EnemyBase(void);

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

protected:

    // リソースロード
    void InitLoad(void) override {}

    // 大きさ、回転、座標の初期化
    void InitTransform(void) override {}

    // 衝突判定の初期化
```

```
    void InitCollider(void) override {}

    // アニメーションの初期化
    void InitAnimation(void) override {}

    // 初期化後の個別処理
    void InitPost(void) override {}

};
```

EnemyBase.cpp

```
#include "EnemyBase.h"

EnemyBase::EnemyBase(void)
    :
    CharactorBase()
{
}

EnemyBase::~EnemyBase(void)
{
}
```

EnemyRat.h

```
#pragma once
#include <DxLib.h>
#include "EnemyBase.h"

class EnemyRat : public EnemyBase
{

public:

    // アニメーション種別
    enum class ANIM_TYPE
    {
        IDLE,
    };
```

```
    // コンストラクタ
    EnemyRat(void);

    // デストラクタ
    ~EnemyRat(void) override;

protected:

    // リソースロード
    void InitLoad(void) override;

    // 大きさ、回転、座標の初期化
    void InitTransform(void) override;

    // 衝突判定の初期化
    void InitCollider(void) override;

    // アニメーションの初期化
    void InitAnimation(void) override;

    // 初期化後の個別処理
    void InitPost(void) override;

    // 更新系
    void UpdateProcess(void) override;
    void UpdateProcessPost(void) override;

private:

    // モデルの大きさ
    static constexpr float SCALE = 0.5f;

    // モデルのローカル回転
    static constexpr VECTOR ROT = { 0.0f, 180.0f * DX_PI_F / 180.0f, 0.0f };

    // 衝突判定用線分開始
    static constexpr VECTOR COL_LINE_START_LOCAL_POS = { 0.0f, 80.0f, 0.0f };

    // 衝突判定用線分終了
    static constexpr VECTOR COL_LINE_END_LOCAL_POS = { 0.0f, -10.0f, 0.0f };
```

```
    // 衝突判定用カプセル上部球体
    static constexpr VECTOR COL_CAPSULE_TOP_LOCAL_POS = { 0.0f, 110.0f, 0.0f };

    // 衝突判定用カプセル下部球体
    static constexpr VECTOR COL_CAPSULE_DOWN_LOCAL_POS = { 0.0f, 30.0f, 0.0f };

    // 衝突判定用カプセル球体半径
    static constexpr float COL_CAPSULE_RADIUS = 20.0f;

};
```

EnemyRat.h

```
#pragma once
#include <DxLib.h>
#include "EnemyBase.h"

class EnemyRat : public EnemyBase
{

public:

    // アニメーション種別
    enum class ANIM_TYPE
    {
        IDLE,
    };

    // コンストラクタ
    EnemyRat(void);

    // デストラクタ
    ~EnemyRat(void) override;

protected:

    // リソースロード
    void InitLoad(void) override;

    // 大きさ、回転、座標の初期化
    void InitTransform(void) override;
```

```
    // 衝突判定の初期化
    void InitCollider(void) override;

    // アニメーションの初期化
    void InitAnimation(void) override;

    // 初期化後の個別処理
    void InitPost(void) override;

    // 更新系
    void UpdateProcess(void) override;
    void UpdateProcessPost(void) override;

private:

    // 衝突判定用線分開始
    static constexpr VECTOR COL_LINE_START_LOCAL_POS = { 0.0f, 80.0f, 0.0f };

    // 衝突判定用線分終了
    static constexpr VECTOR COL_LINE_END_LOCAL_POS = { 0.0f, -10.0f, 0.0f };

    // 衝突判定用カプセル上部球体
    static constexpr VECTOR COL_CAPSULE_TOP_LOCAL_POS = { 0.0f, 110.0f, 0.0f };

    // 衝突判定用カプセル下部球体
    static constexpr VECTOR COL_CAPSULE_DOWN_LOCAL_POS = { 0.0f, 30.0f, 0.0f };

    // 衝突判定用カプセル球体半径
    static constexpr float COL_CAPSULE_RADIUS = 20.0f;

};
```

EnemyManager.h

```
#pragma once
#include <vector>
class EnemyBase;

class EnemyManager
{
```

```
public:

    // コンストラクタ
    EnemyManager(void);

    // デストラクタ
    ~EnemyManager(void);

    // 初期化
    void Init(void);

    // 更新
    void Update(void);

    // 描画
    void Draw(void);

    // 解放
    void Release(void);

    // エネミー
    const std::vector<EnemyBase*>& GetEemies(void) const { return enemies_; }

    // 衝突対象となるコライダを登録
    void AddHitCollider(const ColliderBase* hitCollider);

private:

    // エネミー
    std::vector<EnemyBase*> enemies_;

};
```

EnemyManager.cpp

```
void EnemyManager::AddHitCollider(const ColliderBase* hitCollider)
{
    for (auto& enemy : enemies_)
    {
        enemy->AddHitCollider(hitCollider);
```

```
    }
}
```

【要件】

EnemyRat.cpp、EnemyManager.cppを完成させ、
ネズミの３Ｄモデルをステージに描画し、地面に接地すること。

[EnemyRat]

- ResourceManagerにリソース設定を追加すること
  SRC　　：　ENEMY_RAT
  パス　　：　"Enemy/Rat/Rat.mvl"

- モデルのロードは複製ロードを行うこと

- モデルの大きさを 0.5倍にすること

- ローカル回転を行い、正しいモデルの向きにすること

- 座標を、一旦、{ 0.0f, 100.0f, 1500.0f }に設定すること

- 地面との衝突を行う線分コライダを作成すること

- 木や岩との衝突を行うカプセルコライダを作成すること

- 初期アニメーションでIDLEを再生すること
  (FBX内のアニメーションインデックスの8番)

[EnemyManager]

- Init関数でEnemyRatのインスタンスを1体生成すること

- Update、Draw、Release関数にて、
  それぞれ enemies_ の更新、描画、メモリ解放を行うこと

[GameScene]

- エネミーにステージのモデルコライダを登録すること

【目標】

ネズミのモデルが、プレイヤーモデルよりも小さくなっており、
Zの正方向を向いて、プレイヤー前方方向の位置に描画されていること。

接地もできており、アニメーションも再生されていること。