ゲームエンティティ

Game Programming B #10

向井智彦

本日：続・カスタムエンティティ制作

1. テクスチャ付き球体
   – テクスチャファイルの作り方

2. ライトエンティティの制作（解説とデモ）
   – Gameクラスの設計の問題点
   – ライトエンティティの設計
   – SimpleGameクラスからの利用

3. カメラエンティティの制作（演習）

ライトとカメラの問題点

- 全て Game クラス内に記述されている
  - Gameクラスは、あらゆるゲームの基本のはず
  - 各ゲームでライトやカメラを操作するたびに、基本クラスを編集するのは設計が間違っている

```
bool Game::InitLight()
{
    vLightDiffuse = glm::vec4(1.0f, 1.0f, 1.0f, 1.0f); // (R,G,B,A) =
    vLightPosition = glm::vec4(0, 1.0f, 1.0f, 1.0f);
    return true;
}

bool Game::InitCamera()
{
    vCameraPosition = glm::vec4(0, 2.0f, 2.0f, 1.0f);
    mView = glm::lookAt(
        glm::vec3(0, 2.0f, 2.0f),// 視点
        glm::vec3(0, 0, 0),      // 注視点
        glm::vec3(0, 1.0f, 0));  // カメラローカル座標鉛直上向きベクトル
```

リファクタリングの目的

- Game クラスには「ゲームには必ず1つ以上のカメラがある」「ゲームにはライトがある」という情報のみ記述

- 実際の操作は継承先（SimpleGame）で行う
  - ゲームごとにGameクラスを編集しなくて良いようにする

作業の手順

1. Entityクラスを継承して、LightEntityクラスを作成
   - Init, Update, Render, Release をオーバーライド
   - ライト操作に必要なメンバ変数・関数を追加
     - ライトの色？位置？種類？など
2. SimpleGameクラスのメンバ関数にLightEntityオブジェクトを追加
3. SimpleGameからライトを操作する

作業手順詳細1

- LightEntity::Init
  - ライトの初期値を設定
- LightEntity::Update
  - 全ライトに共通する更新処理を書く
    - 複数のライト間で異なるような個別機能は書かない
      - LightEntityのメンバ変数によって違いを表すか、さらにMyLightEntityのように継承するなど
- LightEntity::Render
  - おそらく何も描画しない（つまり何も書かない）？

作業手順詳細2

- LightEntity::Release
  - ライトの終了処理を書く

- ライト特有のメンバ変数・関数
  - ライト位置
  - 色
  - ライトにも色々種類があるので興味があれば
    - directional light, point light, spot lightなど

カメラエンティティの作成（BASIC）

1. Entityクラスを継承しCameraEntityクラスを作成
   - Init, Update, Render, Release をオーバーライド
   - カメラ操作に必要なメンバ変数・関数を追加
     - カメラ位置？注視点？ズーム操作？など
2. （Gameクラスに共通カメラ機能を追加（後述））
3. SimpleGameクラスのメンバ関数にCameraEntityオブジェクトを追加
4. SimpleGameからカメラを操作する

演習の達成判定

- SimpleGameクラス内で、カメラ位置を変更できる
- SimpleGameクラス内で、カメラの注視位置を変更できる
- 上記2つの操作が、レンダリング結果に反映されている

演習詳細1

- CameraEntity::Init
  - カメラの初期値を設定
- CameraEntity::Update
  - 全カメラに共通する更新処理を書く
    - 複数のカメラ間で異なるような個別機能は書かない
      - CameraEntityのメンバ変数によって違いを表すか、さらにThirdPersonCameraEntityのように継承するなど
- CameraEntity::Render
  - おそらく何も描画しない（つまり何も書かない）？

演習詳細2

- CameraEntity::Release
  - カメラの終了処理を書く

- カメラ特有のメンバ変数・関数
  - カメラ位置、カメラ注視点
  - カメラ行列 glm::mat4 mView; // mCamera;
  - カメラの操作（オプション）
    - ティルト、ドリー、タンブル、パン、クレーン等々

Gameクラスの改変（アドバンス）

- メンバ変数に以下を追加

  std::vector<CameraEntity*> cameras;

  CameraEntity *activeCamera;

- メンバ関数に以下を追加

  CameraEntity* ActiveCamera(void);

  void AddCamera(CameraEntity *camera);

  void SwitchCamera(int cameraID);

ゲームには1つ以上のカメラが存在する。
同時に使用するカメラは1つのみ。