プログラマブルシェーダ２

Game Programming B #09

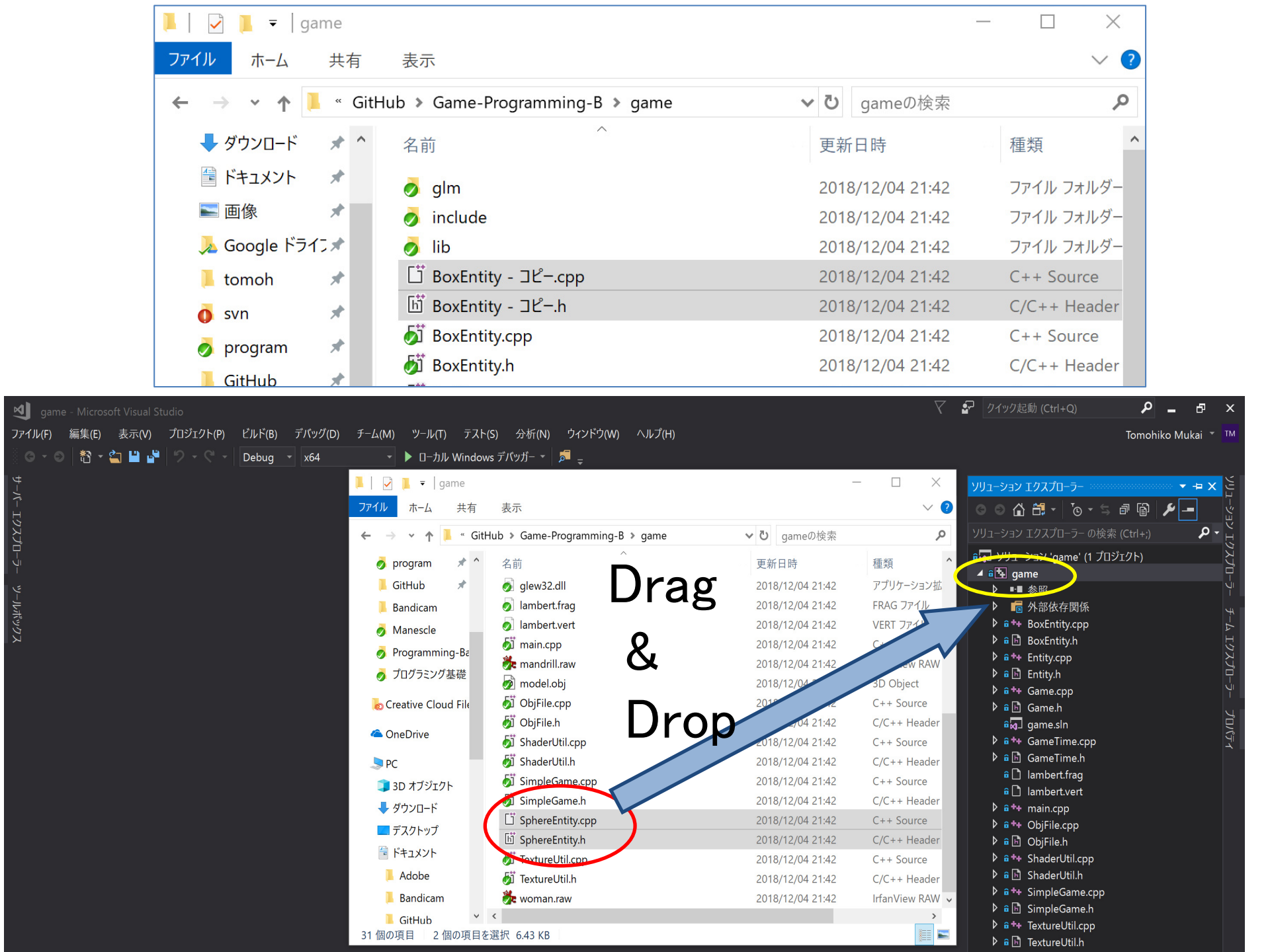
向井智彦

本日の演習:カスタムエンティティ制作

1. モデリング - 好みの形を作る
2. シェーディング - 好みの色を出す
   – 時間に応じて or イベントに応じて色を変える等も
3. エンティティクラス作成
   – 作ったモデルを読み込む
   – 作ったシェーダを読み込む
4. エンティティオブジェクト生成
5. ゲームクラスに登録して表示

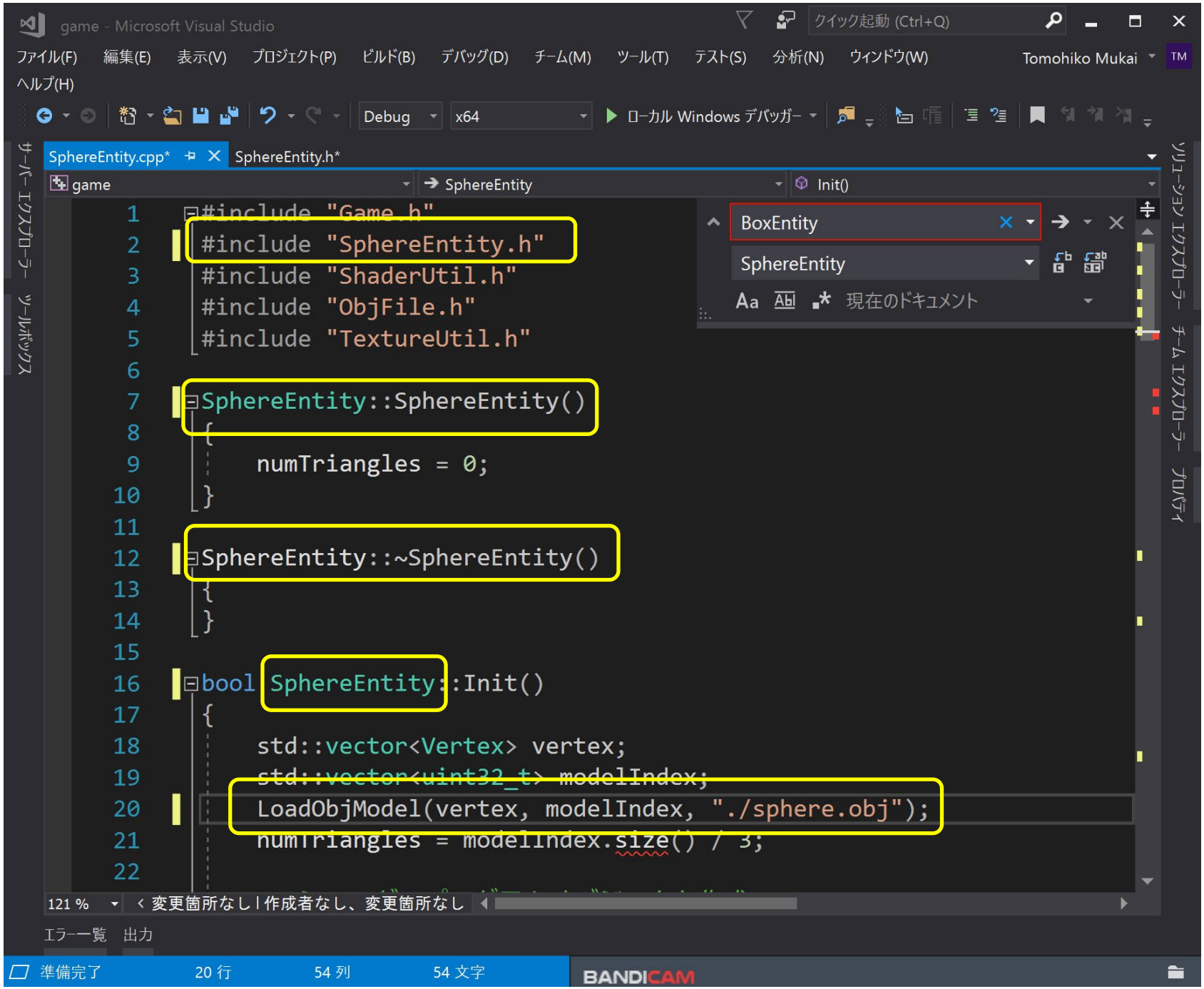
作業手順1

1. BoxEntity.cpp&BoxEntity.h をコピー＆リネーム
   - 例えば、SphereEntity.cpp と SphereEntity.h に
2. gameプロジェクトに追加
   - ファイルをVisual Studio にdrag&drop
3. SphereEntity.cpp と SphereEntity.h 内、「BoxEntity」文字列を「SphereEntity」で置換


game
ファイル
ホーム
共有
表示
GitHub › Game-Programming-B › game
gameの検索
名前
更新日時
種類
glm 2018/12/04 21:42 ファイル フォルダー
include 2018/12/04 21:42 ファイル フォルダー
lib 2018/12/04 21:42 ファイル フォルダー
BoxEntity - コピー.cpp 2018/12/04 21:42 C++ Source
BoxEntity - コピー.h 2018/12/04 21:42 C/C++ Header
BoxEntity.cpp 2018/12/04 21:42 C++ Source
BoxEntity.h 2018/12/04 21:42 C/C++ Header
game - Microsoft Visual Studio
Debug
x64
ローカル Windows デバッガー
Tomohiko Mukai
Drag
&
Drop
SphereEntity.cpp 2018/12/04 21:42 C++ Source
SphereEntity.h 2018/12/04 21:42 C/C++ Header
31 個の項目
2 個の項目を選択 6.43 KB
ソリューション エクスプローラー
game
BoxEntity.cpp
BoxEntity.h
Entity.cpp
Entity.h
Game.cpp
Game.h
game.sln
GameTime.cpp
GameTime.h
lambert.frag
lambert.vert
main.cpp
ObjFile.cpp
ObjFile.h
ShaderUtil.cpp
ShaderUtil.h
SimpleGame.cpp
SimpleGame.h
TextureUtil.cpp
TextureUtil.h

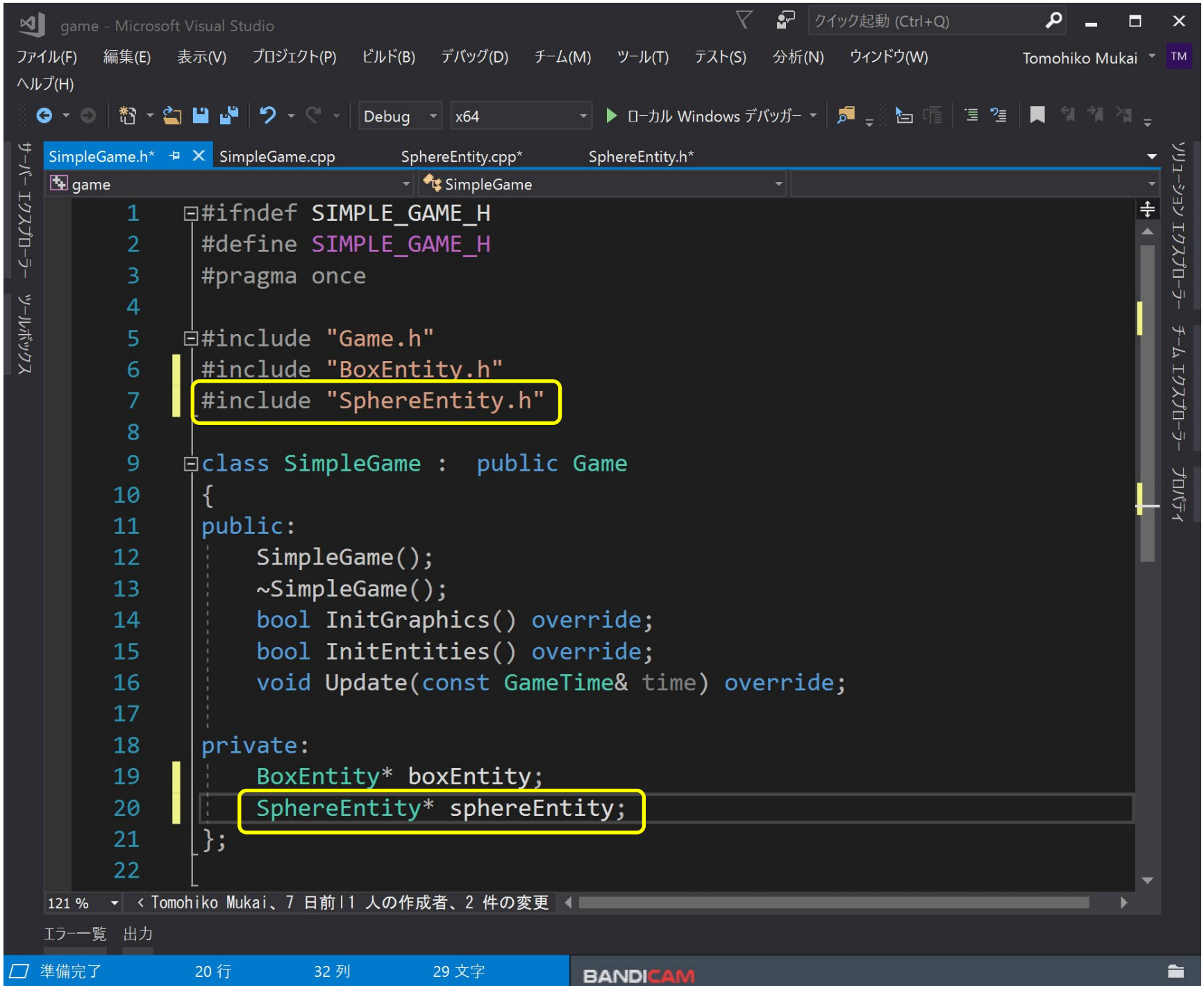
作業手順2

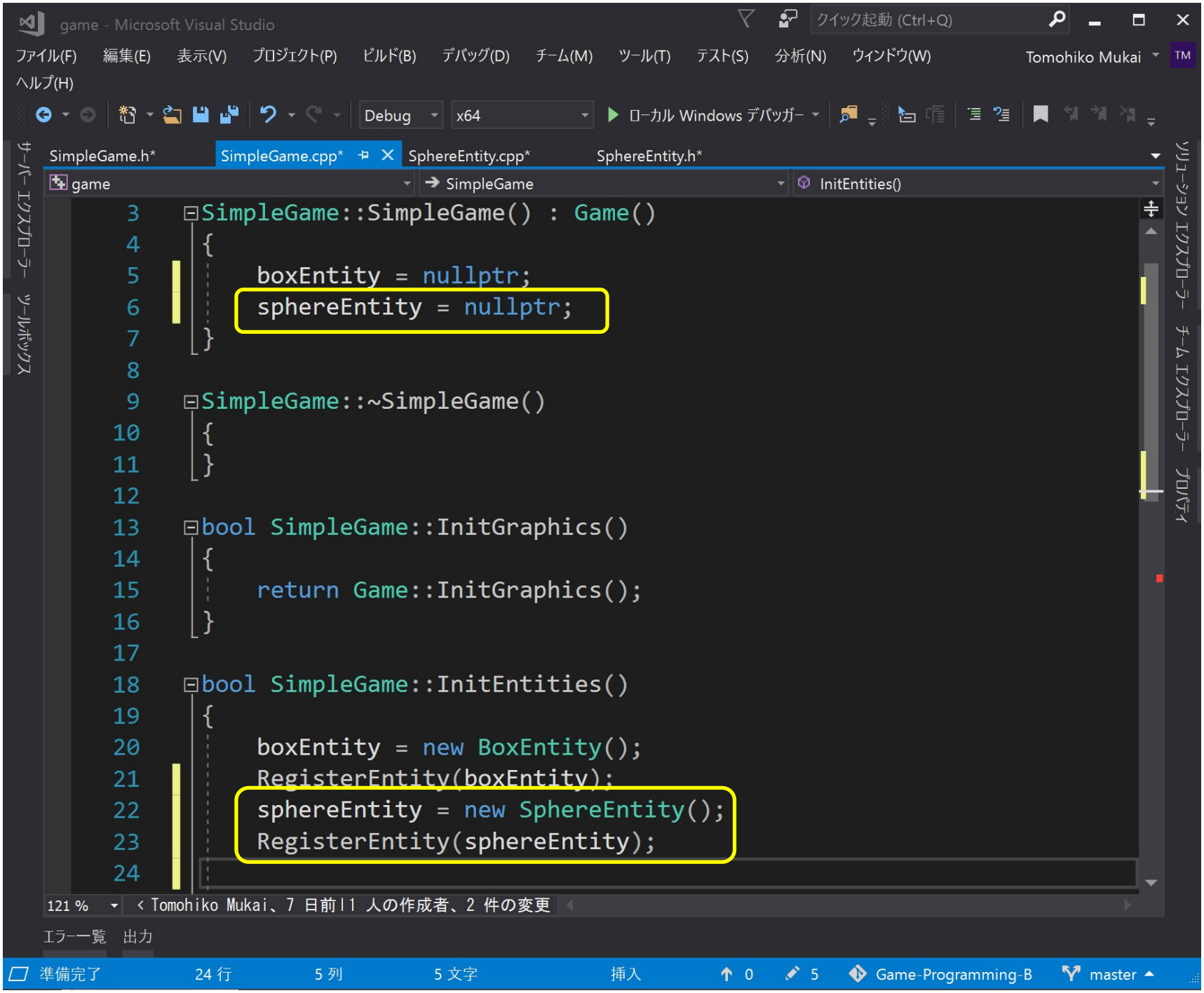
4. 表示する物体形状をモデリング
   - Game-Programming-B の wikiを参照
5. SphereEntity.cpp内、ロードするOBJファイル名を適切に変更
6. （シェーダーの変更が必要な場合は、lambert.vertやlambert.fragをコピー＆リネーム→内容を編集→SphereEntity.cpp内で適切に指定、という操作を行う）


game - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) Tomohiko Mukai
ヘルプ(H)
Debug x64 ローカル Windows デバッガー
SphereEntity.cpp* SphereEntity.h*
game SphereEntity Init()
BoxEntity
SphereEntity
現在のドキュメント
#include "Game.h"
#include "SphereEntity.h"
#include "ShaderUtil.h"
#include "ObjFile.h"
#include "TextureUtil.h"
SphereEntity::SphereEntity()
{
numTriangles = 0;
}
SphereEntity::~SphereEntity()
{
}
bool SphereEntity::Init()
{
std::vector<Vertex> vertex;
std::vector<uint32_t> modelIndex;
LoadObjModel(vertex, modelIndex, "./sphere.obj");
numTriangles = modelIndex.size() / 3;
121 %
変更箇所なし | 作成者なし、変更箇所なし
エラー一覧 出力
準備完了 20 行 54 列 54 文字
BANDICAM

作業手順3

7. SimpleGame.h内
   - SphereEntity.hをinclude
   - SphereEntityへのポインタ型のメンバ変数を追加
8. SimpleGame.cpp内
   - コンストラクタ内でメンバ変数を nullptr で初期化
   - InitEntity内で、SphereEntityオブジェクトをnewしてメンバ変数に代入
   - RegisterEntity関数を用いてエンティティをゲームに登録


game - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) Tomohiko Mukai
ヘルプ(H)
Debug x64 ローカル Windows デバッガー
SimpleGame.h* SimpleGame.cpp SphereEntity.cpp* SphereEntity.h*
game SimpleGame
#ifndef SIMPLE_GAME_H
#define SIMPLE_GAME_H
#pragma once
#include "Game.h"
#include "BoxEntity.h"
#include "SphereEntity.h"
class SimpleGame : public Game
{
public:
SimpleGame();
~SimpleGame();
bool InitGraphics() override;
bool InitEntities() override;
void Update(const GameTime& time) override;
private:
BoxEntity* boxEntity;
SphereEntity* sphereEntity;
};
121 % Tomohiko Mukai、7 日前 | 1 人の作成者、2 件の変更
エラー一覧 出力
準備完了 20 行 32 列 29 文字
BANDICAM


game - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) Tomohiko Mukai
ヘルプ(H)
Debug x64 ローカル Windows デバッガー
SimpleGame.h* SimpleGame.cpp* SphereEntity.cpp* SphereEntity.h*
game SimpleGame InitEntities()
3 SimpleGame::SimpleGame() : Game()
4 {
5 boxEntity = nullptr;
6 sphereEntity = nullptr;
7 }
8
9 SimpleGame::~SimpleGame()
10 {
11 }
12
13 bool SimpleGame::InitGraphics()
14 {
15 return Game::InitGraphics();
16 }
17
18 bool SimpleGame::InitEntities()
19 {
20 boxEntity = new BoxEntity();
21 RegisterEntity(boxEntity);
22 sphereEntity = new SphereEntity();
23 RegisterEntity(sphereEntity);
24
121 % Tomohiko Mukai、7 日前 | 1 人の作成者、2 件の変更
エラー一覧 出力
準備完了 24 行 5 列 5 文字 挿入 0 5 Game-Programming-B master

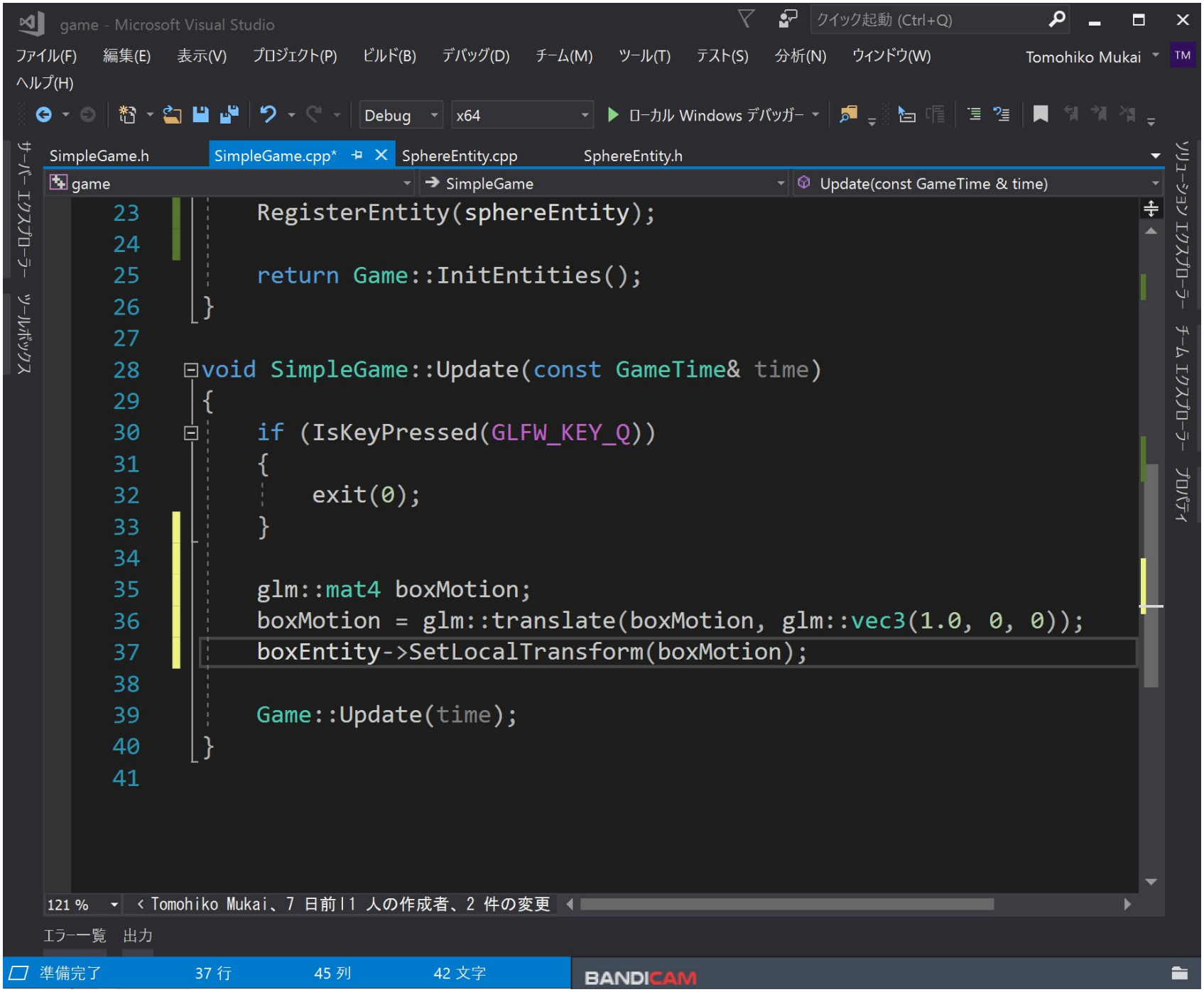
エンティティごとに色を変える方法

1. 各エンティティクラス内、vModelDiffuse（モデルの色）を適宜変更する
   - さらにはSimpleGameクラスから色変更できるようにする　＝　各Entityに色変更用メンバ関数を追加してSimpleGameクラス内から操作

2. シェーダー内に色変更コードを書く
   - 少々アドバンスなやり方
   - どんなイベントが起きたらどんな色になるか、シェーダー内に記述する（前回の応用）

エンティティを動かす方法

1. 各Entityクラス内のUpdate関数に動作を記述する
   - メリット:　同じ動きをするエンティティの場合、楽
   - デメリット: 上記の逆
2. SimpleGameクラス内のUpdate関数に動作を記述する
   - メリット: エンティティ同士の関係を考慮できる
   - デメリット: エンティティの数が増えると大変


game - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Tomohiko Mukai
Debug
x64
ローカル Windows デバッガー
SimpleGame.h
SimpleGame.cpp*
SphereEntity.cpp
SphereEntity.h
game
SimpleGame
Update(const GameTime & time)
23 RegisterEntity(sphereEntity);
24
25 return Game::InitEntities();
26 }
27
28 void SimpleGame::Update(const GameTime& time)
29 {
30 if (IsKeyPressed(GLFW_KEY_Q))
31 {
32 exit(0);
33 }
34
35 glm::mat4 boxMotion;
36 boxMotion = glm::translate(boxMotion, glm::vec3(1.0, 0, 0));
37 boxEntity->SetLocalTransform(boxMotion);
38
39 Game::Update(time);
40 }
41
121 %
Tomohiko Mukai、7 日前 | 1 人の作成者、2 件の変更
エラー一覧 出力
準備完了 37 行 45 列 42 文字
BANDICAM
サーバー エクスプローラー ツールボックス
ソリューション エクスプローラー チーム エクスプローラー プロパティ