内容

[１．作品概要 2](#_Toc69894480)

[２．オブジェクトのデータ管理 3](#_Toc69894481)

[３．オフスクリーンレンダリング 4](#_Toc69894482)

[４．デコボコ処理 5](#_Toc69894483)

[５．塊にくっついた時のオブジェクトの挙動 6](#_Toc69894484)

[６．塊とオブジェクトの巻き込み判定 7](#_Toc69894485)

[７．Cascade Shadow Map 8](#_Toc69894486)

[８．Strategyパターン 9](#_Toc69894487)

[９．鏡面反射 10](#_Toc69894488)

[１０．Qiitaの記事 11](#_Toc69894489)

[１１．URL 13](#_Toc69894490)

**河原電子ビジネス専門学校 ゲームクリエイター科**

**小村　篤**

# １．作品概要

カタタママリシイ



・対応ハード

　PC Windows10

・製作人数

　１人

・製作期間

　2019年５月～ 2020年３月、2020年7月～2020年8月

・製作環境

エンジン

　 学校内製のエンジン(DirectX11)

ツール

　　 Microsoft VisualStudio2019

　3DSMAX2020

　FireAlpaca

　GitHub

　　　Effekseer

RenderDoc

　 使用言語

　　 C++　Python3.7

・担当箇所

Gameプロジェクト、このポートフォリオに記載しているエンジン部分

# ２．オブジェクトのデータ管理

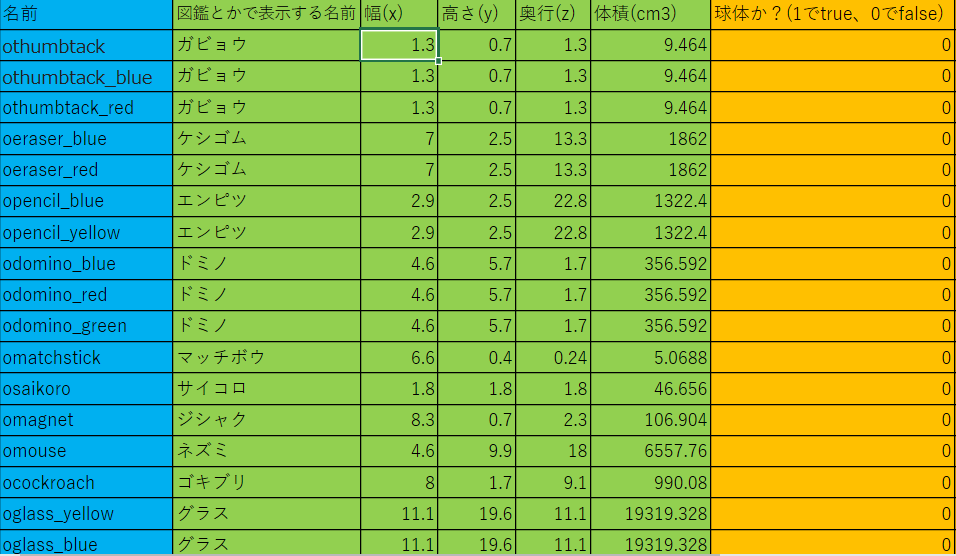
　・参照

PythonApplication1 Game/Object/ObjectData.h,cpp

ゲーム内には様々なオブジェクトがありますが、それらのデータはExcelで管理しています。そのデータをPythonでテキストドキュメントに

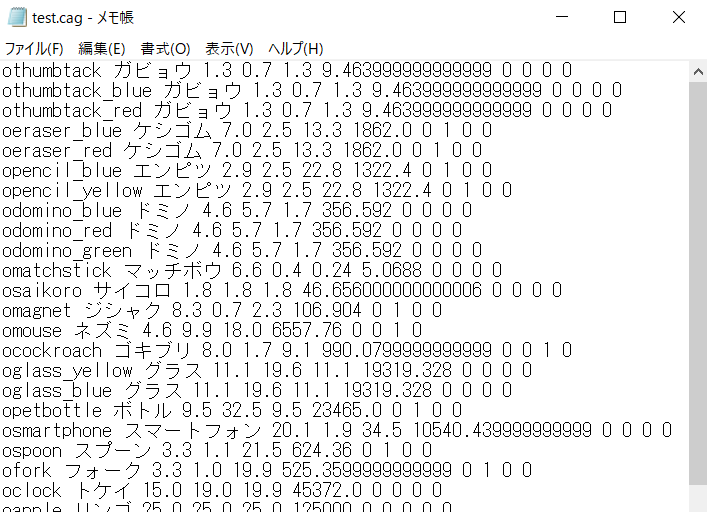
変換して、そのテキストドキュメントをゲームで読み込んでいます。

Excel



Python

テキストドキュメント



ゲームで読み込む

# ３．オフスクリーンレンダリング

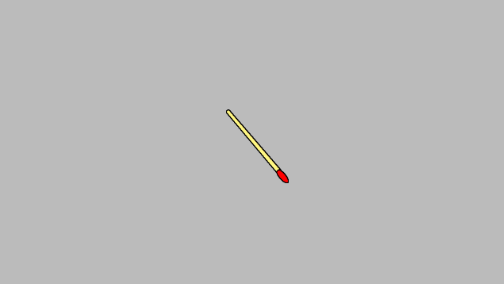
　・参照

Game/OffScreen.h .cpp Game/Object/Collection.h .cpp

ゲーム中に右下に巻き込んだものを表示していますが、これはフレ

－ムバッファとは別のレンダリングターゲットにモデルを描画した

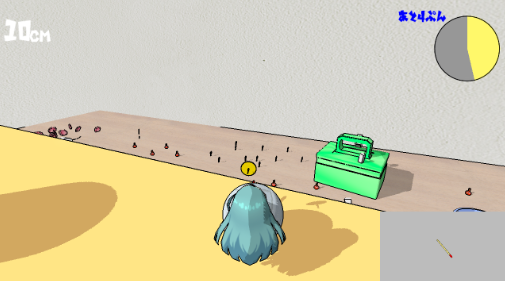
のを、フレームバッファに合成しています。

 フレームバッファ 　　オフスクリーン

テクスチャとして合成



　　　　　　フレームバッファ

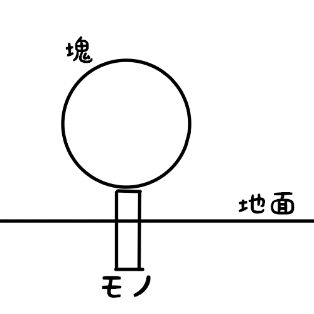


# ４．デコボコ処理

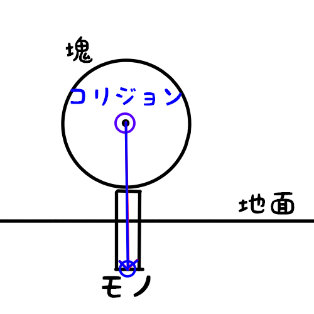
あるオブジェクトを巻き込むと、オブジェクトの形に合わせて塊が動く

　・参照

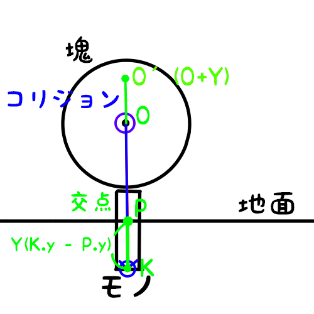
Game/LineSegment.h .cpp

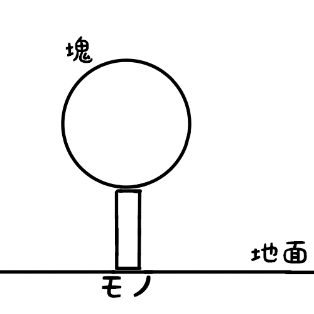


オブジェクトがこのように塊にくっついている場合、



コリジョンを塊の中央からオブジェクトの先端まで移動させます。





この時コリジョンが地面と衝突したら、オブジェクトの先端から交点までの高さを求めて、その高さを塊の座標に加算します。

# ５．塊にくっついた時のオブジェクトの挙動

　・参照

Game/Object/Obj.h .cpp

スキニングとよく似ているのですが、オブジェクトが塊にくっついた時、まず1フレーム目に

**オブジェクトのワールド行列×塊の逆ワールド行列**

**＝塊座標系のオブジェクトのローカル行列**

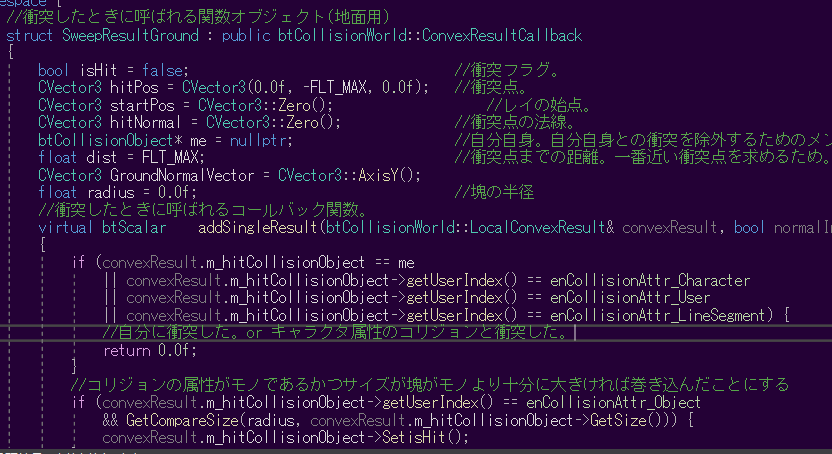
　をして塊を中心としたオブジェクトのローカル行列を求めます。

　そして毎フレーム

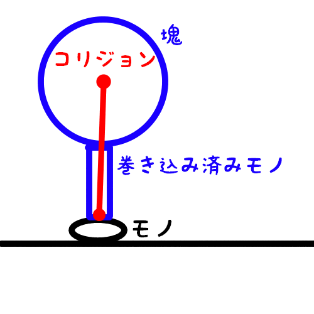
**塊座標系のオブジェクトのローカル行列×塊のワールド行列＝オブジェクトのワールド行列**

をしてオブジェクトのワールド行列を求めています。

# ６．塊とオブジェクトの巻き込み判定



これが塊のコリジョンが他のコリジョンと衝突した際のコールバック関数の一部なのですが、オブジェクトのコリジョンと衝突した時に、塊のサイズがオブジェクトより十分に大きければ、巻き込んだことにしています。



またこのように尖っているオブジェクトを巻き込んでいる場合には、ガタガタ処理のためにコリジョンをオブジェクトの先端まで移動させるのですが、その際も同様の処理をしています。これにより、尖ったオブジェクトを巻き込んでいてもオブジェクトを巻き込みやすくしています。

# ７．Cascade Shadow Map

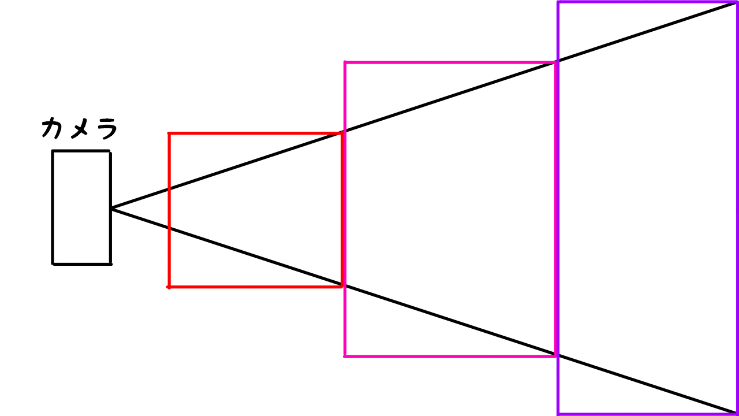
・参照

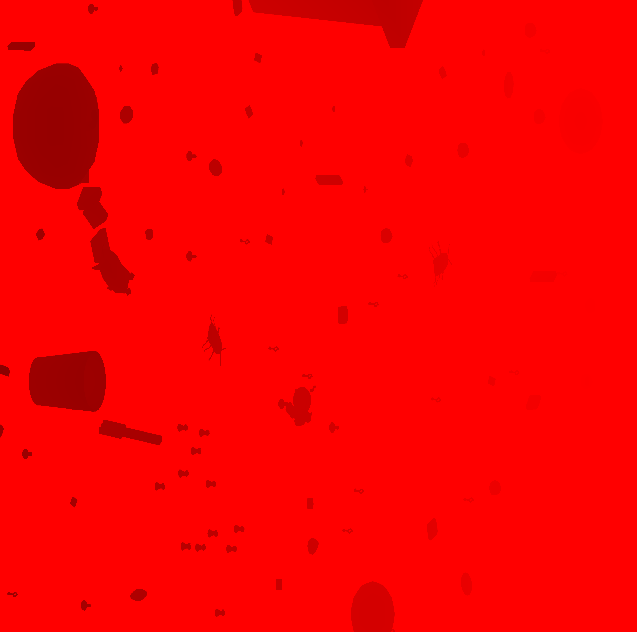
kgEngine/Graphics/shadow/CascadeShadowMap.h .cpp

Game/Assets/shader/model.fx

視錘台をカメラに近い場所から３つに分割して、シャドウマップを生成しています。

　　　　　　　　　　 ライト









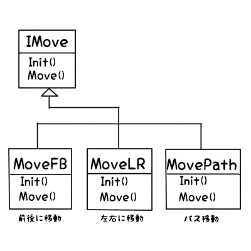
メモリの使用量を抑えるために、カメラから距離があるシャドウマップは解像度を落としています。

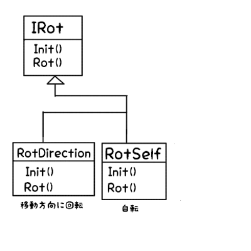
# ８．Strategyパターン

・参照

Game/Object/Obj.cpp

オブジェクトの移動や回転の挙動をStrategyパターンで実装しています。





# ９．鏡面反射

・参照

Game/Assets/shader/model.fx

ステージに登場するオブジェクトの中で、宝石類には鏡面反射ライティングを適用して金属のような表現をしています。



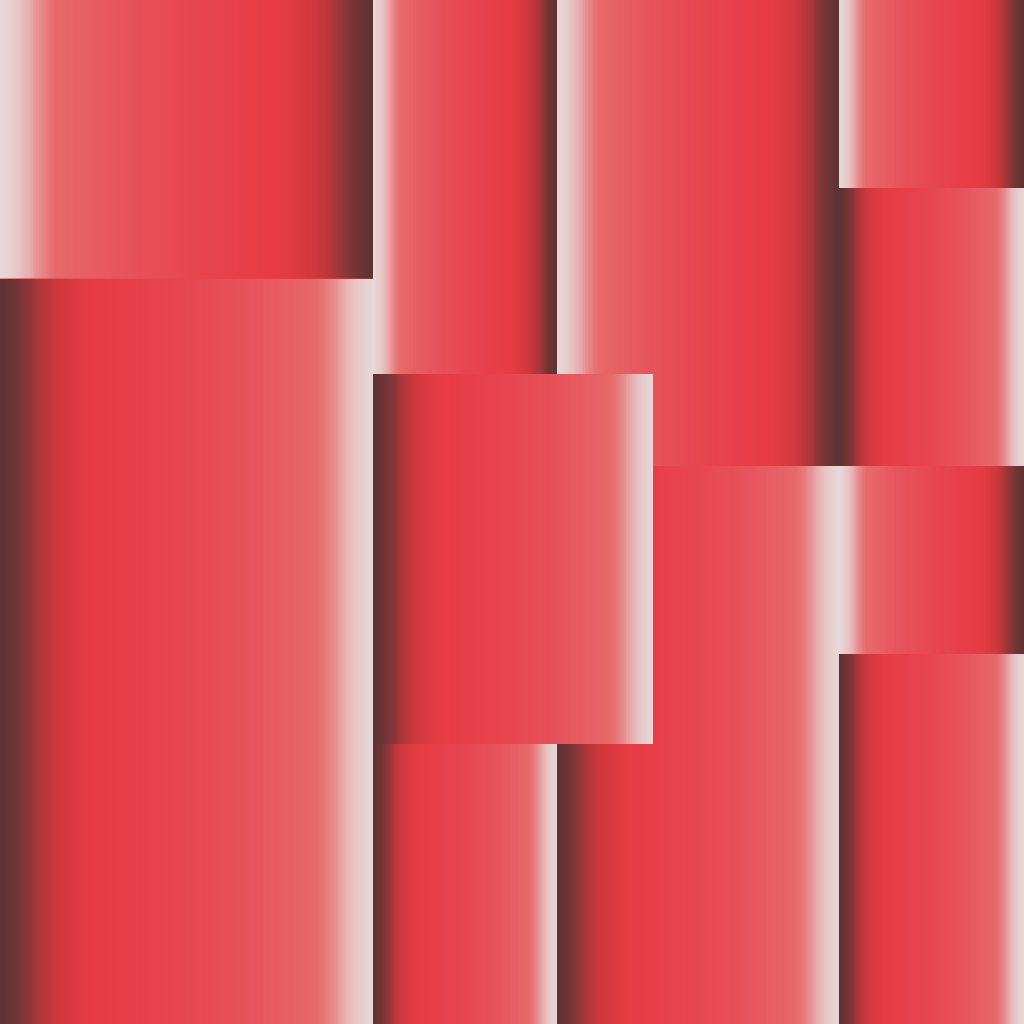
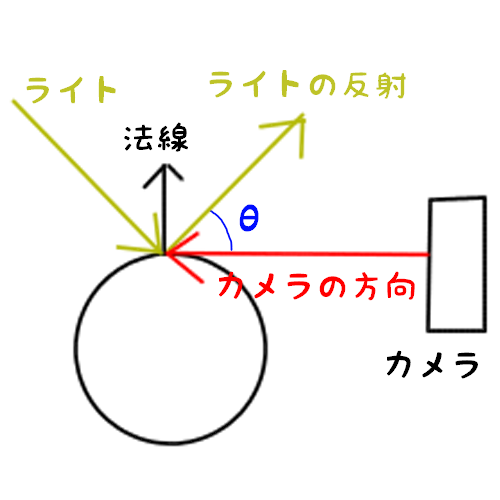
カメラの方向によって光

り方が違う



ライトの方向と法線からライトの反射ベクトルRを求めて、Rとカメラの方向ベクトルの角度θを求めます。θとスペキュラマップを用いて、光の強さを決定しています。

　　　　　　　　　　　　　　　　　　　　　スペキュラマップ



# １０．Qiitaの記事

以下Qiitaで私が書いた記事のURLです。

実際にカタタママリシイに実装しています。

・エッジ検出

　・参照

kgEngine/graphics/normal/EdgeDetection.h .cpp

Game/Assets/shader/edgedelection.fx

<https://qiita.com/akurobit/items/b5231ffae738810b63e7>

・円形ゲージ

・参照

kgEngine/graphics/2D/Sprite.h .cpp

Game/Assets/shader/sprite.fx

<https://qiita.com/akurobit/items/ab90b88088678f706e3a>

・トゥーンシェーダー

・参照

Game/Assets/shader/model.fx

<https://qiita.com/akurobit/items/a016ef4a022eed15c268>

・被写体が全部移るためのカメラ設定

・参照

Game/Collection.cpp

<https://qiita.com/akurobit/items/a6dd03baef6c05d7eae8>

・画面分割

・参照

kgEngine/graphics/GraphicsEngine.h .cpp

<https://qiita.com/akurobit/items/1619bc26010441b8008c>

# １１．URL

GitHubのURLです。このゲームのリポジトリです。

<https://github.com/komura-athushi/kgEngine.git>

プレイ動画

ソロプレイです。

<https://www.youtube.com/watch?v=jlXwaBkYB7E>

対戦プレイです。

<https://www.youtube.com/watch?v=2ZYI-CVe75c>