17　OpenGL を使ったゲーム作成

遠藤　英雄

指導教員　小笠原　祐治

1.　はじめに

私は，以前からゲームを作成したいと思っていた．また，私が志望している会社が OpenGL を使える人を募集していたので，OpenGL を用いて 3D ゲームを作成することをテーマに選定した．

2.　研究概要

2.1　目的

本研究では，OpenGL を習得することを目的に，3D ゲームを作成する．

2.2　開発環境

OS	Windows7
使用言語	C
使用ソフト	Microsoft Visual C++ OpenGL GLUT

表 1．開発環境

3.　OpenGL について

3.1　OpenGL とは

1992 年から規格化が行われている標準的なグラフィックスハードウェアの API である．

3.2　GLUT について

表示を行うための処理は，OS 毎に異なる．Windows と Linux で同じプログラムソースを実行することはできない．OS により異なる部分を吸収するため，AUX ライブラリが用意されている．しかし，AUX ライブラリはもともと学習用であり，実際のアプリケーションを書こうとすると機能に不足がある．

そこで AUX ライブラリを改良したものが GLUT (The OpenGL Utilitiy Toolkit)である．

3.3　描画処理の流れ

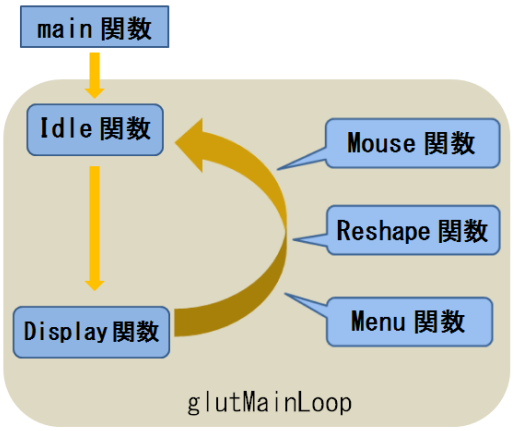

図 1．Main 関数のフローチャート

3.4　OpenGL ライブラリ

＜図 1 の関数の役割＞

・main 関数

初期化（ウィンドウ生成，関数も登録），ループ開始（glutMainLoop）

・Idle 関数

再描画指示による Dispaly 関数の呼び出し(glutRedispalay)

・Display 関数

描画処理（コマ，盤）画の更新

・Mouse 関数

マウスクリックの判定

・Reshape 関数

ウィンドウサイズが変更されたら呼び出す

3.5　glpng について

オセロの盤に張り付けるテクスチャに png 画像を使用した．しかし，GLUT には png 画像を読み込む関数が存在していなかったので，glpng ライブラリを使用した．マッピングの仕方は，pngBind 関数で読み込んだ png 画像を glTexCoord2f 関数を使い，生成したデータにテクスチャを張り付ける．

3.6　アニメーション

アニメーションは、フレーム画像を，短い時間間隔で連続的に描画し，物体が動いているように表示するものである．各フレーム画像は，物体が少しずつ移動している画像である．短い時間間隔で描画するために， glutTimerFunc 関数を用いる．この関数は，指定時間（ミリ秒単位）が経過すると指定した関数を呼び出す設定を行う．経過時間と関数は引数で指定する．

今回は，描画関数 disp を 30 ミリ秒間隔でアニメーションフレームが終了するまで glutTimerFunc で予約し実行しています．

＜アニメーションの仕方＞

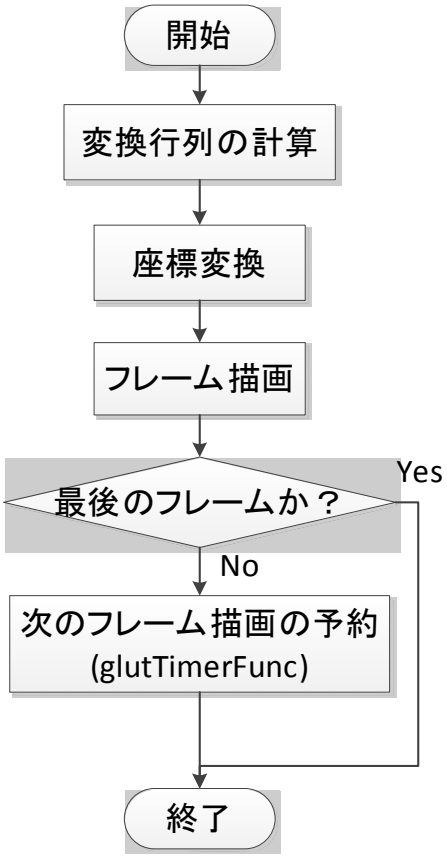

図２．描画手順のフローチャート

フレーム画像におけるコマの描画は，コマの形状データに座標変換行列を掛けて行う．まず，現在の変換行列を保存(glPushMatrix)し，移動(glTranslated), 回転 (glRotated)の変換行列を掛けて行く．最後に，保存していた変換行列に復帰する(glPushMatrix 関数).

4.オセロ概要

4.1　ゲーム画面

ゲーム画面を図３に示す．

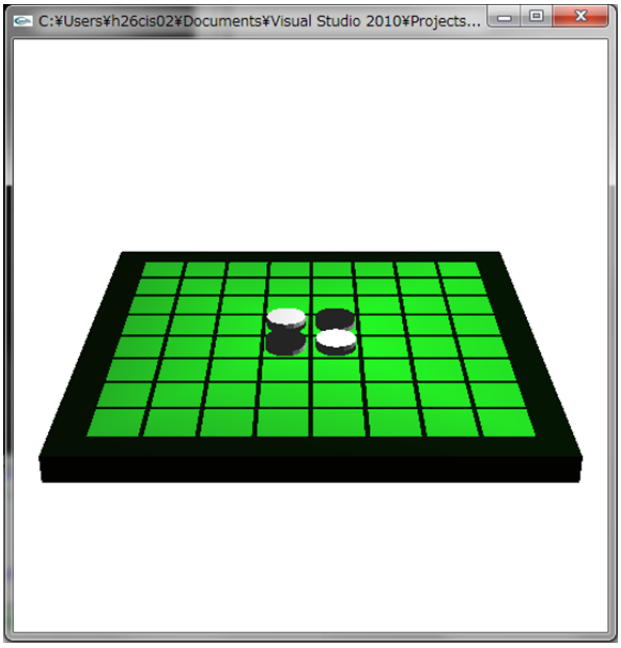

図３．ウィンドウイメージ

4.2　オブジェクト（コマ・盤）

コマや盤のデータは java プログラムを使って作成した．コマの種類は「白黒コマ」,「顔を彫ったコマ」などの複数パターンを用意する。盤面の状況は配列で管理する．

4.3　操作デバイス

基本操作はマウスで行う．

主な操作はメニュー操作，マウス操作切替，コマ置きである．

①　メニュー操作

右クリックでポップアップメニューを表示する．

・視点操作（視点移動の切替）

・コマ選択（コマの種類の設定）

→資料の 4.2 より，2 パターン．

・反転パターン（反転の仕方の設定）

→反転パターンは 2 つ

1. コマが一つずつ反転する．
2. コマが一斉に反転する．

・反転動作（反転アニメーションの設定）
→資料のアニメーションより，4 パターン．
・リスタート（新たにゲームを開始）

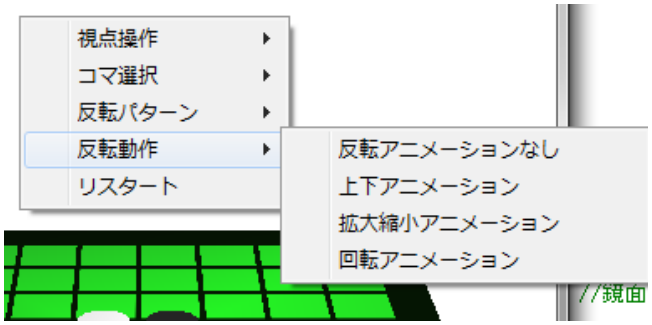

図４．メニューイメージ

② マウス操作切替

マウス操作が，視点移動か，コマ置きかを切り替える．視点移動とは，盤を見る高さを変更するものである．

③ コマ置き

左クリックでコマを置くことができる．

置いた位置で，hanten_count 関数を用いて反転するコマの数を調べる．反転コマ数が 1 以上の場合，そこにコマを置くことができる．反転コマ数を求める際には，図 5 に示すように八方向について調べる．各方向の反転コマ数の合計を求める．方向ごとの反転コマ数を求める関数のフローチャートを図 6 に示す．

x, y はコマを置く位置を示し，mx, my は調べる方向を示す．

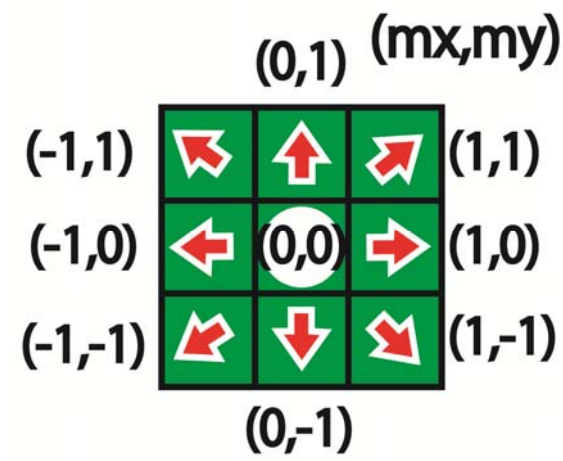

図５．コマ置き探索イメージ

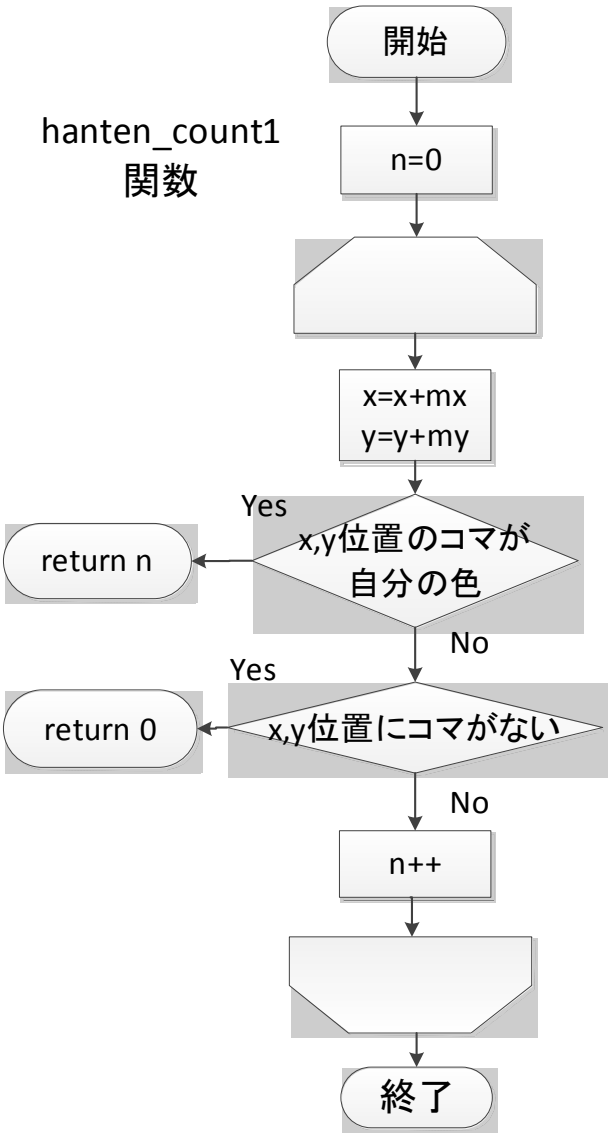

図６．反転検出フローチャート

4.4 反転アニメーション

コマ置きの後，反転するコマのアニメーションを行う．

反転アニメーションは，以下に示す 4 アニメーションパターンがある．

3.6 で示した方法で，1 秒（30 フレーム）のアニメーションを行う．

① 反転アニメーションなし

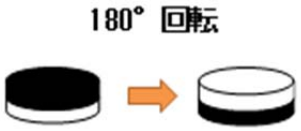

図７．アニメーションパターン 1

② 上下アニメーション

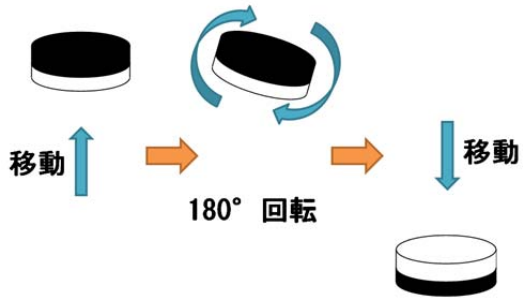

図８．アニメーションパターン２

③ 拡大縮小アニメーション

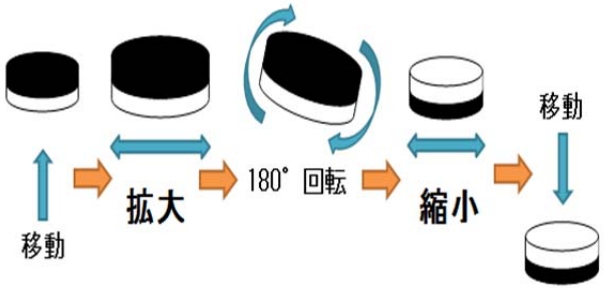

図９．アニメーションパターン３

④ 回転アニメーション

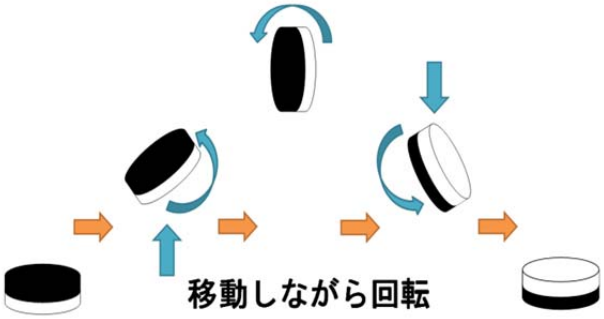

図１０．アニメーションパターン４

4.5 CPU 側のアルゴリズムについて

CPU 側は自分のコマが最も多くなるところにコマを置く．

盤上の全てのマス目で hanten_count 関数を用いて反転コマ数を求める．最も反転コマ数が多いマス位置に自分のコマを置く．

置いた後，反転アニメーションを行い，プレイヤーにコマ置きの主導権を譲渡する．

5. 終わりに

視点移動操作，拡大縮小と回転のアニメーションがまだ実装できていないことと，ゲームの終了条件などの処理がまだできていない．卒業までに必ず完成させたい．

6. 参考文献

https://ja.wikipedia.org/wiki/OpenGL
http://blender.jp/modules/newbb/viewtopic.php?viewmode=flat&topic_id=1123&forum=3
http://individuals.iii.u-tokyo.ac.jp/~yoichiro/acg/docs/tutorialgl.pdf