演習課題 08 (05月31日）レポート

交換留学(文学部)　ES19-0013 ジョユンサン  
課題 8

発展課題8

*// Created by Jho on 31/05/2019.*

*// Copyright © 2019 Jho. All rights reserved.*

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include "cglec.h"

#define WIDTH 400

#define HEIGHT 400

#define PI 3.14

**struct** Point3D *//3*次元の座標点

{

**double** x;

**double** y;

**double** z;

};

**void** AffineTransform(Point3D p[], **int** n, **double** m[4][4]) *//n*個の点をアフィン行列*m*で変換

{

**int** i;

**for** (i = 0; i < n; i++)

{

**double** x = p[i].x, y = p[i].y, z = p[i].z;

p[i].x = m[0][0] \* x + m[0][1] \* y + m[0][2] \* z + m[0][3];

p[i].y = m[1][0] \* x + m[1][1] \* y + m[1][2] \* z + m[1][3];

p[i].z = m[2][0] \* x + m[2][1] \* y + m[2][2] \* z + m[2][3];

}

}

*// struct Point, CglDrawLine(), CglDrawLines()*は*"cglec.h"*で定義済み

*// struct Point3D, AffineTransform()*はこれ以前で定義・宣言しておく

**int** main(**void**)

{

**unsigned** **char**\*\* data = (**unsigned** **char**\*\*)malloc(**sizeof**(**unsigned** **char**\*) \* WIDTH);

**for** (**int** i = 0; i < HEIGHT; i++)

{

data[i] = (**unsigned** **char**\*)malloc(HEIGHT \* **sizeof**(**char**));

}

Image img = { (**unsigned** **char**\*)data, WIDTH, HEIGHT };

Point3D cube[] = { *//*立方体の線図形データ

{1, -1, 1}, {1, -1, -1}, {1, -1, -1}, {-1, -1, -1}, *//*底面

{-1, -1, -1}, {-1, -1, 1}, {-1, -1, 1},{1, -1, 1},

{1, 1, 1}, {1, 1, -1}, {1, 1, -1}, {-1, 1, -1}, *//*上面

{-1, 1, -1}, {-1, 1, 1}, {-1, 1, 1},{1, 1, 1},

{1, -1, 1}, {1, 1, 1}, {1, -1, -1}, {1, 1, -1}, *//*縦線

{-1, -1, -1}, {-1, 1, -1}, {-1, -1, 1}, {-1, 1, 1} };

**int** N = 24; *//* 点データの数は*24*個

**double** r = 15 \* PI / 180;

**double** T[4][4] =

{ 1,0,0,200,

0,1,0,200,

0,0,1,200,

0,0,0,1};

**double** S[4][4] =

{100,0,0,0,

0,100,0,0,

0,0,100,0,

0,0,0,1};

**double** Rx[4][4] =

{ 1,0,0,0,

0,cos(r),-sin(r),0,

0,sin(r),cos(r),0,

0,0,0,1 };

**double** Ry[4][4] =

{cos(r),0,sin(r),0,

0,1,0,0,

-sin(r),0,cos(r),0,

0,0,0,1 };

CglSetAll(img, 0);

AffineTransform(cube, N, Rx);

AffineTransform(cube, N, Ry);

AffineTransform(cube, N, S);

AffineTransform(cube, N, T);

Point p[24]; *//* 正投影後の点データを入れる*2*次元の配列

**for** (**int** i = 0; i < N; i++)

{

p[i].x = (**int**)cube[i].x;

p[i].y = (**int**)cube[i].y;

}

CglDrawLines(img, p, N, 255);

CglSaveGrayBMP(img, "cube.bmp");

}

