LAB 7. SOLAR SYSTEM

컴퓨터 그래픽스

남병규 교수님

201002506 진재연

LAB의 목적

OpenGL의 행렬 스택, 라이팅 등을 종합적으로 사용하여 태양계를 구현해보는 것이 목적입니다.

소스코드

//

// main.cpp

// GLProgramming

//

// Created by jinjae-yeon on 2014. 11. 19..

// Copyright (c) 2014년 DARAM WORLD. All rights reserved.

//

#include <GLUT/GLUT.h>

#include <OpenGL/OpenGL.h>

#include <vector>

#include <sys/time.h>

class Planet

{

friend void idle ();

friend void display ();

public:

Planet ( float distanceFrom = 0, float rotationAxis = 0, float rotationRate = 0, float skiddingRate = 0 ) : m\_distanceFrom ( distanceFrom ), m\_rotationAmount ( 0 ), m\_skiddingAmount ( 0 ), m\_rotationAxis ( rotationAxis ), m\_rotationRate ( rotationRate ), m\_skiddingRate ( skiddingRate )

{

innerPlanet = nullptr;

nextPlanet = nullptr;

}

virtual ~Planet ()

{

if ( innerPlanet != nullptr )

delete innerPlanet;

if ( nextPlanet != nullptr )

delete nextPlanet;

}

public:

virtual void update () { }

virtual void render () { }

private:

void onUpdate ()

{

m\_skiddingAmount += m\_skiddingRate;

m\_rotationAmount += m\_rotationRate;

update ();

if ( innerPlanet != nullptr )

innerPlanet->onUpdate ();

if ( nextPlanet != nullptr )

nextPlanet->onUpdate ();

}

void onRender ()

{

glPushMatrix ();

glRotatef(m\_rotationAmount, 0, 1, 0);

glRotatef(-m\_rotationAxis, 0, 0, 1);

glTranslatef(m\_distanceFrom, 0, 0);

glRotatef(m\_skiddingRate, 0, 1, 0);

glRotatef(m\_rotationAxis, 0, 0, 1);

if ( innerPlanet != nullptr )

innerPlanet->onRender ();

render ();

glPopMatrix ();

if ( nextPlanet != nullptr )

nextPlanet->onRender ();

}

protected:

float m\_rotationAxis;

float m\_rotationRate;

float m\_skiddingRate;

float m\_distanceFrom;

protected:

Planet \* innerPlanet;

Planet \* nextPlanet;

private:

float m\_skiddingAmount;

float m\_rotationAmount;

};

class Neptune : public Planet

{

public:

Neptune () : Planet ( 240, 0, 3.22f, 0 )

{

}

void render ()

{

GLfloat ambient[] = {0.4, 0.4, 0.9, 1.0};

GLfloat diffuse[] = {0.4, 0.4, 0.9, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 6, 40, 40 );

}

};

class Uranos : public Planet

{

public:

Uranos () : Planet ( 200, 0, 2.22f, 0 )

{

nextPlanet = new Neptune ();

}

void render ()

{

GLfloat ambient[] = {0.2, 0.2, 0.9, 1.0};

GLfloat diffuse[] = {0.2, 0.2, 0.9, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 6, 40, 40 );

}

};

class Saturn : public Planet

{

public:

Saturn () : Planet ( 160, 5.25f, 3.22f, 0 )

{

nextPlanet = new Uranos ();

}

void render ()

{

GLfloat ambient[] = {0.6, 0.4, 0.3, 1.0};

GLfloat diffuse[] = {0.6, 0.4, 0.3, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 12, 40, 40 );

}

};

class Ganymede : public Planet

{

public:

Ganymede () : Planet ( 30, 0, 2.22f, 0 )

{

}

void render ()

{

GLfloat ambient[] = {0.6, 0.4, 0.3, 1.0};

GLfloat diffuse[] = {0.6, 0.4, 0.3, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 2, 40, 40 );

}

};

class Io : public Planet

{

public:

Io () : Planet ( 40, 0, 8.22f, 0 )

{

nextPlanet = new Ganymede ();

}

void render ()

{

GLfloat ambient[] = {0.6, 0.4, 0.3, 1.0};

GLfloat diffuse[] = {0.6, 0.4, 0.3, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 3, 40, 40 );

}

};

class Jupiter : public Planet

{

public:

Jupiter () : Planet ( 120, 0, 4.22f, 0 )

{

nextPlanet = new Saturn ();

innerPlanet = new Io ();

}

void render ()

{

GLfloat ambient[] = {0.8, 0.5, 0.2, 1.0};

GLfloat diffuse[] = {0.8, 0.5, 0.2, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 15, 40, 40 );

}

};

class Mars : public Planet

{

public:

Mars () : Planet ( 80, 0, 5.55f, 0 )

{

nextPlanet = new Jupiter ();

}

void render ()

{

GLfloat ambient[] = {0.8, 0.4, 0.2, 1.0};

GLfloat diffuse[] = {0.8, 0.4, 0.2, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 5, 40, 40 );

}

};

class Moon : public Planet

{

public:

Moon () : Planet ( 10, 0, 8.0f, 0 )

{

}

void render ()

{

GLfloat ambient[] = {0.6, 0.4, 0.3, 1.0};

GLfloat diffuse[] = {0.6, 0.4, 0.3, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 3, 40, 40 );

}

};

class Earth : public Planet

{

public:

Earth () : Planet ( 60, 0, 7.0f, 0 )

{

nextPlanet = new Mars ();

innerPlanet = new Moon ();

}

void render ()

{

GLfloat ambient[] = {0.5, 0.4, 0.75, 1.0};

GLfloat diffuse[] = {0.5, 0.4, 0.75, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 7, 40, 40 );

}

};

class Venus : public Planet

{

public:

Venus () : Planet ( 40, 0, 7.25f, 0 )

{

nextPlanet = new Earth ();

}

void render ()

{

GLfloat ambient[] = {0.3, 0.6, 0.2, 1.0};

GLfloat diffuse[] = {0.3, 0.6, 0.2, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 7, 40, 40 );

}

};

class Mercury : public Planet

{

public:

Mercury () : Planet ( 20, 0, 8.5f, 0 )

{

nextPlanet = new Venus ();

}

void render ()

{

GLfloat ambient[] = {0.7, 0.4, 0.2, 1.0};

GLfloat diffuse[] = {0.7, 0.4, 0.2, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0, 0, 0, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 2, 40, 40 );

}

};

class Sun : public Planet

{

public:

Sun ()

{

innerPlanet = new Mercury ();

}

void render ()

{

GLfloat ambient[] = {0.75, 0.4, 0.4, 1.0};

GLfloat diffuse[] = {0.75, 0.4, 0.4, 1.0};

GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

GLfloat shine = 10.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shine);

GLfloat emission[] = {0.4, 0.1, 0.1, 1};

glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);

glutSolidSphere ( 10, 40, 40 );

}

};

void idle ();

void display ();

void reshape ( int width, int height );

void keyboardDown ( unsigned char key, int x, int y );

void keyboardUp ( unsigned char key, int x, int y );

Planet \* sun;

int camera\_x, camera\_y = 240, camera\_z = 260;

bool isKeyDown[256];

int main ( int argc, const char \* argv [] )

{

// GLUT 초기화

glutInit(&argc, (char\*\*)argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(640, 480);

glutCreateWindow("201002506 Jin Jae-yeon");

// 콜백 함수 등록

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboardDown);

glutKeyboardUpFunc(keyboardUp);

glutIdleFunc(idle);

// 라이팅 켜기

glEnable(GL\_LIGHTING);

// 깊이 테스트 켜기

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

sun = new Sun ();

// GLUT 메시지 루프 시작

glutMainLoop();

delete sun;

return 0;

}

void idle ()

{

static long long lastTime;

static unsigned day = 1;

timeval ts;

gettimeofday(&ts, nullptr);

long long currentTime = ts.tv\_sec \* 1000 + ts.tv\_usec / 1000;

if ( ( currentTime - lastTime ) / 1000.0f >= ( 1 / 30.0f ) )

{

char title [ 128 ];

sprintf ( title, "%s (Day: %d)", "201002506 Jin Jae-yeon", ++day );

glutSetWindowTitle(title);

float unit = 2;

if(isKeyDown['w']) camera\_z -= unit;

if(isKeyDown['s']) camera\_z += unit;

if(isKeyDown['a']) camera\_x -= unit;

if(isKeyDown['d']) camera\_x += unit;

if(isKeyDown['q']) camera\_y += unit;

if(isKeyDown['e']) camera\_y -= unit;

sun->onUpdate ();

glutPostRedisplay ();

lastTime = currentTime;

}

}

void display ()

{

glClearColor ( 0, 0, 0, 1 );

glClear ( GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT );

glMatrixMode ( GL\_MODELVIEW );

glLoadIdentity ();

gluLookAt(camera\_x, camera\_y, camera\_z, 0, 0, 0, 0, 1, 0);

glEnable(GL\_LIGHT0);

GLfloat diffuse0[]={1.0, 0.6, 0.6, 0.6};

GLfloat ambient0[]={1.0, 1.0, 1.0, 1.0};

GLfloat specular0[]={1.0, 0.3, 0.3, 0.3};

GLfloat light0\_pos[]={0, 0, 0, 1.0};

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_pos);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient0);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse0);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular0);

sun->onRender ();

glFlush ();

}

void reshape ( int width, int height )

{

glViewport ( 0, 0, width, height );

glMatrixMode ( GL\_PROJECTION );

glLoadIdentity ();

gluPerspective ( 45, width / ( float ) height, 1, 1000 );

}

void keyboardDown ( unsigned char key, int x, int y )

{

isKeyDown[key] = true;

}

void keyboardUp ( unsigned char key, int x, int y )

{

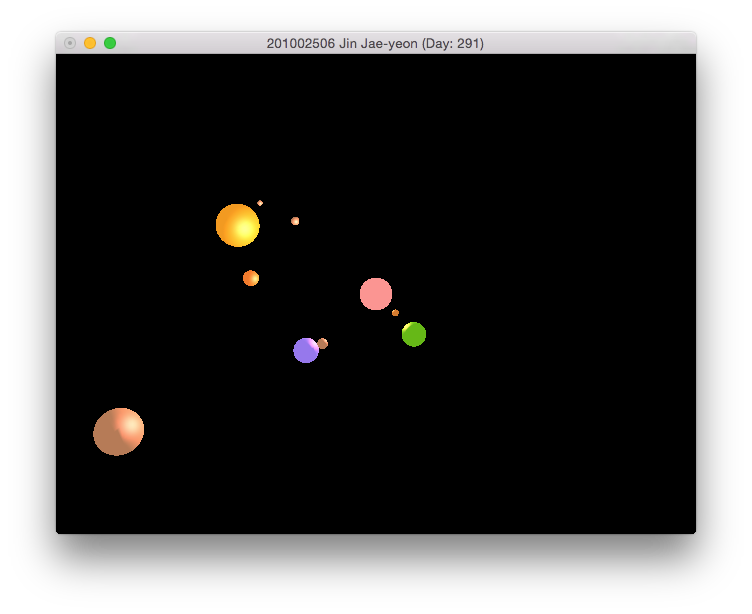
isKeyDown[key] = false;

}

각 행성을 구현하기 위해 C++ Class를 사용하여 행성을 객체화하였으며, 행성마다 기울기와 공전속도, 자전 속도, 모체 행성으로부터의 거리 등을 가질 수 있습니다.

카메라도 W, A, S, D, Q, E키를 이용해 각각 전진, 좌측이동, 후진, 우측이동, 위로, 아래로 움직일 수 있습니다.

결과



논의

이 LAB의 키는 무엇인가?

그동안 공부했던 OpenGL의 행렬 스택과 라이팅을 이용해 행성계를 구현하는 것입니다.

무슨 실수를 하고 무엇을 배웠는가?

행성계가 너무 크고 행성이 많아 카메라의 범위를 잡는데에 어려움이 있었습니다.

프로그램을 어떻게 향상시킬 수 있겠는가?

행성에 무늬를 주어 진짜 행성처럼 보이게 하면 더 멋진 행성계를 구현할 수 있을 것입니다.