LAB 10. TEXTURED SOLAR SYSTEM

컴퓨터 그래픽스

남병규 교수님

201002506 진재연

LAB의 목적

Lab 7에서 구현한 Solar System에 텍스쳐를 붙여 진짜 행성처럼 보이게 하는 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>

#pragma pack (push, 1)

struct BITMAPFILEHEADER

{

unsigned short bfType;

unsigned int bfSize;

unsigned short bfReserved1;

unsigned short bfReserved2;

unsigned int bfOffBits;

} \_\_attribute\_\_((packed));

struct BITMAPINFOHEADER

{

unsigned int biSize;

int biWidth;

int biHeight;

unsigned short biPlanes;

unsigned short biBitCount;

unsigned int biCompression;

unsigned int biSizeImage;

int biXPelsPerMeter;

int biYPelsPerMeter;

unsigned int biClrUsed;

unsigned int biClrImportant;

} \_\_attribute\_\_((packed));

#pragma pack (pop)

GLuint readTexture(const char \* filename)

{

GLuint tex;

glGenTextures(1, &tex);

FILE \* fp = fopen(filename, "rb");

BITMAPFILEHEADER fileHeader;

BITMAPINFOHEADER infoHeader;

fread(&fileHeader, 1, sizeof(BITMAPFILEHEADER), fp);

if(fileHeader.bfType != 0x4D42 || fileHeader.bfReserved1 != 0 || fileHeader.bfReserved2 != 0)

{

glDeleteTextures(1, &tex);

return 0;

}

fread(&infoHeader, 1, sizeof(BITMAPINFOHEADER), fp);

if(infoHeader.biPlanes != 1)

{

glDeleteTextures(1, &tex);

return 0;

}

fseek(fp, fileHeader.bfOffBits, SEEK\_SET);

GLubyte \* pixels = new GLubyte [ 256 \* 256 \* 4 ];

fread(pixels, 1, 256 \* 256 \* 4, fp);

fclose(fp);

glBindTexture(GL\_TEXTURE\_2D, tex);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glTexEnvf(GL\_TEXTURE\_2D, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, 256, 256, 0, GL\_BGRA, GL\_UNSIGNED\_BYTE, pixels);

delete [] pixels;

return tex;

}

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 ), m\_texture ( 0 )

{

innerPlanet = nullptr;

nextPlanet = nullptr;

}

virtual ~Planet ()

{

if ( innerPlanet != nullptr )

delete innerPlanet;

if ( nextPlanet != nullptr )

delete nextPlanet;

if(m\_texture)

glDeleteTextures(1, &m\_texture);

}

public:

virtual void update () { }

virtual void render () { }

private:

void onUpdate ()

{

m\_skiddingAmount += m\_skiddingRate / 3;

m\_rotationAmount += m\_rotationRate / 3;

update ();

if ( innerPlanet != nullptr )

innerPlanet->onUpdate ();

if ( nextPlanet != nullptr )

nextPlanet->onUpdate ();

}

void onRender ()

{

glMatrixMode(GL\_MODELVIEW);

glPushMatrix ();

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

glTranslatef(m\_distanceFrom, 0, 0);

if ( innerPlanet != nullptr )

innerPlanet->onRender ();

glMatrixMode(GL\_TEXTURE);

glLoadIdentity();

//glTranslatef(0.5, 0, 0);

glTranslatef(m\_rotationAmount / 100, 0, 0);

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

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

//glTranslatef(-0.5, 0, 0);

glMatrixMode(GL\_MODELVIEW);

// 텍스쳐 매핑

glBindTexture(GL\_TEXTURE\_2D, m\_texture);

if(m\_texture)

{

glEnable(GL\_TEXTURE\_GEN\_S);

glEnable(GL\_TEXTURE\_GEN\_T);

GLfloat plane\_coef\_s [] = { 1, 0, 0, 1 };

GLfloat plane\_coef\_t [] = { 0, 1, 0, 1 };

glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_SPHERE\_MAP);

glTexGeni(GL\_T, GL\_TEXTURE\_GEN\_MODE, GL\_SPHERE\_MAP);

glTexGenfv(GL\_S, GL\_OBJECT\_PLANE, plane\_coef\_s);

glTexGenfv(GL\_T, GL\_OBJECT\_PLANE, plane\_coef\_t);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

}

render ();

glDisable(GL\_TEXTURE\_GEN\_S);

glDisable(GL\_TEXTURE\_GEN\_T);

glPopMatrix ();

if ( nextPlanet != nullptr )

nextPlanet->onRender ();

}

protected:

float m\_rotationAxis;

float m\_rotationRate;

float m\_skiddingRate;

float m\_distanceFrom;

protected:

Planet \* innerPlanet;

Planet \* nextPlanet;

protected:

GLuint m\_texture;

private:

float m\_skiddingAmount;

float m\_rotationAmount;

};

class Neptune : public Planet

{

public:

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

{

m\_texture = readTexture("neptune.bmp");

}

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 Uranus : public Planet

{

public:

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

{

nextPlanet = new Neptune ();

m\_texture = readTexture("uranus.bmp");

}

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\*/30, 3.22f, 3.22f )

{

nextPlanet = new Uranus ();

m\_texture = readTexture("saturn.bmp");

}

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, 2.22f )

{

}

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, 8.22f )

{

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, 7.1f, 4.22f )

{

nextPlanet = new Saturn ();

innerPlanet = new Io ();

m\_texture = readTexture("jupiter.bmp");

}

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, 5.55f )

{

nextPlanet = new Jupiter ();

m\_texture = readTexture("mars.bmp");

}

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, 8.0f )

{

m\_texture = readTexture("moon.bmp");

}

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, 7.0f )

{

nextPlanet = new Mars ();

innerPlanet = new Moon ();

m\_texture = readTexture("earth.bmp");

}

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, 7.25f )

{

nextPlanet = new Earth ();

m\_texture = readTexture("venus.bmp");

}

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, 8.5f )

{

nextPlanet = new Venus ();

m\_texture = readTexture("mercury.bmp");

}

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 () : Planet ( 0, 0, 1.0, 0 )

{

innerPlanet = new Mercury ();

m\_texture = readTexture("sun.bmp");

}

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 = 0, camera\_z = 240;

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);

// 텍스쳐 켜기

glEnable(GL\_TEXTURE\_2D);

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 / 60.0f ) )

{

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, camera\_x, camera\_y, camera\_z - 240, 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;

}

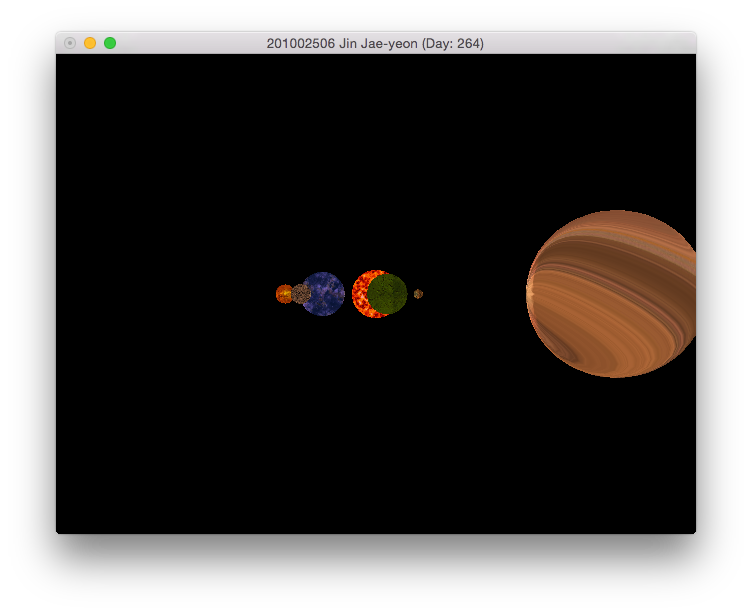
void keyboardUp ( unsigned char key, int x, int y )

{

isKeyDown[key] = false;

}

기존에 제출했던 과제에 있던 Planet 클래스의 버그를 일부 수정하였고 그러면서 행성에 텍스쳐를 입힐 수 있는 기능을 추가하여 상속받은 클래스마다 다른 텍스쳐를 입힐 수 있도록 하였습니다.

또한 BMP 파일을 읽어오는 기능을 직접 구현하여 PDF 파일에서 제시한 라이브러리를 사용하지 않는 방향에서 텍스쳐를 읽어올 수 있도록 하였습니다.

결과

논의

이 LAB의 키는 무엇인가?

그동안 공부했던 OpenGL의 행렬 스택과 라이팅을 이용해 행성계를 구현한 과제에 텍스쳐를 입히는 것입니다.

무슨 실수를 하고 무엇을 배웠는가?

행성이 회전할 때 텍스쳐가 같이 회전되지 않아 해당 부분을 고치는데 어려움이 있었습니다.

프로그램을 어떻게 향상시킬 수 있겠는가?

행성에 원래의 크기와 거리 등을 적용하여 실제 태양계를 시뮬레이션해보면 더욱 재미있을 것 같습니다.