과제5 코드 전제부분 16011024 주수헌

#include <GL/glut.h>

#include <GL/gl.h>

#include <GL/glu.h>

#define GAP 16

void redisplay\_all();

GLuint window, world, screen, command;

GLfloat fDelta = 0.0f;

GLboolean bBlend = false;

GLboolean bSmooth = false;

GLfloat Red = 0.0;

GLfloat Green = 0.0;

GLfloat Blue = 0.0;

GLint Index = 0;

unsigned char PALETTE[16][3] = {

{ 255, 255, 255 }, // WHITE

{ 0, 255, 255 }, // CYAN

{ 255, 0, 255 }, // PURPLE

{ 0, 0, 255 }, // BLUE

{ 192, 192, 192 }, // LIGHT GRAY

{ 128, 128, 128 }, // DARK GRAY

{ 0, 128, 128 }, // DARK TEAL

{ 128, 0, 128 }, // DARK PURPLE

{ 0, 0, 128 }, // DARK BLUE

{ 255, 255, 0 }, // YELLOW

{ 0, 255, 0 }, // GREEN

{ 128, 128, 0 }, // DARK YELLOW

{ 0, 128, 0 }, // DARK GREEN

{ 255, 0, 0 }, // RED

{ 128, 0, 0 }, // DARK RED

{ 0, 0, 0 }, // BLACK

};

void world\_display() {

Red = PALETTE[Index][0] / 255.0f; //어떤 인덱스든 빨간부분 나타냄

Green = PALETTE[Index][1] / 255.0f; //어떤 인덱스든 초록부분 나타냄

Blue = PALETTE[Index][2] / 255.0f; //어떤 인덱스든 파랑부분 나타냄

glColor3f(Red, Green, Blue); //위에 코드에서 정한 rgb 색상으로 한다

if (bBlend) { //혼합 함수: 인접화소의 색을 주어진 화소에 스며들게하여 에일리어싱감쇄

glEnable(GL\_BLEND); //혼합기능 활성화

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA); //혼합모드를 설정

}

//GL\_SRC\_ALPHA의 블렌딩 함수는 새로운 색상 요소에 0.4를 곱하고

//GL\_ONE\_MINUS\_SRC\_ALPHA의 블렌딩 함수는 기존 색상 요소에 1-0.4=0.6을 곱하여

//두 연산결과를 더하여 최종색상을 결정한다

if (bSmooth) {

glEnable(GL\_LINE\_SMOOTH); //부드러운 선 기능 활성화

glHint(GL\_LINE\_SMOOTH\_HINT, GL\_NICEST);//여기에 옵션 다른것도 줄수 있음

//glHint(GL\_LINE\_SMOOTH\_HINT, GL\_FASTEST);

//glHint(GL\_LINE\_SMOOTH\_HINT, GL\_DONT\_CARE);

//추가점수부분

}

//GL\_NICEST은 조금 느리더라도 질이 좋은 안티 에일리어싱을 가해줌

//GL\_FASTEST: 질이 떨어져도 가장 빠르게 그리기 원할시 사용한다

//GL\_DONT\_CARE: 디폴드 값이며 OpenGL이 알아서 속도와 품질을 결정하게 된다

glBegin(GL\_LINES);

glVertex3f(-1.0 + fDelta, 1.0, 0.0); //2개의 점을 이어 선 표시

glVertex3f(1.0 - fDelta, -1.0, 0.0);

glVertex3f(-1.0, -1.0 + fDelta, 0.0); //2개의 점을 이어 선 표시

glVertex3f(1.0, 1.0 - fDelta, 0.0);

glEnd();

//그리고 싶은 선을 다 그렸으면 스무딩함수와 혼합함수를 모두 disable한다

if (bSmooth)

glDisable(GL\_LINE\_SMOOTH);

if (bBlend)

glDisable(GL\_BLEND);

glutSwapBuffers();

}

void world\_timer(int Value) {

if (fDelta< 2.0f) {

fDelta = fDelta + 0.1;

}

else {

fDelta = 0.0;

if (++Index >= 15) {

Index = 0;

glClear(GL\_COLOR\_BUFFER\_BIT);//이함수로 색을 바꿀때마다 화면에 지워지게 된다

}

}

redisplay\_all();

glutTimerFunc(100, world\_timer, 1);

}

void screen\_display() {

static unsigned char buffer[128 \* 128 \* 4 \* 2];

glutSetWindow(world); //현재 윈도우 설정

glReadPixels(0, 0, 128, 128, GL\_RGBA, GL\_UNSIGNED\_BYTE, buffer);

//gl ReadPixels()는 프레임 버퍼로부터 픽셀블럭을 읽어낸다

//즉, 작은 창에 그려진 프레임 버퍼 내용을 메모리로 복사하는 함수이다

//프레임 버퍼로부터 읽은 첫번째 픽셀의 윈도우 창좌표 0,0으로 구체화한다

//그 위치는 픽셀 사각형 좌하단 기준으로 한다

//픽셀 사각형 가로세로를 128,128로 정한다

//GL\_RGBA은 픽셀 데이터 포멧이다. GL\_UNSIGNED\_BYTEd은 픽셀 데이터 타입이다

//반환하는 픽셀 데이터: buffer

glutSetWindow(screen); //screen이라는 서브윈도우를 설정

glPixelZoom(4, 4); //픽셀 확대 비율을 정한다

//픽셀 쓰기 작업에 대한 x y 확대비율을 정한다

glDrawPixels(128, 128, GL\_RGBA, GL\_UNSIGNED\_BYTE, buffer);

//이 함수에 의해 큰 창에 해당하는 프레임 버퍼로 기록된다

glPixelZoom(0, 0);

glutSwapBuffers();

}

void main\_display() {

glClearColor(0.8, 0.8, 0.8, 0.0);

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

glutSwapBuffers();

}

void main\_keyboard(unsigned char key, int x, int y) {

switch (key) {

case 'b': bBlend = !bBlend; break; //혼합 함수 사용

case 's': bSmooth = !bSmooth; break; //스무딩 함수 사용

case 't': if (!bBlend)

bBlend = bSmooth = true; //두가지 함수 모두 사용

else

bBlend = bSmooth = false; //두가지 함수 모두 사용안함

break;

//case 27: ex(0);

}

}

void redisplay\_all() {

// glutSetWindow(command);

// glutPostRedisplay();

glutSetWindow(world);

glutPostRedisplay();

glutSetWindow(screen);

glutPostRedisplay();

}

int main(int argc, char\*\* argv) {

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

glutInitWindowSize(128 + 512 + GAP \* 3, 512 + GAP \* 2);

glutInitWindowPosition(0, 0);

glutInit(&argc, argv);

window = glutCreateWindow("Line Antialiasing Demo");

glutDisplayFunc(main\_display);

glutKeyboardFunc(main\_keyboard);

//윈도우창에 2개 서브 윈도우 창을 만듦

world = glutCreateSubWindow(window, GAP, GAP, 128, 128);

glutDisplayFunc(world\_display);

glutKeyboardFunc(main\_keyboard);

screen = glutCreateSubWindow(window, GAP + 128 + GAP, GAP, 512, 512);

glutDisplayFunc(screen\_display);

glutKeyboardFunc(main\_keyboard);

redisplay\_all();

glutTimerFunc(100, world\_timer, 1);

glutMainLoop();

return 0;

}