#include <cstdlib>

#include <cstdio>

#include <ctime>

#include <cmath>

//#include <windows.h>

#pragma comment(lib, "winmm.lib")

#include <iostream>

using namespace std;

#include <GLUT/glut.h>

//#include "template.h"

#define W 600 //window Sz=600x600 = WxH

#define H 600 //window Sz=600x600 = WxH

#define M glPushMatrix(); // MACROS

#define m glPopMatrix();

#define T glTranslatef(

#define R glRotatef(

#define S glScalef(

#define L glEnable(GL\_LIGHTING);

#define l glDisable(GL\_LIGHTING);

#define C glColor3f( //set a color: C r,g,b); //added

#define L0 glEnable(GL\_LIGHT0); //red // E/Disable four lights 0..3

#define l0 glDisable(GL\_LIGHT0);

#define L1 glEnable(GL\_LIGHT1); //red

#define l1 glDisable(GL\_LIGHT1);

#define L2 glEnable(GL\_LIGHT2); //yellow

#define l2 glDisable(GL\_LIGHT2);

#define L3 glEnable(GL\_LIGHT3); //move

#define l3 glDisable(GL\_LIGHT3);

#define L4 glEnable(GL\_LIGHT4); //grey // E/Disable four lights 0..3

#define l4 glDisable(GL\_LIGHT4);

#define L5 glEnable(GL\_LIGHT5); //move light

#define l5 glDisable(GL\_LIGHT5);

#define L6 glEnable(GL\_LIGHT6); //red light

#define l6 glDisable(GL\_LIGHT6);

#define L7 glEnable(GL\_LIGHT7); // red

#define l7 glDisable(GL\_LIGHT7);

#define L8 glEnable(GL\_LIGHT8); // no can be set to a color

#define l8 glDisable(GL\_LIGHT8);

#define L9 glEnable(GL\_LIGHT9); // no "

#define l9 glDisable(GL\_LIGHT9);

#define Scub glutSolidCube( //four primitives

#define Ssph glutSolidSphere(

#define Scon glutSolidCone(

#define Stor glutSolidTorus(

#define Wcub glutWireCube(

#define Wsph glutWireSphere(

#define Wcon glutWireCone(

#define Wtor glutWireTorus(

GLfloat mat\_ambient[] = { .0,.0,.0, .0 };

GLfloat mat\_specular[] = { 0.,0.,0., .0 };

GLfloat mat\_diffuse0[]={1.,1.,1., 1.}; //white

GLfloat mat\_diffuse1[]={1.,.0,.0, 1.}; //red

GLfloat mat\_diffuse2[]={0.,0.,1., 1.}; //blue

GLfloat mat\_diffuse3[]={0.,0.,1., .5}; //blue

GLfloat mat\_diffuse4[]={0.,1.,1., .5}; //sky

GLfloat mat\_diffuse5[]={0.,0.,1., .5}; //violet

GLfloat mat\_diffuse6[]={1.,.5,0., .5}; //brown

GLfloat mat\_diffuse7[]={1.,1.,0., .5}; //yellow

GLfloat mat\_shininess[]={128.}; //this needs specular (work together)

GLfloat light0\_position[]={0., 190., 130., 1.0};

GLfloat light1\_position[]={0., 150., 150., 1.0};

GLfloat light2\_position[]={0., 150., 150., 1.0};

GLfloat light3\_position[]={0., 150., 150., 1.0};

GLfloat light4\_position[]={0., 150., 150., 1.0};

GLfloat light5\_position[]={0., 150., 150., 1.0};

GLfloat light6\_position[]={0., 150., 150., 1.0};

GLfloat light7\_position[]={0., 150., 150., 1.0};

float XXX = 0., YYY = 0.; //pointing: mouse position in window coordinates

float an=0., da=10. ;

void pointing(float xx, float yy)

{ int Wid = glutGet(GLUT\_WINDOW\_WIDTH); int Hei = glutGet(GLUT\_WINDOW\_HEIGHT); cout << "\nwindowWH : " << Wid << ',' << Hei << endl;

cout << "mouse pixels: xx, yy : " << xx << ",\t\t" << yy << endl;

//glutGet(GLUT\_WINDOW\_X); glutGet(GLUT\_WINDOW\_Y); //https://www.opengl.org/resources/libraries/glut/spec3/node70.html

XXX = (xx / Wid)\*W; //-0 because the window origin is the same as the coords origin (left, top and botum corner )

YYY = ((Hei-yy) / Hei)\*H;

cout << "mouse coords: XXX,YYY : " << XXX << ",\t" << YYY << endl;

}

void display(void)

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

M T W/4., H/4., 0.); Wcub 90.); l2 m

M T W/4., 3.\*H /4., 0.); Scub 60.); l1 m

M T 3.\*W/4., H /4., 0);Wsph 40.,16,8);l3 m

M T 3.\*W/4., 3.\*H/4., 0.); Ssph 40., 16,8); l4 m

//M T W/4., H/4., 0.); R an , 1.,0,0); Wcub 90.);m

// M T W/4., 3.\*H /4., 0.); R an, 1., 0,0); L L1 Scub 60.); I1 I m

//M T 3.\*W/4., 3.\*H/4

// M T 3.\*W/4., H /4., 0);

glutSwapBuffers();

}

void spinDisplay() {glutPostRedisplay(); }

void keyboard(unsigned char key, int xx, int yy)

{ switch (key)

{

// case 'm': PlaySound(TEXT("music.wav"), NULL, SND\_ASYNC); break;

//case 'M': PlaySound(NULL, NULL, NULL); break;

}

glutPostRedisplay();

}

void special(GLint key, GLint x, GLint y)

{ //if (key == GLUT\_KEY\_F2)

glutPostRedisplay();

}

void mouse(int button, int state, int xx, int yy)

{ pointing(xx, yy);

switch (button)

{

case GLUT\_LEFT\_BUTTON: if(state == GLUT\_DOWN) { glutIdleFunc(spinDisplay); } break;

case GLUT\_RIGHT\_BUTTON: if(state == GLUT\_UP) { glutIdleFunc(NULL); } break;

}

glutPostRedisplay();

}

void reshape(int w, int h)

{ glViewport(0,0,(GLsizei)w,(GLsizei)h);

glMatrixMode(GL\_PROJECTION); glLoadIdentity();

glOrtho(0,W,0,H, -W,H);

glMatrixMode(GL\_MODELVIEW); glLoadIdentity();

}

void init() {

// glClearColor(1., 1., 1., 1.); initColoring();

}

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

{ glutInit(&argc, argv);

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

glutInitWindowSize (W, H); glutInitWindowPosition(0,0);

glutCreateWindow("Dr. Tannouri's easiest volume visualization");

init();

glutDisplayFunc(display);

glutMouseFunc(mouse);

glutKeyboardFunc(keyboard);

glutSpecialFunc(special);

glutReshapeFunc(reshape);

glutMainLoop();

}

void initColoring(void)

{ //glClearColor(1., 1., 1., 1.);

glShadeModel(GL\_SMOOTH); //default: SMOOTH not FLAT

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, mat\_shininess);

//glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, mat\_ambient);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, mat\_ambient);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_position); //three positions for light sources

glLightfv(GL\_LIGHT1, GL\_POSITION, light1\_position);

glLightfv(GL\_LIGHT2, GL\_POSITION, light2\_position);

glLightfv(GL\_LIGHT3, GL\_POSITION, light3\_position);

glLightfv(GL\_LIGHT4, GL\_POSITION, light4\_position);

glLightfv(GL\_LIGHT5, GL\_POSITION, light5\_position);

glLightfv(GL\_LIGHT6, GL\_POSITION, light6\_position);

glLightfv(GL\_LIGHT7, GL\_POSITION, light7\_position);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, mat\_diffuse0); //eight different colors for material lighting

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, mat\_diffuse1);

glLightfv(GL\_LIGHT2, GL\_DIFFUSE, mat\_diffuse2);

glLightfv(GL\_LIGHT3, GL\_DIFFUSE, mat\_diffuse3);

glLightfv(GL\_LIGHT4, GL\_DIFFUSE, mat\_diffuse4);

glLightfv(GL\_LIGHT5, GL\_DIFFUSE, mat\_diffuse5);

glLightfv(GL\_LIGHT6, GL\_DIFFUSE, mat\_diffuse6);

glLightfv(GL\_LIGHT7, GL\_DIFFUSE, mat\_diffuse7);

glEnable(GL\_LIGHTING); glEnable(GL\_LIGHT1); //enable some lighting

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

}

/\* void pointing(float xx, float yy)

{ cout << "\n\nmouse position xx,yy (in pixels):\n mouseXY : " << xx << "," << yy << endl;

int Wid = glutGet(GLUT\_WINDOW\_WIDTH); int Hei = glutGet(GLUT\_WINDOW\_HEIGHT); cout << "windowWH : " << Wid << ',' << Hei << endl;

//glutGet(GLUT\_WINDOW\_X); glutGet(GLUT\_WINDOW\_Y); //https://www.opengl.org/resources/libraries/glut/spec3/node70.html

M T xx, yy, 0.); Wcub 5.); m

XXX = (xx / Wid)\*500. - 250.; YYY = 250. + (-yy / Hei) \* 500;

cout<<" mouse position: "<< (xx/Wid)\*500.-250. <<'\_'<<250.+ (-yy/Hei)\*500. <<endl;

}

\*///////////////////////////////////////////////////////////////

static GLuint texName[6];

void readBMPfile(int txk)

{ FILE \*fil = NULL; unsigned char info[54];

// if(txk==0) fopen\_s(&fil, "bg.bmp", "rb"); else

// if(txk==1) fopen\_s(&fil, "bg1.bmp", "rb"); else

// if(txk==2) fopen\_s(&fil, "bg2.bmp", "rb");

if (!fil) { cout << fil <<" txk= "<<txk<< " cannot open this file: fil\n"; getchar(); exit(0); }

else fread(info, sizeof(unsigned char), 54, fil); // Read info: 54-byte\_header; info[18] and info[22] //Problem ?

int width = \*(int\*)&info[18]; // extract image\_width from header

int height = \*(int\*)&info[22]; // extract image\_height from header

int size = 3 \* width \* height;

unsigned char \*data = new unsigned char[3 \* width\*height];

fread(data, sizeof(unsigned char), size, fil); // Read the rest of the data (image data)

fclose(fil);

///////////////////////// S W A P R,B colors

for (int i=0; i<size; i+=3) { unsigned char tmp; tmp = data[i]; data[i] = data[i + 2]; data[i + 2] = tmp; }

/////////////////////////

glGenTextures(2, &texName[txk]); // genTexture(2D,texName[txk])

glBindTexture(GL\_TEXTURE\_2D, texName[txk]); // bind(texture,texName)

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

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, data);

}

void drawTexture(int txk) {

float h = 1000.; short int w = 1000., d = 300.;

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);

glBindTexture(GL\_TEXTURE\_2D, texName[txk]);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_REPLACE); //D..,BLEND,MODULATE,REPLACE);//

glBegin(GL\_QUADS); //x,y,z

glTexCoord2f(0., 0.); glVertex3f(-w, -h, d);

glTexCoord2f(0., 1.); glVertex3f(-w, h, d);

glTexCoord2f(1., 1.); glVertex3f(w, h, d);

glTexCoord2f(1., 0.); glVertex3f(w, -h, d);

glEnd();

glDisable(GL\_TEXTURE\_2D);

glPopMatrix();

}