Grafika Komputerowa

Autor:
Jacek Wieczorek

Prowadzący:
Dr inż. Tomasz Kapłon

Wydział Elektroniki
III rok
Pn TP 8.15 - 11.00

26 grudnia 2011

1 Cel laboratorium

Celem ćwiczenia było poznanie podstawowych technik teksturowania powierzchni z wykorzystaniem *OpenGL* i *GLUT*.

2 Wspólne funkcje programu dla każdego z zadań

2.1 Wczytanie tekstury

```
GLbyte *LoadTGAImage(const char *FileName, GLint *ImWidth, GLint *ImHeight, GLint *ImComponents,
{
    #pragma pack(1)
    typedef struct
    {
        GLbyte    idlength;
        GLbyte    colormaptype;
        GLbyte    datatypecode;
        unsigned short    colormapstart;
        unsigned short    colormaplength;
        unsigned char     colormapdepth;
        unsigned short    x_orgin;
        unsigned short    y_orgin;
        unsigned short    width;
        unsigned short    height;
        GLbyte    bitsperpixel;
        GLbyte    descriptor;
    }TGAHEADER;
    #pragma pack(8)


    FILE *pFile;
    TGAHEADER tgaHeader;
    unsigned long lImageSize;
    short sDepth;
    GLbyte    *pbitsperpixel = NULL;

    *ImWidth = 0;
    *ImHeight = 0;
    *ImFormat = GL_BGR_EXT;
    *ImComponents = GL_RGB8;

    pFile = fopen(FileName, "rb");
    if(pFile == NULL)
    return NULL;

    fread(&tgaHeader, sizeof(TGAHEADER), 1, pFile);

    *ImWidth = tgaHeader.width;
    *ImHeight = tgaHeader.height;
    sDepth = tgaHeader.bitsperpixel / 8;


    if(tgaHeader.bitsperpixel != 8 && tgaHeader.bitsperpixel != 24 && tgaHeader.bitsperpixel != 3
        return NULL;
    lImageSize = tgaHeader.width * tgaHeader.height * sDepth;
    pbitsperpixel = (GLbyte*)malloc(lImageSize * sizeof(GLbyte));

    if(pbitsperpixel == NULL)
        return NULL;

    if(fread(pbitsperpixel, lImageSize, 1, pFile) != 1)
    {
        free(pbitsperpixel);
        return NULL;
    }

    switch(sDepth)
    {
        case 3:
            *ImFormat = GL_BGR_EXT;
            *ImComponents = GL_RGB8;
            break;
        case 4:
            *ImFormat = GL_BGRA_EXT;
            *ImComponents = GL_RGBA8;
            break;
        case 1:
            *ImFormat = GL_LUMINANCE;
            *ImComponents = GL_LUMINANCE8;
            break;
    };

    fclose(pFile);
    return pbitsperpixel;
}
```

2.2 Sterowanie za pomocą myszki

```
void Mouse(int btn, int state, int x, int y)
{
   if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        x_pos_old = x;
        y_pos_old = y;
        status = 1;
    }
        else if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN){
                zoom = y;
                status = 2;
        }
        else
        status = 0;
}

void Motion( GLsizei x, GLsizei y )
{

    delta_x= x - x_pos_old;
        x_pos_old = x;
        delta_y = y - y_pos_old;
        y_pos_old = y;
        delta_zoom = y - zoom;
    zoom = y;

    glutPostRedisplay();
}
```

2.3 Funkcja *MyInit*

```
void MyInit(void)
{
        GLbyte *pBytes;
        GLint ImWidth, ImHeight, ImComponents;
        GLenum ImFormat;
        if(which !=2 )
                glEnable(GL_CULL_FACE);

        pBytes = LoadTGAImage("C:\\Users\\jacek\\Desktop\\D1_t.tga", &ImWidth, &ImHeight, &ImCom
        glTexImage2D(GL_TEXTURE_2D, 0, ImComponents, ImWidth, ImHeight, 0, ImFormat, GL_UNSIGNED_
        free(pBytes);
        glEnable(GL_TEXTURE_2D);
        glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        GLfloat mat_ambient[]  = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_diffuse[]  = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_shininess  = {20.0};
    GLfloat light_position[] = {0.0, 0.0, 10.0, 1.0};
    GLfloat light_ambient[] = {0.1, 0.1, 0.1, 1.0};
    GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat light_specular[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat att_constant  = {1.0};
    GLfloat att_linear    = {0.05};
    GLfloat att_quadratic  = {0.001};
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
        glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);
        glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}
```

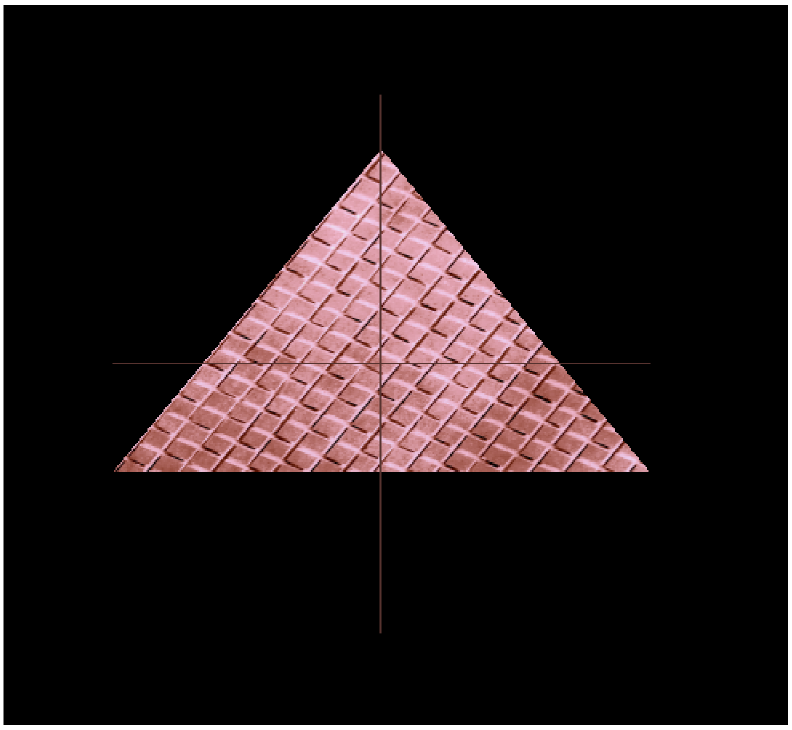
3 Zadanie 1

Pierwsze zadanie polegało na narysowaniu trójkąta i wypełnbienu go dowolnie wybraną teksturą zapisaną w pliku o *TGA*.

3.1 Kod programu

```
glBegin(GL_TRIANGLES);
  glTexCoord2f(0.0f, 0.0f);
  glVertex3f(0.0f, 4.0f,0.0f);

  glTexCoord2f(1.0f, 0.0f);
  glVertex3f(-5.0f, -2.0f, 0.0f);

  glTexCoord2f(0.5f, 1.0f);
  glVertex3f(5.0f,-2.0f, 0.0f);
glEnd();
```

3.2 Przykładowy obraz



Rysunek 1: Tekstura nałożona na trójkąt

4 Zadanie 2

Celem drugie zadania było zbudowanie i teksturowanie jednostronne ostrosłupa o kwadratowej podstawie.

By umozliwić sprawdzenie jednostronnego teksturowania piramidy, zaimplementowana została mozliwość wybierania za pomoca klawiatury, która ściana piramidy ma być wyświetlana.

4.1 Kod programu

4.1.1 Sterowanie klawiaturą

```
void keys(unsigned char key, int x, int y)
{
    if(key == 'p') model = 1;
    if(key == 'w') model = 2;
    if(key == 's') model = 3;

        if(key == 'z') kk = 0;
        if(key == 'x') kk = 1;
        if(key == 'c') kk = 2;
        if(key == 'v') kk = 3;
        if(key == 'b') kk = 4;
        if(key == 'n') kk = 5;
        if(key == 'm') kk = 6;


    RenderScene();
}
```

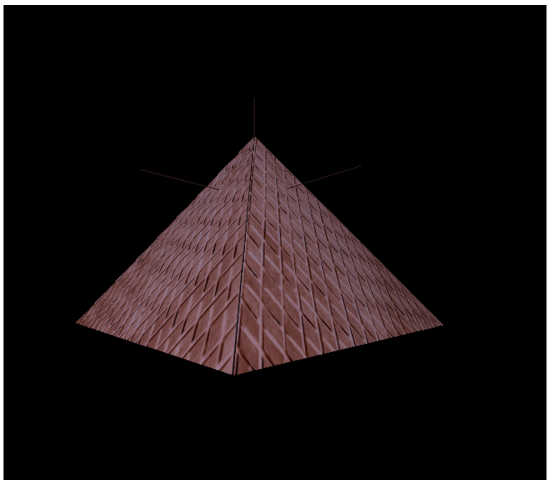
4.1.2 Rysowanie piramidy

```
if(status == 1)                     // je?li lewy klawisz myszy wci?ni?ty
{
        thetax += delta_x * pix2angle / 30.0;
        thetay += delta_y * pix2angle / 30.0;
}
else if(status ==2)
{
        theta_zoom += delta_zoom/10.0;
}

if(thetay > 3.1415)
        thetay -= 2*3.1415;
else if(thetay <= -3.1415)
        thetay += 2*3.1415;

if(thetay > 3.1415/2 || thetay < -3.1415/2)
{
        p = -1.0;
}
else
{
p = 1.0;
```

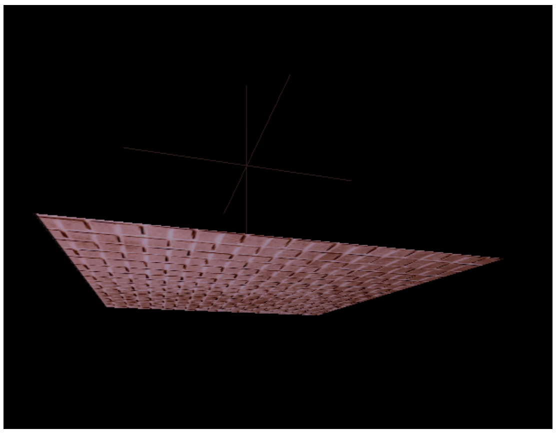
```
}

viewer[0] = theta_zoom*cos(thetax)*cos(thetay);
viewer[1] = theta_zoom*sin(thetay);
viewer[2] = theta_zoom*sin(thetax)*cos(thetay);

if(kk == 0) {
        glBegin(GL_QUADS);                                  // draw square
        glTexCoord2f(0.0f, 0.0f);                   // set color to green
        glVertex3f(-5.0f, -5.0f, 5.0f);
        glTexCoord2f(0.0f, 1.0f);                   // set color to white
        glVertex3f( -5.0f, -5.0f, -5.0f);
        glTexCoord2f(1.0f, 1.0f);                   // set color to blue
        glVertex3f( 5.0f, -5.0f, -5.0f);
        glTexCoord2f(1.0f, 0.0f);                   // set color to yellow
        glVertex3f( 5.0f, -5.0f, 5.0f);
        glEnd();
}

if(kk == 1) {
        glBegin(GL_TRIANGLES);              // draw triangle
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(  0.0f,  3.0f, 0.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(-5.0f, -5.0f, 5.0f);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f( 5.0f, -5.0f, 5.0f);
        glEnd();
}

if(kk == 2) {
        glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(  0.0f,  3.0f, 0.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f( 5.0f, -5.0f, 5.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f( 5.0f, -5.0f, -5.0f);
        glEnd();
}

if(kk == 3) {
        glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(  0.0f,  3.0f, 0.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f( -5.0f, -5.0f, -5.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(-5.0f, -5.0f, 5.0f);
        glEnd();
}

if(kk == 4) {
        glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(  0.0f,  3.0f, 0.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f( -5.0f, -5.0f, -5.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(-5.0f, -5.0f, 5.0f);
        glEnd();
```

```
    }

    if(kk == 5){
            glBegin(GL_TRIANGLES);
            glTexCoord2f(0.0f, 0.0f);
            glVertex3f(  0.0f,  3.0f, 0.0f);
            glTexCoord2f(1.0f, 0.0f);
            glVertex3f(-5.0f, -5.0f, 5.0f);
            glTexCoord2f(1.0f, 1.0f);
            glVertex3f(-5.0f, -5.0f, 5.0f);
            glEnd();
    }


    if(kk == 6) {
            glBegin(GL_TRIANGLE_FAN);          // draw triangle
            glTexCoord2f(0.0f, 0.0f);
            glVertex3f(  0.0f,  3.0f, 0.0f);
            glTexCoord2f(1.0f, 0.0f);
            glVertex3f(-5.0f, -5.0f, 5.0f);
            glTexCoord2f(0.0f, 1.0f);
            glVertex3f( 5.0f, -5.0f, 5.0f);
            glTexCoord2f(1.0f, 1.0f);
            glVertex3f( 5.0f, -5.0f, -5.0f);
            glTexCoord2f(1.0f, 0.0f);
            glVertex3f( -5.0f, -5.0f, -5.0f);
            glTexCoord2f(0.0f, 1.0f);
            glVertex3f(-5.0f, -5.0f, 5.0f);
            glEnd();



            glBegin(GL_QUADS);                                 // draw square
            glTexCoord2f(0.0f, 0.0f);                  // set color to green
            glVertex3f(-5.0f, -5.0f, 5.0f);
            glTexCoord2f(0.0f, 1.0f);                  // set color to white
            glVertex3f( -5.0f, -5.0f, -5.0f);
            glTexCoord2f(1.0f, 1.0f);                  // set color to blue
            glVertex3f( 5.0f, -5.0f, -5.0f);
            glTexCoord2f(1.0f, 0.0f);                  // set color to yellow
            glVertex3f( 5.0f, -5.0f, 5.0f);
            glEnd();
    }
```

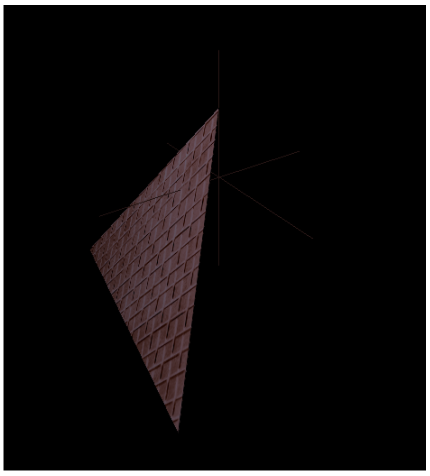
4.2 Przykładowe obrazy



Rysunek 2: Tekstura nałożona na piramidę



Rysunek 3: Tekstura nałożona na podstawę piramidy



Rysunek 4: Tekstura nałożona na jeden z boków piramidy

5 Zadanie 3

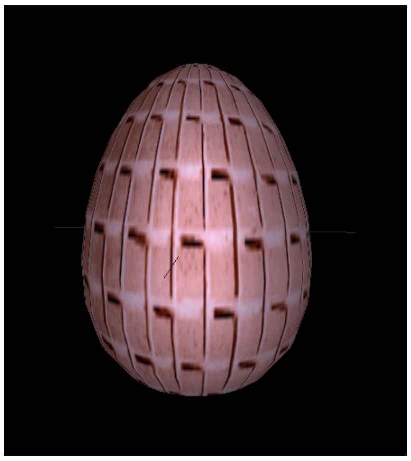
Ostatnie zadanie polegało na nałożeniu na jajko wykorzystywane podczas poprzednich laboratoriów tekstur.

5.1 Kod programu

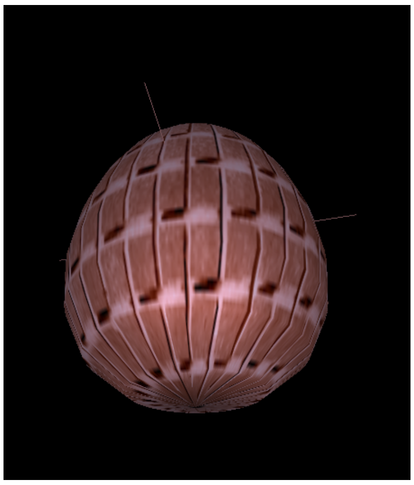
```
if(status == 1)                    // je?li lewy klawisz myszy wci?ni?ty
{
        thetax += delta_x * pix2angle / 30.0;
        thetay += delta_y * pix2angle / 30.0;
}
else if(status ==2)
{
        theta_zoom += delta_zoom/10.0;
}

if(thetay > 3.1415)
        thetay -= 2*3.1415;
else if(thetay <= -3.1415)
        thetay += 2*3.1415;

if(thetay > 3.1415/2 || thetay < -3.1415/2)
{
        p = -1.0;
}
else
{
p = 1.0;
}


viewer[0] = theta_zoom*cos(thetax)*cos(thetay);
viewer[1] = theta_zoom*sin(thetay);
```

```
viewer[2] = theta_zoom*sin(thetax)*cos(thetay);

EggsTriangles();

//funkcja EggsTriangles

void EggsTriangles(){
        float div = N * 1.0f;
        for(int i=0; i<N-1; i++){
        for(int j=0; j<N-1; j++)
                {
                        glBegin(GL_TRIANGLES);
                        //glColor3fv(col[i][j]);
                        glNormal3fv(nor[i][j]);
                        glTexCoord2f(i / div, j/div);
                        glVertex3fv(tab[i][j]);


                        //glColor3fv(col[i+1][j]);
                        glNormal3fv(nor[i+1][j]);
                        glTexCoord2f((i+1)/div, j/div);
                        glVertex3fv(tab[i+1][j]);


                        //glColor3fv(col[i][j+1]);
                        glNormal3fv(nor[i][j+1]);
                        glTexCoord2f(i/div, (j+1)/div);
                        glVertex3fv(tab[i][j+1]);
                        glEnd();

                        glBegin(GL_TRIANGLES);

                        glNormal3fv(nor[i+1][j+1]);
                        glTexCoord2f((i+1)/div, (j+1)/div);
                        glVertex3fv(tab[i+1][j+1]);


                        //glColor3fv(col[i+1][j]);
                        glNormal3fv(nor[i+1][j]);
                        glTexCoord2f((i+1)/div, j/div);
                        glVertex3fv(tab[i+1][j]);


                        //glColor3fv(col[i][j+1]);
                        glNormal3fv(nor[i][j+1]);
                        glTexCoord2f(i/div, (j+1)/div);
                        glVertex3fv(tab[i][j+1]);
                        glEnd();
                }
    }
}
```

5.2 Przykładowy obraz



Rysunek 5: Tekstura nałożona na jajko



Rysunek 6: Tekstura nałożona na jajko

6 Wnioski

Teksturowanie obiektów z wykorzystaniem biblioteki *OpenGL* z rozszerzeniem *GLUT* nie jest trudnym zadaniem. Po zapoznaniu sie z listą parametrów i możliwości jakie daje nam wyżej wymieneiona biblioteka, możemy w praktycznie dowolny sposób nakładać tekstury na obiekty, definiować czy mają być jednostronne czy dwustronne, ładowac dowolne obrazy.