Grafika komputerowa

Autor:
Tymon Tobolski (181037)

Prowadzący:
Dr inż. Tomasz Kapłon

Wydział Elektroniki
III rok
Pn TP 08.15 - 11.00

12 grudnia 2011

1 Cel laboratorium

Celem laboratorium było zaprezentowanie podstawowych technik teksturowania powierzchni obiektów z wykorzystaniem biblioteki OpenGL wrz z rozszerzeniem GLUT.

2 Wczytywanie tekstury

Podana w instrukcji funkcja realizujaca wczytywanie tekstury z pliku TGA została przygotowana dla systemu Windows. W takiej postanie nie działa niestety pod systemem Mac OS X. Różnice w odczycie mogą wynikać z różnej implementacji systemu plików (NTFS/HFS+) jak też z nieco innej implementacji samej biblioteki OpenGL. Wymagana modyfikacja funkcji została opatrzona stosownym komentarzem.

```
// Funkcja wczytujaca teksture

GLbyte *LoadTGAImage(const char *FileName, GLint *ImWidth, GLint *ImHeight,
    GLint *ImComponents, GLenum *ImFormat){
    typedef struct {
        GLbyte    idlength;
        GLbyte    colormaptype;
        GLbyte    datatypecode;
        unsigned short    colormapstart;
        unsigned short    colormaplength;
        //unsigned char     colormapdepth; // Wymagana modyfikacja dla
            systemu Mac OS X
        unsigned short    x_orgin;
        unsigned short    y_orgin;
        unsigned short    width;
        unsigned short    height;
        GLbyte    bitsperpixel;
        GLbyte    descriptor;
    } TGAHEADER;


    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;
```

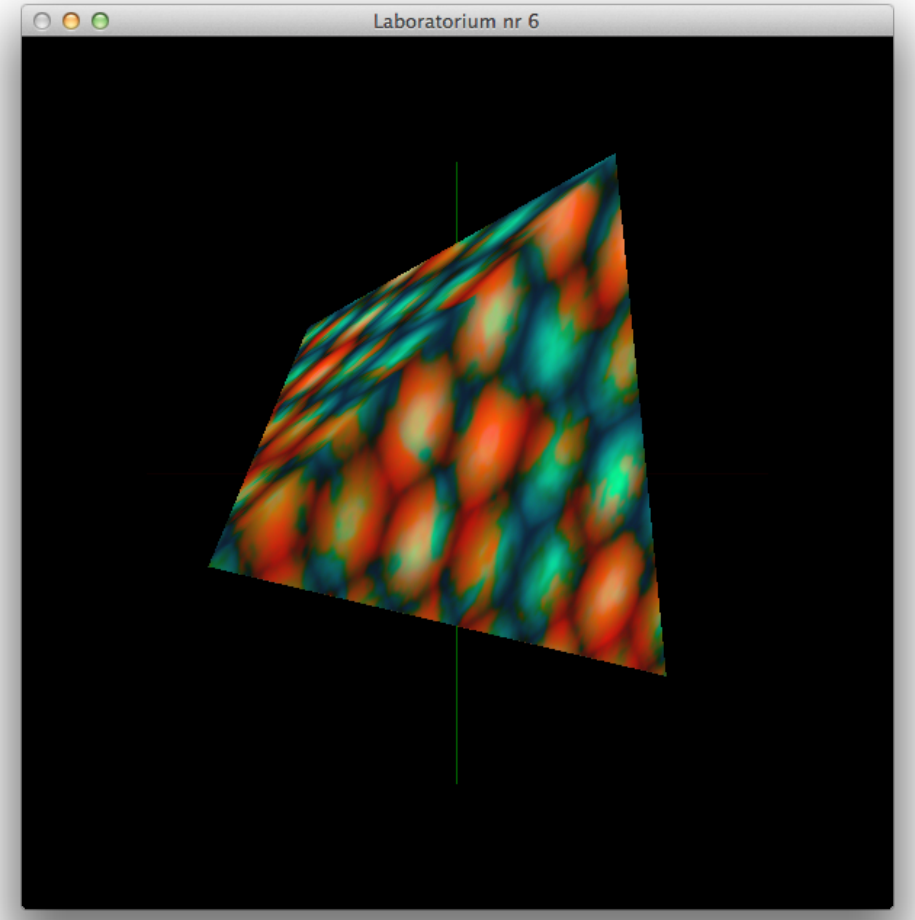
```
        sDepth = tgaHeader.bitsperpixel / 8;

41
        if(tgaHeader.bitsperpixel != 8 && tgaHeader.bitsperpixel != 24 &&
            tgaHeader.bitsperpixel != 32)
            return NULL;

        lImageSize = tgaHeader.width * tgaHeader.height * sDepth;
        pbitsperpixel = (GLbyte*)malloc(lImageSize * sizeof(GLbyte));

        if(pbitsperpixel == NULL)
            return NULL;

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

        switch(sDepth)
        {
            case 3:
                *ImFormat = GL_BGR_EXT;
                *ImComponents = GL_RGB8;
61              break;
            case 4:
                *ImFormat = GL_BGRA_EXT;
                *ImComponents = GL_RGBA8;
                break;
            case 1:
                *ImFormat = GL_LUMINANCE;
                *ImComponents = GL_LUMINANCE8;
                break;
        };
71
        fclose(pFile);

        return pbitsperpixel;
    }



    // Wczytanie tekstury (czesc funkcji init())
    GLbyte *pBytes;
81  GLint ImWidth, ImHeight, ImComponents;
    GLenum ImFormat;

    pBytes = LoadTGAImage("/Users/teamon/Downloads/t_1024.tga", &ImWidth, &
        ImHeight, &ImComponents, &ImFormat);
    glTexImage2D(GL_TEXTURE_2D, 0, ImComponents, ImWidth, ImHeight, 0, ImFormat,
         GL_UNSIGNED_BYTE, pBytes);
    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);
91  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

3 Teksturowana piramida

Pierwsze zadanie polegało na narysowaniu otekstorowanej piramidy. Należało też zaimplementować możliwość pokazywania i chowania poszczególnych ścian piramidy.

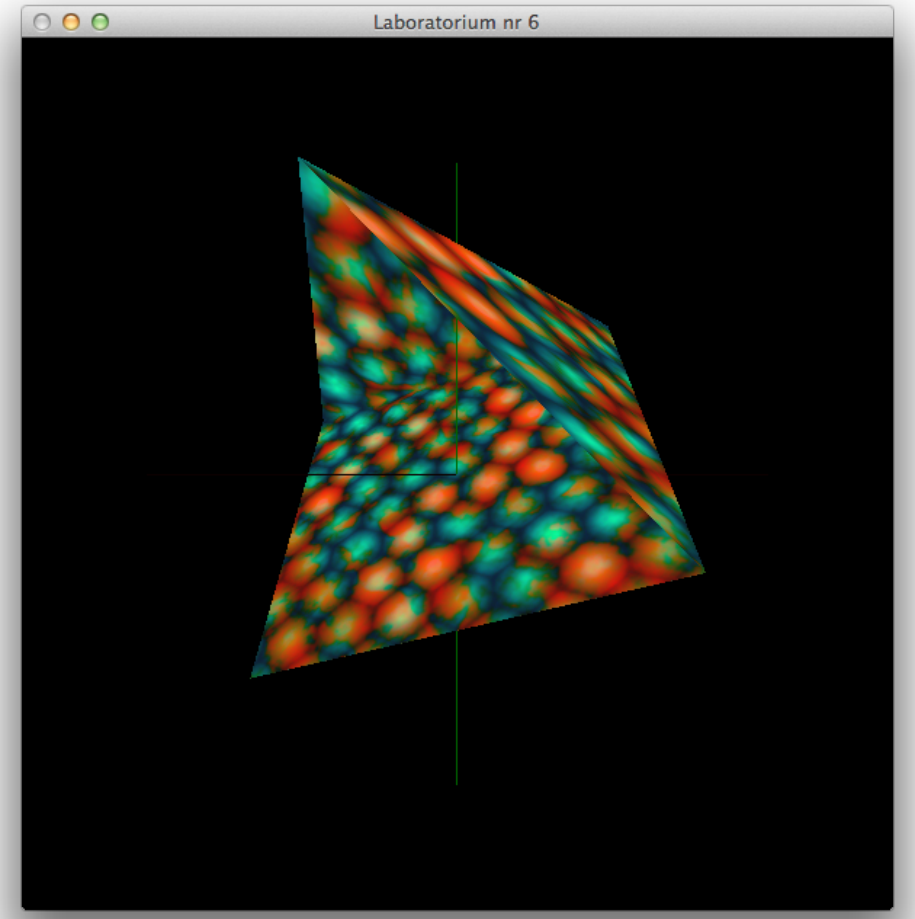
```
// Funkcja rysujaca teksturowana piramide

void piramid(){
    if(show[0]){
        glBegin(GL_QUADS);
        glTexCoord2f(0.0, 1.0);
        glVertex3f(-3.0,  3.0, 0.0); // D
        glTexCoord2f(1.0, 1.0);
        glVertex3f( 3.0,  3.0, 0.0); // C
        glTexCoord2f(1.0, 0.0);
        glVertex3f( 3.0, -3.0, 0.0); // B
        glTexCoord2f(0.0, 0.0);
        glVertex3f(-3.0, -3.0, 0.0); // A
        glEnd();
    }

    glBegin(GL_TRIANGLES);

    if(show[1]){
        glTexCoord2f(0.0, 0.0);
        glVertex3f(-3.0, -3.0, 0.0); // A
        glTexCoord2f(1.0, 0.0);
        glVertex3f( 3.0, -3.0, 0.0); // B
        glTexCoord2f(0.5, 0.5);
        glVertex3f( 0.0,  0.0, 5.0);
    }

    if(show[2]){
        glTexCoord2f(1.0, 0.0);
        glVertex3f( 3.0, -3.0, 0.0); // B
        glTexCoord2f(1.0, 1.0);
        glVertex3f( 3.0,  3.0, 0.0); // C
        glTexCoord2f(0.5, 0.5);
        glVertex3f( 0.0,  0.0, 5.0);
    }

    if(show[3]){
        glTexCoord2f(1.0, 1.0);
        glVertex3f( 3.0,  3.0, 0.0); // C
        glTexCoord2f(0.0, 1.0);
        glVertex3f(-3.0,  3.0, 0.0); // D
        glTexCoord2f(0.5, 0.5);
        glVertex3f( 0.0,  0.0, 5.0);
    }

    if(show[4]){
        glTexCoord2f(0.0, 1.0);
        glVertex3f(-3.0,  3.0, 0.0); // D
        glTexCoord2f(0.0, 0.0);
        glVertex3f(-3.0, -3.0, 0.0); // A
        glTexCoord2f(0.5, 0.5);
        glVertex3f( 0.0,  0.0, 5.0);
    }
```

```
    glEnd();
}
```



Rysunek 1: Oteksturowana piramida

Warto przy okazji zwrócic uwage na parametr `GL_CULL_FACE`, który odpowiada za teksturowanie obu stron lub tylko jednej wieloboku. Ustawienie tego



Rysunek 2: Oteksturowana piramida z widocznymi trzema ścianami

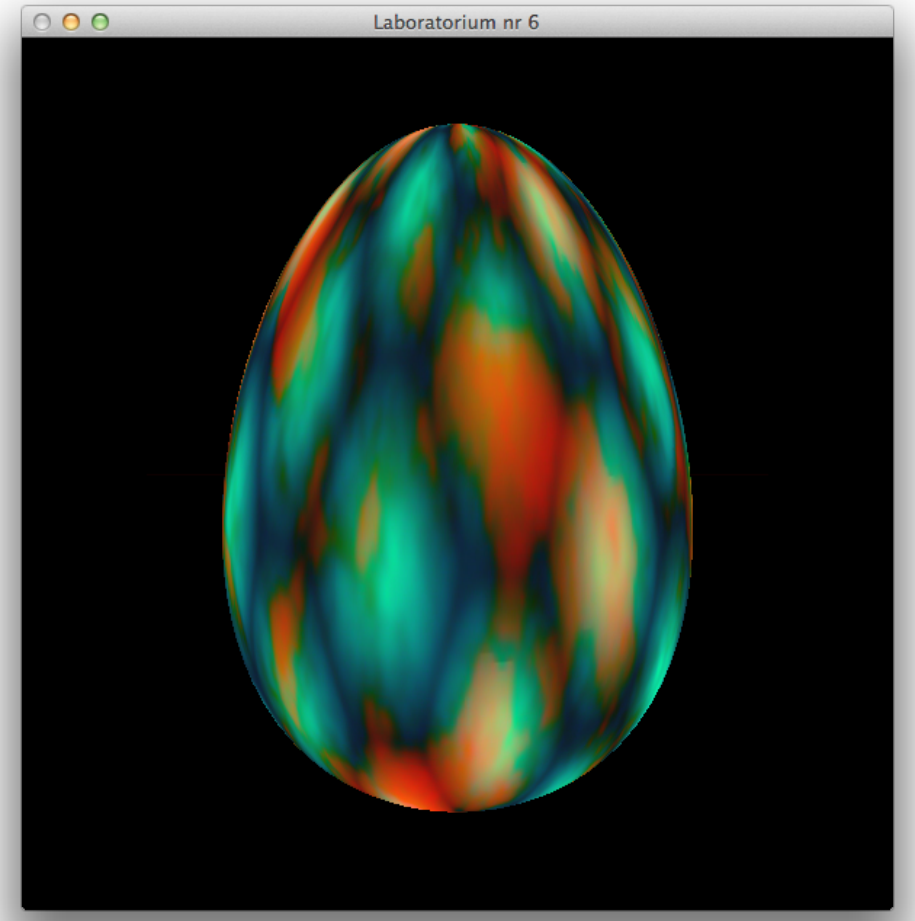
parametru poprzez wywołanie funkcji `glEnable(GL_CULL_FACE)` pozwala na uniknięcie obliczeń teksturowania wewnętrznych ścian obiektów.

4 Teksturowane jajko

Kolejnym zadaniem było narysowanie jajka z poprzednich zajęc tym razem z wykorzystaniem tekstury. W tym celu należało zamienić wywołania funkcji `glColor3v` na `glTexCoord2f` z odpowiednimi parametrami.

```
// Obliczenie punktow jajka oraz wektorow normalnych
void calc(){
  float u = 0.0, v=0.0;

  for(int i=0; i< N; i++){
    u = ((float)i)/(N-1);
    for(int j=0; j<N; j++){
      v = ((float)j)/(N-1);
      tab[i][j][0] = (-90*pow(u, 5.0) + 225*pow(u, 4.0) - 270*pow(u, 3.0) +
          180*pow(u, 2.0) - 45*u) * cos(3.14 * v) ;
      tab[i][j][1] =  160*pow(u, 4.0) - 320*pow(u, 3.0) + 160*pow(u, 2.0) -
          5.0;
      tab[i][j][2] = (-90*pow(u, 5.0) + 225*pow(u, 4.0) - 270*pow(u, 3.0) +
          180*pow(u, 2.0) - 45*u) * sin(3.14 * v);

      GLfloat xu = (-450*pow(u,4) + 900*pow(u,3) - 810*pow(u,2) + 360*u -
          45)*cos(3.14 * v);
      GLfloat yu = 640*pow(u,3) - 960*pow(u,2) + 320*u;
      GLfloat zu = (-450*pow(u,4) + 900*pow(u,3) - 810*pow(u,2) + 360*u -
          45)*sin(3.14 * v);

      GLfloat xv = 3.14 * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3) - 180*
          pow(u,2) + 45*u)*sin(3.14 * v);
      GLfloat yv = 0;
      GLfloat zv = -3.14 * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3) - 180*
          pow(u,2) + 45*u)*cos(3.14 * v);

      normal[i][j][0] = yu*zv - zu*yv;
      normal[i][j][1] = zu*xv - xu*zv;
      normal[i][j][2] = xu*yv - yu*xv;

      GLfloat len = sqrt(pow(normal[i][j][0],2) + pow(normal[i][j][1],2) +
          pow(normal[i][j][2],2));
      normal[i][j][0] /= len;
      normal[i][j][1] /= len;
      normal[i][j][2] /= len;

      if(i >= N/2) {
        normal[i][j][0] *= -1;
        normal[i][j][1] *= -1;
        normal[i][j][2] *= -1;
      }
    }
  }
}

// Fragment funkcji rysujacej jajko
for(int i=0; i<N/2; i++){
    for(int j=0; j<N-1; j++){
        glNormal3fv(normal[i][j]);
        glTexCoord2f(i/NF, j/NF);
        glVertex3fv(tab[i][j]);
```

```
            glNormal3fv(normal[i+1][j]);
            glTexCoord2f((i+1)/NF, j/NF);
            glVertex3fv(tab[i+1][j]);

            glNormal3fv(normal[i][j+1]);
            glTexCoord2f(i/NF, (j+1)/NF);
            glVertex3fv(tab[i][j+1]);

            glNormal3fv(normal[i][j+1]);
            glTexCoord2f(i/NF, (j+1)/NF);
            glVertex3fv(tab[i][j+1]);

            glNormal3fv(normal[i+1][j+1]);
            glTexCoord2f((i+1)/NF, (j+1)/NF);
            glVertex3fv(tab[i+1][j+1]);

            glNormal3fv(normal[i+1][j]);
            glTexCoord2f((i+1)/NF, j/NF);
            glVertex3fv(tab[i+1][j]);
        }
    }

    for(int i=N-1; i>N/2; i--){
        for(int j=0; j<N-1; j++){
            glNormal3fv(normal[i][j]);
            glTexCoord2f((N-i-1)/NF, (N-j)/NF);
            glVertex3fv(tab[i][j]);

            glNormal3fv(normal[i-1][j]);
            glTexCoord2f((N-i+1-1)/NF, (N-j)/NF);
            glVertex3fv(tab[i-1][j]);

            glNormal3fv(normal[i][j+1]);
            glTexCoord2f((N-i-1)/NF, (N-j-1)/NF);
            glVertex3fv(tab[i][j+1]);

            glNormal3fv(normal[i][j+1]);
            glTexCoord2f((N-i-1)/NF, (N-j-1)/NF);
            glVertex3fv(tab[i][j+1]);

            glNormal3fv(normal[i-1][j+1]);
            glTexCoord2f((N-i+1-1)/NF, (N-j-1)/NF);
            glVertex3fv(tab[i-1][j+1]);

            glNormal3fv(normal[i-1][j]);
            glTexCoord2f((N-i+1-1)/NF, (N-j)/NF);
            glVertex3fv(tab[i-1][j]);
        }
    }
```



Rysunek 3: Oteksturowane jajko

5 Wnioski

Teksturowanie obiektów 3-D z użyciem biblioteki OpenGL nie jest skomplikowanym zadaniem. Należy jednak pamiętać o odpowiednim przeliczaniu parametrów punktów tekstury. Problemem okazało się też samo wczytywanie tekstury wynikające z różnic implmentacji pod systemami Windows oraz Max OS X.