

Grafika Komputerowa. Podstawy teksturowania w OpenGL

Aleksander Denisiuk

Polsko-Japońska Akademia Technik Komputerowych

Wydział Informatyki w Gdańsku

ul. Targ Drzewny 9/11

80-894 Gdańsk

denisiuk@pja.edu.pl

Podstawy teksturowania w OpenGL

Modelowanie Torusa

Teksturowanie

Implementacja

Najnowsza wersja tego dokumentu dostępna jest pod adresem

<http://users.pja.edu.pl/~denisjuk>

Modelowanie Torusa

Torus oteksturowany

Modelowanie torusa

Niewidoczne ściany

Teksturowanie

Implementacja

Modelowanie Torusa

Torus oteksturowany

Modelowanie Torusa

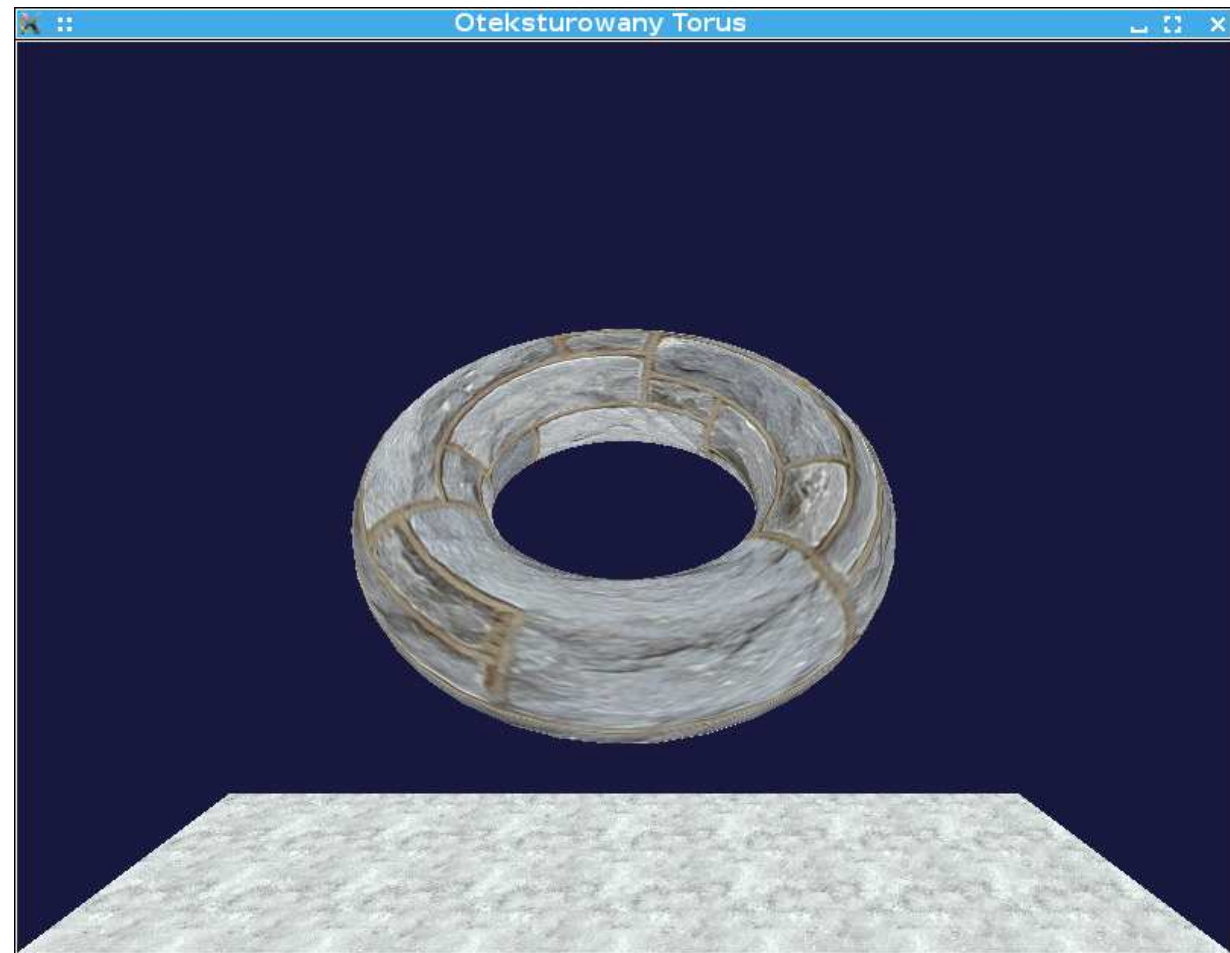
Torus oteksturowany

Modelowanie torusa

Niewidoczne ściany

Teksturowanie

Implementacja



Modelowanie Torusa

Torus otekstuwany

Modelowanie torusa

Niewidoczne ściany

Teksturowanie

Implementacja

- $P(\theta, \varphi) = \begin{pmatrix} (R + r \cos \varphi) \sin \theta \\ r \sin \varphi \\ (R + r \cos \varphi) \cos \theta \end{pmatrix},$
- $0 \leq \theta, \varphi \leq 2\pi$
- Mapowanie toroidalne: $\begin{cases} s = \frac{\theta}{2\pi}, \\ t = \frac{\varphi}{2\pi}. \end{cases}$
- Węzły: $\begin{cases} \theta_j = \frac{2\pi}{M}j, & j = 0, \dots, M, \\ \varphi_i = \frac{2\pi}{N}i, & i = 0, \dots, N. \end{cases}$

Triangulacja torusa (GL_TRIANGLE_STRIP)

Modelowanie Torusa

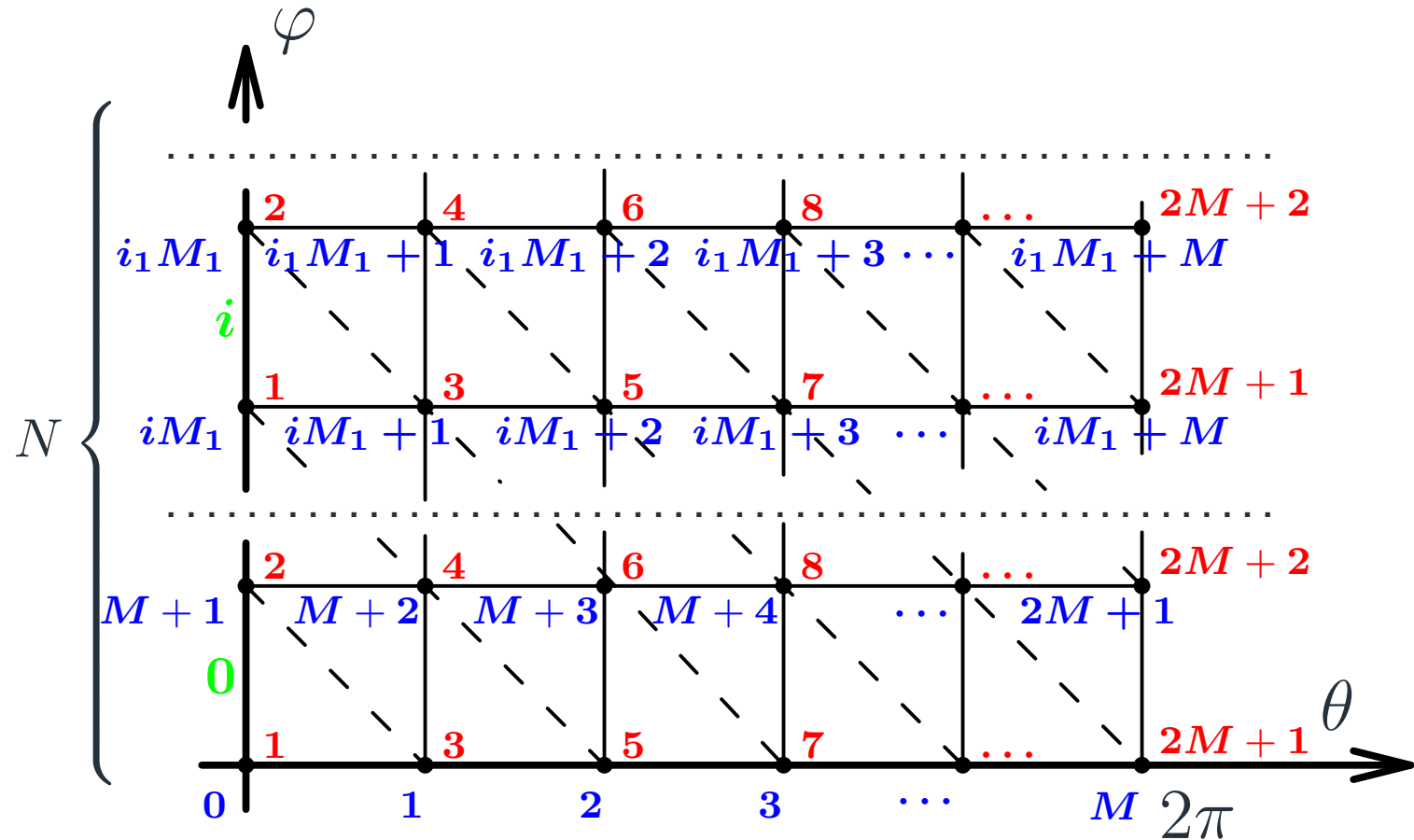
Torus oteksturowany

Modelowanie torusa

Niewidoczne ściany

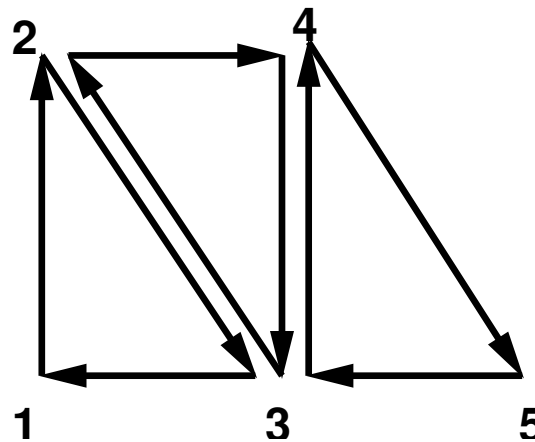
Teksturowanie

Implementacja



- wierzchołki, $M_1 = M + 1, i_1 = i + 1$
- indeksy
- paski (GL_TRIANGLE_STRIP)

- Ściany trójkątów, które są wewnątrz torusa nigdy nie będą widoczne
- Opcja `GL_CULL_FACE` pozwala na wyłączenie części ścian z renderowania
- Wyłączone ściany określa się przez orientację
 - ☐ `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
 - ☐ `GL_CW`, `GL_CCW`



Modelowanie Torusa

Torus otekstutowany

Modelowanie torusa

Niewidoczne ściany

Teksturowanie

Implementacja

```
glEnable(GL_CULL_FACE);
```

```
glCullFace(GL_BACK);
```

```
glFrontFace(GL_CW);
```

```
.....
```

```
glDisable(GL_CULL_FACE);
```


Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

Teksturowanie

Modelowanie Torusa

Teksturowanie

Tekstura

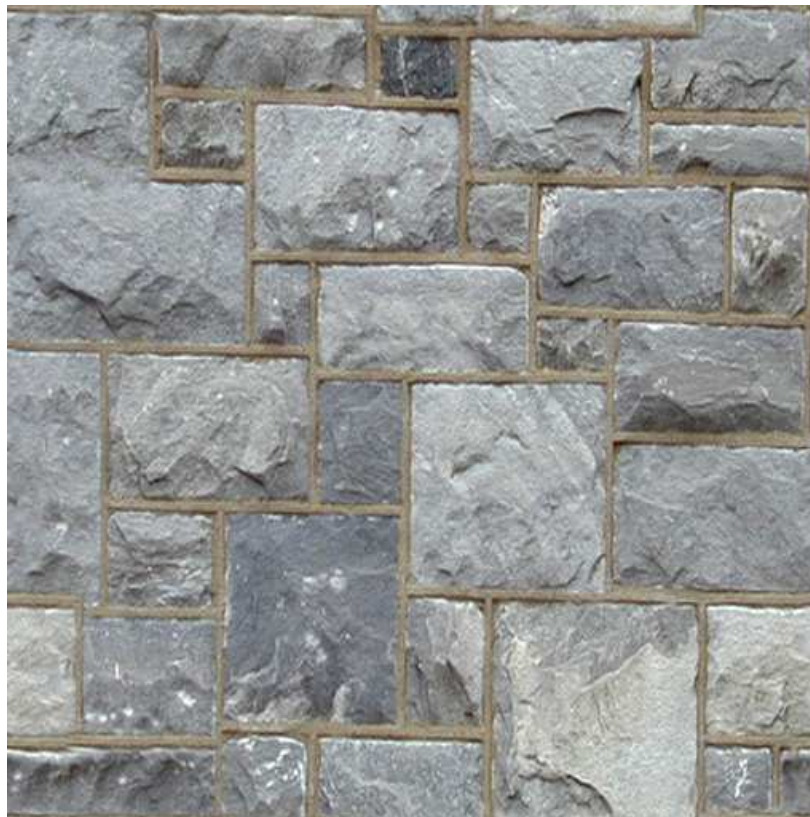
Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

- Obiekt tekstury na GPU
- Unit teksturowy na GPU
- Przykładowy obrazek:



Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

- `texture` jest zmienną typu `GLuint`

```
// nowy obiekt teksturowy  
glGenTextures(1, &texture);
```

```
// aktywizacja  
glBindTexture(GL_TEXTURE_2D, texture);
```

funkcja glBindTexture

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
void glBindTexture( enum target, uint texture );
```

- GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D, GL_TEXTURE_1D_ARRAY, GL_TEXTURE_2D_ARRAY, GL_TEXTURE_RECTANGLE, GL_TEXTURE_BUFFER, GL_TEXTURE_CUBE_MAP, GL_TEXTURE_2D_MULTISAMPLE, oraz GL_TEXTURE_2D_MULTISAMPLE_ARRAY

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
void glTexImage2D( enum target, int level,
                  int internalformat,
                  sizei width, sizei height,
                  int border, enum format,
                  enum type, const void *data );
```

- `border = 0`
- `width, height` — w każdej implementacji ograniczenie do co najmniej 1024
- 2^k
- `format` — `GL_RED`, `GL_RG`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, oraz `GL_BGRA`

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
glDeleteTextures(1, &texture);
```

Intrapolacja i ekstrapolacja aktywnej tekstury

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
// parametry interpolacji tekstury
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                GL_LINEAR);

// parametry ekstrapolacji tekstury
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
                GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
                GL_CLAMP_TO_EDGE);
```

- w shaderze — zmienna uniform (typu `sampler2D`)
`uniform sampler2D Texture;`
- przypisać numer unitu (0, 1, etc), jak zwykłej zmiennej uniform
- aktywizacja unitu teksturowego i dowiązanie do niego obiektu tekstury w programie

```
glActiveTexture(GL_TEXTURE0);  
glBindTexture(GL_TEXTURE_2D, texture);
```

- ☐ ten sam unit może działać z różnymi teksturami
- ☐ zazwyczaj tego samego typu

- pobieranie teksela (w shaderze)

```
out_Color = texture(Texture, Texcoord);
```

- ☐ gdzie `Texcoord` jest zmienną typu `vec2`

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

- inaczej rezerwuje się miejsce w GPU
- przy wysyłaniu tekstury karta graficzna może utworzyć mipmapę

```
glBindTexture(GL_TEXTURE_2D, texture);
```

```
glTexStorage2D(GL_TEXTURE_2D, levels, internal_format,  
               width, height);
```

```
glTexSubImage2D(GL_TEXTURE_2D, levels,  
                x_offset, y_offset,  
                width, height, format, type, image);
```

```
glGenerateMipmap(GL_TEXTURE_2D);
```

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
glBindTexture(GL_TEXTURE_2D, texture);

glTexStorage2D(GL_TEXTURE_2D, 7, GL_RGBA,
               512, 512);

glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 512,
                512, GL_RGBA, GL_UNSIGNED_BYTE,
                (const GLvoid*)image);

glGenerateMipmap(GL_TEXTURE_2D);
```

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
glTexParameteri(GL_TEXTURE_2D,  
                GL_TEXTURE_MIN_FILTER,  
                GL_LINEAR_MIPMAP_LINEAR);
```

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

```
glfwWindowHint(GLFW_SAMPLES, 8);  
glEnable(GL_MULTISAMPLE);
```

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

- opracowany przez Truevision w 1986
- jest bardzo prostym formatem, bez patentów
- składa się z:
 - ☐ nagłówka
 - ☐ danych

Modelowanie Torusa

Teksturowanie

Tekstura

Mipmapping

Nadpróbkowanie

Format TGA

Implementacja

1. (1 byte) długość ID
2. (1 byte) typ mapy kolorów
3. (1 byte) typ obrazka (2 — bez kompresji, bez mapy kolorów)
4. (5 bytes) specyfikacja mapy kolorów
5. (10 bytes) specyfikacja obrazka
 - (a) (2 bytes) współrzędna x lewego dolnego rogu
 - (b) (2 bytes) współrzędna y lewego dolnego rogu
 - (c) (2 bytes) szerokość obrazka w pikselach
 - (d) (2 bytes) wysokość obrazka w pikselach
 - (e) (1 byte) bitów na pixel (24 — rgb, 32 — rgb α)
 - (f) (1 byte)
 - i. bity 3–0 określają kanał α (bitów na pixel),
 - ii. bity 5–4 określają kierunek, (w prawo, w dół)
 - iii. bity 8–7 są zerowe

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

`vertices.h`

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

Implementacja

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
#version 430 core
```

```
layout(location=0) in vec4 in_position;
```

```
layout(location=2) in vec2 in_texture;
```

```
uniform mat4 model_matrix;
```

```
uniform mat4 view_matrix;
```

```
uniform mat4 projection_matrix;
```

```
out vec2 tex_coord;
```

```
void main(void){
```

```
    gl_Position = (projection_matrix * view_matrix  
        * model_matrix) * in_position;
```

```
    tex_coord = in_texture;
```

```
}
```


Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
#version 430 core
```

```
layout (location = 0) out vec4 color;
```

```
in vec2 tex_coord;
```

```
uniform sampler2D texture_unit;
```

```
void main(void){  
    color = texture(texture_unit, tex_coord);  
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

- `TextureModel` — model z teksturą
- `Torus` — torus
- `Plane` — płaszczyzna
- `TextureCameraProgram` — nowy program, z teksturą
- `Texture` — tekstura

Nowa struktura dla wierzchołków

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
typedef struct TextureVertex{  
    float position[4];  
    float texture[2];  
} TextureVertex;
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

TextureModel

texture_unit_ : GLuint

texture_ : GLuint

+ SetTexture (t: GLuint)

+ SetTextureUnit (t: GLuint)

```
class TextureModel{
public:
    void SetTextureUnit(GLuint t){texture_unit_=t;}
    void SetTexture(GLuint t){texture_ = t;}
protected:
    GLuint texture_unit_;
    GLuint texture_;
};
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

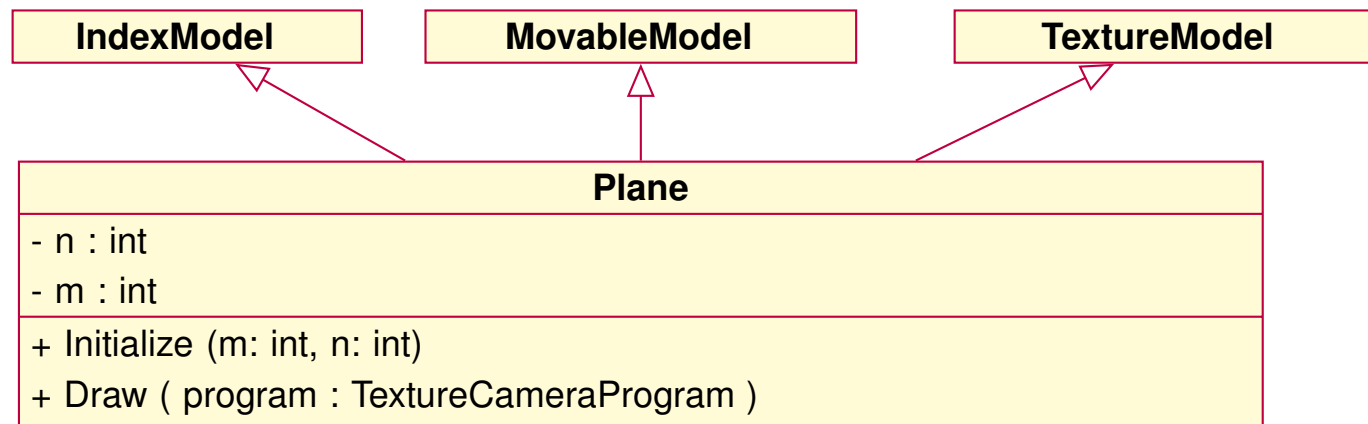
Plane

Torus

TextureCameraProgram

Texture

Window



```
class Plane : public IndexModel,
              public MovableModel, public TextureModel{
public:
    void Initialize(int m, int n);
    void Draw(const Program & program) const;
private:
    int n_; // mesh parameter
    int m_; // mesh parameter
};
```

Inicjalizacja. Wierzchołki i indeksy

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Plane::Initialize(int m, int n){
    m_=m;
    n_=n;
    model_matrix_.Translate(0, -4, 0);
    TextureVertex vertices[4]={
        {{-(float)(m),0.0f,-(float)(n),1.0f},
          {-(float)(m),-(float)(n)}}},
        {{(float)(m),0.0f,-(float)(n),1.0f},
          {(float)(m),-(float)(n)}}},
        {{(float)(m),0.0f,(float)(n),1.0f},
          {(float)(m),(float)(n)}}},
        {{-(float)(m),0.0f,(float)(n),1.0f},
          {-(float)(m),(float)(n)}}}
    };
    GLuint indices[4]={ 0, 1, 3, 2};
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
glGenVertexArrays(1, &vao_);
glBindVertexArray(vao_);
glGenBuffers(1, &vertex_buffer_);
glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer_);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices),
             vertices, GL_STATIC_DRAW);
glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE,
                    sizeof(vertices[0]), (GLvoid*)0);
glEnableVertexAttribArray(0);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE,
                    sizeof(vertices[0]),
                    (GLvoid*)sizeof(vertices[0].position));
glEnableVertexAttribArray(2);
glGenBuffers(1, &index_buffer_);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, index_buffer_);
glBufferData(GL_ELEMENT_ARRAY_BUFFER,
            sizeof(indices), indices, GL_STATIC_DRAW);
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Plane::Draw(const Program & program) const{
    glBindVertexArray(vao_);
    glUseProgram(program);
    program.SetModelMatrix(model_matrix_);
    glActiveTexture(texture_unit_);
    glBindTexture(GL_TEXTURE_2D, texture_);

    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glFrontFace(GL_CW);
    glDrawElements(GL_TRIANGLE_STRIP, 4,
                  GL_UNSIGNED_INT, (GLvoid*)0);
    glDisable(GL_CULL_FACE);

    glBindVertexArray(0);
    glUseProgram(0);
}
```


Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

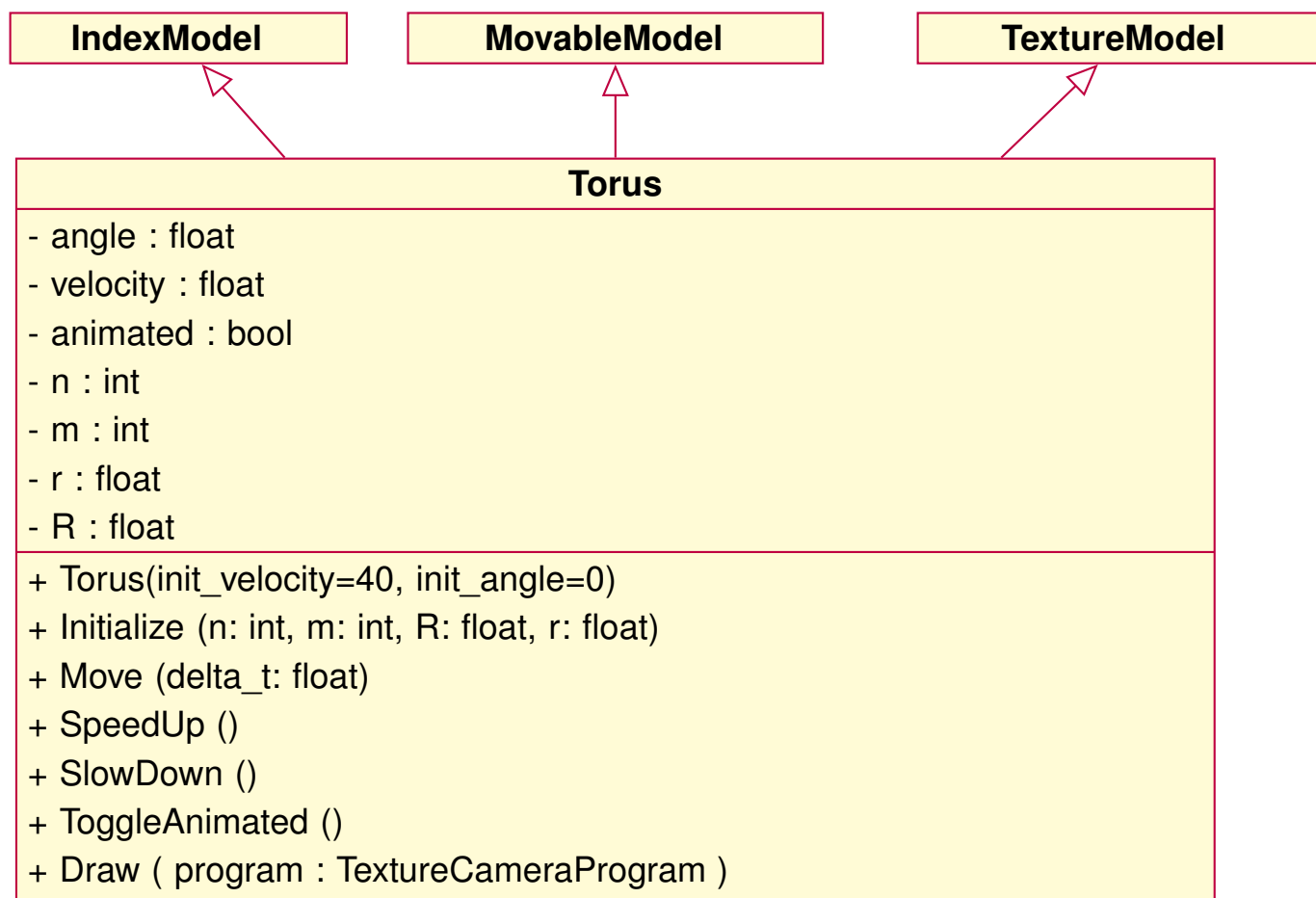
Plane

Torus

TextureCameraProgram

Texture

Window



Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
TextureVertex* vertices
    =new TextureVertex[(m_+1)*(n_+1)];
int i,j;
for (i=0;i<=n_;i++) {
    float phi=2*M_PI/(float)n_*i;
    for (j=0;j<=m_;j++){
        float theta=2*M_PI/(float)m_*j;
        vertices[i*(m_+1)+j].position[0]
            =(R + r*cos(phi)) * sin(theta);
        vertices[i*(m_+1)+j].position[1]=r*sin(phi);
        vertices[i*(m_+1)+j].position[2]
            =(R + r*cos(phi))*cos(theta);
        vertices[i*(m_+1)+j].position[3]=1.0f;
        vertices[i*(m_+1)+j].texture[0]=(float)j/(float)m_;
        vertices[i*(m_+1)+j].texture[1]=(float)i/(float)n_;
    }
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
GLuint * indices=new GLuint[2*n_*(m_ + 1)];

unsigned int k=0;

for(i=0; i<=n_ - 1; i++){
    for(j=0; j<=m_; j++){
        indices[2*(i*(m_ + 1)+j)]=k;
        indices[2*(i*(m_ + 1)+j)+1]=k+m_+1;
        k++;
    }
}
```

Inicjalizacja. VAO i VBO dla wierzchołków

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
glGenVertexArrays(1, &vao_);  
glBindVertexArray(vao_);
```

```
glGenBuffers(1, &vertex_buffer_);  
glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer_);  
glBufferData(GL_ARRAY_BUFFER,  
             (m_+1)*(n_+1)*sizeof(vertices[0]),  
             vertices,  
             GL_STATIC_DRAW);
```

Inicjalizacja. Argumenty shadera wierzchołków

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
glVertexAttribPointer(0, 4, GL_FLOAT,  
    GL_FALSE, sizeof(vertices[0]), (GLvoid*)0);  
glEnableVertexAttribArray(0);  
glVertexAttribPointer(2, 2, GL_FLOAT,  
    GL_FALSE, sizeof(vertices[0]),  
    (GLvoid*)sizeof(vertices[0].position));  
glEnableVertexAttribArray(2);
```

- dla współrzędnych tekstury wykorzystywany argument 2
- argument 1 zostawmy dla kolorów

Inicjalizacja. VBO dla indeksów

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
glGenBuffers(1, &index_buffer_);  
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,  
              index_buffer_);  
glBufferData(GL_ELEMENT_ARRAY_BUFFER,  
              (m_+1)*n_*2*sizeof(GLuint),  
              indices,  
              GL_STATIC_DRAW  
              );
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
delete [] vertices;  
delete [] indices;  
glBindBuffer(GL_ARRAY_BUFFER, 0);  
glBindVertexArray(0);  
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Torus::Draw(const Program &program) const{
    glBindVertexArray(vao_);
    glUseProgram(program);
    program.SetModelMatrix(model_matrix_);
    glActiveTexture(texture_unit_);
    glBindTexture(GL_TEXTURE_2D, texture_);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glFrontFace(GL_CW);
    for (int i=0;i<n_;i++){
        glDrawElements(GL_TRIANGLE_STRIP,
            2*(m_ + 1), GL_UNSIGNED_INT,
            (GLvoid*)(sizeof(GLuint)*2*i*(m_+1)));
    }
    glDisable(GL_CULL_FACE);
    glBindVertexArray(0);
    glUseProgram(0);
}
```


Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

ModelProgram

TextureCameraProgram

- texture_unit_location_ : GLint

+ Initialize (vertex_shader_file, fragment_shader_file)

+ SetTextureUnit (t : GLuint)

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void TextureCameraProgram::Initialize(  
    const char *vertex_shader_file,  
    const char *fragment_shader_file){  
    ModelProgram::Initialize(  
        vertex_shader_file,  
        fragment_shader_file);  
    texture_unit_location_  
        = glGetUniformLocation("texture_unit");  
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void TextureCameraProgram::SetTextureUnit(  
    GLuint t) const{  
    glUniform1i(texture_unit_location_, t);  
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
class Texture{
public:
    ~Texture();
    void Initialize(const char *filename);
    operator GLuint() const {return texture_;}
private:
    GLuint texture_;
    void LoadTGAFileOrDie(GLenum target,
                          const char * filename);
};
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Texture::Initialize(const char * filename){
    glGenTextures(1, &texture_);
    // aktywacja
    glBindTexture(GL_TEXTURE_2D, texture_);
    LoadTGAFileOrDie(GL_TEXTURE_2D, filename);
    // parametry interpolacji tekstury
    glTexParameteri(GL_TEXTURE_2D,
                    GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D,
                    GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    // parametry ekstrapolacji tekstury
    glTexParameteri(GL_TEXTURE_2D,
                    GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D,
                    GL_TEXTURE_WRAP_T, GL_REPEAT);
    glBindTexture(GL_TEXTURE_2D, 0);
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
Texture::~Texture(){  
    glDeleteTextures(1, &texture_);  
}
```

Wczytywanie z pliku. Struktura nagłówka TGA

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
typedef struct TGAHeader{
    uint8_t    idlength;
    uint8_t    colormap;
    uint8_t    datatype;
    uint8_t    colormapinfo[5];
    uint16_t   xorigin;
    uint16_t   yorigin;
    uint16_t   width;
    uint16_t   height;
    uint8_t    bitperpel;
    uint8_t    description;
} TGAHeader;
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Texture::LoadTGAFileOrDie(GLenum target,
                                const char * filename){
    TGAHeader* header;
    GLchar* buffer;
    size_t file_size;
    GLuint format;
    GLuint internal_format;
```


LoadTGAFileOrDie. Wczytywanie pliku

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
ifstream file (filename,
                ios::in|ios::ate|ios::binary);
if (file.is_open()) {
    file_size = file.tellg();
    buffer = new GLchar [file_size];
    file.seekg (0, ios::beg);
    file.read (buffer, file_size);
    file.close();
}
else{ //file was not opened
    cerr<<"Could not open the texture file "
         << filename << endl;
    exit(EXIT_FAILURE);
}
```

LoadTGAFileOrDie. Testowanie danych

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
if (file_size <= sizeof(TGAHeader)) {
    cerr<<"Too small texture file "
        <<filename<<endl;

    delete [] buffer;
    glfwTerminate();
    exit(EXIT_FAILURE);
}

header = (TGAHeader*)buffer;
// tylko nieskompresowany RGB lub RGBA obraz
if (header->datatype != 2
    || (header->bitperpel != 24
        && header->bitperpel != 32)){
    cerr<<"Wrong TGA format "<<filename<<endl;
    delete [] buffer;
    glfwTerminate();
    exit(EXIT_FAILURE);
}
```

LoadTGAFileOrDie. Wysyłanie danych i zakończenie

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
format =
    (header->bitperpel == 24 ? GL_BGR : GL_BGRA);
internal_format =
    (format == GL_BGR ? GL_RGB8 : GL_RGBA8);

glTexImage2D(target, 0, internal_format,
    header->width, header->height, 0, format,
    GL_UNSIGNED_BYTE,
    (const GLvoid*)(buffer
        + sizeof(TGAHeader) + header->idlength));

delete [] buffer;
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
private:
    Torus torus_;
    Plane plane_;
    Texture color_texture_;
    void InitTextures();
```

.....

Stałe w pliku window.h

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
const char* kVertexShader="TextureShader.vertex.glsl";
const char* kFragmentShader="TextureShader.fragment.glsl";
const char* kColorTextureFile="texture.tga";
const char* kIceTextureFile="ice.tga";
const int kPlaneM=10;
const int kPlaneN=10;
const int kTorusM=32;
const int kTorusN=32;
const float kTorusR=2.5f;
const float kTorusr=0.5f;
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

`InitTextures();`

`InitModels();`

`InitPrograms();`

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Window::InitTextures(){
    color_texture_.Initialize(kColorTextureFile);
    ice_texture_.Initialize(kIceTextureFile);
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Window::InitModels(){
    torus_.Initialize(kTorusN, kTorusM,
                     kTorusR, kTorusr);
    torus_.SetTexture(color_texture_);
    torus_.SetTextureUnit(GL_TEXTURE0);

    plane_.Initialize(kPlaneM, kPlaneN);
    plane_.SetTexture(ice_texture_);
    plane_.SetTextureUnit(GL_TEXTURE0);
}
```


Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Window::InitPrograms(){
    program_.Initialize(kVertexShader,
                       kFragmentShader);
    program_.SetTextureUnit(0);
}
```

Modelowanie Torusa

Teksturowanie

Implementacja

Shadery

Nowe klasy C++

vertices.h

Hierarchia modeli

Plane

Torus

TextureCameraProgram

Texture

Window

```
void Window::Run(void){
    while (!glfwWindowShouldClose(window_)){
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        clock_t now = clock();
        if (last_time_ == 0) last_time_ = now;
        torus_.Move((float)(now - last_time_)
                    / CLOCKS_PER_SEC );

        last_time_ = now;

        torus_.Draw(program_);
        plane_.Draw(program_);

        glfwSwapBuffers(window_);
        glfwPollEvents();
    }
}
```