

Zaawansowane systemy programowania grafiki. Światła punktowe, kierunkowe, spot. Tłumienie Światła

Aleksander Denisiuk
Uniwersytet Warmińsko-Mazurski
Olsztyn, ul. Słoneczna 54
denisjuk@matman.uwm.edu.pl

3 marca 2021

Światła punktowe, kierunkowe, spot. Tłumienie Światła

Model Oświetlenia

Shadery

Implementacja

Najnowsza wersja tego dokumentu dostępna jest pod adresem

<http://wmii.uwm.edu.pl/~denisjuk/uwm>

Model Oświetlenia

- ❖ Trzy światła
- ❖ Obliczenia

Shadery

Implementacja

Model Oświetlenia

Światło punktowe z tłumieniem

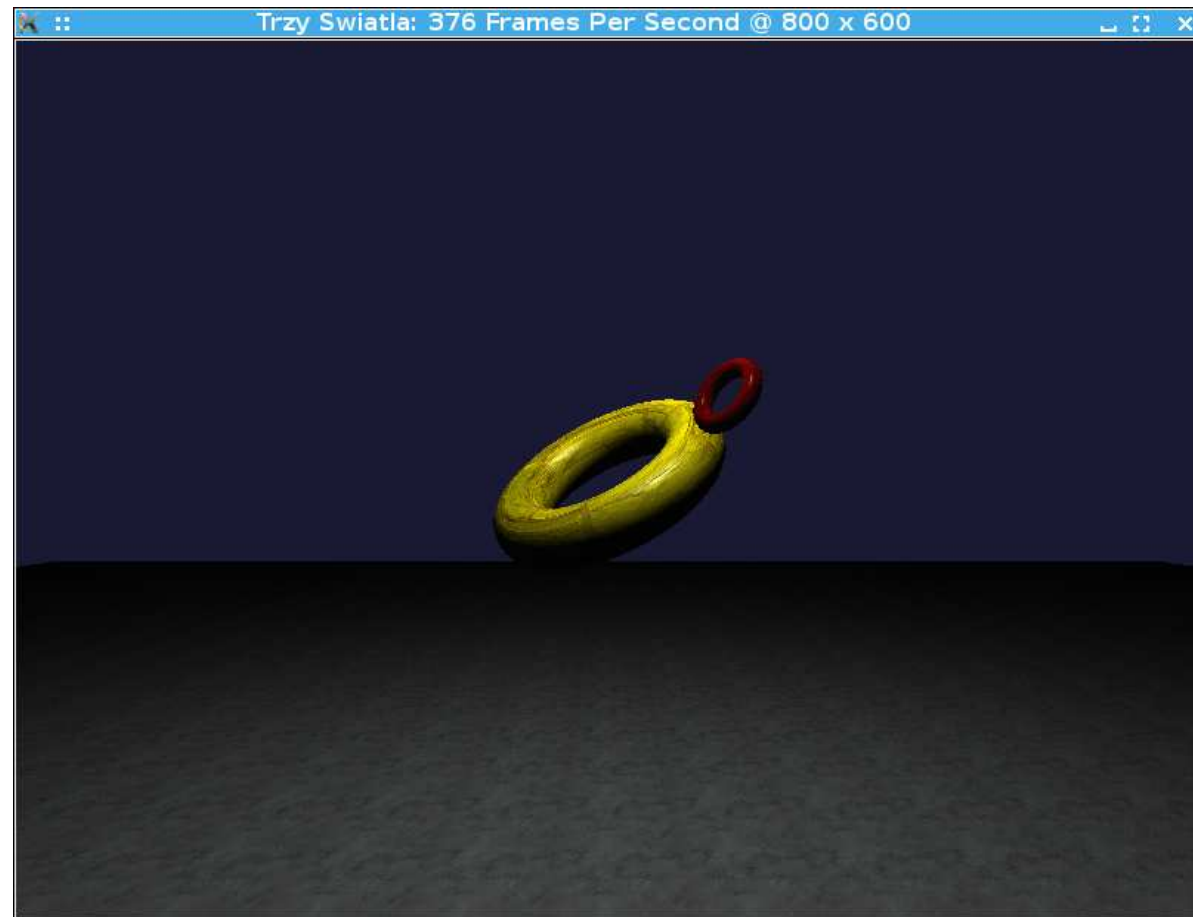
Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja



Światło kierunkowe

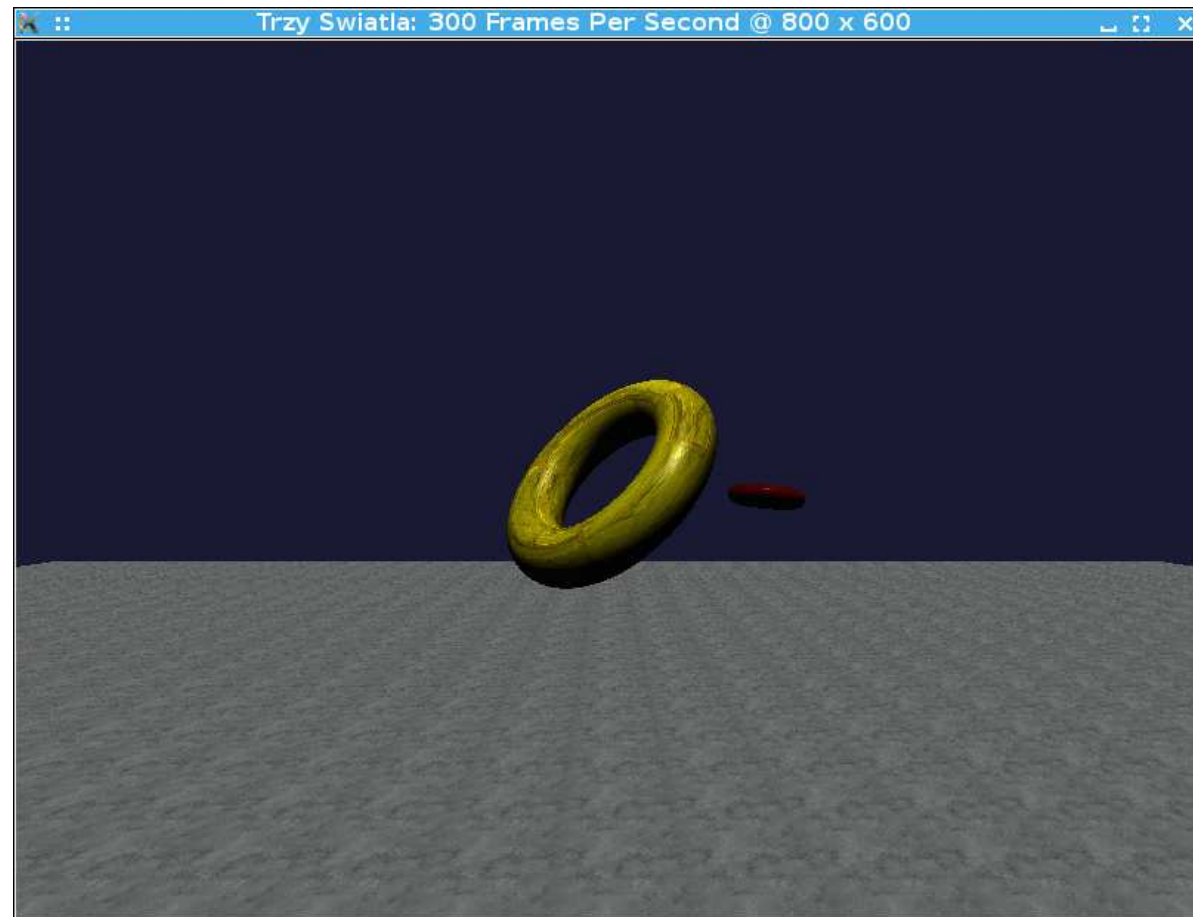
Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja



Światło spot

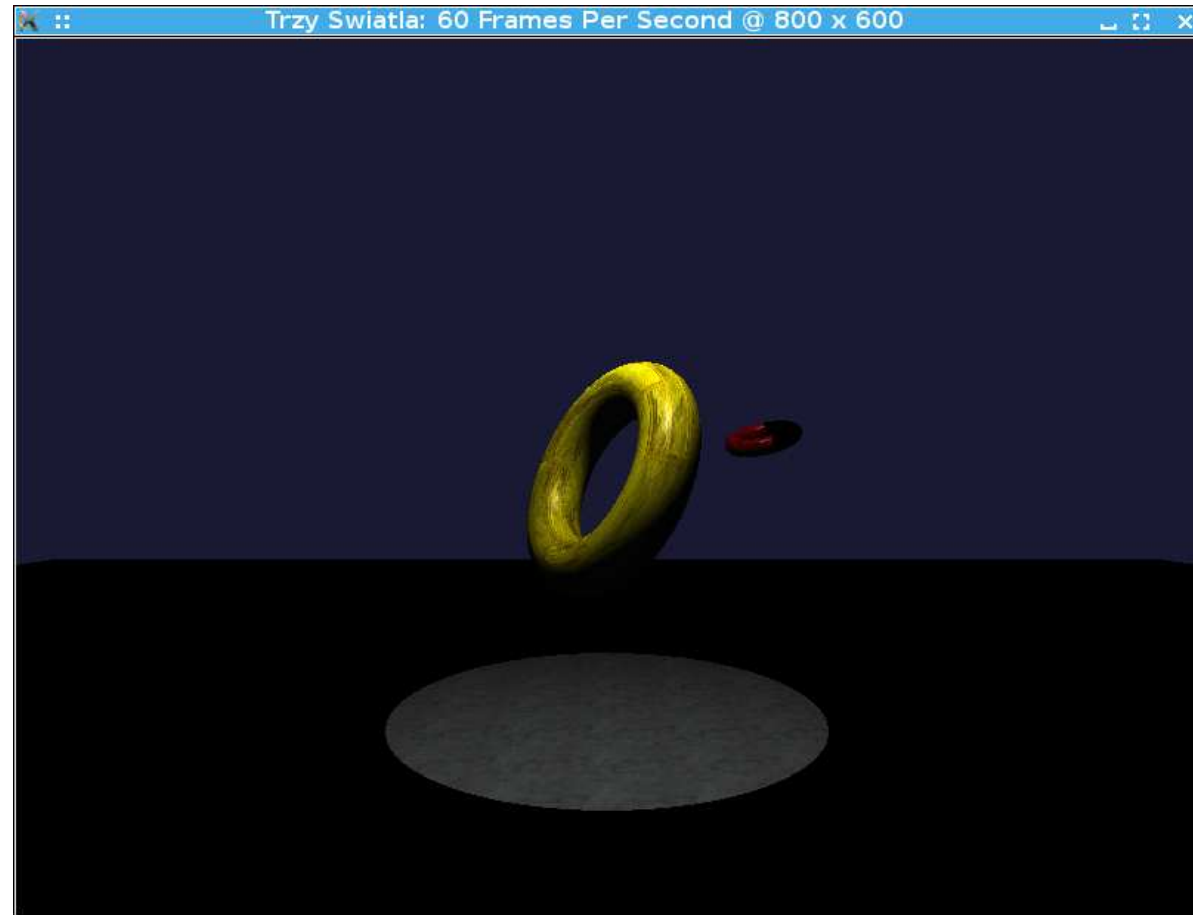
Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja



Światło punktowe

Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja

- $I = I_s + I_d + I_a + I_e$
 - ◆ $I_d = \rho_d I_d^{In} \cos \theta$
 - ◆ $I_s = \rho_s I_s^{In} (\cos \varphi)^f$
 - ◆ $I_a = \rho_a I_a^{In}$
 - ◆ $I_e = \text{Const}$

- Tłumienie światła

- ◆ współczynnik tłumienia

$$a = \frac{1}{a_0 + a_1 d + a_2 d^2},$$

- ◆ gdzie d jest odległością od źródła światła

Światło kierunkowe (Sun)

Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja

- jak światło punktowe
- źródło światła umieszczone jest w nieskończoności
- $(x_0, y_0, z_0, 0)$
- brak tłumienia (czemu?)

Światło spot

Model Oświetlenia

❖ Trzy światła

❖ Obliczenia

Shadery

Implementacja

- jak światło punktowe
- kierunek
- kąt obcinania (cutoff), ψ_0
- wskaźnik tłumienia, p
-

$$I = \begin{cases} I_0(\cos \psi)^p, & \text{jeżeli } \psi < \psi_0 \\ 0 & \end{cases}$$

- ◆ Uwaga: $\psi < \psi_0 \iff \cos \psi > \cos \psi_0$
- ◆ Do shadera prześlemy cosinus (czemu?)

Model Oświetlenia

Shadery

- ❖ Point Light
- ❖ SunLight
- ❖ SpotLight

Implementacja

Shadery

Vertex Shader. Dane wejściowe i zmienne uniform

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
#version 430 core
```

```
layout (location=0) in vec4 in_position;  
layout (location=2) in vec2 in_texture;  
layout (location=3) in vec3 in_normal;
```

```
uniform mat4 model_matrix;  
uniform mat3 normal_matrix;  
uniform mat4 view_matrix;  
uniform mat4 projection_matrix;
```

Światło i struktura wyjściowa

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
uniform struct PointLight{  
    vec4 position;  
    vec4 ambient;  
    vec4 diffuse;  
    vec4 specular;  
    vec3 attenuation;  
}  
light;
```

```
out struct Vertex {  
    vec2 texcoord;  
    vec3 normal;  
    vec3 light_dir;  
    vec3 view_dir;  
    float dist;  
}  
frag_vertex;
```

main

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
void main(void) {  
    vec4 vertex = model_matrix * in_position;  
    frag_vertex.light_dir = (light.position.xyz  
                             - vertex.xyz);  
    frag_vertex.normal = normal_matrix * in_normal;  
    vec4 camera =  
        -view_matrix * vec4(0.0, 0.0, 0.0, 1);  
    frag_vertex.view_dir = camera.xyz - vertex.xyz;  
    frag_vertex.dist = distance(light.position,  
                                vertex);  
    gl_Position = (projection_matrix  
                   * view_matrix) * vertex;  
    frag_vertex.texcoord = in_texture;  
}
```

Fragment Shader. Dane wejściowe i wyjściowe

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
#version 430 core

layout (location = 0) out vec4 color;

in struct Vertex {
    vec2 texcoord;
    vec3 normal;
    vec3 light_dir;
    vec3 view_dir;
    float dist;
} frag_vertex;
```

Światło i materiał

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
uniform struct PointLight{  
    vec4 position;  
    vec4 ambient;  
    vec4 diffuse;  
    vec4 specular;  
    vec3 attenuation;  
}light;
```

```
uniform struct Material{  
    vec4 ambient;  
    vec4 diffuse;  
    vec4 specular;  
    vec4 emission;  
    float shininess;  
}material;
```

```
uniform sampler2D texture_unit;
```

main Tłumienie i wektory

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
void main(void) {  
    float attenuation = 1.0 /  
        (light.attenuation[0] +  
         light.attenuation[1] * frag_vertex.dist +  
         light.attenuation[2] * frag_vertex.dist  
         * frag_vertex.dist);  
  
    vec3 normal  
        = normalize(frag_vertex.normal);  
    vec3 light_dir  
        = normalize(frag_vertex.light_dir);  
    vec3 view_dir  
        = normalize(frag_vertex.view_dir);  
}
```


Obliczenie oświetlenia

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
color = material.ambient * light.ambient;
float n_dot_l
    = max(dot(normal, light_dir), 0.0);
color
    += material.diffuse * light.diffuse * n_dot_l;
float r_dot_v_pow
    = pow( max(
        dot(reflect(-light_dir, normal), view_dir),
        0.0), material.shininess);
color += material.specular
    * light.specular * r_dot_v_pow;
color *= attenuation;
color += material.emission;
color *=
    texture(material.color_texture,
            frag_vertex.texcoord);
```

SunLightShader.vertex.glsl

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
uniform struct SunLight {  
    vec4 position;  
    vec4 ambient;  
    vec4 diffuse;  
    vec4 specular;  
} light;
```

```
out Vertex {  
    vec2 texcoord;  
    vec3 normal;  
    vec3 light_dir;  
    vec3 view_dir;  
} frag_vertex;
```

.....

```
frag_vertex.light_dir = light.position.xyz;
```

Shader fragmentów

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

- jak w shaderze wierzchołków
 - ◆ bez tłumienia (*attenuation*)

Shader wierzchołków

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

- jak światło punkowe

Shader fragmentów

Model Oświetlenia

Shadery

❖ Point Light

❖ SunLight

❖ SpotLight

Implementacja

```
float spot_effect
    = dot(normalize(light.direction), -light_dir);
spot_effect
    = max(pow(spot_effect, light.exponent), 0.0);
float spot
    = float(spot_effect > light.cutoff);
float attenuation = spot * spot_effect / (
    light.attenuation[0]
    + light.attenuation[1] * frag_vertex.dist
    + light.attenuation[2] * frag_vertex.dist
        * frag_vertex.dist);

.....

color *= attenuation;
```

Model Oświetlenia

Shadery

Implementacja

- ❖ Nowe klasy C++
- ❖ `light.h`
- ❖ `material.h`
- ❖ `matma.h`
- ❖ `vertices.h`
- ❖ Hierarchia modeli
- ❖ `Tori`
- ❖ `Plane`
- ❖ `Plane`
- ❖ `LightProgram`
- ❖ `PointLightProgram`
- ❖ Inicjalizacja
- ❖ `SunLightProgram`,
`SpotLightProgram`
- ❖ `Window`
- ❖ Zdarzenia myszki

Implementacja

Nowe klasy C++

Model Oświetlenia

Shadery

Implementacja

❖ **Nowe klasy C++**

❖ `light.h`

❖ `material.h`

❖ `matma.h`

❖ `vertices.h`

❖ Hierarchia modeli

❖ `Tori`

❖ `Plane`

❖ `Plane`

❖ `LightProgram`

❖ `PointLightProgram`

❖ Inicjalizacja

❖ `SunLightProgram`,
`SpotLightProgram`

❖ `Window`

❖ Zdarzenia myszki

- `LightableModel` — model z macierzą `normal_matrix`
- `MaterialModel` — model z materiałem
- `Tori` — dwa torusy
- `Plane` — płaszczyzna
- `LightProgram` — klasa bazowa dla programów ze światłami
- `PointLightProgram` — światło punktowe
- `SunLightProgram` — światło kierunkowe
- `SpotLightProgram` — światło spot
- `light.h` — struktury dla światła
- `material.h` — struktury dla materiałów

Światło punktowe

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ **light.h**

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
typedef struct PointLight {  
    GLfloat position[4];  
    GLfloat ambient[4];  
    GLfloat diffuse[4];  
    GLfloat specular[4];  
    GLfloat attenuation[3];  
} PointLight;
```


Światło kierunkowe

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ **light.h**

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
typedef struct SunLight {  
    GLfloat position[4];  
    GLfloat ambient[4];  
    GLfloat diffuse[4];  
    GLfloat specular[4];  
} SunLight;
```

Światło spot

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
typedef struct SpotLight {  
    GLfloat position[4];  
    GLfloat ambient[4];  
    GLfloat diffuse[4];  
    GLfloat specular[4];  
    GLfloat attenuation[3];  
    GLfloat direction[3];  
    GLfloat cutoff;  
    GLfloat exponent;  
} SpotLight;
```

Struktura materiału

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
typedef struct Material {  
    GLfloat ambient[4];  
    GLfloat diffuse[4];  
    GLfloat specular[4];  
    GLfloat emission[4];  
    GLfloat shininess;  
} Material;
```

Macierz 3×3

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
class Mat3{
```

```
public:
```

```
    Mat3(); // Unit matrix
```

```
    operator const float* () const{return matrix_;
```

```
    void RotateAboutX(GLfloat angle); //gedrees
```

```
    void RotateAboutY(GLfloat angle); //gedrees
```

```
    void RotateAboutZ(GLfloat angle); //gedrees
```

```
    void Scale(GLfloat x_scale,  
               GLfloat y_scale, GLfloat z_scale);
```

```
    void SetUnitMatrix();
```

```
    void Log();
```

```
private:
```

```
    GLfloat matrix_[9]; // column-major
```

```
    void MultiplyBy(const Mat3 &);
```

```
    explicit Mat3(float);
```

```
};
```

Nowa struktura dla wierzchołków

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
typedef struct NormalTextureVertex{  
    float position[4];  
    float texture[2];  
    float normal[3];  
} NormalTextureVertex;
```

Model z macierzą `normal_matrix`

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ `light.h`

❖ `material.h`

❖ `matma.h`

❖ `vertices.h`

❖ Hierarchia modeli

❖ `Tori`

❖ `Plane`

❖ `Plane`

❖ `LightProgram`

❖ `PointLightProgram`

❖ Inicjalizacja

❖ `SunLightProgram`,
`SpotLightProgram`

❖ `Window`

❖ Zdarzenia myszki

LightableModel

```
# normal_matrix_ : Mat3
```

```
class LightableModel {  
protected:  
    Mat3 normal_matrix_;  
};
```

Model z materiałem

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

MaterialModel

material_ : Material

+ SetMaterial (mat: Material)

```
class MaterialModel{
public:
    void SetMaterial(const Material & material){
        material_=material;}
protected:
    Material material_;
};
```

Dwa torusy

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

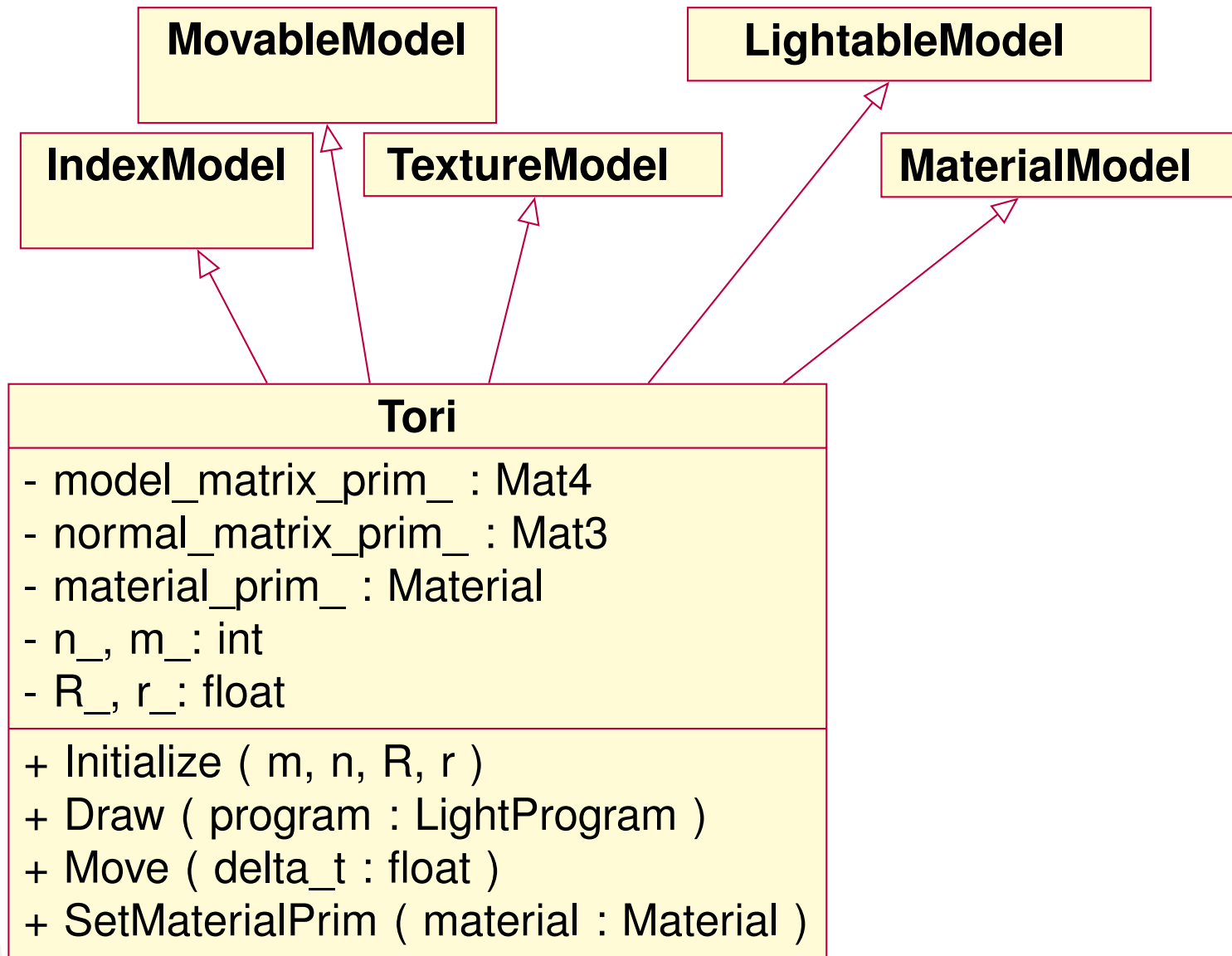
❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki



Deklaracja

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
class Tori: public IndexModel,
           public TextureModel,
           public MovableModel, public LightableModel,
           public MaterialModel{
public:
    void SetMaterialPrim(const Material
                        & material){
        material_prim_ = material;}

    .....

private:
    Mat4  model_matrix_prim_;
    Mat3  normal_matrix_prim_;
    Material material_prim_;
```

Uzupełnienie inicjalizacji

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
NormalTextureVertex * vertices
    =new NormalTextureVertex[(m_ + 1)*(n_ + 1)];
for (i=0; i<=n_; i++) {
    GLfloat phi=2*M_PI/(float)n_*i;
    for (j=0; j<=m_; j++) {
        GLfloat 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].normal[0]=cos(phi)*sin(theta);
        vertices[i*(m_+1)+j].normal[1]=sin(phi);
        vertices[i*(m_+1)+j].normal[2]=cos(phi)*cos(theta);
    }
}
```

Trzeci argument dla shadera wierzchołków

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
glVertexAttribPointer(3, 3, GL_FLOAT,  
                      GL_FALSE, sizeof(vertices[0]),  
                      (GLvoid*) (sizeof(vertices[0].position)  
                                + sizeof(vertices[0].texture)));  
glEnableVertexAttribArray(3);
```

Move () . Pierwszy torus

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Tori::Move(GLfloat delta_t) {  
    if (!animated_) return;  
    angle_ += delta_t * velocity_  
    if (angle_ > 360) angle_ -= 360;  
    if (angle_ < -360) angle_ += 360;  
  
    model_matrix_.SetUnitMatrix();  
    model_matrix_.RotateAboutX(angle_);  
    model_matrix_.RotateAboutY(angle_);  
    normal_matrix_.SetUnitMatrix();  
    normal_matrix_.RotateAboutY(-angle_);  
    normal_matrix_.RotateAboutX(-angle_);  
}
```

Move () . Drugi torus

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
model_matrix_prim_.SetUnitMatrix();  
model_matrix_prim_.Scale(0.2, 0.2, 0.2);  
model_matrix_prim_.RotateAboutX(2*angle_);  
model_matrix_prim_.Translate(-3, 0, 2);  
model_matrix_prim_.RotateAboutY(angle_);  
normal_matrix_prim_.SetUnitMatrix();  
normal_matrix_prim_.RotateAboutY(-angle_);  
normal_matrix_prim_.RotateAboutX(-2*angle_);  
normal_matrix_prim_.Scale(5.0, 5.0, 5.0);  
}
```

Draw () . Działania wstępne

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Tori::Draw(const Program &program) const{
```

```
    glBindVertexArray(vao_);
```

```
    glUseProgram(program);
```

```
    glActiveTexture(texture_unit_);
```

```
    glBindTexture(GL_TEXTURE_2D, texture_);
```

```
    glEnable(GL_CULL_FACE);
```

```
    glCullFace(GL_BACK);
```

```
    glFrontFace(GL_CW);
```

Draw () . Pierwszy torus

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
program.SetModelMatrix(model_matrix_);
program.SetMaterial(material_);
program.SetNormalMatrix(normal_matrix_);
for (int i=0; i<n_; i++) {
    glDrawElements(GL_TRIANGLE_STRIP,
        2 * (m_ + 1),
        GL_UNSIGNED_INT,
        (GLvoid*) (sizeof(GLuint) * 2 * i * (m_ + 1))
    );
}
```

Draw () . Drugi torus

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ **Tori**

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
program.SetModelMatrix(model_matrix_prim_);  
program.SetMaterial(material_prim_);  
program.SetNormalMatrix(normal_matrix_prim_);  
for (int i=0; i<n_; i++) {  
    glDrawElements(GL_TRIANGLE_STRIP,  
        2 * (m_ + 1),  
        GL_UNSIGNED_INT,  
        (GLvoid*) (sizeof(GLuint) * 2 * i * (m_ + 1))  
    );  
}
```


Płaszczyzna

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ **Plane**

❖ Plane

❖ LightProgram

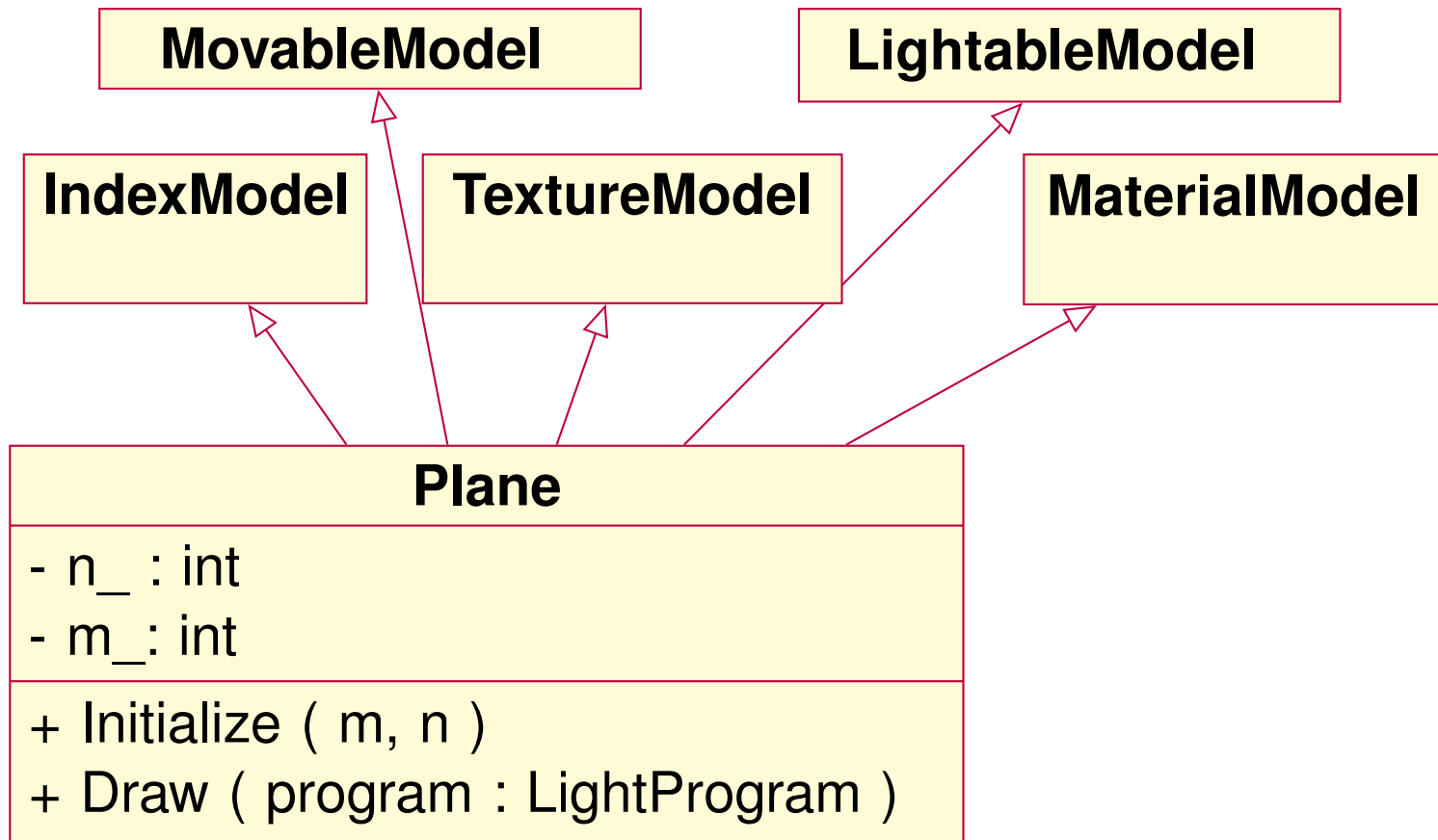
❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki



Zmiany w inicjalizacji

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
NormalTextureVertex* vertices
    = new NormalTextureVertex[ (m_ + m_ + 1)
                                * (n_ + n_ + 1) ];

for (i=n_; i>= -n_; i--) {
    for (j=-m_; j<=m_; j++) {
        int pos=(n_-i) * (m_+m_+1) + j+m_;
        vertices[pos].position[0]=j;
        vertices[pos].position[1]=0.0f;
        vertices[pos].position[2]=i;
        vertices[pos].position[3]=1.0f;
        vertices[pos].texture[0]=(float) j / (float) m_;
        vertices[pos].texture[1]=(float) i / (float) n_;
        vertices[pos].normal[0]=0.0f;
        vertices[pos].normal[1]=1.0f;
        vertices[pos].normal[2]=0.0f;
    }
}
```

Indeksy

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ **Plane**

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
GLuint* indices = new GLuint[4*n_*(2*m_+1)];
```

```
int k=0;
```

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

Wyświetlenie

Model Oświetlenia

Shadery

Implementacja

- ❖ Nowe klasy C++
- ❖ light.h
- ❖ material.h
- ❖ matma.h
- ❖ vertices.h
- ❖ Hierarchia modeli
- ❖ Tori
- ❖ Plane
- ❖ **Plane**
- ❖ LightProgram
- ❖ PointLightProgram
- ❖ Inicjalizacja
- ❖ SunLightProgram, SpotLightProgram
- ❖ Window
- ❖ Zdarzenia myszki

```
glBindVertexArray(vao_);
glUseProgram(program);
program.SetModelMatrix(model_matrix_);
program.SetNormalMatrix(normal_matrix_);
program.SetMaterial(material_);
glActiveTexture(texture_unit_);
glBindTexture(GL_TEXTURE_2D, texture_);
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);
glFrontFace(GL_CW);
for (int i=0; i<2*n_; i++) {
    glDrawElements(GL_TRIANGLE_STRIP, 2*(2*n_+1),
        GL_UNSIGNED_INT,
        (GLvoid*)(sizeof(GLuint) * 2*i*(2*m_+1)));
}
```

LightProgram

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

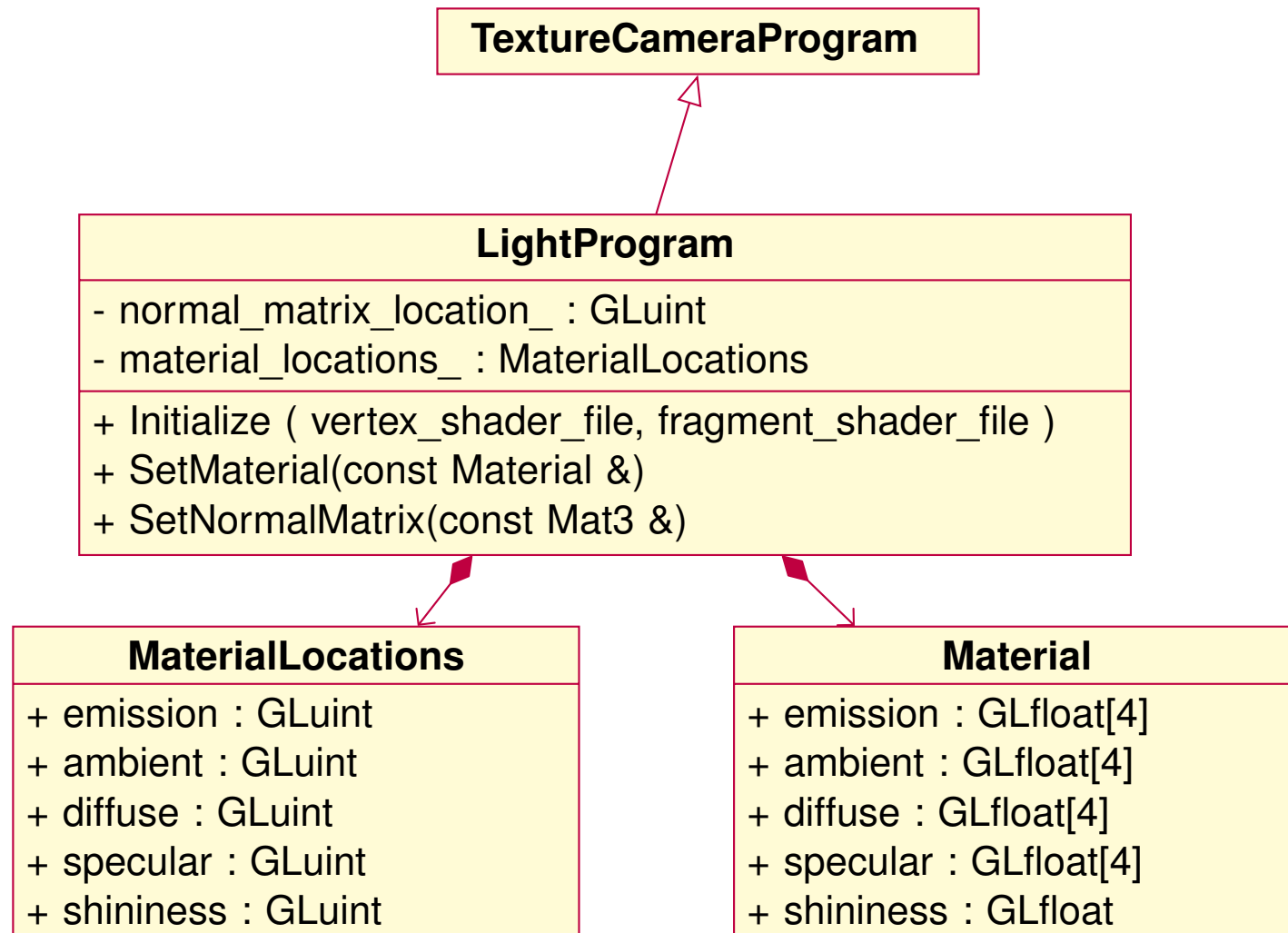
❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki



Inicjalizacja

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void LightProgram::Initialize(  
    const char *vertex_shader_file,  
    const char *fragment_shader_file) {  
    TextureCameraProgram::Initialize(  
        vertex_shader_file, fragment_shader_file);  
    normal_matrix_location_  
        = glGetUniformLocationOrDie("normal_matrix");  
    material_locations_.emission  
        = glGetUniformLocationOrDie("material.emission");  
    material_locations_.ambient  
        = glGetUniformLocationOrDie("material.ambient");  
    material_locations_.diffuse  
        = glGetUniformLocationOrDie("material.diffuse");  
    material_locations_.specular  
        = glGetUniformLocationOrDie("material.specular");  
    material_locations_.shininess  
        = glGetUniformLocationOrDie("material.shininess");  
}
```

Set...

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void LightProgram::SetMaterial(  
    const Material & material) const{  
    glUniform4fv(material_locations_.ambient,  
        1, material.ambient);  
    glUniform4fv(material_locations_.emission,  
        1, material.emission);  
    glUniform4fv(material_locations_.diffuse,  
        1, material.diffuse);  
    glUniform4fv(material_locations_.specular,  
        1, material.specular);  
    glUniform1f(material_locations_.shininess,  
        material.shininess);  
}  
  
void LightProgram::SetNormalMatrix(  
    const Mat3 & matrix) const{  
    glUniformMatrix3fv(normal_matrix_location_,  
        1, GL_TRUE, matrix);  
}
```

Program ze światłem punktowym

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

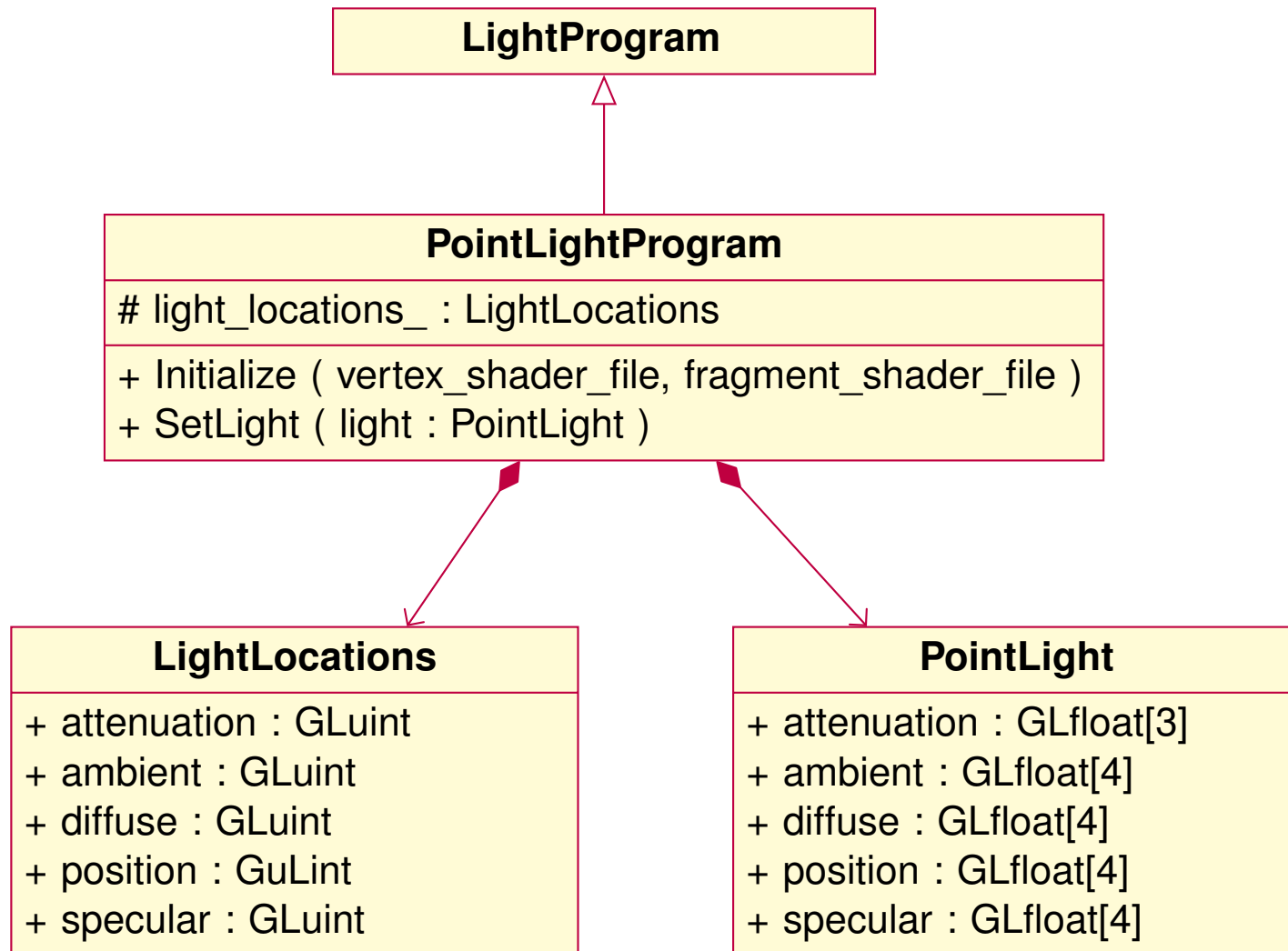
❖ **PointLightProgram**

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki



Inicjalizacja

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,

SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void PointLightProgram::Initialize(  
    const char *vertex_shader_file,  
    const char *fragment_shader_file) {  
    LightProgram::Initialize(vertex_shader_file,  
        fragment_shader_file);  
    light_locations_.ambient  
        = GetUniformLocationOrDie("light.ambient");  
    light_locations_.attenuation  
        = GetUniformLocationOrDie("light.attenuation");  
    light_locations_.diffuse  
        = GetUniformLocationOrDie("light.diffuse");  
    light_locations_.position  
        = GetUniformLocationOrDie("light.position");  
    light_locations_.specular  
        = GetUniformLocationOrDie("light.specular");  
}
```

SetLight

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void PointLightProgram::SetLight (  
    const PointLight & light) const {  
    glUniform4fv(light_locations_.ambient,  
        1, light.ambient);  
    glUniform4fv(light_locations_.diffuse,  
        1, light.diffuse);  
    glUniform4fv(light_locations_.specular,  
        1, light.specular);  
    glUniform4fv(light_locations_.position,  
        1, light.position);  
    glUniform3fv(light_locations_.attenuation,  
        1, light.attenuation);  
}
```

SunLightProgram, SpotLightProgram

Model Oświetlenia

Shadery

Implementacja

- ❖ Nowe klasy C++
- ❖ `light.h`
- ❖ `material.h`
- ❖ `matma.h`
- ❖ `vertices.h`
- ❖ Hierarchia modeli
- ❖ `Tori`
- ❖ `Plane`
- ❖ `Plane`
- ❖ `LightProgram`
- ❖ `PointLightProgram`
- ❖ Inicjalizacja
- ❖ `SunLightProgram`,
`SpotLightProgram`
- ❖ `Window`
- ❖ Zdarzenia myszki

● Analogicznie

Stałe dla materiałów

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
const Material kYellowMaterial={
    {0.2f, 0.2f, 0.2f, 1.0f}, //Ambient
    {1.0f, 1.0f, 0.0f, 1.0f}, //Diffuse
    {0.6f, 0.6f, 0.6f, 1.0f}, //Specular
    {0.0f, 0.0f, 0.0f, 1.0f}, //Emission
    60.0f
};
```

```
const Material kIceMaterial={
    {0.4f, 0.4f, 0.4f, 1.0f}, //Ambient
    {0.95f, 0.95f, 0.99f, 1.0f}, //Diffuse
    {0.99f, 0.99f, 0.99f, 1.0f}, //Specular
    {0.0f, 0.0f, 0.0f, 1.0f}, //Emission
    10.0f
};
```

Stałe dla świateł

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
const PointLight kPointLight={
    {0.0f, 7.5f, 3.0f, 1.0f}, //position
    {0.1f, 0.1f, 0.1f, 1.0f}, //ambient
    {1.0f, 1.0f, 1.0f, 1.0f}, //diffuse
    {1.0f, 1.0f, 1.0f, 1.0f}, //specular
    {0.5f, 0.005f, 0.0125f}    //attenuation
};
```

```
const SunLight kSunLight={
    {0.0f, 7.5f, 3.0f, 0.0f}, //position
    {0.1f, 0.1f, 0.1f, 1.0f}, //ambient
    {1.0f, 1.0f, 1.0f, 1.0f}, //diffuse
    {1.0f, 1.0f, 1.0f, 1.0f} //specular
};
```

Światło spot

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
const SpotLight kSpotLight={
    {0.0f, 7.5f, 3.0f, 1.0f}, //position
    {0.3f, 0.3f, 0.3f, 1.0f}, //ambient
    {1.0f, 1.0f, 1.0f, 1.0f}, //diffuse
    {1.0f, 1.0f, 1.0f, 1.0f}, //specular
    {0.5f, 0.005f, 0.0125f}, //attenuation
    {0.0f, -6.0f, -3.0f}, //direction
    20, //cutoff, degrees
    20 //exponent
};
```

Uzupełnienie klasy

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

public:

```
void MouseButton(int button,  
                 int state, int x_coord, int y_coord);  
void MouseMove(int x_coord, int y_coord);
```

private:

```
PointLightProgram point_program_;  
SunLightProgram sun_program_;  
SpotLightProgram spot_program_;
```

```
int x_origin_;  
int y_origin_;
```

```
enum {POINT_PROGRAM,  
      SUN_PROGRAM, SPOT_PROGRAM} current_program_;
```

```
void SetViewMatrix();  
void SetProjectionMatrix();
```

InitModels

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::InitModels() {
    tori_.Initialize(kTorusN, kTorusM,
                    kTorusR, kTorusr);
    tori_.SetTexture(color_texture_);
    tori_.SetTextureUnit(GL_TEXTURE0);
    tori_.SetMaterial(kBlueMaterial);
    tori_.SetMaterialPrim(kYellowMaterial);

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


InitPrograms

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::InitPrograms() {  
    view_matrix_.Translate(0, 0, -10);  
    projection_matrix_  
        = Mat4::CreateProjectionMatrix(60,  
            (float)width_ / (float)height_,  
                                           0.1f, 100.0f);  
}
```

InitPrograms. PointProgram

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
//point
```

```
point_program_.Initialize(kPointLightVertexShader,  
                           kPointLightFragmentShader);
```

```
glUseProgram(point_program_);
```

```
point_program_.SetLight(kPointLight);
```

```
point_program_.SetTextureUnit(0);
```

```
point_program_.SetProjectionMatrix(projection_matrix_);
```

```
point_program_.SetViewMatrix(view_matrix_);
```

InitPrograms. SunProgram, SpotProgram

Model Oświetlenia

Shadery

Implementacja

- ❖ Nowe klasy C++
- ❖ `light.h`
- ❖ `material.h`
- ❖ `matma.h`
- ❖ `vertices.h`
- ❖ Hierarchia modeli
- ❖ `Tori`
- ❖ `Plane`
- ❖ `Plane`
- ❖ `LightProgram`
- ❖ `PointLightProgram`
- ❖ Inicjalizacja
- ❖ `SunLightProgram`,
`SpotLightProgram`
- ❖ `Window`
- ❖ Zdarzenia myszki

● Analogicznie

SetViewMatrix

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram, SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::SetViewMatrix() {  
    glUseProgram(point_program_);  
    point_program_.SetViewMatrix(view_matrix_);  
    glUseProgram(sun_program_);  
    sun_program_.SetViewMatrix(view_matrix_);  
    glUseProgram(spot_program_);  
    spot_program_.SetViewMatrix(view_matrix_);  
    glUseProgram(0);  
}
```

SetProjectionMatrix

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::SetProjectionMatrix() {  
    glUseProgram(point_program_);  
    point_program_.SetProjectionMatrix(  
        projection_matrix_);  
    glUseProgram(sun_program_);  
    sun_program_.SetProjectionMatrix(  
        projection_matrix_);  
    glUseProgram(spot_program_);  
    spot_program_.SetProjectionMatrix(  
        projection_matrix_);  
    glUseProgram(0);  
}
```

Resize

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::Resize(int new_width,  
                    int new_height) {  
    width_ = new_width;  
    height_ = new_height;  
    projection_matrix_  
        = Mat4::CreateProjectionMatrix(60,  
        (float)width_ / (float)height_, 0.1f, 100.0f);  
    SetProjectionMatrix();  
    glViewport(0, 0, width_, height_);  
}
```

Zmiany w Run

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
switch (current_program_) {  
case POINT_PROGRAM:  
    tori_.Draw(point_program_);  
    plane_.Draw(point_program_);  
break;  
case SUN_PROGRAM:  
    tori_.Draw(sun_program_);  
    plane_.Draw(sun_program_);  
break;  
case SPOT_PROGRAM:  
    tori_.Draw(spot_program_);  
    plane_.Draw(spot_program_);  
break;  
}
```

Zmiany w KeyEvent

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
case GLFW_KEY_F1:
    current_program_ = POINT_PROGRAM;
break;
case GLFW_KEY_F2:
    current_program_ = SUN_PROGRAM;
break;
case GLFW_KEY_F3:
    current_program_ = SPOT_PROGRAM;
break;
```


Rejestracja callbacków w `main.cpp`

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ `light.h`

❖ `material.h`

❖ `matma.h`

❖ `vertices.h`

❖ Hierarchia modeli

❖ `Tori`

❖ `Plane`

❖ `Plane`

❖ `LightProgram`

❖ `PointLightProgram`

❖ Inicjalizacja

❖ `SunLightProgram`,
`SpotLightProgram`

❖ `Window`

❖ Zdarzenia myszki

```
void MouseScroll(GLFWwindow* /*window*/,  
    double xoffset, double yoffset) {  
    window.MouseScroll(xoffset, yoffset);  
}  
  
void MouseButton(GLFWwindow* /*window*/,  
    int button, int action, int mods) {  
    window.MouseButton(button, action, mods);  
}  
  
void MouseMove(GLFWwindow* /*window*/,  
    double x_pos, double y_pos) {  
    window.MouseMove(x_pos, y_pos);  
}
```

.....

```
glfwSetScrollCallback(window, MouseScroll);  
glfwSetMouseButtonCallback(window, MouseButton);  
glfwSetCursorPosCallback(window, MouseMove);
```

Sterowanie kamerą za pomocą kółka myszy

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::MouseScroll(double /* xoffset */,  
                        double yoffset) {  
    view_matrix_.Translate(0, 0, -0.25f*yoffset);  
    SetViewMatrix();  
}
```

Lewy przycisk

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::MouseButton(int button, int action,  
                           int /*mods*/) {  
    // only start motion if the left button is pressed  
    if (button == GLFW_MOUSE_BUTTON_LEFT) {  
        // when the button is released  
        if (action == GLFW_RELEASE) {  
            x_origin_ = -1;  
            y_origin_ = -1;  
        }  
        else { // state = GLFW_PRESS  
            double x_pos, y_pos;  
            glfwGetCursorPos(*this,  
                             &x_pos, &y_pos);  
            x_origin_ = x_pos;  
            y_origin_ = y_pos;  
        }  
    }  
}
```

Przemieszczenia

Model Oświetlenia

Shadery

Implementacja

❖ Nowe klasy C++

❖ light.h

❖ material.h

❖ matma.h

❖ vertices.h

❖ Hierarchia modeli

❖ Tori

❖ Plane

❖ Plane

❖ LightProgram

❖ PointLightProgram

❖ Inicjalizacja

❖ SunLightProgram,
SpotLightProgram

❖ Window

❖ Zdarzenia myszki

```
void Window::MouseMove(double x_pos,
                       double y_pos) {

    float delta_x_angle=0;
    float delta_y_angle=0;
    if (x_origin_ >= 0 && y_origin_ >=0) {
        // update deltaAngle
        delta_x_angle = (x_pos - x_origin_) * 0.1f;
        delta_y_angle = (y_pos - y_origin_) * 0.1f;
        x_origin_=x_pos;
        y_origin_=y_pos;

        // update camera's direction
        view_matrix_.RotateAboutY(delta_x_angle);
        view_matrix_.RotateAboutX(delta_y_angle);
        SetViewMatrix();
    }
}
```