Zaawansowane systemy programowania grafiki. Modelowanie nieba

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

12 kwietnia 2021

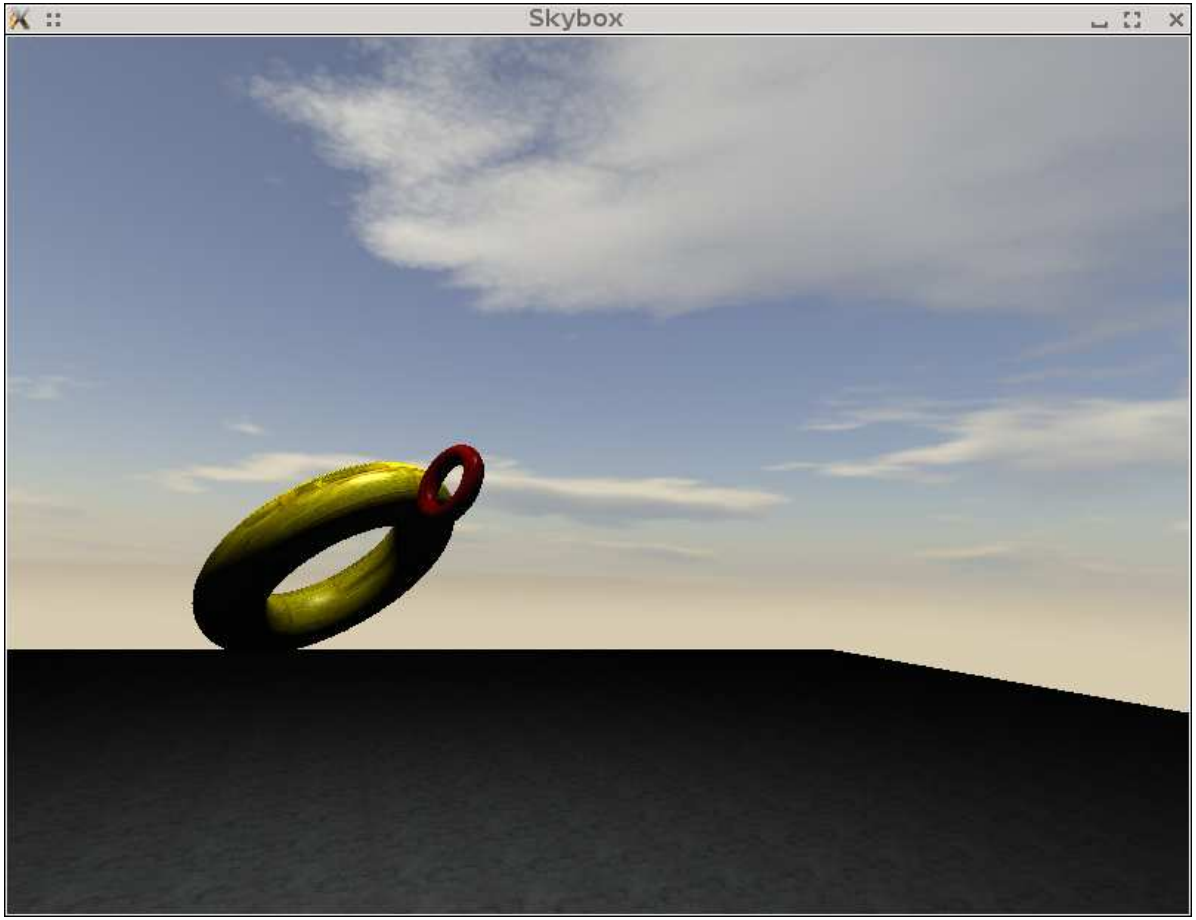
Modelowanie nieba

Najnowsza wersja tego dokumentu dostępna jest pod adresem

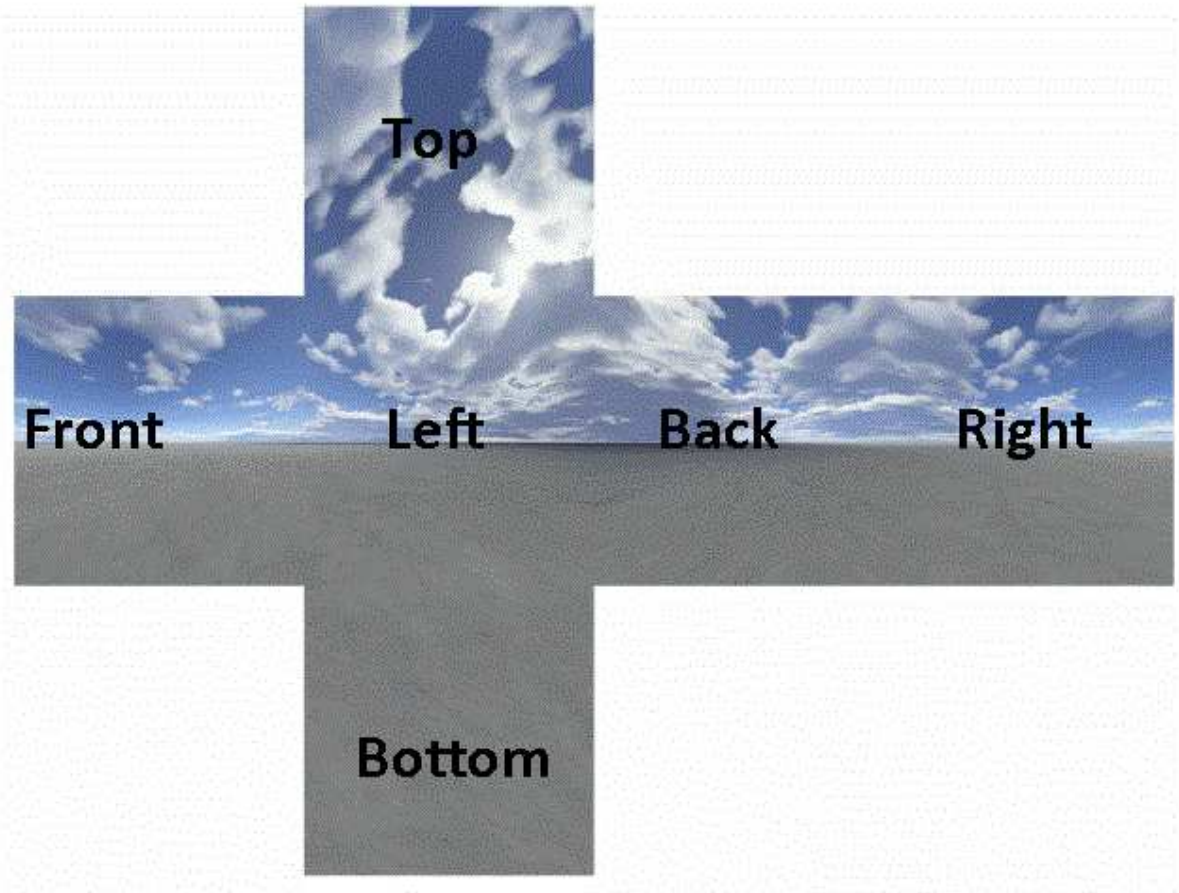
`http://wmii.uwm.edu.pl/~denisjuk/uwm`

Modelowanie nieba

Modelowanie nieba



Mapowanie sześcienne



- Współrzędne teksturowe są trójwymiarowe

Czynności

- utworzyć teksturę `GL_CUBE_MAP`
- zamodelowć sześcian
- oddzielny program cieniujący
  - ✦ brak oświetlenia, materiału, macierzy `model_matrix`

Implementacja *skyboksa*

Vertex Shader


```
#version 430 core

layout(location=0) in vec4 in_position;
layout(location=2) in vec3 in_texture;


uniform mat4 view_matrix;
uniform mat4 projection_matrix;

out vec3 tex_coord;

void main(void){
    tex_coord = in_texture;
    gl_Position = (projection_matrix
      * view_matrix) * in_position;
}
```

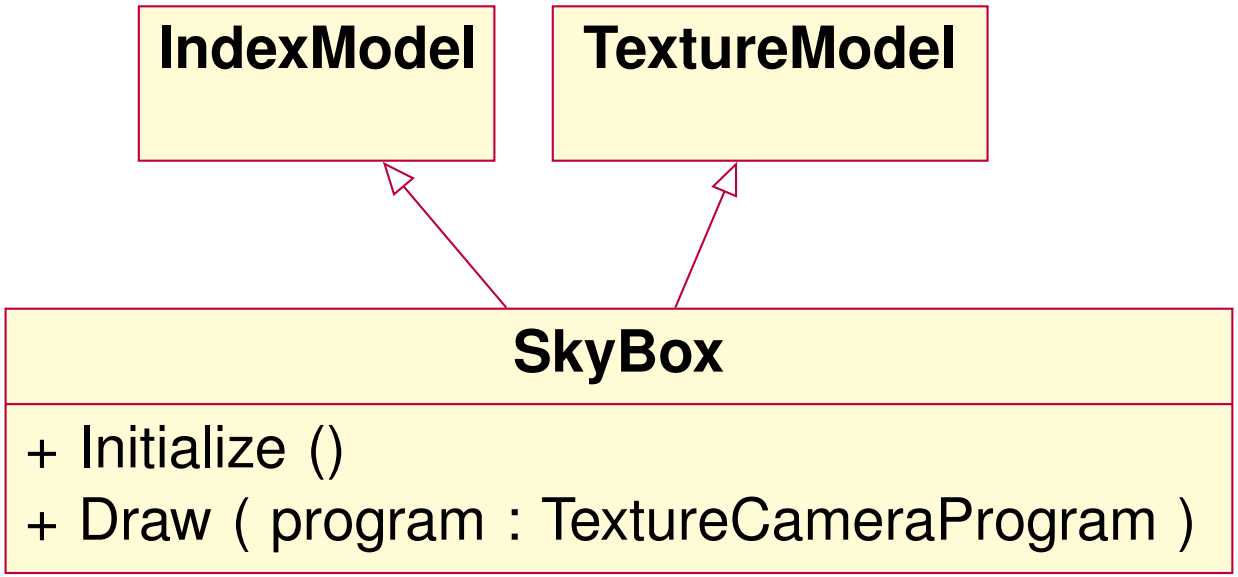
Fragment Shader


```
#version 430 core

in vec3 tex_coord;

out vec4 out_Color;

uniform  samplerCube texture_unit;

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

Program

- Klasa `TextureCameraProgram`

Model



Struktura dla wierzchołków


```
typedef struct SkyTextureVertex{
    float position[4];
    float texture[3];
} SkyTextureVertex;
```

Wierzchołki


```
const SkyTextureVertex kVertices[8] = {
//front
{{-50.0f, 50.0f, 50.0f, 1.0f}, {-1.0f, 1.0f, 1.0f
{{ 50.0f, 50.0f, 50.0f, 1.0f}, {1.0f, 1.0f, 1.0f}
{{ 50.0f,-50.0f, 50.0f, 1.0f}, {1.0f, -1.0f, 1.0f
{{-50.0f,-50.0f, 50.0f, 1.0f}, {-1.0f, -1.0f, 1.0
// back
{{ 50.0f, 50.0f,-50.0f, 1.0f}, {1.0f, 1.0f, -1.0f
{{-50.0f, 50.0f,-50.0f, 1.0f}, {-1.0f, 1.0f, -1.0
{{-50.0f,-50.0f,-50.0f, 1.0f}, {-1.0f, -1.0f, -1.
{{ 50.0f,-50.0f,-50.0f, 1.0f}, {1.0f, -1.0f, -1.0
};
```

Indeksy


```
const GLuint kIndices[36] = {
0, 1, 3,  1, 2, 3, // front
4, 5, 7,  5, 6, 7, // back
5, 4,0,  4,1,0, // top
7,6,2, 6,3,2, // bottom
5,0,6, 0,3,6, // left
1,4,2, 4,7,2  // right
};
```

VAO i bufory


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

glGenBuffers(1, &vertex_buffer_);
glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer_);
glBufferData(GL_ARRAY_BUFFER, sizeof(kVertices),
   kVertices, GL_STATIC_DRAW);

glGenBuffers(1, &index_buffer_);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,
             index_buffer_);
glBufferData(GL_ELEMENT_ARRAY_BUFFER,
   sizeof(kIndices), kIndices, GL_STATIC_DRAW);
```

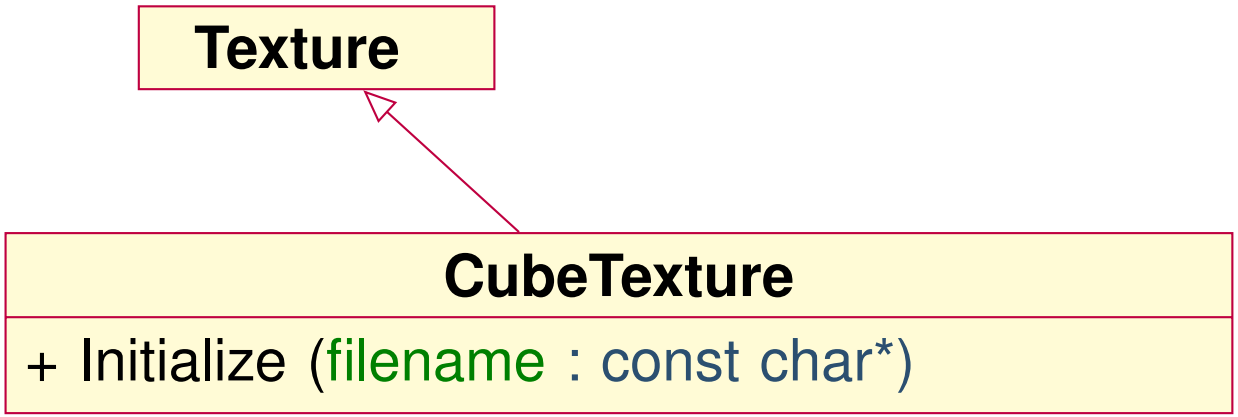
Argumenty shadera wierzchołków


```
glEnableVertexAttribArray(0);
glEnableVertexAttribArray(2);

glVertexAttribPointer(0, 4, GL_FLOAT,
   GL_FALSE, sizeof(kVertices[0]), (GLvoid*)0);
glVertexAttribPointer(2, 3, GL_FLOAT,
   GL_FALSE, sizeof(kVertices[0]),
   (GLvoid*)sizeof(kVertices[0].position));

glBindVertexArray(0);
```

Wyświetlenie


```
glUseProgram(pr);
glBindVertexArray(vao_);
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);
glFrontFace(GL_CCW);
glActiveTexture(texture_unit_);
glBindTexture(GL_TEXTURE_CUBE_MAP, texture_);

glDepthMask(0);
glDrawElements(GL_TRIANGLES, 36,
      GL_UNSIGNED_INT, (GLvoid*)0);
glDepthMask(1);

glDisable(GL_CULL_FACE);
glBindTexture(GL_TEXTURE_CUBE_MAP, 0);
glBindVertexArray(0);
glUseProgram(0);
```

Tektrura sześcienna



- Zakładamy, że teksura zapisana jest w sześciu plikach o nazwach `filenamen.tga`, gdzie

$$n = \begin{cases} 1 & \text{dla prawej ściany} \\ 2 & \text{dla tylnej ściany} \\ 3 & \text{dla górnej ściany} \\ 4 & \text{dla lewej ściany} \\ 5 & \text{dla przedniej ściany} \\ 6 & \text{dla dolnej ściany} \end{cases}$$

Inicjalizacja tekstury


```
glGenTextures(1, &texture_);
//    glActiveTexture(GL_TEXTURE1);
// aktywizacja
glBindTexture(GL_TEXTURE_CUBE_MAP, texture_);
// parametry interpolacji tekstury
glTexParameteri(GL_TEXTURE_CUBE_MAP,
   GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP,
   GL_TEXTURE_MAG_FILTER, GL_LINEAR);
// parametry
glTexParameteri(GL_TEXTURE_CUBE_MAP,
   GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP,
   GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP,
   GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
```

Przygotowywanie nazwy pliku


```
char * filename1;

int len=strlen(filename);

filename1 = new char[len+6];

strcpy(filename1,filename);
char n='1';
filename1[len+1]='.';
filename1[len+2]='t';
filename1[len+3]='g';
filename1[len+4]='a';
filename1[len+5]='\0';
```

Ładowanie obrazów


```
GLenum target;

for(char i=0;i<6;i++){
 filename1[len]=n++;
   switch(i){
      case 0:
   //                right;
       target = GL_TEXTURE_CUBE_MAP_POSITIVE_X;
      break;
   ..................................
      case 5:
   //                bottom;
       target = GL_TEXTURE_CUBE_MAP_NEGATIVE_Y;
      break;
 }
LoadTGAFileOrDie(target, filename1);
}
```

Renderowanie

- Niebo — jako pierwsze:

```
sky_.Draw(sky_program_);

tori_.Draw(point_program_);
plane_.Draw(point_program_);
```

Inicjalizacja tekstury nieba


```
void Window::InitTextures(){
.....................................
    sky_.Initialize();
    sky_.SetTexture(sky_texture_);
    sky_texture_.Initialize(kSkyTextureFile);
}
```

Inicjalizacja nieba


```
void Window::InitModels(){
.....................................
    sky_.Initialize();
    sky_.SetTexture(sky_texture_);
    sky_.SetTextureUnit(GL_TEXTURE1);
}
```

Inicjalizacja programu sześciwnnego


```
void Window::InitPrograms(){
.....................................
    sky_program_.Initialize(kSkyBoxVertexShader,
      kSkyBoxFragmentShader);
    glUseProgram(sky_program_);
    sky_program_.SetTextureUnit(1);
    sky_program_.SetProjectionMatrix(
      projection_matrix_);
    sky_program_.SetViewMatrix(view_matrix_);
}
```

- Odpowiednio uzupełnione funkcje
  `void Window::SetViewMatrix()` oraz
  `void Window::SetProjectionMatrix()`