Laborator 10

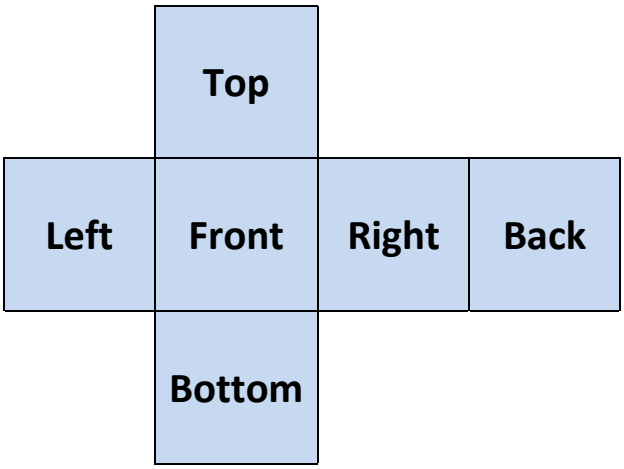
1 Obiective

Acest laborator vă prezintă noțiunile de bază despre conceptul de cubemaps (care sunt folosite pentru definirea skybox-urilor) și proprietățile de reflecție ale obiectelor 3D.

2 Fundament teoretic

2.1 Cubemap

În multe situații, dorim să afișăm o imagine "de fundal" reprezentând partea din scena 3D pe care nu am modelat-o. Putem realiza acest lucru prin includerea scenei noastre 3D în interiorul unui cub cu texturi diferite aplicate pe fiecare față. Acest cub se numeste skybox și este compus din 6 texturi urmând acest model:



Reprezentarea OpenGL a acestui tip de textură este un **cubemap**, care conține 6 texturi individuale 2D. Prin combinarea acestor texturi și maparea acestora pe un cub unitar, putem foarte ușor să eșantionăm o valoare a texturii utilizând un vector de direcție. Acest vector poate fi preluat din poziția interpolată a vârfului cubului.

Pentru a crea un cubemap, trebuie să generăm un id de textură și să îl legăm la ținta GL_TEXTURE_CUBE_MAP.

```
GLuint textureID;
glGenTextures(1, &textureID);
glBindTexture(GL_TEXTURE_CUBE_MAP, textureID);
```

Pentru fiecare față a cubemap-ului, trebuie să apelam funcția glTexImage2D și să setăm un parametru țintă specific pentru fața curentă:

Target	Orientation
GL_TEXTURE_CUBE_MAP_POSITIVE_X	Right
GL_TEXTURE_CUBE_MAP_NEGATIVE_X	Left
GL_TEXTURE_CUBE_MAP_POSITIVE_Y	Top
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y	Bottom
GL_TEXTURE_CUBE_MAP_POSITIVE_Z	Back
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z	Front

Putem incrementa liniar ținta pentru a seta toate texturile cubului:

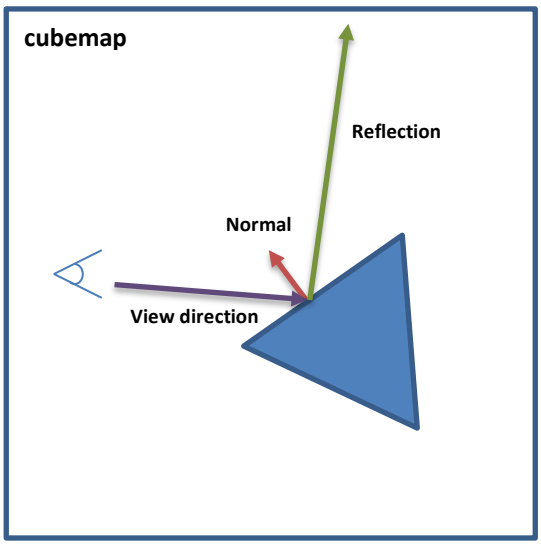
```
for(GLuint i = 0; i < skyBoxFaces.size(); i++)
{
	image = stbi_load(skyBoxFaces[i], &width, &height, &n, force_channels);
	if (!image) {
		fprintf(stderr, "ERROR: could not load %s\n", skyBoxFaces[i]);
		return false;
	}
	glTexImage2D(
		GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0,
		GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
}
```

În **fragment shader**, folosim un "sampler" diferit în acest caz (**samplerCube**), dar folosim aceeași funcție ("texture") pentru a eșantiona valoarea de culoare. Aici, variabila textureCoordinates reprezintă vectorul de direcție.

```
#version 410 core

in vec3 textureCoordinates;
out vec4 color;

uniform samplerCube skybox;

void main()
{
    color = texture(skybox, textureCoordinates);
}
```

2.2 Reflecții

Având acum un mediu înconjurător (reprezentat de cubemap), putem specifica proprietățile reflexive ale diferitelor obiecte. Reflecția se bazează pe unghiul privitorului. Acest vector de reflecție poate fi calculat utilizând funcția GLSL numită "**reflect**". Trebuie să specificăm vectorul normală și vectorul de direcție a camerei. Vectorul rezultat este folosit pentru a eșantiona din textura cubemap.




```
vec3 viewDirectionN = normalize(viewDirection);
vec3 normalN = normalize(normal);
vec3 reflection = reflect(viewDirectionN, normalN);
vec3 colorFromSkybox = vec3(texture(skybox, reflection));
```

3 Tutorial

Începeți prin a descărca resursele și a le include în proiectul dvs. Clasa SkyBox va fi utilizată pentru încărcarea și desenarea cubemap-urilor.

Adăugați într-un vector texturile (mai exact, calea lor) care vor fi mapate pe cubemap. Creați o funcție void numită **initSkybox()** pentru următorul cod și apelați funcția după ce inițializați obiectele din scenă.

```
std::vector<const GLchar*> faces;
faces.push_back("skybox/right.tga");
faces.push_back("skybox/left.tga");
faces.push_back("skybox/top.tga");
faces.push_back("skybox/bottom.tga");
faces.push_back("skybox/back.tga");
faces.push_back("skybox/front.tga");

mySkyBox.Load(faces);
```

Definiți următoarele variabile globale care vor reprezenta skybox-ul nostru și shader-ul folosit pentru desenarea acestuia.

```
gps::SkyBox mySkyBox;
gps::Shader skyboxShader;
```

În funcția **initShaders()**, încărcați shader-ul skybox-ului.

```
skyboxShader.loadShader("shaders/skyboxShader.vert", "shaders/skyboxShader.frag");
skyboxShader.useShaderProgram();
```

În funcția **renderScene()** apelați funcția de desenare a skybox-ului după ce ați apelat toate celelalte funcții pentru desenarea restului obiectelor. Dacă proiectul conține umbre generate folosind tehnica shadow mapping, aveți grijă să nu desenați skybox-ul în trecerea de trasare pentru crearea hărții de adâncimi, ci în trecerea de trasare normală a scenei.

```
mySkyBox.Draw(skyboxShader, view, projection);
```

4 Temă

- Urmați tutorialul pentru a afișa un skybox.
- Încărcați un alt obiect 3D în scenă și calculați culoarea obiectului, luând în considerare și reflecțiile provenite de la skybox.