

Základy počítačové grafiky

Přednáška 8

Martin Němec

VŠB-TU Ostrava

2024

Aktuální stav

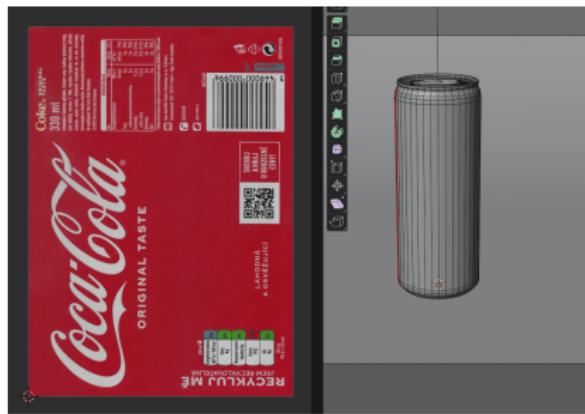
- Vytváříme scény doplněné o světla;
- Umíme se pohybovat, rozhlížet a transformovat objekty;
- Používáme více shaderů a materiálů;

Je něco, co byste si přáli zopakovat?



Textury

Textury umožňují to, aby povrch objektu vypadaly realističtěji bez nutnosti zvýšení počtu polygonů. Jsou zásadní pro vzhled a výkon aplikací.



Textury

Textury umožňují přidávat detailly tělesům, ovlivňují nejen základní barvu ale i další (nejen) optické vlastnosti.

Dělení podle dimenze:

- Jednodimenzionální (1D);
- Dvojdimenzionální (2D);
- Třídimenzionální (3D) – (volume textures).

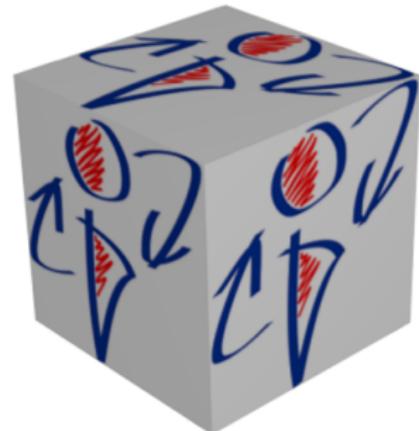
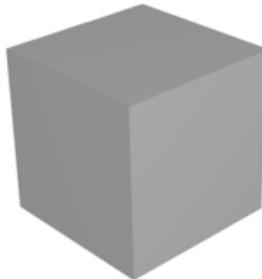
Základní jednotky:

- bitmapa – pixel (picture element);
- objemové data – voxel (volumetric element);
- textury - texel (texture element).

Textury

Textury dělíme podle toho, co ovlivňují:

- difúzní textura (diffuse texture);
- odlesková (zrcadlová) textura (specular texture);
- normálová textura (normal texture);
- textura lomu (distortion / refraction);
- ambientní texturu (ambient texture);
- PBR materiály (albedo, roughness, metallic atd.).



Získávání textur

Textury lze získat různými způsoby, od modelování, přes snímání reálných modelu, až po generování pomocí AI. Textury lze dělit podle různých vlastnosti:

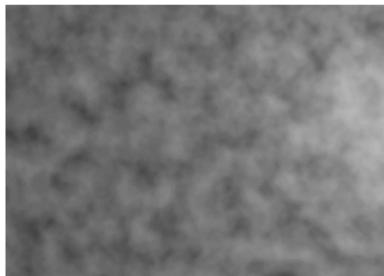
- Rozlišení (1024x1024, 2048x2048, 4096x4096 apod.).
- Vlastnosti (Opakování (Tiling), bez hran (Seamless), atd.)
- Formát a komprese (PNG, JPEG, DDS, EXR).
- Průhlednost (Alpha channel).



Procedurální textury

Načítat obrázek (rastrovou texturu) nemusí být vždy nevhodnější a jediný způsob. Procedurální textury jsou matematicky generovány přímo v aplikaci, podle potřeb.

- Můžeme přizpůsobit rozlišení.
- Použití šumových funkcí (normálová textura).
- Lze využít fraktální geometrii (nejen pro generování textur).
- Použití pro mraky, terén, stromy apod.



Třídimenzionální textury

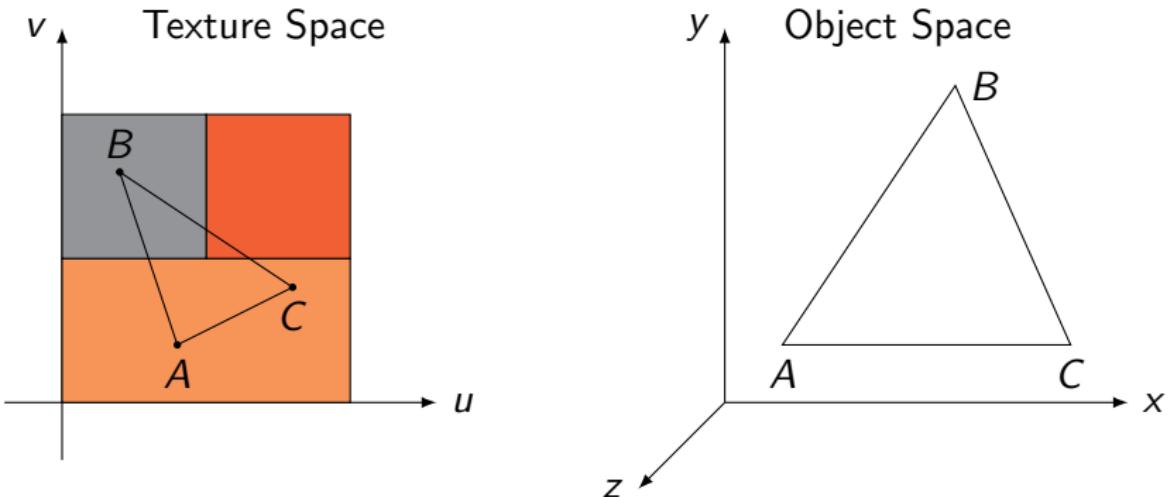
Zobrazování objemových dat v medicíně (CT, MR), GIS.

Základní element voxel. Mřížka s 1024 voxely, zabírá:
 $1024 \times 1024 \times 1024 \times 4\text{byte}$, což je asi 4,29 GB paměti.



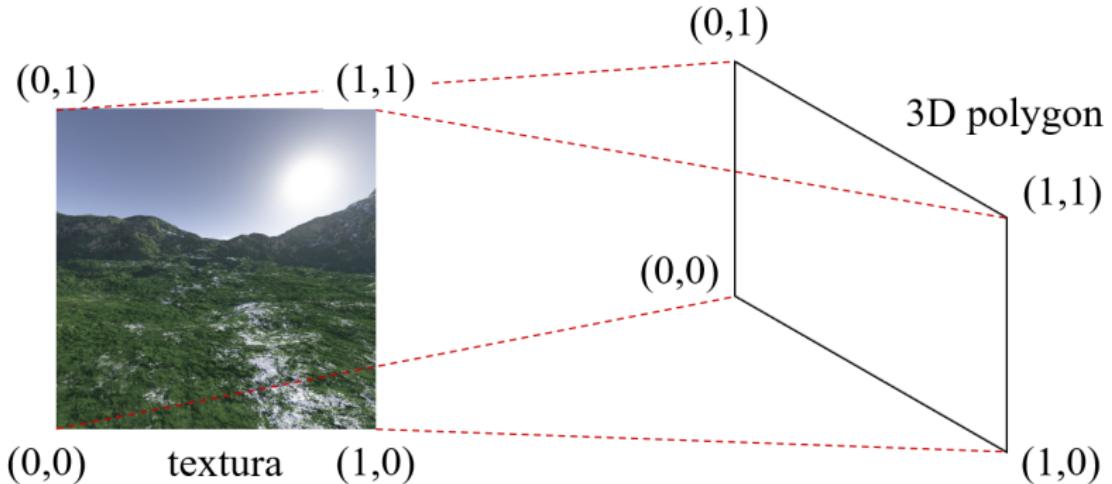
Magnetická rezonance (supravodivé magnety, teplota pod 10kelv., chlazení tekuté hélium).

Textury – mapování



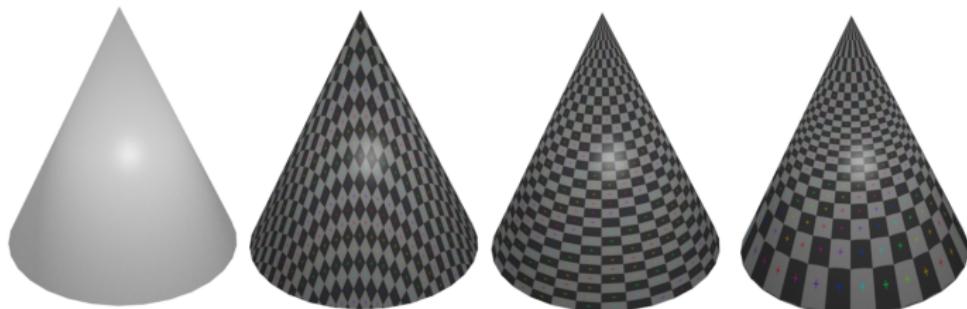
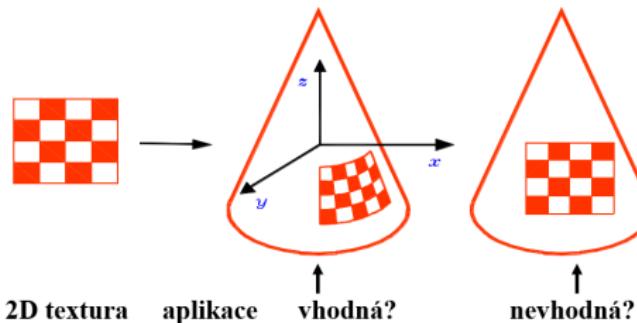
Texturovací souřadnice - 2D souřadnice (u, v) , které odpovídají pozici bodu v textuře, obraz má obvykle rozsah $u, v \in \langle 0, 1 \rangle$

Textury – mapování



Texturovací souřadnice - 2D souřadnice (u,v) , které odpovídají pozici bodu v textuře, obraz má obvykle rozsah $u, v \in \langle 0, 1 \rangle$

Mapování a problémy



Typy mapování

- **Přímé mapování** – bereme data z textury a přiřazujeme 3D bod tělesa. (texture space - screen space).
- **Inverzní mapování** – bereme fragmenty z obrazovky a přiřazujeme bod na textuře (screen space - texture space).

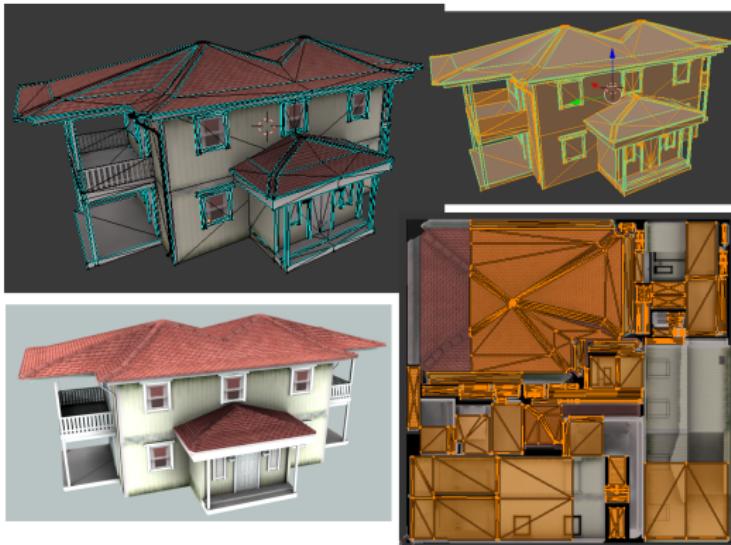
Nejčastěji se v praxi využívá inverzní mapování, kdy nanášení textury je dáno funkcí $M(x, y, z)$, která přiřazuje každému bodu povrchu tělesa bod z textury.

Textury

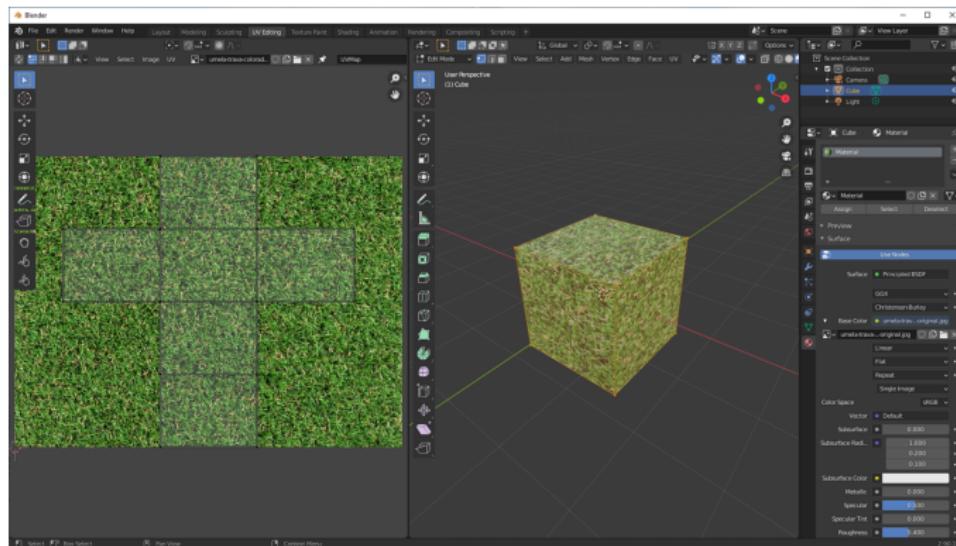
```
const float plain[] = {
    1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f,
    1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f,
    -1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f,
    -1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f,
    1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f,
    -1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f
};

const float plain[] = {
    1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.0f,
    1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f,
    -1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 1.0f,
    -1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.0f,
    -1.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f, 1.0f, 1.0f
};
```

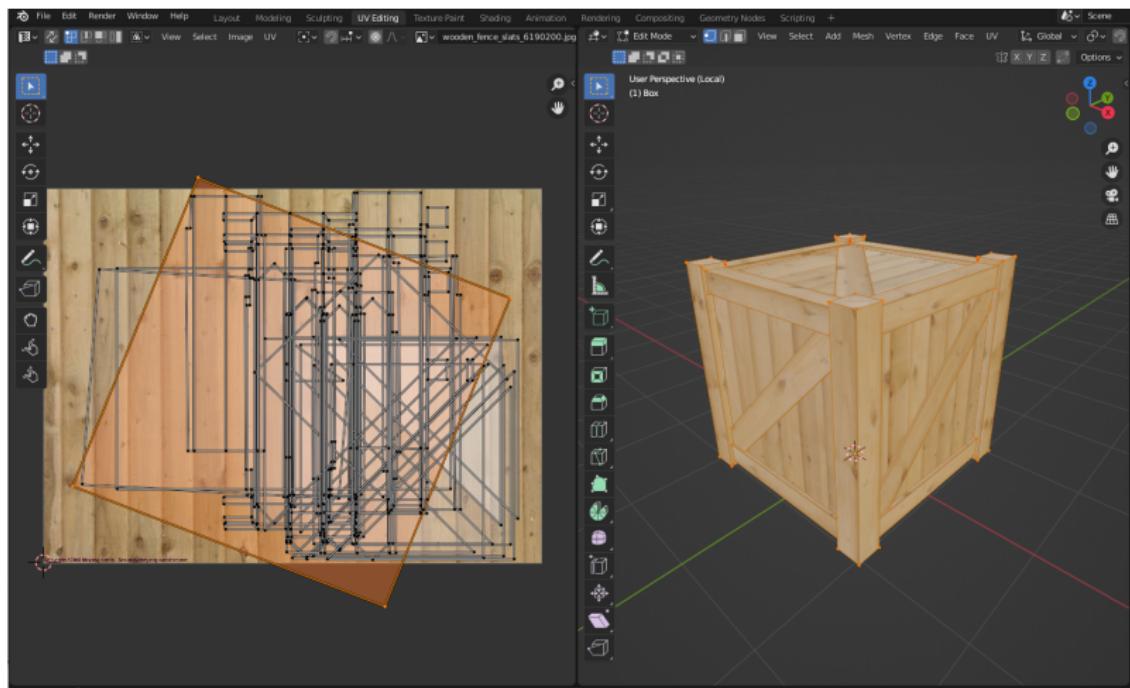
Textury – mapování



Blender ukázka mapování



Blender



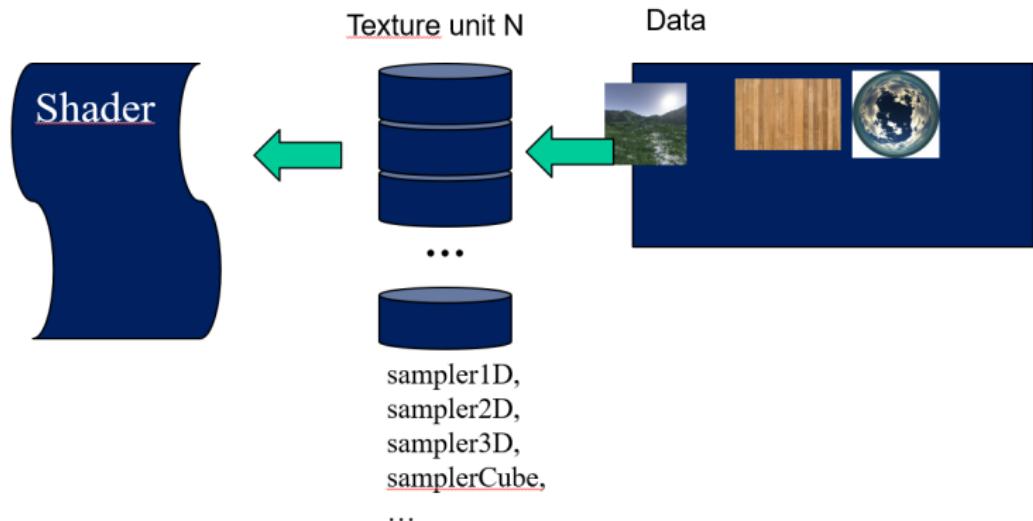
Render



Texturovací jednotka

Texturovací jednotka – určuje barvu podle souřadnic (u,v,s,t). Dva základní přístupy použití:

- můžeme propojovat textury s jednou texturovací jednotkou;
- každá textura je spojena s jednou jednotkou, nastavujeme konkrétní texturovací jednotku.



Texture OpenCV

```
cv::Mat image;
image = cv::imread("../wooden_planks.jpg", 1 ); //load the image (RGB format)
//cv::imshow("im1",image);
cv::flip(image, image, 0); //correct orientation

glGenTextures(1, &texture_id); //reference to texture
glActiveTexture(GL_TEXTURE0); //active texture unit 0 
glBindTexture(GL_TEXTURE_2D,texture_id); //object texture_id is 2D texture

glTexImage2D(GL_TEXTURE_2D,           // Type of texture 1D, 2D, 3D
             0,                  // Pyramid level (for mip-mapping) - 0 is the top level
             GL_RGB,              // Internal colour format to convert to
             image.cols,          // Image width i.e. 640
             image.rows,          // Image height i.e. 480
             0,                  // Border width in pixels (can either be 1 or 0)
             GL_BGR,              // Input format (GL_RGB, GL_RGBA, GL_BGR etc.)
             GL_UNSIGNED_BYTE,    // Image data type
             image.ptr());        // The actual image data itself

glGenerateMipmap(GL_TEXTURE_2D); //Generate mipsmaps now!!!
image.release(); //release image file  
```

fragment shader

```
#version 450
in vec2 texCoord;
out vec4 gl_FragColor;

uniform sampler2D textura; //1D,2D,3D,samplerCube , atd.

void main()
{
    gl_FragColor = texture(textura, texCoord);
}
```

Souřadnicový systém textury musí být otočen horizontálně aby byl správně interpretován v OpenGL.



```
cv::flip(image, image, 0); //OpenCV
```

Načtení souboru SOIL

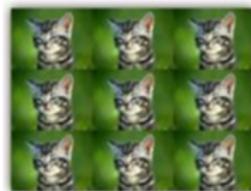
Existuje celá řada knihoven, které umožňují načíst soubor s obrázkem do paměti počítače. Můžete použít např.: OpenCV, SOIL, LibKTX, Libjpg, LodePNG atd.

```
//Bind the first texture to the first texture unit.  
//glActiveTexture specifies active texture unit.  
glActiveTexture(GL_TEXTURE0); //texture unit 0  
  
//Load texture and generate textureID  
GLuint textureID = SOIL_load_OGL_texture("D:\\test.png",  
    SOIL_LOAD_RGBA, SOIL_CREATE_NEW_ID, SOIL_FLAG_INVERT_Y);  
glBindTexture(GL_TEXTURE_2D, textureID);  
  
//Set texture unit to fragment shader  
GLint uniformID = glGetUniformLocation(  
    shaderProgram, "textureUnitID");  
glUniform1i(uniformID, 0); //texture unit 0
```

Nastavení textur

Souřadnice mohou být i mimo rozsah textury (mimo $<0, 1>$).

```
// opakovani textury
glTexParameteri(
    GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(
    GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```



GL_REPEAT



GL_MIRRORED_REPEAT



GL_CLAMP_TO_EDGE

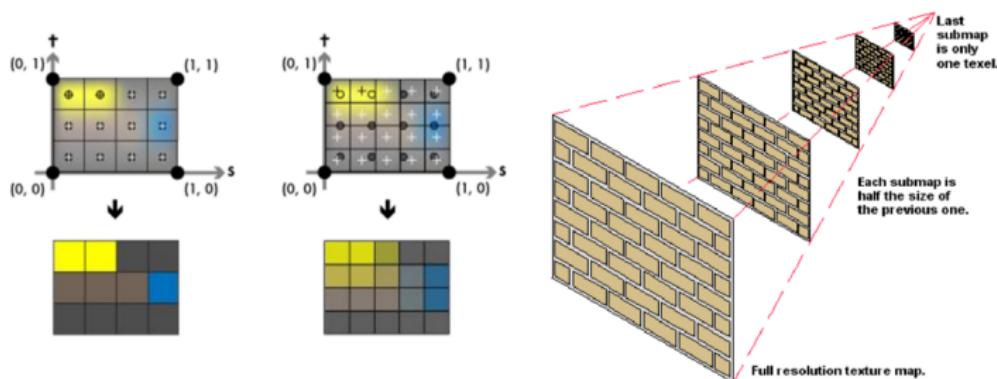


GL_CLAMP_TO_BORDER

Mipmapy

Modely mohou mít ve scéně různé velikosti (podle vzdálenosti od kamery), bude potřebná přizpůsobit velikost textury (textura má obvykle pevně danou velikost).

```
// opakovani textury
glTexParameteri(
    GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(
    GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

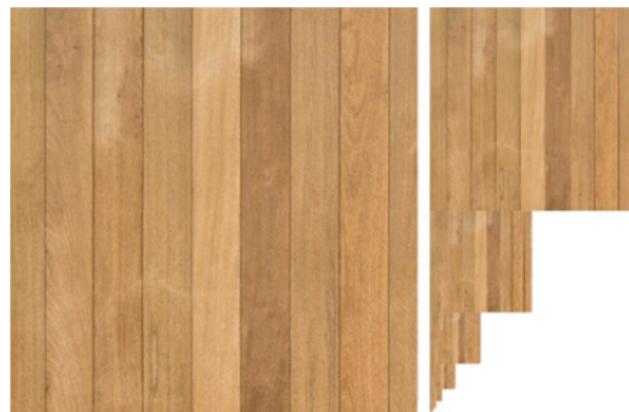


Generování mipmap

MIP lat. "multum in parvo" (mnoho v málu)

```
glGenerateMipmap(GL_TEXTURE_2D);
```

256x256 → 128x128 → 64x64 → 32x32 atd.



Vlastní mipmapy

Level **n** is the **n**th mipmap reduction image.

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 64, 64, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[0]);  
glTexImage2D(GL_TEXTURE_2D, 1, GL_RGBA, 32, 32, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[1]);  
glTexImage2D(GL_TEXTURE_2D, 2, GL_RGBA, 16, 16, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[2]);  
glTexImage2D(GL_TEXTURE_2D, 3, GL_RGBA, 8, 8, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[3]);  
glTexImage2D(GL_TEXTURE_2D, 4, GL_RGBA, 4, 4, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[4]);  
glTexImage2D(GL_TEXTURE_2D, 5, GL_RGBA, 2, 2, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[5]);  
glTexImage2D(GL_TEXTURE_2D, 6, GL_RGBA, 1, 1, 0,  
             GL_RGBA, GL_UNSIGNED_BYTE, MIPLevels[6]);
```

Volba filtrov (lineární, nejbližší)

MIN, MAG – bližší a vzdálenější textura v mipmapě.



GL_NEAREST



GL_LINEAR

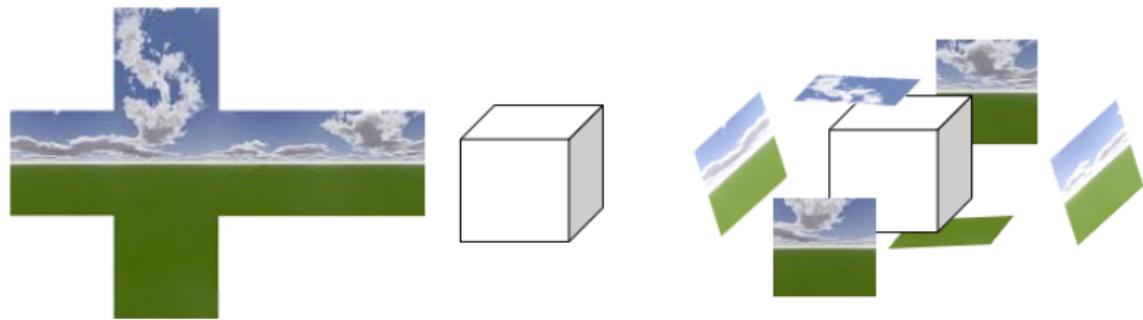
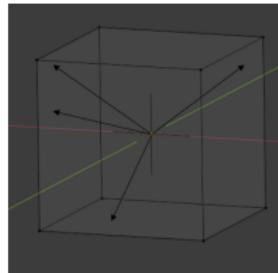
Volba filtru

Další možnosti volby filtru:

- GL_NEAREST_MIPMAP_NEAREST - nejbližší pixel z menší nebo větší textury.
- GL_LINEAR_MIPMAP_NEAREST - bilineární interpolaci nejbližších texelů v bližší textuře. Při změně velikosti může nastat viditelný přechod (přechod na jinou bližší texturu).
- GL_NEAREST_MIPMAP_LINEAR – vyberou se dva nejbližší texely z větší a menší textury mezi kterými se provede lineární interpolace.
- GL_LINEAR_MIPMAP_LINEAR provádí se bilineární interpolace na každé textuře zvlášť a výsledky se opět interpolují (trilineární interpolace). Nejnáročnější na výpočet, ale nejlepší výsledky.

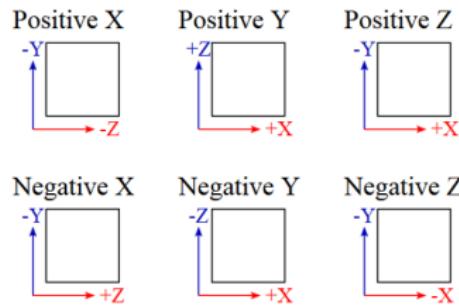
Cube map

3D mapování pomocí krychle (šesti základních textur).



Cube mapa

```
glActiveTexture(i);
glBindTexture(GL_TEXTURE_CUBE_MAP, texture_id);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,
    0, GL_RGBA, ..., img.ptr());
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y,
    0, GL_RGBA, ..., img.ptr());
...
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z,
    0, GL_RGBA, ..., img.ptr());
```



fragment shader

Texturovací souřadnice (vec3) budou mít 3 složky (u,v,w).

```
#version 330
in vec3 TexCoord;
out vec4 gl_FragColor;

uniform samplerCube Texture; //samplerCube

void main()
{
    gl_FragColor = texture(Texture, TexCoord);
}
```

Nebe ve scéně

Základní techniky z pohledu vykreslovaného modelu:

- SkyBox - krychle, která má obraz oblohy promítnutý na šest ploch.
- SkyDome - kopule (kulová, polokulová) umístěná ve scéně, na které je namapovaná obloha.



Ukázka nebe a pozadí ze hry Half-Life.

Dva základní přístupy ve vykreslení:

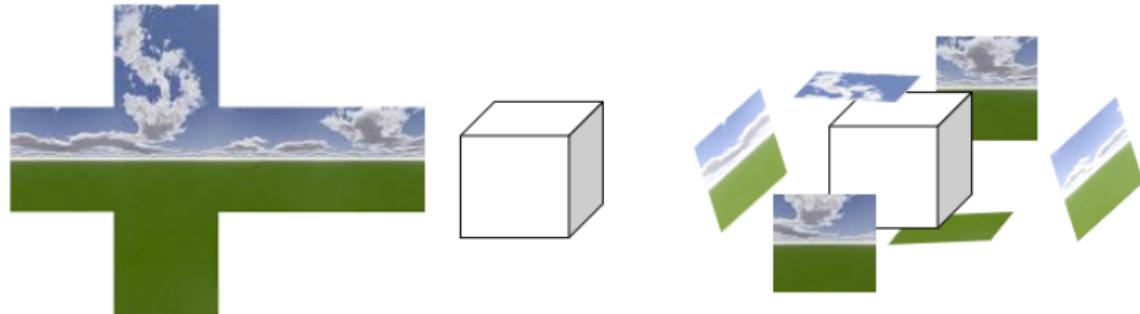
- těleso přes celou scénu - problém s ořezáváním (perspektiva a rovina z-far).
- "malé" těleso na pozici kamery - těleso (např. skyBox) mění pozici podle kamery (zakážeme zápis do z-bufferu nebo ho pak vymažeme).

```
while (!glfwWindowShouldClose(window)) {  
    // clear color and depth buffer  
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  
    scene->drawSkyBox();  
    // clear depth buffer  
    glClear(GL_DEPTH_BUFFER_BIT);  
    scene->draw();  
    ...  
}
```

SkyBox

- Rychlé, jednoduché pro implementování, obtížnější získat správně navazující texturu.
- Obvykle je základem krychle (Cube mapping).

GL_TEXTURE_CUBE_MAP_POSITIVE_X
GL_TEXTURE_CUBE_MAP_NEGATIVE_X
GL_TEXTURE_CUBE_MAP_POSITIVE_Y
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y
GL_TEXTURE_CUBE_MAP_POSITIVE_Z
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z



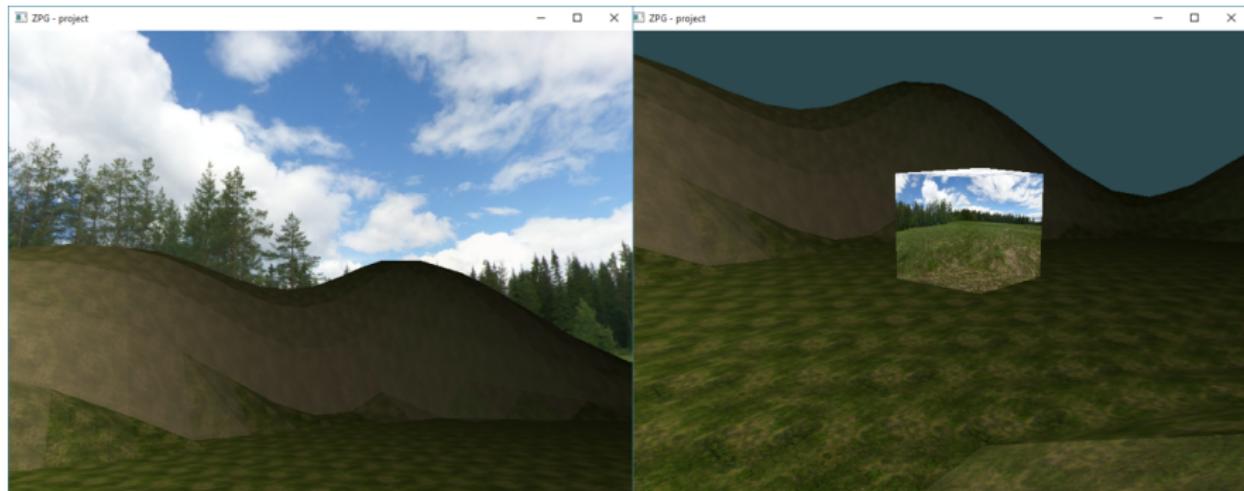
SkyBox

Skybox (CubeMap) pro u,v,s se používá přímo pozice v lokálním souřadném systému.

```
// krychle pro skybox
const float skycube[108] = {
    -1.0f, -1.0f, -1.0f,
    -1.0f, -1.0f, 1.0f,
    -1.0f, 1.0f, 1.0f,
    1.0f, 1.0f, -1.0f,
    -1.0f, -1.0f, -1.0f,
    -1.0f, 1.0f, -1.0f,
    1.0f, -1.0f, 1.0f,
    -1.0f, -1.0f, -1.0f,
    ...
    -1.0f, 1.0f, 1.0f,
    1.0f, -1.0f, 1.0f
};
```

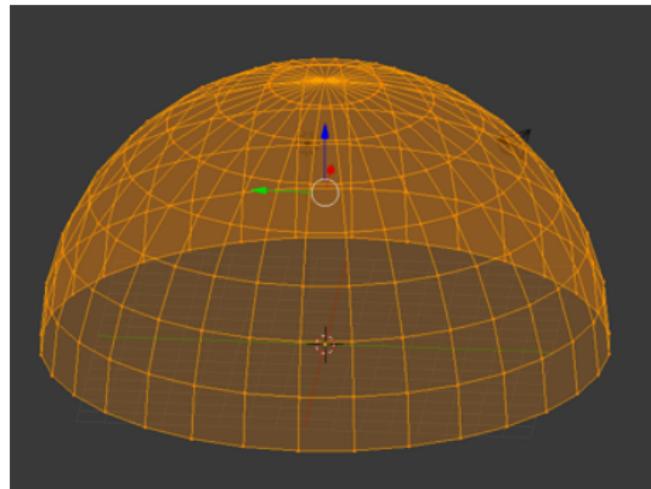
SkyBox

Velikost – kombinace "depth map", srovnání se zapnutým a vypnutým posunem SkyBoxu.



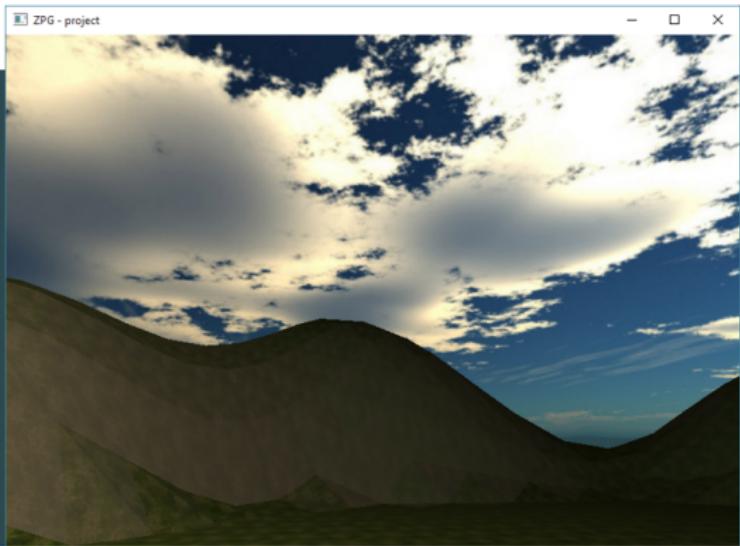
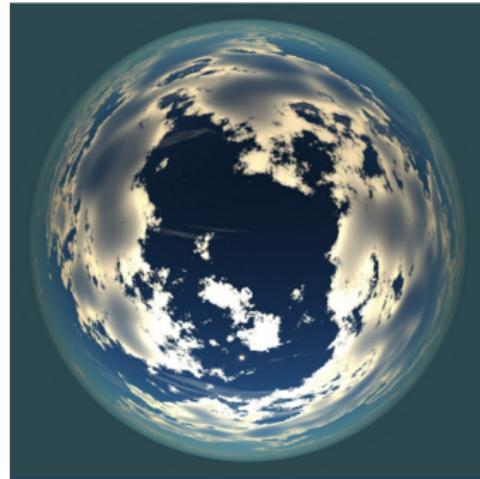
SkyDome

Jednodušeji získatelná textura, vykreslujeme více vrcholů



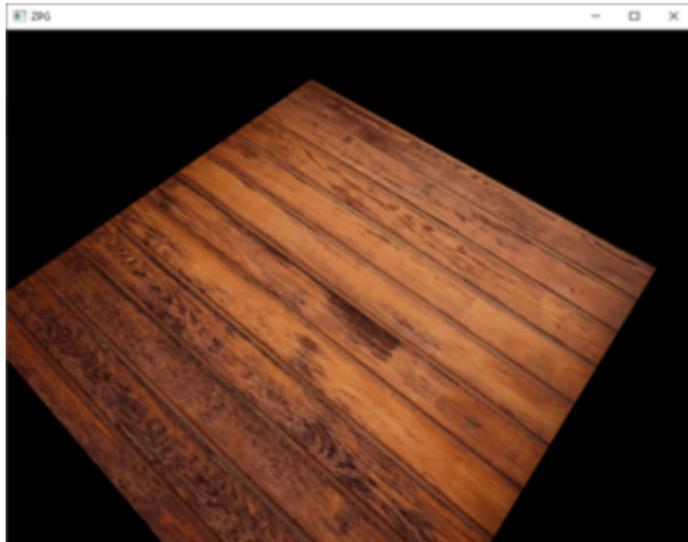
SkyDome

Přirozenější pohyb, jakou velikost nastavit ?



Cvičení

- Načtěte texturu (např. knohovna SOIL) a následně ji namapujte na těleso.
- Vyzkoušejte si obě varianty 2D i CubeMap.
- Vytvořte SkyBox (SkyDome příště, až budeme umět načítat modely).



Dotazy?

Ukázka

