

LMS Moodle VŠB-TUO: 460-2021/03 Základy počítačové grafiky (2024/2025 ZS): Sekce: Cvičení 10

1. Pokuste se dodělat identifikaci modelů a detekci pozice v globálním souřadném systému (kam jste klikli, pomocí funkce `unProject()`).
Můžete vytvořit funkci programu tak, že kam klikneme do scény, tak tam naroste strom.



2. Nejprve musíte při vykreslování přidat identifikátor do stencil bufferu

```
//přidání ID do stencil bufferu
glEnable(GL_STENCIL_TEST);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
for (všechny kreslené objekty){
    glStencilFunc(GL_ALWAYS, objekt->getID(), 0xFF);
    drawArray(...);
}
```

3. Následně po kliknutí načtete data z frame bufferu a zjistíme těleso na aktuální pozici.

```
//načtení ID a pozice ve světových souřadnicích

GLbyte color[4];
GLfloat depth;
GLuint index;

GLint x = (GLint)cursor.x;
GLint y = (GLint)cursor.y;

int newy = camera->getResolution().y - y;

glReadPixels(x, newy, 1, 1, GL_RGBA, GL_UNSIGNED_BYTE, color);
glReadPixels(x, newy, 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT, &depth);
glReadPixels(x, newy, 1, 1, GL_STENCIL_INDEX, GL_UNSIGNED_INT, &index);

printf("Clicked on pixel %d, %d, color %02hhx%02hhx%02hhx%02hhx, depth %f, stencil index %u\n", x, y, color[0], color[1], color[2], color[3], depth, index);

//Můžeme nastavit vybrané těleso scena->setSelect(index-1);

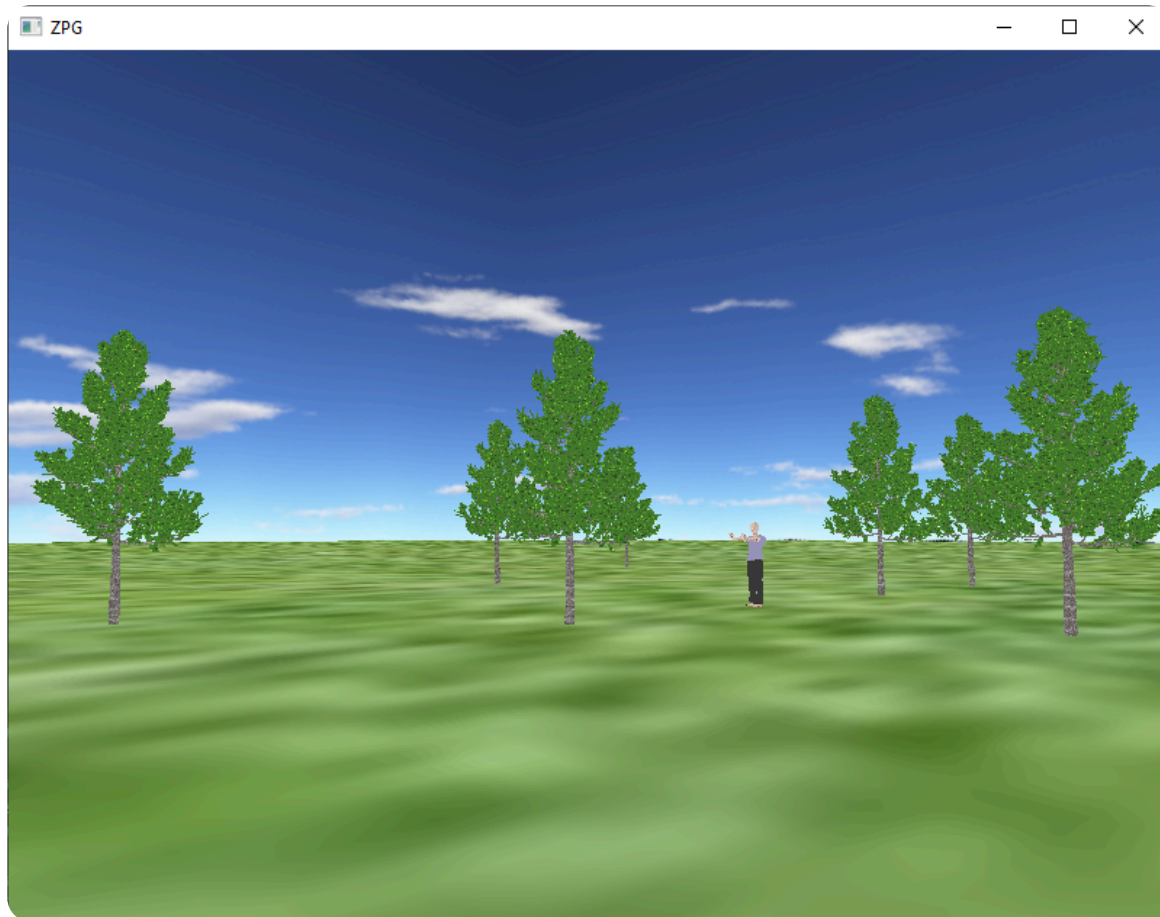
//Můžeme vypočítat pozici v globálním souřadném systému.
```

```
glm::vec3 screenX = glm::vec3(x, newy, depth);
glm::mat4 view;
glm::mat4 projection;
glm::vec4 viewport = glm::vec4(0, 0, getResolution().x, getResolution().y);
glm::vec3 pos = glm::unProject(screenX, view, projection, viewport);
```

```
printf("unProject [%f,%f,%f]\n", pos.x, pos.y, pos.z);
```

4. Funkce knihovny GLM [unProject\(\)](#) umožňuje pohodlně převést bod zpět do lokálního nebo globálního souřadného systému.

Vypočteme inverzní matice a provedeme inverzní transformace (souřadnice bodu získaného z obrazovky a zetová souřadnice ze z-bufferu) nám to umožňují.



5. Využijte funkci unProject u pohybu vybraného modelu tak, že zadáte řídicí body Bézierovy křivky (kubiky), v případě více bodů, budeme pohyb provádět po obloucích (jednotlivých kubikách).

```
glm::mat4 A = glm::mat4(glm::vec4(-1.0, 3.0, -3.0, 1.0),
                          glm::vec4(3.0, -6.0, 3.0, 0),
                          glm::vec4(-3.0, 3.0, 0, 0),
                          glm::vec4(1, 0, 0, 0));

std::cout << glm::to_string(A) << std::endl;

glm::mat4x3 B = glm::mat4x3( glm::vec3(0, 0, 0),
                              glm::vec3(2, 2, 0),
                              glm::vec3(4, 4, 0),
                              glm::vec3(6, 0, 0));

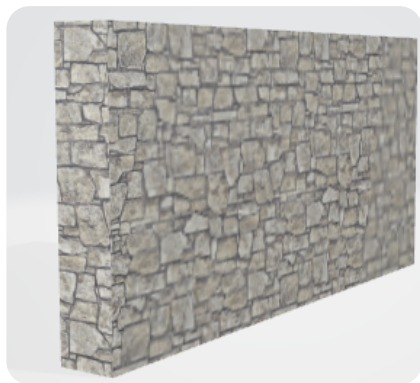
float t = 0.5f;

glm::vec4 parameters = glm::vec4(t * t * t, t * t, t, 1.0f);

glm::vec3 p = parameters * A * glm::transpose(B);
cout << "t = " << t << " P=[ " << p[0] << " , " << p[1] << " , " << p[2] << "]" << endl;
```

V souboru bezier.cpp je jednoduchá ukázka pohybu modelu po Bézierově křivce. Cílem je samozřejmě vhodně objektově tuto část zakomponovat do projektu.


Pro případné rozčlenění scény můžete použít model stěny.




 [TreeObj.zip](#) 

 [teren.zip](#) 

 [simplePicking.zip](#) 

 [bezier.cpp](#) 

 [zed.zip](#) 

Čeština (cs) 

© 2012 - 2025 [VŠB-TUO](#)

[Kontaktovat technickou podporu](#)

Běží na technologii [Moodle Pty Ltd](#)