Nepovinné úkoly na procvičování po 1. cvičení

Úkoly jsou pouze pro procvičení látky, není to písemka, nebudou kontrolovány. Případné dotazy směřujte na diskuzní fórum nebo pošlete e-mail cvičícím.

Poznámka: pokud se na nějakém cvičení nestihne udělat příklad na *aspect*, tak ho tady ignorujte.

Obecné procvičování shaderů

Úkol 1: Posuňte objekty na obrazovce o kousek doleva/doprava/nahoru/dolů, třeba o 0,2.

Poznámky:

- uděláte to tak, že přičtete hodnotu 0,2 k souřadnici *x* nebo *y* při výpočtu výsledné pozice vrcholu, podobně jako při použití *aspect*
- je nutné nejdříve přičíst a potom výsledek vydělit tím *aspect*, ne naopak
- k zamyšlení: co se změní, proč to bude špatně, když nejdřív vydělíte podle *aspect* a až teprve potom přičtete konstantu?

Úkol 2: Zmenšete objekty na polovinu (50% velikosti).

Poznámky:

- místo přičítání konstanty hodnotu násobím 0,5
- vyzkoušejte neuniformní zmenšení, neboli zmenšete v každé ose jinak
- ačkoliv tady to nic nezmění, násobte opět před použitím *aspect*, podobně jako u posouvání. Pouze z principu, dělá se to v tomto pořadí :-)
- zmenšete objekty na 0,5 a posuňte je o 0,2
- všimněte si, co se stane, když nejdříve posunete a potom zvětšíte a naopak
- upravení pomocí *aspect* bude vždy až poslední. Správné pořadí je zmenšit, pak posunout, pak *aspect*
- k zamyšlení: co se vlastně děje, když zvětšujete po posouvání, a v čem a proč se to liší od posouvání po zvětšení?

Úkol 3: Proveďte zrcadlení, vyměňte osy *x* a *y*.

Poznámky:

- pozor, co musíte dělit pomocí *aspect*? (Odpověď: nové *x*, tedy staré *y*)

Úkol 4: Otočte objekty o 1 radián proti směru hodinových ručiček.

Poznámky:

- rotace je složitější, kdo chce, může si odvodit vzorec, kdo nechce, řešení je:

$$x_{out} = x_{in} \cdot \cos(\alpha) - y_{in} \cdot \sin(\alpha)$$
$$y_{out} = x_{in} \cdot \sin(\alpha) + y_{in} \cdot \cos(\alpha)$$

- můžete použít funkce sin a cos, které jsou přístupné v shaderech. Jejich argumentem je úhel v radiánech.
- opět, *aspect* je až poslední.

Úkol 5: Zmenšete, otočte a posuňte objekty.

- jaké je pořadí? (odpověď: zmenšit, otočit, posunout, *aspect*)

Procvičování uniform proměnných

Úkoly 6-10: Realizujte předchozí příklady, ale použijte uniform proměnné místo konstant. To, o kolik se posunete / otočíte / zvětšíte, nastavujte z kódu aplikace (C++/Javy).

Úkol 11: Místo dvou floatů použijte `vec2` pro posunutí a pro zvětšení (pokud to tak už neděláte :-)).

Úkol 12: Posuňte dům a hvězdu jinak, tedy například dům doleva a hvězdu doprava.

Nápověda:

- nastavte posun pro dům, vykreslete dům, pak nastavte nový posun pro hvězdu a vykreslete hvězdu.

Úkol 13: Vykreslete několik hvězd na obloze, každou jinde.

Procvičování animace a uživatelského rozhraní

Úkol 14: Vytvořte animaci, nechte dům otáčet proti směru hodinových ručiček.

Nápověda:

- vytvořte globální proměnnou / atribut třídy, která bude uchovávat aktuální úhel a bude se inkrementovat každý snímek.

Úkol 15: Posuňte dům o kousek vpravo, když uživatel stiskne klávesu X, a o kousek vlevo, když stiskne klávesu Z.

Úkol 16: Nechte dům se pohybovat vpravo, dokud uživatel drží klávesu X, a vlevo, dokud uživatel drží klávesu Z.

Nápověda:

- použijte funkci `glutKeyboardUpFunc` v C++ a metodu `glfwSetKeyCallback` v Javě. Podrobnosti hledejte na Internetu, např. https://github.com/LWJGL/lwjgl3-wiki/wiki/2.6.3-Input-handling-with-GLFW.

Procvičování vykreslovacích příkazů

Úkol 17: Pokud se to nestihlo na cvičení, přidejte další objekty, jako jsou třeba dveře, okna, komín, plot, tráva, cesta apod. Použijte vykreslovací mód `GL_TRIANGLE_STRIP`.

Vzorové řešení vybraných příkladů

Úkol 1: Posuňte objekty na obrazovce o kousek doleva/doprava/nahoru/dolů, třeba o 0,2.

Řešení:

- upravíme výpočet výsledné pozice vrcholu ve vertex shaderu
- kód (vertex.glsl) s vyznačenými upravenými místy:

```
...
void main() {
    ...
    gl_Position = vec4(position.x + 0.2, position.y, 0.0, 1.0);
    ...
}
...
```

Úkol 6: Posuňte objekty na obrazovce o kousek doleva/doprava/nahoru/dolů, třeba o 0,2. Použijte uniform proměnnou.

Řešení:

- do shaderu přidáme novou uniform proměnnou *shift* typu `float`
- upravíme výpočet výsledné pozice vrcholu ve vertex shaderu
- kód (vertex.glsl) s vyznačenými upravami:

```
#version 330
...
uniform float shift;
...
void main() {
    ...
    gl_Position = vec4(position.x + shift, position.y, 0.0, 1.0);
    ...
}
...
```

- z kódu aplikace (C++/Java) nastavíme při vykreslování hodnotu uniform proměnné *shift*
- kód (Cv1.java) s vyznačenými úpravami:

```
...
glUseProgram(program);
...
int shiftLoc = glGetUniformLocation(program, "shift");
glUniform1f(shiftLoc, 0.2f);
...
glBindVertexArray(geometryArray);
glDrawArrays(...);
...
```