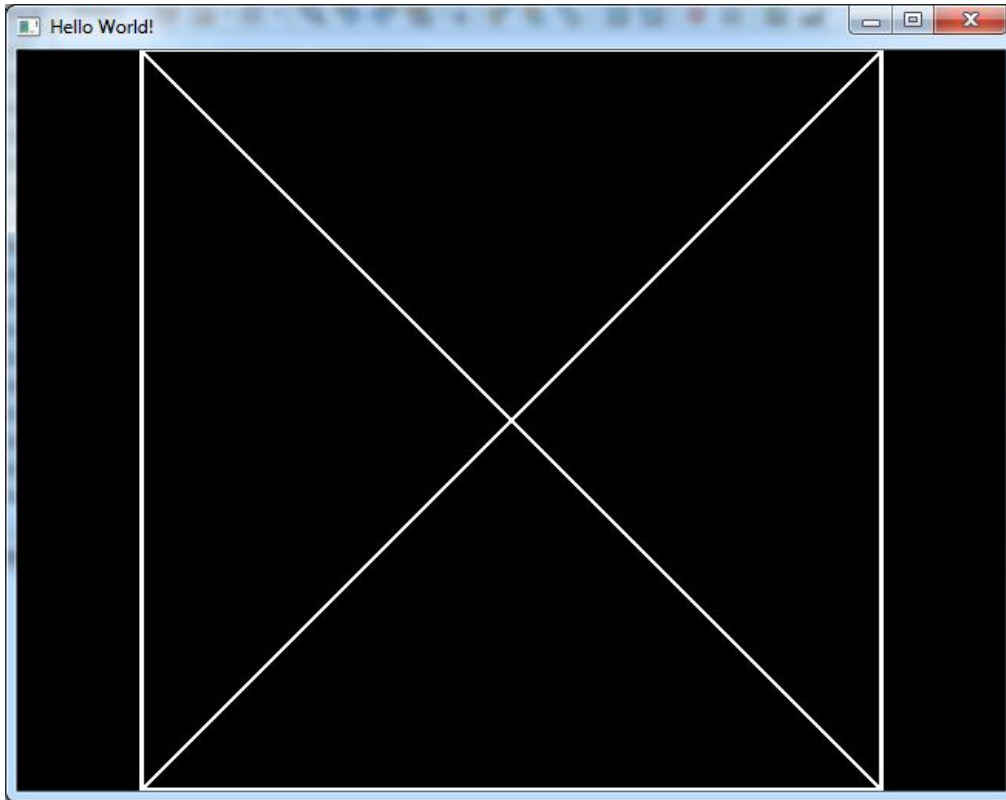


PV112 Java - 2. cvičení

Stáhněte si cv2.zip ze studijních materiálů. Rozbalte a složku otevřete v NetBeans `File -> Open project...` (`Ctrl + Shift + O`). Po úspěšném otevření, zkompilování a spuštění projektu (`Ctrl + F5`) by se mělo zobrazit okno 640x480 pixelů uprostřed obrazovky s nadpisem "Hello World!", černým pozadím a drátěným modelem krychle přes většinu obrazovky.



Dnes budeme pracovat převážně v souboru `Cv2.java`, který v NetBeans najdete v projektovém okně pod `cv2 -> Source packages -> cz.muni.fi.pv112.cv2 -> C v2.java`. Všimněte si také, že pro dosažení stejného měřítka souřadnic `X` a `Y` je použit `aspect` jako v 1. cvičení, viz.

```
model.vs.gls1.
```

Po dostatečném zorientování ve struktuře třídy doporučuji stiskem `Ctrl + Shift + Minus` sbalit všechny bloky a rozbalit si pouze metody `init()` a `render()`.

Task 1 – zmenšení krychle a transformace do NDC

Při práci s OpenGL se běžně používá pravoruký souřadnicový systém

(https://www.ntu.edu.sg/home/ehchua/programming/opengl/images/Graphics3D_RHS.png). Ve

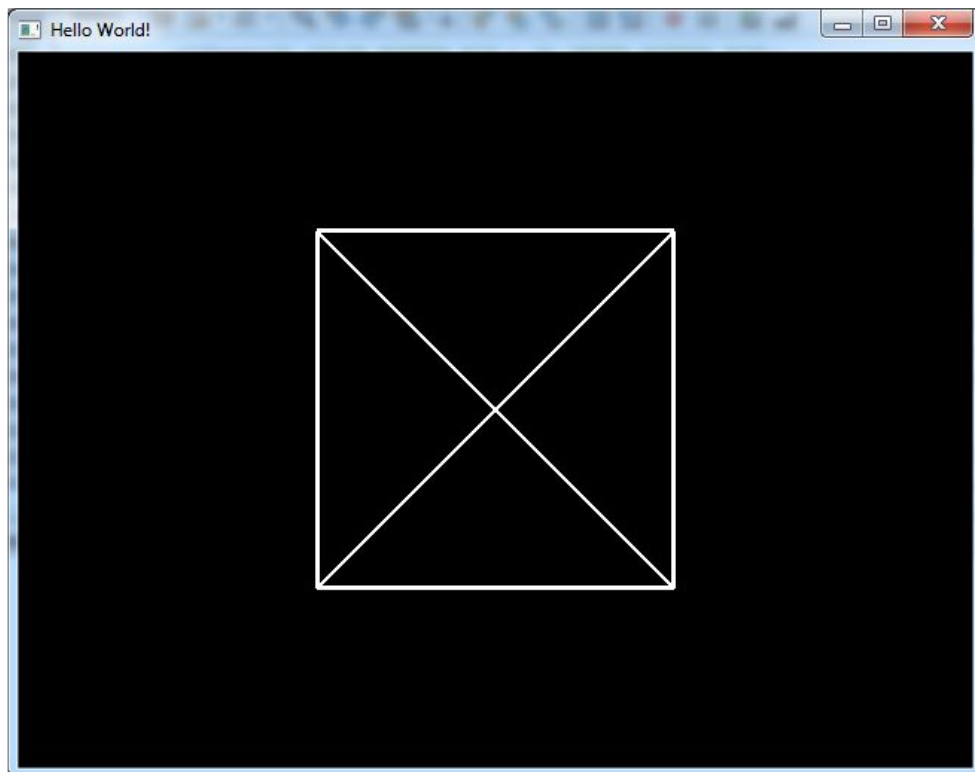
skutečnosti ale OpenGL očekává, že trojúhelníky určené k převodu na pixely (rasterizaci) jsou mu dodány v tzv. Normalized Device Coordinates (NDC) prostoru. Tento prostor je v OpenGL krychle s rozsahem $[-1, 1]$ ve všech třech (`X`, `Y`, `Z`) souřadnicích. NDC je možné nazývat také prostor ořezu, neboť části trojúhelníků ležící mimo tento prostor nejsou vykreslovány. Na prvním cvičení jsme pracovali se souřadnicemi, které v tomto prostoru již ležely, takže k žádnému ořezu nedocházelo.

Pro převod souřadnic z pravorukého souřadnicového systému do NDC je potřeba vynásobit `Z` souřadnici vrcholu `-1`, protože osa `Z` v NDC míří opačným směrem než v našem počátečním pravorukém systému.

Aby byla krychle ve scéně lépe vidět, zmenšíme obsah scény na 50% násobením všech souřadnic vrcholů `0,5`. Díky těmto násobením budeme moci používat vlastní souřadnicový systém, který bude

pravoruký a bude dvakrát „větší“ než NDC. Oba převody (transformace) se provedou násobením pozic vrcholů maticemi ve vertex shaderu, viz. `model.vs.glsl`. Potřebné matice se vytvoří a nastaví z Javy v `Cv2.java` → `render()`. Pro práci s maticemi použijeme knihovnu JOML (<https://joml-ci.github.io/JOML/apidocs/index.html>) a konkrétně třídu `Matrix4f`.

Zajistěte vše potřebné pro převod do NDC a zmenšení objektů ve scéně, nápovědy jsou umístěny v komentářích v kódu (Task 1).

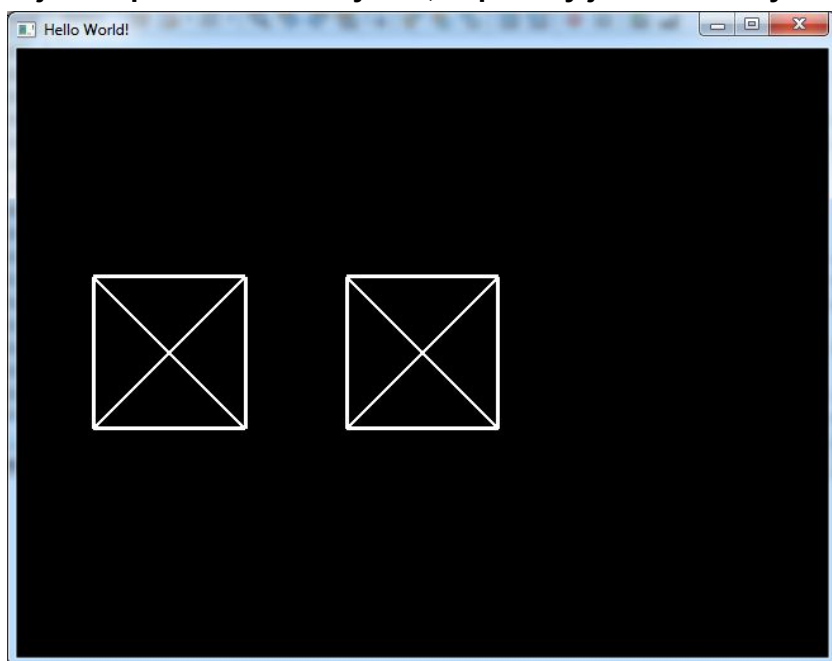


Task 2 – přidání druhé krychle

Nyní zmenšíme objekty ve scéně na 25% původní velikosti a přidáme druhou krychli, která leží nalevo od první a bude umístěna dále ve směru od pozorovatele.

Všimněte si, že ve výsledném obraze vypadají obě krychle, že leží ve stejné vzdálenosti od pozorovatele.

Zajistěte přidání druhé krychle, nápovědy jsou umístěny v komentářích (Task 2).

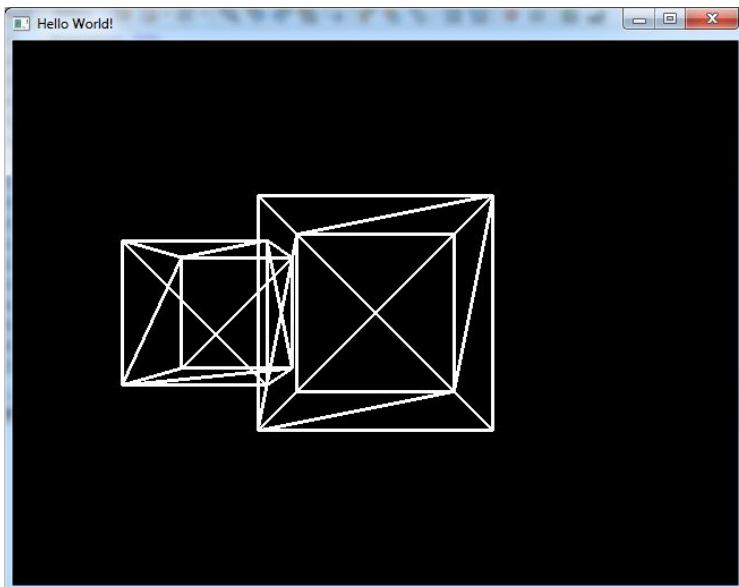


Task 3 – perspektivní promítání

Nyní vytvoříme a na již existující transformaci aplikujeme matici perspektivního promítání, takže vzdálenější objekty se po promítnutí (projekci) budou jevit menší. Matice perspektivního promítání nahradí naší vlastní projekci z Task 1, takže ji zakomentujeme nebo odstraníme. Toto promítání nám vytvoří nový počáteční souřadnicový prostor, takže část našich objektů bude ležet mimo. Je to proto, že perspektivní promítání promítne pohledový jehlan

(http://www.songho.ca/opengl/gl_projectionmatrix.html) na krychli NDC, takže není vidět vše, co leží mimo tento jehlan. Celý pohledový jehlan bude oříznut rovinami kolmými na Z a ležícími na souřadnicích -1 a 500. Aby byly naše krychle vidět celé, posuneme je kousek dozadu (v záporném směru Z). **Navíc** zrušíme dělení souřadnice X v shaderu `model.vs.glsl`, protože toto nyní zajistí perspektivní promítání.

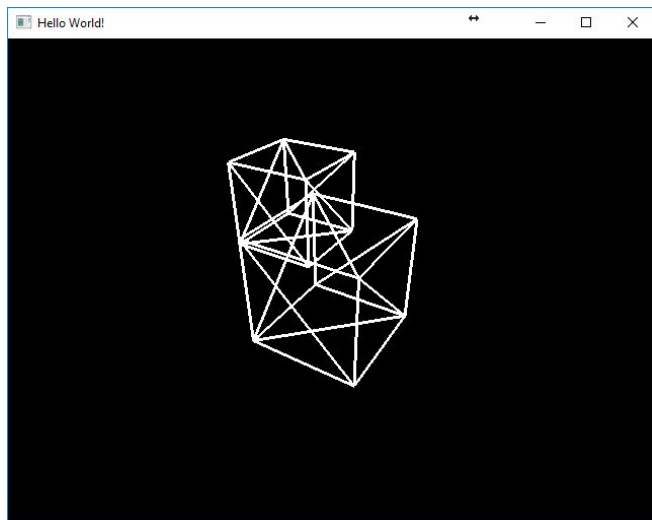
Zajistěte vše potřebné, nápovědy opět v komentářích (Task 3).



Task 4 – oddělení často používaných transformací

Jde tak trochu o refactoring. Doteď jsme používali pouze jednu matici, pomocí které jsme transformovali oba objekty. Běžné však je, že si matice opakovaně používaných (základních) transformací uložíme zvlášť, abychom je nemuseli vytvářet pokaždé znova. Právě takovými základními transformacemi jsou promítání a transformace pohledem (kamera). Podle komentářů v kódu tyto dvě matice vytvoříme a použijeme pro vykreslování našich dvou krychlí – je potřeba zvolit jinou metodu `drawModel(...)`, kterou pro vykreslení krychle používáme.

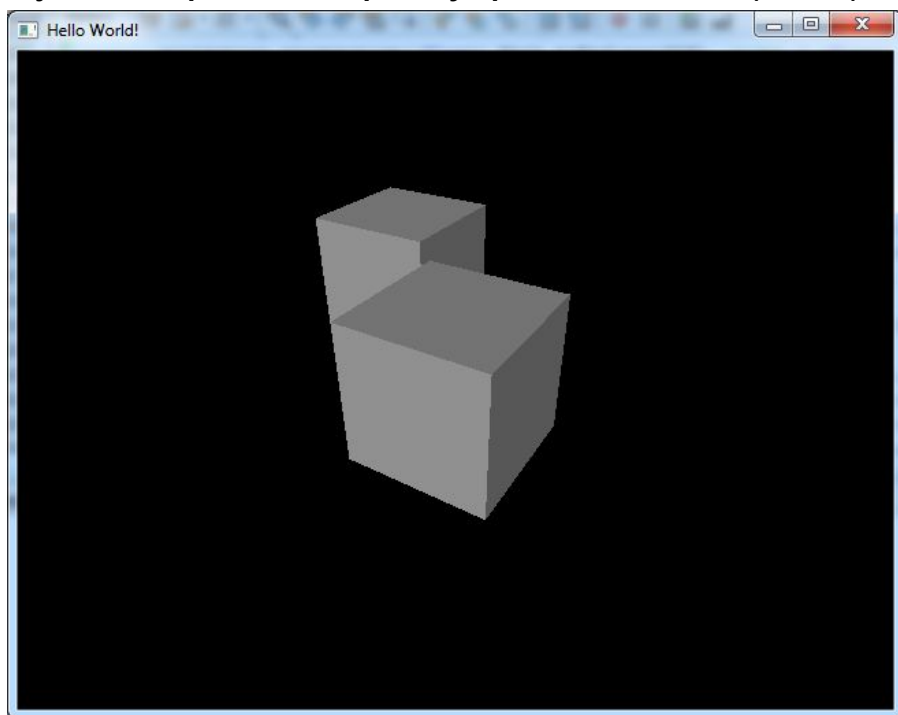
Zajistěte vše potřebné, nápovědy opět v komentářích (Task 4).



Task 5 – triviální osvětlení plných trojúhelníků

Doteď jsme naschvál kreslili pouze drátěný model. Důvodem bylo zjednodušení, abychom se nemuseli zabývat osvětlením a rozeznali hrany krychle i bez něj. V Java kódu zapneme `GL_DEPTH_TEST` a nahradíme konstantu `GL_LINE` atributem `mode`, který je možné měnit klávesami F (fill) a L (line). Dále zapneme triviální osvětlení v kódu fragment shaderu v `model.fs.glsl`.

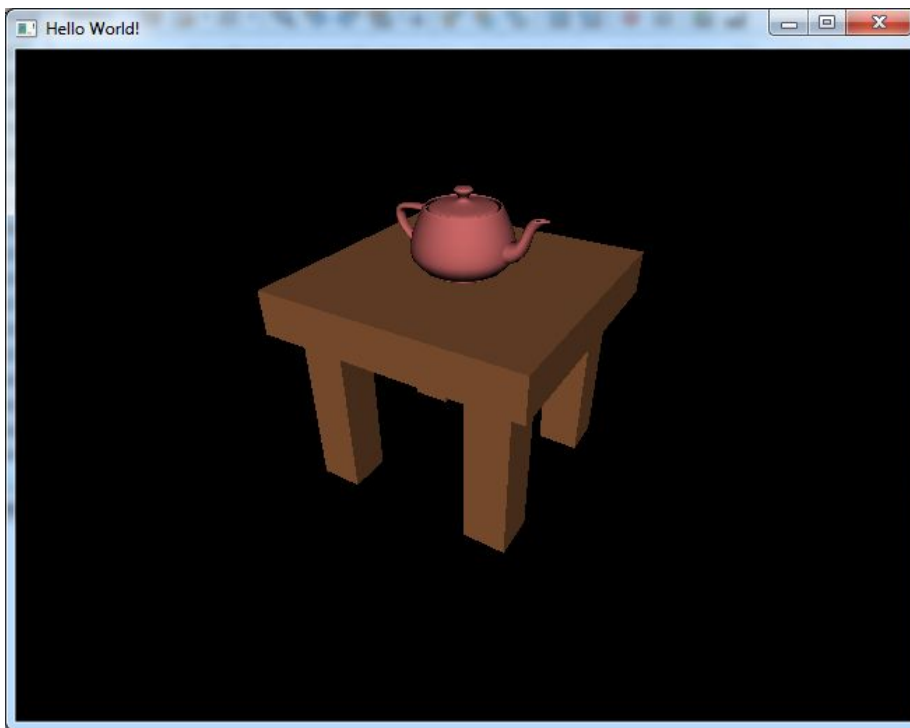
Zajistěte vše potřebné, nápovědy opět v komentářích (Task 5).



Task 6 – stůl s konvičkou

Nakreslete stůl jako pět kvádrů a čajovou konvičku na něm. Pro vykreslení použijte metody `drawTablePart` a `drawModel`.

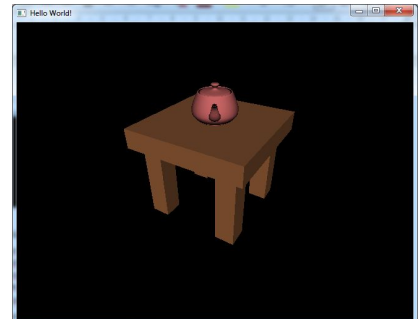
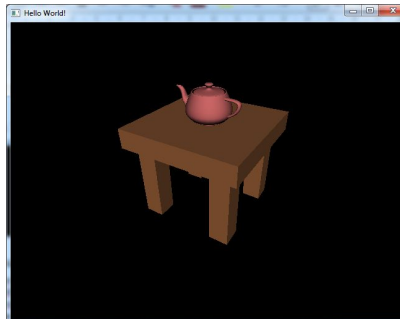
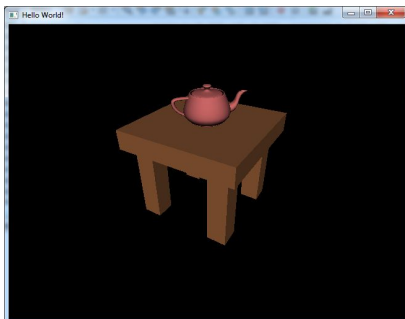
Zajistěte vše potřebné, nápovědy opět v komentářích (Task 6).



Task 7 – otáčení konvičky

Doteď byla scéna statická, v každém snímku se vykresloval stejný obraz. Do scény přidáme animaci otáčení konvičky. Otáčení konvičky docílíme použitím rotační matice okolo osy Y. Plynulost rotace je zajištěna nastavením programu tak, že překresluje pravidelně podle obnovovací frekvence monitoru, a minimální náročností naší scény pro GPU, takže si můžeme dovolit hodnotu v atributu t zvyšovat o pevný krok.

Zajistěte vše potřebné, nápovědy opět v komentářích (Task 7).



Task 8 (bonus) – vykreslení knížky

Doteď nebylo nutné o pořadí transformací příliš přemýšlet. To v běžné praxi neplatí neboť především pořadí transformací posunutí vs. rotace často způsobuje nechtěné výsledky. V zásadě jsou dva způsoby, jak lze o složených transformacích přemýšlet: lokální a globální. Například mějme transformace posunutí T a otočení R v kódu v tomto pořadí:

```
m.translate(...)
m.rotate(...)
```

Při transformacích používáme pravoruký souřadnicový systém, ve kterém se transformace vektoru v provádí násobením maticí M jako $v' = Mv$, tj. násobí se transformační maticí zleva. Ve výše uvedeném příkladě se složí matice $M = TR$ a tou se násobí $v' = TRv$. V lokálním přístupu k transformacím lze tedy přemýšlet tak, že se nejprve objekt otočí (R) a následně posunutí (T) se provede v novém lokálním souřadném systému daném R . V rámci globálního přístupu lze složenou transformaci chápat tak, že se

objekt nejprve posune na místo určené T a potom se celý prostor včetně již posunutého objektu otočí okolo svého středu $[0, 0, 0]$ podle R .

Nyní je vaším úkolem vykreslit žlutou knížku (jako kvádr) tak, aby ležela na nejbližším rohu stolu (k pozorovateli) a aby byla sama otočená okolo svého středu o 60° . Pro debugging transformovaných prostorů je možné využít vykreslení jejich souřadného systému jako osičky (X, Y, Z), viz `drawAxes`. **Zajistěte vše potřebné, nápovědy opět v komentářích (Task 8).**

