

PV112 Java - 6. cvičení

Stáhněte si cv6.zip ze studijních materiálů. Rozbalte a složku otevřete v NetBeans `File -> Open project...` (`Ctrl + Shift + O`). Po úspěšném otevření, zkompilování a spuštění projektu by se mělo zobrazit okno 640x480 pixelů uprostřed obrazovky s nadpisem "Hello World!" a konvičkou. Můžete ovládat pozici kamery; při držení levého tlačítka myši se můžete otáčet kolem středu, a při držení pravého tlačítka můžete přibližovat/oddalovat scénu.

Post-processing

Velké množství vykreslovacích technik a efektů vyžaduje více kroků ke zpracování dat a vygenerování kýženého obrazu. K tomu se často využívá možnost vykreslit data do textury a s ní dál pracovat/počítat. Naimplementujeme jednoduchý post-processing, kdy si vykreslíme scénu ne do paměti okna, ale do oddělené textury.

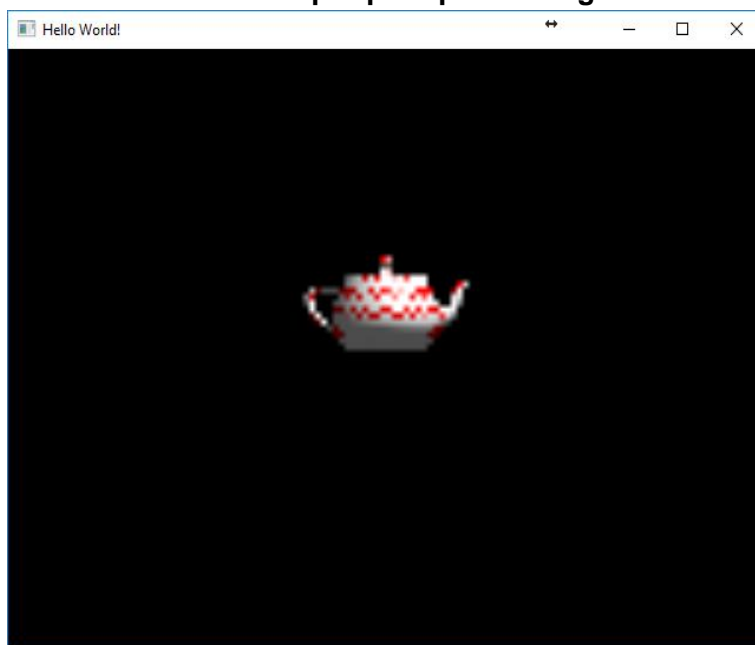
Task 1 - vykreslování mimo obrazovku do FBO a vykreslení textury na obrazovku přes fullscreen quad

Na vykreslování do textury se používá Frame Buffer Object (FBO), který umožňuje vytvořit vedlejší paměť, do které se bude kreslit, a u které např. lze nastavit, jak má vypadat paměť pro výstup; tzn. kolik textur (a jejich velikost) má být k dispozici a jaký je jejich formát (bitová hloubka jednotlivých složek, ...).

Povinná je pouze textura pro hloubku, vytvoříme si ale texturu pro barvu, protože tu cheme následně upravit a zobrazit. Do této paměti si vykreslíme scénu a budeme tedy mít její barevný výstup připojené textuře.

Takto vykreslená scéna do textury se ale sama nikde nezobrazí; vykreslíme ji tak, že vytvoříme obdélník, který pokryje celé okno a jeho fragmenty obarvíme podle uložené textury. Na to jsou potřeba dvě věci - geometrie obdélníku a shadery, které správně přetransformují vrcholy a obarví fragmenty.

Zajistěte vše potřebné k získání mezikroku pro post-processing.



Další výhodou vedlejší paměti pro vykreslování je i její nezávislá velikost. Můžeme např. kreslit do pětikrát menší textury – je to textura jako každá jiná a při vzorkování se její hodnoty mohou dopočítat.

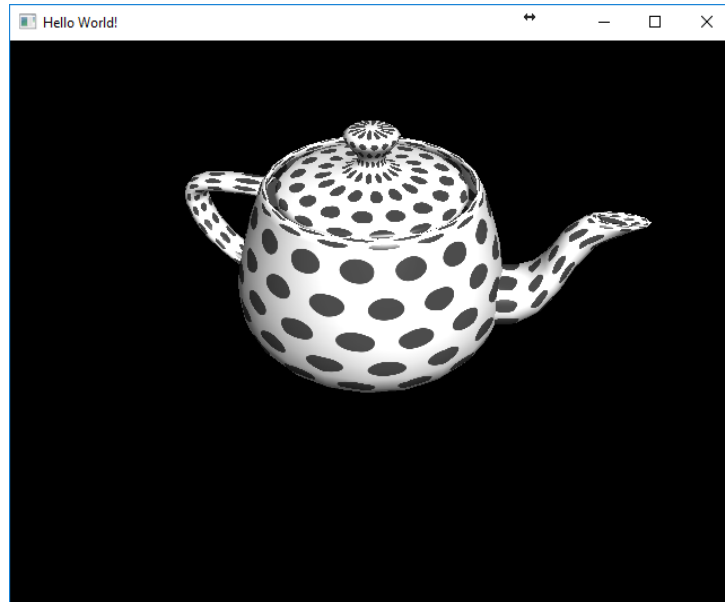
Výsledek lze vidět na následujícím obrázku. Zajímavější ale často je kreslit naopak do větší textury. K čemu to může být dobré?

Task 2 - úprava barev

Jedním z typických post-processing efektů je úprava výsledné barvy. Lze např. zobrazit jen jednotlivé barevné složky. Dalšími úpravami může být inverze barev nebo zvýraznění některých barevných tónů. Vyzkoušíme si převod na černobílý obraz, který není nic jiného než zobrazení pouze jasové složky světla. Naivní implementace by byla jednoduché zprůměrování barvy, ale lidé vnímají různé vlnové délky jinak intenzivně, proto je potřeba vážený průměr.

Doporučované váhy jsou různé, často se ale používají ty ze standardu YIQ/NTSC a to 0.299, 0.587 a 0.114 pro R, G a B složky.

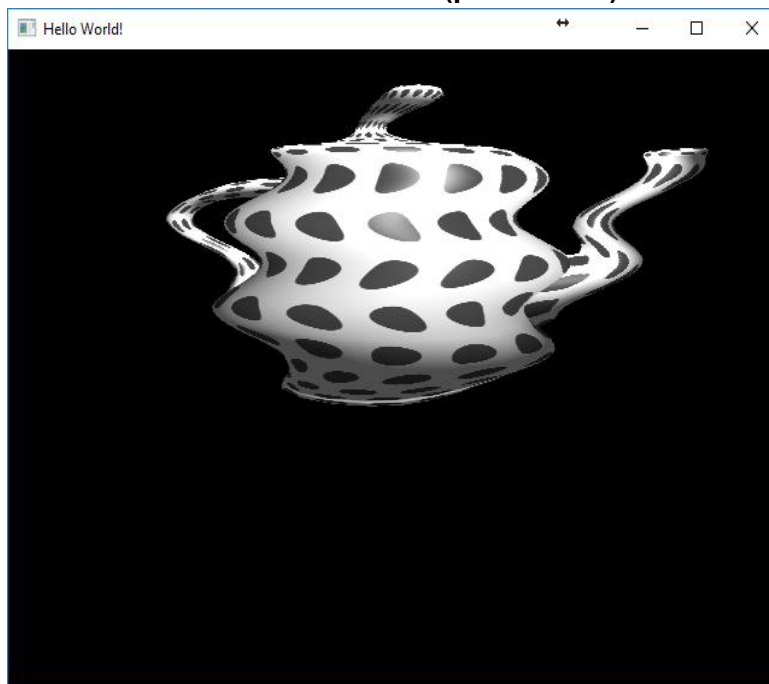
Zprůměrujte barvy z textury, aby byl vykreslen černobílý obraz.



Task 3 - Vlnky (deformování obrazu posunem)

Předchozí efekt by šel naimplementovat i bez post-processingu; ale pro následující se bez něj už neobejdete. Vzhledem k tomu, že máme k dispozici všechny vykreslené pixely, můžeme si pomocí texturovacích souřadnic vzít i takový, který původně byl na jiném místě.

Vytvořte zvlnění obrazu. Zkuste ho animovat v čase (po stisku A).



Task 4 - kreslení drátěného modelu

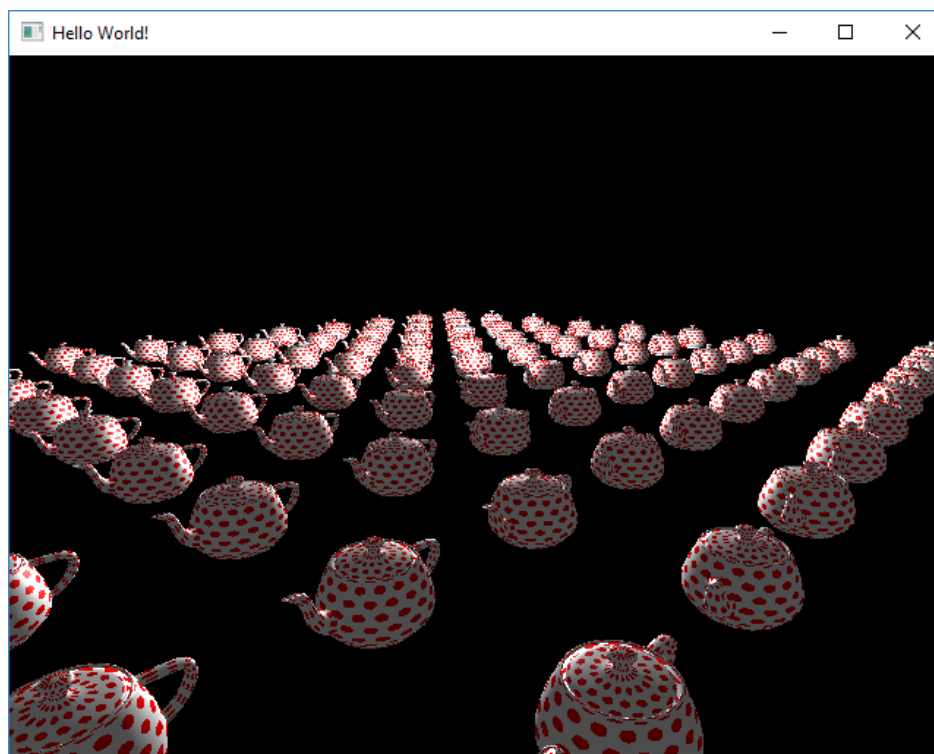
Pokud teď stiskneme L pro vykreslování čar místo vyplněných trojúhelníků, uvidíme drátěný model 2 trojúhelníků:



Proč se to děje? Rozhodně se nejedná o chtěný výsledek – chtěli bychom vidět zvlněný černobílý drátěný model konvičky. **Upravte vykreslování tak, aby byl vidět správný drátěný model.**

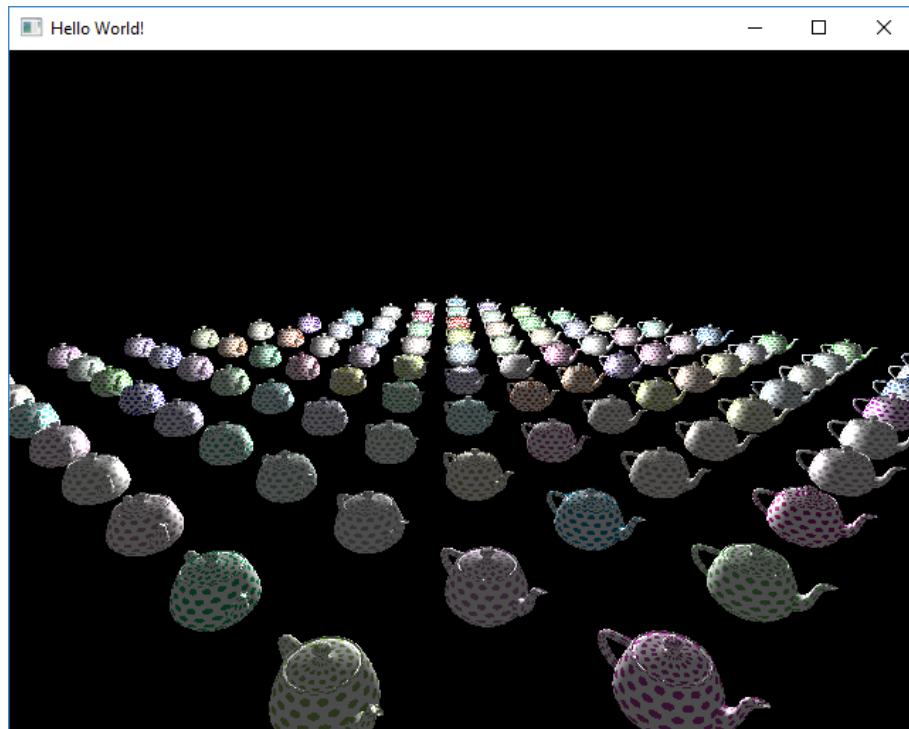
Task 5 – Instacování

Instacování slouží k vykreslení velkého množství objektů pomocí jednoho vykreslovacího příkazu, tak, že každý vykreslovaný objekt může mít svoje vlastní parametry, které se pro vykreslení použijí. Pomocí instancování vykreslete 100 konviček, rozložených do mřížky 10x10 tak, aby byly rozmístěné okolo středu (0,1,0). Na posunutí o 1 v ose Y nezapomeňte, bude se později hodit!



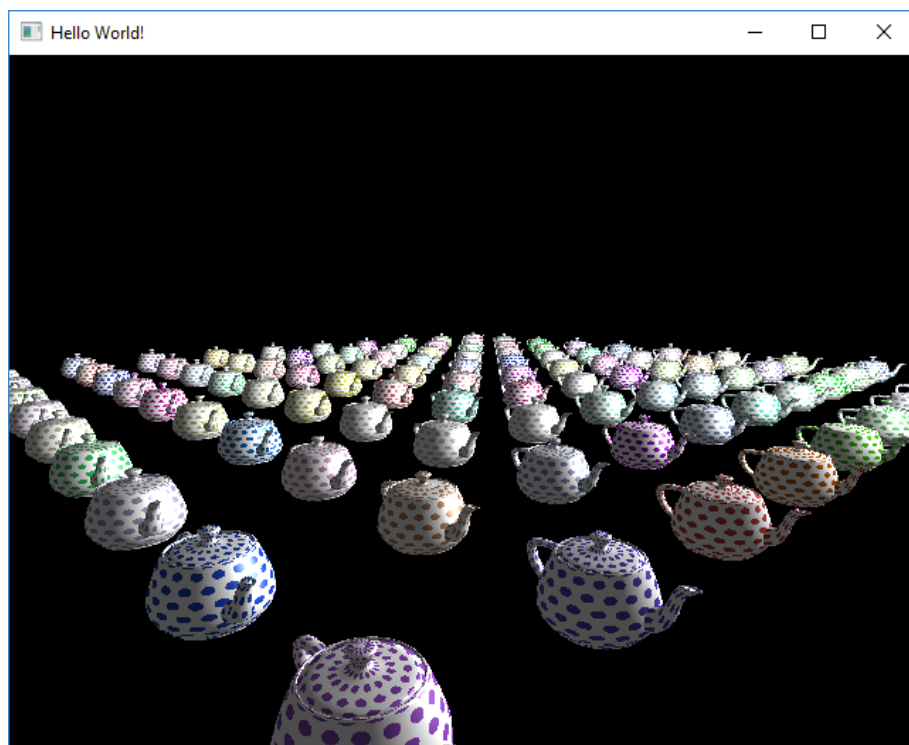
Task 6 – Různá barva pro každou instanci

Kromě pozice zkuste pro každou instanci i jinou barvu. Pro získání náhodných barev můžete použít funkci, kterou jsme vám již připravili, `randomColor()`.



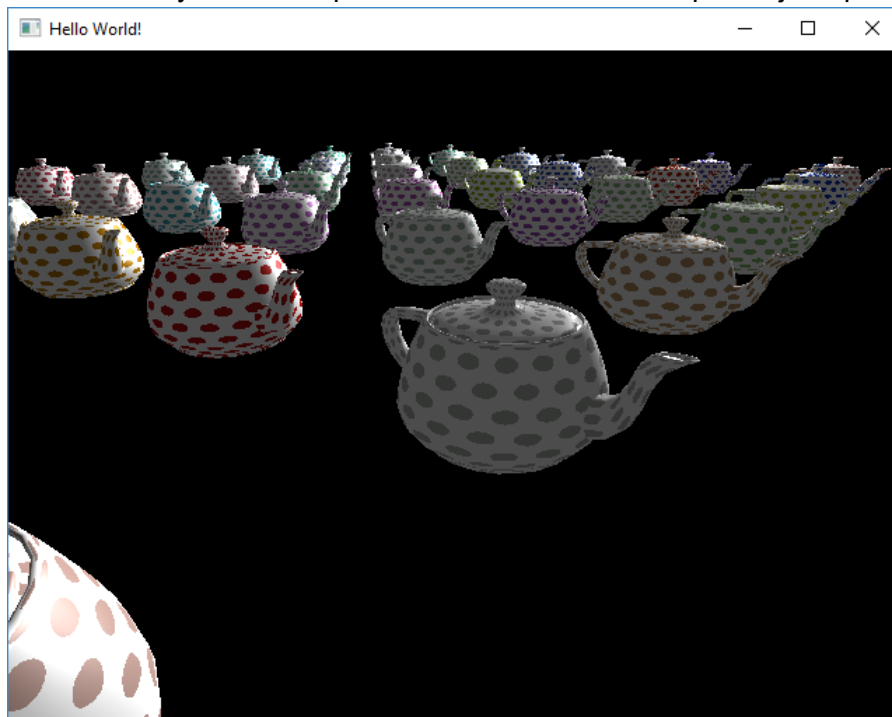
Task 7 – UBO pro světlo

Nyní předáváme parametry světla jako několik uniform proměnných; můžeme ale nahrát všechny proměnné do .



Task 8 – Změna části UBO

Data v UBO se dají měnit – nicméně přehrávat je všechna by bylo zbytečné, a lze změnit pouze část. Zkuste animovat světlo tak, aby se v UBO pro data se světlem měnila pouze jeho pozice.



Task 9 – Přidání dalších UBO

V tomto kroku přidáme další UBO, který bude obsahovat data o instancích konviček - pozice a barvy, které jsme doposud předávali jako uniformní proměnné. Po tomto kroku se nám vzhledově výsledná scéna nezmění.

Task 10 – Přidání shaderu a modelu navíc

V tomto úkolu si přidáme vykreslení dalšího modelu do scény, který bude používat jinou dvojici shaderů. Abychom nemuseli nastavovat všechny jeho uniformní proměnné znovu, pro osvětlení použijeme již existující UBO s daty o světle. Pro data o instanci si vytvoříme další UBO, do kterého data nahrajeme.

