Zadání projektu – Texturování

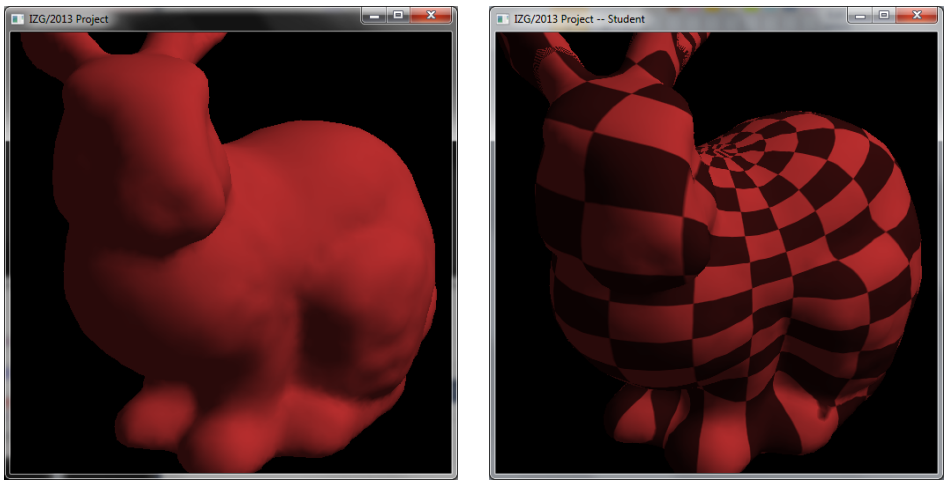
Základy počítačové grafiky (IZG)
ak. rok 2013/2014

Michal Španěl, spanel@fit.vutbr.cz

24.2.2014

1 První seznámení

Cílem projektu je pochopení praktických souvislostí témat přednášek a cvičení. Základem projektu je připravená funkční kostra programu, který realizuje softwarový rendering jednoduchého polygonálního modelu.



Obrázek 1: Ukázka zobrazení modelu *bunny.tri* před vypracováním projektu (vlevo) a po vypracování projektu (vpravo). Doplněno *texturování* s využitím pomocného tělesa - koule.

- Aplikace načítá zobrazovaný model z textových souborů s příponou .TRI. Pro testování máte na výběr několik modelů.

- Standardně zobrazovaný model (BUNNY.TRI) lze změnit zadáním jiného názvu na příkazové řádce:

  Příklad: IZG_PROJ SKULL.TRI

- Základním zobrazovaným primitivem je trojúhelník.
- Aplikace řeší celý proces vykreslování včetně rasterizace trojúhelníku softwarově (bez grafické karty, bez OpenGL).
- Pro manipulaci se scénou je implementován primitivní manipulátor, který pouze otáčí a zvětšuje/zmenšuje zobrazovaný model. Pozice kamery je fixní.
- Renderer řeší veškeré transformace modelu včetně perspektivní projekce.

S naprostou většinou algoritmů, které jsou v projektu použity (transformace, rasterizace, osvětlení, apod.) se postupně seznámíte na přednáškách a cvičeních.

2 Zdrojový kód

Kostra programu je obdobou frameworku, který používáte ve cvičení a je napsána v *čistém C*.

- Kreslíme do vlastního alokovaného frame bufferu.
- Pro vykreslení frame bufferu na obrazovku a reakce na události (myšování, apod.) se používá knihovna *SDL* (*Simple Directmedia Layer*).
  http://www.libsdl.org/

Předkompilované balíčky knihovny SDL (testováno s verzí *1.2.15*) najdete na webu:

http://www.libsdl.org/download-1.2.php

Překlad ve zkratce:

- *Windows/MinGW* – Na windows lze použít překladač *MinGW* (GCC portované na windows) a přiložený *Makefile*. V učebnách fakulty mi funguje:
  - Otevřít okno s příkazovou řádkou.
  - Nastavit cesty k MinGW překladači spuštěním: Q:/MINGW/SET_MINGW.BAT (alternativně: SET PATH=Q:/MINGW/MINGW/BIN;%PATH%)
  - Spustit překlad: MINGW32-MAKE
  - V tomto případě se použije již instalovaná knihovna SDL z adresáře Q:/MINGW/SDL.
- *Windows/MSVC* – Otevřete dodaný solution v *MS Visual Studiu* a nechte proběhnout případnou konverzi solution a projektu. SDL knihovnu nainstalujete takto:
  - Stáhněte si balíček binárních knihoven pro MSVC: http://www.libsdl.org/release/SDL-devel-1.2.15-VC.zip
  - A rozbalte jej do adresáře "vedle"projektu, tzn.:
    .
    ..
    SDL-1.2.15
    IZG_PROJ
  - F7 spustí překlad
- *Linux/GCC* – Pro překlad na linuxu je připravený *Makefile* stačí tedy: MAKE. SDL knihovna bývá součástí většiny Linuxových distribucí, pokud ne:
  - Stáhněte si zdrojové kódy z http://www.libsdl.org/release/SDL-1.2.15.zip
  - Zadejte typickou sekvenci ./CONFIGURE; MAKE; MAKE INSTALL
  - Na Ubuntu funguje také: SUDO APT-GET INSTALL LIBSDL1.2-DEV

V případě potíží s překladem (v uvedeném pořadí)...

1. hledejte chybu na vašem konci klávesnice,
2. zeptejte se kamaráda,
3. nahlédněte do diskusního fóra IZG a případně pošlete dotaz,
4. zatancujte kolem svého počítače, proneste několik magických manter a pak zkuste překlad znovu,
5. napiště mi email.

3 Bodovaný úkol

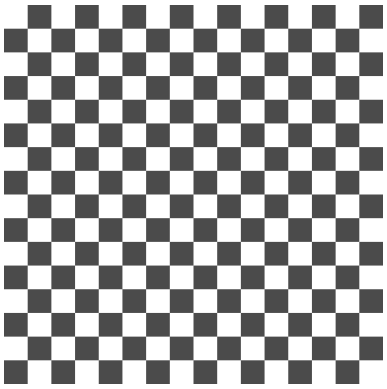
Cílem je doplnit do připraveného *„studentského“* rendereru následující algoritmy/funkce:

- *podporu texturování* s využitím koule jako pomocného tělesa
- nanést na model *texturu šachovnice*.

Studenti pracují na řešení projektu samostatně a každý odevzdá své vlastní řešení. Poraďte si, ale řešení vypracujte samostatně!

3.1 Krok č.1 – Generování textury šachovnice

Do vašeho studentského rendereru přidejte podporu pro vytvoření a uložení 2D textury a v rámci inicializace rendereru procedurálně vygenerujte texturu imitující šachovnici.



Obrázek 2: Ukázka správně připravené textury.

Textura bude pole S_RGBA hodnot, podobně jako frame buffer. Zachovejte rozměr textury v pixelech daný konstantou TEXTURE_SIZE a počet políček NUM_OF_TILESxNUM_OF_TILES. Pro tmavá pole použijte barvu BLACK_TILE a pro světlá WHITE_TILE. Co bude třeba udělat?

- Do struktury S_StudentRenderer doplnit vlastní reprezentaci 2D textury s využitím typu S_RGBA - tzv. aktivní kreslící textura.
- Upravit fce studrenCreate() a studrenRelease() - přidat vytvoření a nakonec uvolnění textury.

- Funkce STUDRENCREATE() – doplnit kód, který vytvoří texturu se vzorem šachovnice.

3.2 Krok č.2 – Přiřazení texturovacích souřadnic vrcholům

Pro všechny vrcholy načteného polygonálního modelu automaticky vygenerovat a přiřadit texturovací souřadnice s využitím mapovani přes pomocné těleso - kouli.

Vzorce pro výpočet texturovacích souřadnic u, v jsou odvozené ze sférických souřadnic a lze najít více různých variant. Můžete předpokládat, že model je umístěn v počátku souřadného systému a posunutí středu koule není třeba řešit.

Je-li (x, y, z) normalizovaný vektor ze středu souř. systému do vrcholu, pak můžete texturovací souřadnice určit takto:

$$\begin{aligned} u &= 0.5 + \frac{arctan(\frac{z}{x})}{2\pi}, \\ v &= 0.5 - \frac{arcsin(y)}{\pi}. \end{aligned}$$

Jak je v grafice zvykem, vygenerované texturovací souřadnice budou v intervalu $< 0, 1 >$, který vždy odpovídá aktuálním rozměrům textury.

- Kód pro přiřazení texturovacích souřadnic doplňte do fce ONINIT().
- Pro uchování texturovacích souřadnic je určeno pole TEXCOORDS ve struktuře S_MODEL.

3.3 Krok č.3 – Získání hodnoty textury s bilineární interpolací

Pro získání hodnoty z textury na souřadnicích u, v využijte bilineární interpolaci čtyř nejbližších pixelů. Hodnoty u, v v intervalu $< 0, 1 >$ odpovídájí rozměrům textury v pixelech.

Pokud vám korektně funguje generování texturovacích souřadnic, není třeba řešit případné opakování textury, protože texturovací souřadnice nebudou mimo tento interval.

- Kód doplňte do STUDRENTEXTUREVALUE().
- Alpha složku výsledné barvy nebudeme využívat, můžete ji nastavit na 255.

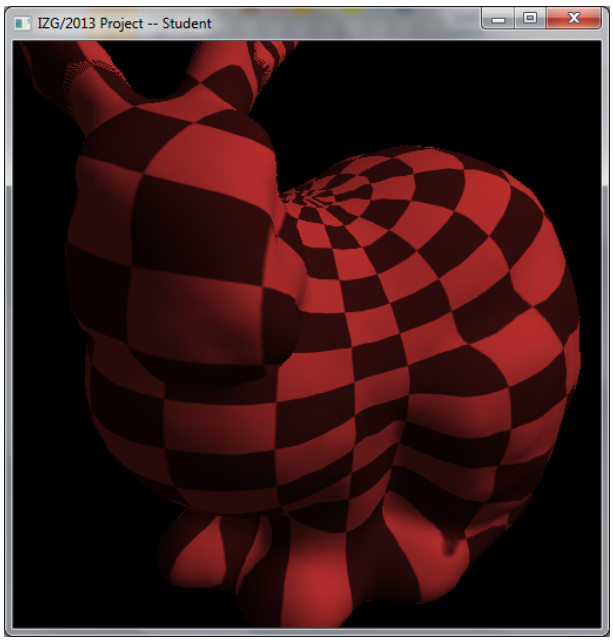
3.4 Krok č.4 – Podpora texturování při rasterizaci

Posledním krokem implementace texturování je úprava samotné rasterizace trojúhelníku, kdy je nutné interpolovat texturovací souřadnice pomocí barycentrických souřadnic a následně zkombinovat barvu získanou výpočtem osvětlovacího modelu s hodnotou textury v daném bodě pomocí *modulace*.

Modulace je princip aplikace textury, kdy textura dále strukturuje povrch objektu daný materiálem:

$$\begin{aligned} R &= R_l * R_t, \\ G &= G_l * G_t, \\ B &= B_l * B_t. \end{aligned}$$

kde $R_l G_l B_l$ je barva určená osvětlovacím modelem a $R_t G_t B_t$ hodnota textury ($R_t, G_t, B_t \in < 0, 1 >$).



Obrázek 3: Výstup po implementaci texturování.

Bude tedy třeba:

- Upravit fci studrenProjectTriangle() – proti původní verzi přidat volání vaší upravené fce studrenDrawTriangle().

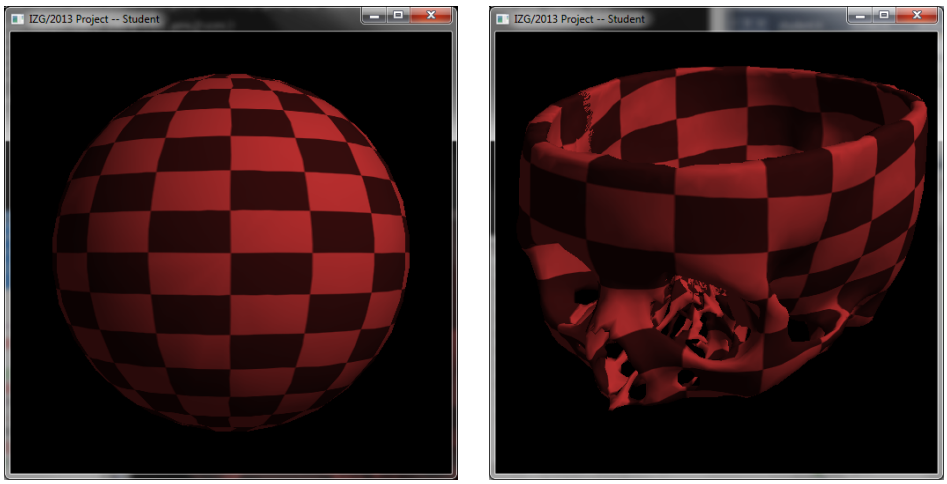
- Upravit fci STUDRENDRAWTRIANGLE() – doplnit interpolaci texturovacích souřadnic pomocí barycentrických souřadnic podobně jako se ve stávající funkci interpoluje souřadnice z.
- Hodnotu textury získat pomocí vaší fce STUDRENTEXTUREVALUE().
- Doplnit modulaci barvy trojúhelníku pomocí textury.

Pozn.: Není třeba řešit perspektivní korekci textury.

3.5 Poznámky k řešení

Pro vaše modifikace je připravená oddělená varianta rendereru v souborech STUDENT.H a STUDENT.C. S projektem experimentujte dle libosti (měňte a modifikujte co chcete), ale pamatujte, že odevzdané soubory STUDENT.H a STUDENT.C musí být funkční s originálním frameworkem! Pro odevzdané řešení je povoleno modifikovat pouze tyto dva soubory.

Jak přepnout na zobrazování pomocí studentského rendereru? Konkrétní renderer se vytváří ve fci MAIN() v souboru MAIN.C. Počáteční volbu lze ovlivnit definováním makra USE_STUDENT_RENDERER na začátku MAIN.C a opětovným překladem. Renderer lze také přepnout za běhu pomocí kláves **O** a **P**.

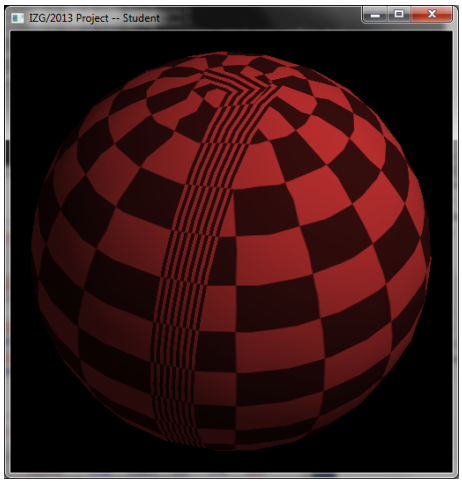


Obrázek 4: Ukázky dalších modelů...

4 Tipy pro vypracování

- Doporučuji prostudovat princip základního rendereru (soubory RENDER.H a RENDER.C), velká část projektu je o menších modifikacích již existujících funkcí...
- Pro kontrolu, zda máte texturu správně inicializovanou, můžete využít fci SAVEBITMAP() z BMP.H.

Současná verze rendereru a reprezentace 3D modelů neumožňuje snadno a korektně vyřešit přiřazení texturovacích souřadnic trojúhelníkům, jejichž hrany přecházejí přes hranu textury $1 - 0$. Následná interpolace u a v v rasterizaci způsobuje potíže viditelné na Obrázku 5. Toto nebude považováno za chybu!



Obrázek 5: Ukázka „povolená chyba“.

5 Odevzdání

viz. web

6 Závěrem

Ať se dílo daří a ať vás grafika alespoň trochu baví!!!