Rendskell

Haskell renderer

David Němeček

17. 12. 2008

Obsah

Popis projektu

Zadání

Zadáním projektu bylo vytvořit renderovací program v Haskellu, který jako vstup bere deskriptivní popis scény a jako výstup vytvoří odpovídající obrázek a uloží jej do souboru ve formátu TARGA.

Souhrnný popis řešení

Program pracuje na principu ray-tracingu, vytváří si virtuální matematicky definovanou scénu na základě poskytnutého deskriptivního souboru, z bodu pozice kamery vysílá paprsky (polopřímky) a hledá průsečíky s tělesy scény a z těch skládá výsledný rastrový obrázek, který následně zapíše na disk.

Moduly

Projekt je pro přehlednost a možnost snadného opětovného využití některých funkcí rozdělený do následujících logických celků:

Image

Obsluha rastrové reprezentace obrázku pomocí pixelů a ukládání do souboru.

Modul poskytuje základní datové typy Pixel (barevný bod definovaný trojicí barev – červená, zelená, modrá) a Pixmap (seznam pixelů).

Tyto typy je následně možné pomocí odpovídající funkce převést na binární datový řetězec, spojit s hlavičkou souboru *.tga a celek uložit na disk ve formě výsledného obrázku.

Geometry

Základní geometrické datové typy a funkce pro práci s trojrozměrným prostorem, slouží jako „jádro“ pro samotné vytvoření matematické reprezentace scény.

V modulu jsou definovány datové typy bod, vektor, barva, těleso, paprsek, průsečík paprsku s tělesem, tělesová množina s operacemi „přidat“ a „odebrat“.

Součástí je i sada funkcí na práci s vektory, výpočet průsečíků paprsku s tělesy a jejich další zpracování.

Utils

Různé všeobecně využitelné funkce, jako je výpočet kořenů kvadratické rovnice.

Parser

Zpracování vstupních souborů s popisem scény. Modul poskytuje funkce k načtení dat a jejich postupné převedení do patřičných datových struktur především z geometrického modulu a vytváří tak mezivrstvu, která odděluje funkcionálně „nečistou“ I/O část programu od samotných výpočtů grafického zobrazení scény.

Renderer

Hlavní část ray-tracing enginu, na základě geometrické reprezentace scény vytváří rastrový obrázek určený k uložení.

Modul definuje dodatečné datové typy světla (ambientní, směrové) a kamery.

Renderovací funkce vytvářejí v prostoru pole paprsků, které jsou rovnoběžné, respektive vycházejí z jednoho bodu, v závislosti na tom, jestli jde o paralelní, nebo perspektivní promítání, ty jsou otestovány na průnik objekty ve scéně, v každém paprsku je použit nejbližší průsečík ke kameře, z něj je veden další paprsek směrem do světla, pokud protíná na své cestě nějaký objekt, je bod ve stínu, jinak se použije barva objektu s přihlédnutím na úhel, který svírá normála v tomto bodě s vektorem směrového světla. Z jednotlivých výsledků tohoto pole vzniká výstupní pixelový rastr k uložení do souboru.

Main

Vstupní bod programu, spojující jednotlivé moduly dohromady. Pro každý soubor, který je předáván přes parametry při spuštění, se volá parsovací funkce, výsledná reprezentace scény je předána k renderování, při kterém vznikne pro každou zadanou kameru výsledný obrázek.

Formát vstupu

Příkazová řádka

Program při spouštění bere jako své parametry jména souborů scén.

Příklad: ./rendskell scena1.rsd scena2.rsd

Soubor se scénou

Zpracovatelné soubory se scénou nesou příponu *.rsd (Rendskell Scene Description). Nejde však o závazné pravidlo, spíše o doporučení, na příponě totiž nezáleží. Tento soubor má následující strukturu a požadavky na obsah:

- Jde o čistě textový dokument v kódování UTF-8 rozdělený na řádky pomocí znaku \n (0x0A).
- První řádek obsahuje řetězec „Rendskell“ k základnímu ověření, že jde o soubor určený pro tento program.
- Prázdné řádky a řádky začínající znakem „#“ (0x23) jsou ignorovány, je možné je tedy využít k psaní komentářů a zpřehlednění textu.

- V ostatních případech jde o řádek obsahující příkaz a případně dodatečné parametry ve formátu: jednoslovný příkaz, mezera, parametry oddělené mezerou.
    - **Sphere** – přidá do scény kouli
        1. **akce** (add/sub) – množinová operace: „add“ objekt přidá, „sub“ odečte od aktuální scény
        2. **X Y Z** (3x Float) – pozice středu
        3. **poloměr** (Float)
        4. **R G B** (3x Float) – barva objektu
    - **Camera** – přidá kameru pro renderování
        1. **X Y Z** (3x Float) – pozice kamery
        2. **X Y Z** (3x Float) – vektor první osy promítací roviny
        3. **X Y Z** (3x Float) – vektor druhé osy promítací roviny
        4. **projekce** (paralell/perspective) – volba paralelní nebo perspektivní projekce
        5. **soubor** (String) – jméno souboru, do kterého se obraz kamery uloží (k názvu se automaticky přidá přípona „.tga“)
        6. **šířka výška** (2x Integer) – rozměry obrázku k renderování v pixelech
    - **Light** – přidá do scény osvětlení
        - **ambient** – ambientní osvětlení přidává světlost všem objektům scény
            1. **síla** (Float) – jak moc se toto světlo projeví
            2. **R G B** (3x Float) – barva světla
        - **directional** – směrové světlo je určeno vektorem, ve kterém působí a může tak vytvářet stín
            1. **X Y Z** (3x Float) – směrový vektor světla
            2. **síla** (Float) – jak moc se toto světlo projeví
            3. **R G B** (3x Float) – barva světla
    - **/Scene** – ukončuje čtení souboru, za tímto rádkem mohou následovat jakákoliv data

Seznam souborů

Kromě nutných zdrojových souborů (Main.hs, Parser.hs, Image.hs, Geometry.hs, Util.hs, Renderer.hs) a Makefile ke snadnému překladu programu obsahuje projekt pár ukázek scén a jejich výstupu (caterpillar, showcase1, showcase2, shadetest).

Seznam zdrojů

- http://local.wasp.uwa.edu.au/~pbourke/dataformats/tga/