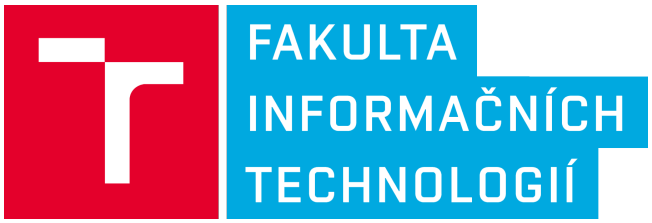


Počítačová grafika

Zobrazení kvadrik pomocí Ray Tracingu

13. prosince 2018

Autoři: Martin Ivančo, xivanc03@stud.fit.vutbr.cz
Petr Buchal, xbucha02@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

Zadání

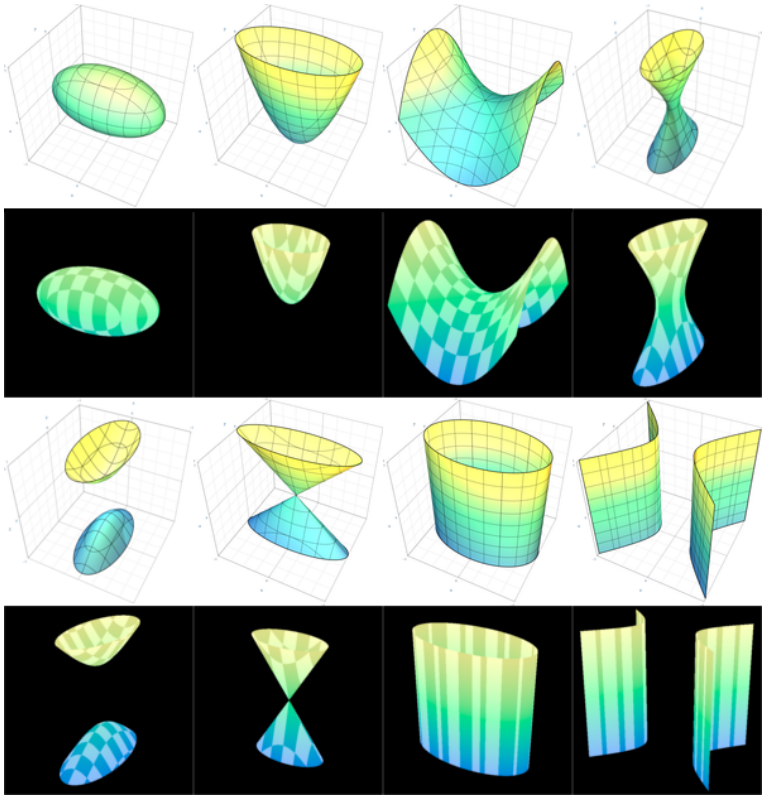
- Vytvořit nebo převzít jednoduchý ray tracer
  - Udělat průzkum, jak náročné je ray tracer naprogramovat
  - Udělat průzkum existujících řešeních
  - Na základě obecného průzkumu vybrat vhodnou cestu
- Zobrazovat obecné kvadriky
  - Možnost zobrazení s různými texturami
  - Transformace zobrazení kvadrik v prostoru
- Program bude interagovat s uživatelem
  - Uživatel si bude moci intuitivně vybrat, jaký typ kvadriky s jakými parametry chce vykreslit
  - Uživatel si bude moci vykreslit kvadriky s ukázkovými parametry

Nejdůležitější dosažené výsledky

V projektu jsme jako základ využili existující ray tracer, který byl schopen rendrovat některá základní primitiva a přidali jsme podporu pro rendrovaní kvadrik. Zobrazení kvadrik lze transformovat pomocí změn v nastavení kamery. Kvadriky lze dále vykreslit s texturou či osvětlením. Všechny tyto vlastnosti je schopen kontrolovat uživatel pomocí grafického uživatelského rozhraní.

Vykreslování různých kvadrik

Výsledný program je schopen vykreslovat 8 druhů reálných kvadrik s různými parametry. Program také obsahuje možnost nastavit bounding box, aby bylo možné oříznout kvadriky a zobrazit také jejich vnitřní část. Při programování jsme se snažili dostat, co nejblíž k matematickým programům, které často umějí kvadriky rendrovat.



Obrázek 1: Porovnání kvadrik vytvořených pomocí ray traceru (černé pozadí) s referenčními kvadrikami na Wikipedii (bílé pozadí)(částečně převzato z [2]).

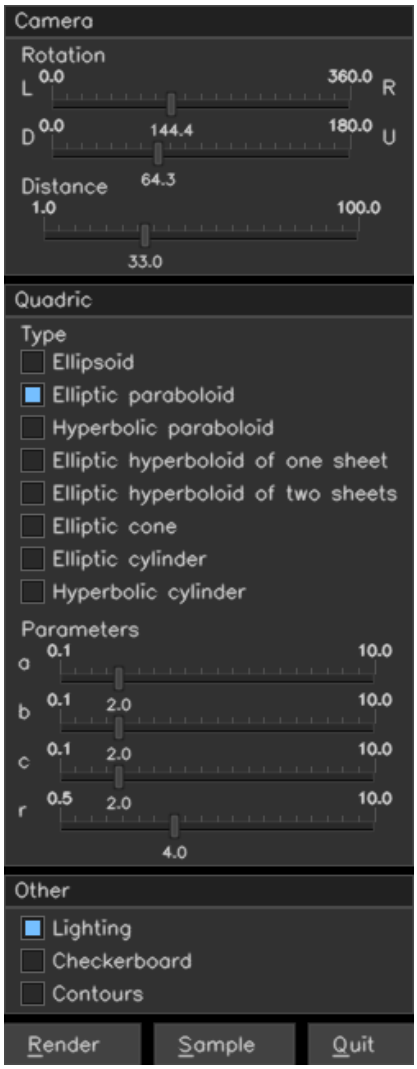
Textury, osvětlení, rotace

Aby jsme dosáhli co největší přehlednosti a čitelnosti kvadrik, využíváme několika procedurálně generovaných textur. Ta nejzákladnější z nich je generována na základě výškové souřadnice daného bodu. Barvy plynule přecházejí od modré přes zelenou až po žlutou u nejvýše položených bodů. Obarvení se samozřejmě přizpůsobuje už zmíněnému bounding boxu kvadriky. Program také podporuje skokovou změnu barvy pro lepší viditelnost výškových vrstevnic kvadriky. Dále program také podporuje vykreslování čtvercové sítě, u které jsou zesvětlovány pixely ze střídajících se čtvercových plošek, aby byl vytvořen šachovnicový vzor.

Kromě těchto procedurálně generovaných textur program také podporuje osvětlení, k čemuž bylo nutné počítat normály k povrchu kvadriky. Blíže je tento proces popsán v následující kapitole. Program umožňuje také intuitivně měnit pozici kamery v prostoru, aby bylo možné kvadriky zobrazit z různých úhlů pohledu. Pozice kamery je počítána ze dvou úhlů určujících natočení a vzdálenosti od počátku souřadnicové soustavy. Kamera je tedy vždy orientována tak, že směruje ke středu souřadnicové soustavy. Toto by bylo samozřejmě možné jednoduše změnit, ale z hlediska intuitivnosti uživatelského prostředí jsme zhodnotili, že toto omezení bude prospěšné.

Grafické uživatelské rozhraní

Uživatelské rozhraní (obrázek 2) jsme navrhli, tak aby uživatel mohl co nejsnáze dostat kvadriky, které chce renderovat. První věc, kterou naše GUI řeší jsou rotace a vzdálenost kamery, tu řeší pomocí tří sliderů, které nastavují rotaci nahoru-dolů, doleva-doprava a poté vzdálenost od kvadriky. Dále si uživatel může vybrat typ kvadriky pomocí osmi radio tlačítek. Pomocí dalších 4 sliderů, je možné nastavit parametry kvadriky a, b, c a poloměr bounding boxu. Následně si pomocí check boxů nastaví, zdali chce použít osvětlení nebo některou z procedurálních textur. Tlačítkem 'Render' uživatel vyrenderuje kvadriku podle nastavených parametrů, tlačítkem 'Sample' uživatel vyrenderuje kvadriku podobnou té na Wikipedii (obrázek 1) a tlačítkem 'Quit' uživatel ukončí aplikaci.



Obrázek 2: Uživatelské rozhraní, kterým je možné nastavit konkrétní druh renderované kvadriky.

Zvláštní použité znalosti

Na přednáškách byly probírány základní techniky a principy ray tracingu. Tyto byly vysvětleny na jednoduchých primitivech jako jsou roviny, koule nebo válce. V tomto projektu však bylo nutné nastudovat problematiku ray tracingu kvadrik, které představují nadmnožinu již probíraných primitiv (například koule je speciální případ elipsoidu). Níže jsou popsány dvě části ray tracingu kvadrik.

Hledání průsečníku paprsku s kvadrikou

Všechny reálné kvadriky je možné zapsat jedním ze dvou tvarů rovnic [1]:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \epsilon_1 \frac{z^2}{c^2} = \epsilon_2$$

$$\frac{x^2}{a^2} + \epsilon_3 \frac{y^2}{b^2} + \epsilon_4 z = \epsilon_5$$

přičemž ϵ_i je -1, 0 nebo 1.

Tento fakt je možné využít a pomocí ϵ_i parametrizovat řešení. Není tedy nutné vytvářet řešení pro každý z 8 podporovaných typů kvadrik zvlášť. Začneme prvním tvarem rovnice. Po jednoduché úpravě získáme:

$$x^2b^2c^2 + y^2a^2c^2 + \epsilon_1 z^2a^2b^2 = \epsilon_2 a^2b^2c^2$$

Při ray tracingu poznáme počátek kvadriky S, počátek paprsku R a směr paprsku $\vec{v}$:

$$S = (x_S, y_S, z_S)$$

$$R = (x_R, y_R, z_R)$$

$$\vec{v} = (x_{\vec{v}}, y_{\vec{v}}, z_{\vec{v}})$$

Hledaný bod (x, y, z) z rovnice kvadriky je průsečníkem dané kvadriky s paprskem. Po dosazení tedy získáme rovnici:

$$(x_R + tx_{\vec{v}} - x_S)^2b^2c^2 + (y_R + ty_{\vec{v}} - y_S)^2a^2c^2 + \epsilon_1(z_R + tz_{\vec{v}} - z_S)^2a^2b^2 - \epsilon_2 a^2b^2c^2 = 0$$

kde jedinou neznámou proměnnou je parametr t. Po roznásobení a upravení získáme tuto kvadratickou rovnici:

$$(x_{\vec{v}}^2b^2c^2 + y_{\vec{v}}^2a^2c^2 + \epsilon_1 z_{\vec{v}}^2a^2b^2)t^2 + ((x_R - x_S)x_{\vec{v}}b^2c^2 + (y_R - y_S)y_{\vec{v}}a^2c^2 + \epsilon_1(z_R - z_S)z_{\vec{v}}a^2b^2)2t$$
$$+ (x_R^2 + x_S^2 - 2x_Rx_S)b^2c^2 + (y_R^2 + y_S^2 - 2y_Ry_S)a^2c^2 + \epsilon_1(z_R^2 + z_S^2 - 2z_Rz_S)a^2b^2 - \epsilon_2 a^2b^2c^2$$

Tady už je potřeba jen spočítat diskriminant a získat kořeny (pokud existují). Je důležité vybrat ten správný kořen – bude to ten menší z nich, hledaný bod ovšem musí ležet před kamerou a tedy kořen musí být kladný. Podobně jsme postupovali i při odvozování rovnic pro druhý tvar rovnice.

Určení normálového vektoru v bodě průniku

Pokud chceme kvadriky zobrazovat také s osvětlením, je nutné pro každý bod který jsme získali v předešlé kapitole spočítat normálový vektor. K tomu využijeme dvě parciální derivace – získáme tím dva vektory, které jsou v daném bodě tečné ke kvadrice. Vektorovým součinem těchto dvou vektorů pak získáme normálový vektor. Ukážeme si to na kvadrice s rovnicí druhého tvaru. Jednoduchou úpravou získáme:

$$z = \frac{\epsilon_5}{\epsilon_4} - \frac{x^2}{a^2\epsilon_4} - \frac{y^2\epsilon_3}{b^2\epsilon_4}$$

Následně spočítáme dvě parciální derivace:

$$\frac{\partial z}{\partial x} = -\frac{2}{a^2\epsilon_4}x$$

$$\frac{\partial z}{\partial y} = -\frac{2\epsilon_3}{b^2\epsilon_4}y$$

Normálový vektor pak získáme vektorovým součinem dvou vektorů určených těmito derivacemi[1]:

$$\vec{n} = (1, 0, -\frac{2}{a^2\epsilon_4}x) \times (0, 1, -\frac{2\epsilon_3}{b^2\epsilon_4}y)$$

Určení normálového vektoru u kvadriky s rovnicí prvního tvaru je o něco komplikovanější, protože je nutno parciálně derivovat složenou funkci, další postup je obdobný.

[1] Do x a y dosadíme souřadnice z bodu průniku

Práce na projektu

Rozdělení práce v týmu

Martin Ivančoudělal prvotní průzkum týkající se existujících řešení ray tracerů a vybral vhodný ke zpracování projektu. K němu implementoval modul, který dokáže renderovat kvadriky. Petr Buchal udělal průzkum týkající se knihoven GUI pro c++, vhodnou vybral a následně v ní implementoval GUI. Jak na propojení ray traceru s GUI, tak na psaní dokumentace jsme se podíleli oba.

Co bylo nejpracnější

Mezi náročnější části projektu určitě patřilo porozumění matematice kvadrik a její následná implementace v ray traceru. Náročné bylo odladění osvětlení, které způsobovalo občasné díry v kvadrikách. Problémy zprvu rovněž způsoboval nefungující překlad v operačním systému Windows. Při propojování ray traceru s GUI způsobila zdržení nevědomost, že API CVUI nad OpenCV nepodporuje RGBA.

Zkušenosti získané řešením projektu

Porozuměli jsme matematice kvadrik. Seznámili jsme se s podrobnějším fungování ray traceru. Seznámili jsme se s možnostmi knihoven GUI pro jazyk c++, podrobně jsme se seznámili s API CVUI nad OpenCV. Vyzkoušeli jsme si práci v týmu, kde každý dělal na různých věcech, které se na konci měli integrovat.

Autoevaluace

Technický návrh (85%): Oba členové týmu provedli průzkum v oblasti, kterou měli na starost a dále navrhli řešení se kterým byli oba spokojení.

Programování (85%): Jak uživatelské prostředí, tak především samotný ray tracer je jednoduše rozšiřitelný. Aplikaci jsme otestovali a nezaznamenali jsme žádné chyby. Čitelnost kódu GUI by nicméně mohla být o něco lepší.

Vzhled vytvořeného řešení (95%): Program kombinuje intuitivní použitelnost s komplexním nastavením jednotlivých parametrů renderování kvadrik. Se vzhledem GUI jsme oba dva spokojení, s renderováním kvadrik taktéž.

Využití zdrojů (90%): Čerpali jsme z přednášek předmětu PGR, různých online zdrojů a dokumentace použitých knihoven.

Hospodaření s časem (65%): Projekt se začal řešit měsíc před odevzdáváním, nicméně vzhledem k náročnosti ostatních předmětů a neúprosnosti termínů, jsme hlavní část projektu dodělávali poslední týden před odevzdáním.

Spolupráce v týmu (80%): Vždy jsme se dohodli na práci, kterou kdo bude dělat a podle toho byla práce udělána.

Celkový dojem (90%): Jsme spokojení s výsledným programem. Byl to rozhodně jeden ze zajímavějších projektů v tomto semestru a máme pocit, že jsme si z něj odnesli cenné zkušenosti na poli ray tracingu.

Ovládání vytvořeného programu

Technologie potřebné pro spuštění programu

- Standartní knihovny jazyka c++
- Knihovna OpenCV (https://github.com/opencv/opencv)
- Nástroj CMake

Programy a služby použité při tvorbě

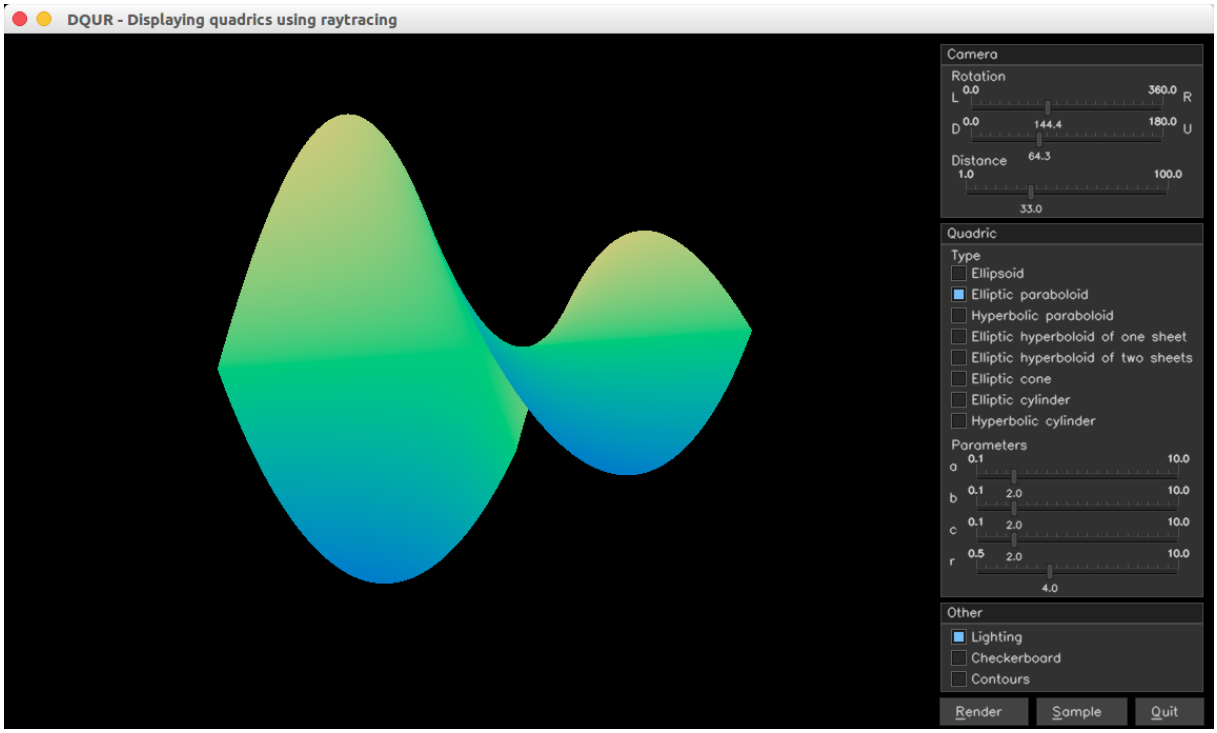
- Editor Sublime text 3
- Visual Studio Code
- Nástroj CMake
- Existující ray tracer simple-raytracer (https://github.com/cem/simple-raytracer)
- API CVUI nad knihovnou OpenCV (https://github.com/Dovyski/cvui)

Obsluha programu

Využití skriptu run.sh

- **./run.sh** - přeloží zdrojový kód a spustí program
- **./run.sh clean** - smaže přeložený zdrojový kód
- **./run.sh rebuild** - smaže přeložený zdrojový kód a přeloží jej znovu

Na obrázku 3 je GUI našeho ray traceru. V levé části se nachází renderovaná kvadrika, v pravé části se nachází nastavení parametrů renderování. Vlevo nahoře se nachází okno ve kterém je možné nastavit rotaci a vzdálenost kamery od kvadriky. Pod ním se nachází okno s nastavením typu kvadriky a jejích parametrů. Dále se zde nachází checkboxy pro zapnutí osvětlení a dvou textur, které je možné na kvadriku nanést. Tlačítko 'Render' vyrenderuje kvadriku podle nastavených hodnot, tlačítko 'Sample' vyrenderuje kvadriku podobnou té na Wikipedii a tlačítko 'Quit' ukončí program.



Obrázek 3: GUI ray traceru.

Doporučení pro budoucí zadávání projektů

K organizaci projektu nemáme žádné výhrady.

Literatura

[1] Silvio Levy. Quadrics. http://www.geom.uiuc.edu/docs/reference/CRC-formulas/node61.html, 1995. [Online; navštíveno 12.12.2018].

[2] Wikipedia. Quadric — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Quadric&oldid=859928276, 2018. [Online; navštíveno 12.12.2018].