VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

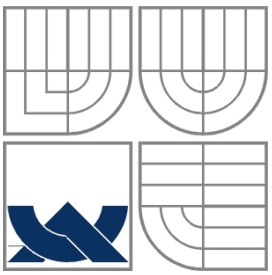
VÝUKOVÝ PROGRAM PRO DEMONSTRACI METOD OSVĚTLENÍ A STÍNOVÁNÍ 3D OBJEKTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

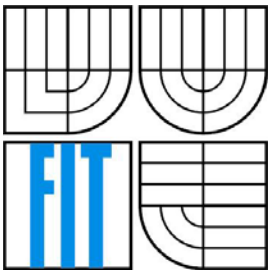
AUTOR PRÁCE
AUTHOR

VÍT CHVÁL

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÝUKOVÝ PROGRAM PRO DEMONSTRACI METOD OSVĚTLENÍ A STÍNOVÁNÍ 3D OBJEKTŮ

EDUCATION COMPUTER PROGRAM FOR DEMONSTRATION METHODS OF 3D OBJECTS ILLUMINATION AND SHADING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

VÍT CHVÁL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ ZUZAŇÁK

BRNO 2008

Abstrakt

Tato práce popisuje tvorbu výukového programu pro demonstraci metod osvětlení a stínování 3D objektů. V textu jsou podrobně popsány modely osvětlení. Těmi jsou Phongův a Lambertův, jako zástupci empirických modelů a BRDF jako zástupce fyzikálního modelu. Další část textu popisuje metody stínování. Je zde zmíněna konstantní, Gouraudova a Phongova metoda. Velká část textu také pojednává o tvorbě výukových programů. Zbytek textu je věnován návrhu aplikace a její implementaci.

Klíčová slova

fyzikální popis, empirický popis, Lambertův model, Phongův model, konstantní metoda, Gouraudova metoda, Phongova metoda, OpenGL, WxWidgets, BRDF, ray-tracing, radiozita, realtime, zrcadlová složka, ambientní složka, difúzní složka, RGB, normála, intenzita záření

Abstract

This work describes an Education Computer Program for Demonstration Methods of 3D Objects Illumination and Shading creation. There are in detail described lighting models in this text. There are Phong and Lambert models as representatives of empiric models and BRDF as representative of physical model. Next part of the text describes types of shading methods. You can find there constant, Gouraud and Phong method. Following part of the text discuss issues of Educational applications. The rest of the text is devoted to design and implementation of the application.

Keywords

Physical description, empiric description, Lambert model, Phong model, constant method, Gouraud method, Phong method, OpenGL, WxWidgets, BRDF, ray-tracing, radiosity, realtime, specular element, ambient element, diffuse element, RGB, normal, emission intesity

.

Citace

Chvál Vít: Výukový program pro demonstraci metod osvětlení a stínování 3D objektů. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Výukový program pro demonstraci metod osvětlení a stínování 3D objektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Zuzaňáka
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Děkuji tímto svému vedoucímu panu Ing. Jiřímu Zuzaňákovi za jeho odborné vedení, rady a pomoc při řešení této bakalářské práce.

Obsah

Úvod

V současné době, kdy na poli grafického hardwaru a také softwaru došlo k nebývalému rozmachu, se objevuje velký zájem o studium této oblasti. Vzniká tedy velká spousta literatury, nejrůznějších tutoriálů a demonstračních programů, které se pokouší jejich uživatelům studium 3D grafiky co nejvíce ulehčit. Většina těchto zdrojů informací se ovšem soustředí pouze na teoretickou část problematiky a málokdy uživateli dokáže v daném okamžiku předvést, jak dané řešení problému vypadá v praxi. Tento nešvar výukových textů a některých programů jsem se snažil v mé bakalářské práci, vytvořením demonstračního programu, odstranit.

Cílem mého snažení je tedy výukový program pro demonstraci metod osvětlení a stínování 3D objektů. Aplikace, která má vzniknout, by měla uživateli nenásilnou formou vysvětlit, jak pracují jednotlivé modely osvětlení a stínování, měla by uživateli předložit teoretickou část v textové, přehledné formě. Aplikace musí obsahovat praktickou část výuky. Tato část bude mít za úkol uživateli vizuální formou předvést to, co popsala část teoretická. Jako každý správný výukový program by i tento měl obsahovat test vědomostí.

První kapitola popisuje teorii modelu osvětlení, jeho rozdělení na fyzikální a zjednodušené empirické popisy. Podrobněji se zde rozepisuji o vlastnostech jednotlivých známých osvětlovacích modelů, jejich použití v praxi a o jejich výhodách i nevýhodách.

Druhá kapitola se zabývá teorií stínování ve 3D. Popisuji zde jednotlivé metody stínování a to postupně konstantní, Gouraudovo a Phongovo. Opět se zabývám praktickým použitím těchto metod, jejich výhodami a nevýhodami.

Třetí kapitola je věnována knihovně OpenGL a toolkitu WxWidgets. Jejich významu, funkcím a důvodu použití právě v mé implementaci.

Čtvrtá kapitola se věnuje problematice výukových programů. Uvádí správné postupy používané při výuce studenta a jejich praktické využití přímo v mé implementaci programu.

Šestá kapitola popisuje podrobněji implementaci mé bakalářské práce. V krátkosti zde uvádím nejdůležitější body své práce s odůvodněním jejich použití.

Sedmá kapitola je určena pro popis práce s programem.

1 Model osvětlení

Tato dlouhá teoretická kapitola se zabývá modelem osvětlení. Ten určuje jak povrch modelovaného tělesa ve 3D odráží dopadající světlo. Pro simulaci tohoto odrazu je nutné definovat povrch těles. Musíme tedy jasně určit jednotlivé vlastnosti materiálu, ze kterých jsou virtuální 3D objekty tvořeny. Těmito vlastnostmi může být hladkost/drsnost , lesklost/matnost a další.

Model osvětlení je určen odrazovou funkcí. Ta vyjadřuje intenzitu rozptýleného světla v závislosti na směru rozptylu, intenzitě a vlnové délce světla vycházejícího ze světelného zdroje.

Model osvětlení se, jak uvidíme později, rozděluje do dvou základních skupin. Těmi jsou fyzikální osvětlovací modely a empirické osvětlovací modely. Prozatím jen podotknu, že v realtime vykreslování se používají empirické modely, které jsou zjednodušením reality.

V následujícím úseku kapitoly podrobně popíšu fyzikální a především empirické modely osvětlení, uvedu jejich použiti a samozřejmě podrobně popíšu zástupce těchto modelů

1.1 Fyzikální a empirické modely osvětlení

Jak jsem již zmínil dříve, osvětlovací modely se v základu rozdělují do dvou skupin. A to na modely fyzikální a empirické. V následujícím odstavci představím oba tyto modely, uvedu jejich zástupce a vyjmenuji výhody a nevýhody jejich použití.

1.1.1 Fyzikální osvětlovací model

Tento model popisuje realistické vlastnosti světla a jeho odrazu. Jde o přesný fyzikální popis reálného osvětlení a je výsledkem složitých rovnic. Výpočet těchto rovnic je ale časově velmi náročný a proto je nevhodný pro realtime vykreslování 3D scény. Tento model uvádím jen pro úplnost, jeho použití je pro výukový program osvětlení a stínovaní ve 3D velmi nevhodné. Zástupcem fyzikálního modelu osvětlení je BRDF.

BRDF

BRDF vychází z anglického názvu bidirectional reflectance distribution function, což je dvousměrová odrazová distribuční funkce. Jde o čtyř dimenzionální funkci, která definuje jak je světlo odráženo na neprůhledném povrchu tělesa.

Funkce pracuje se směrem dopadajícího světla a se směrem světla odraženého, obě jsou definovány s ohledem na povrchové normály, a vrací podíl odraženého záření a ozáření povrchu.

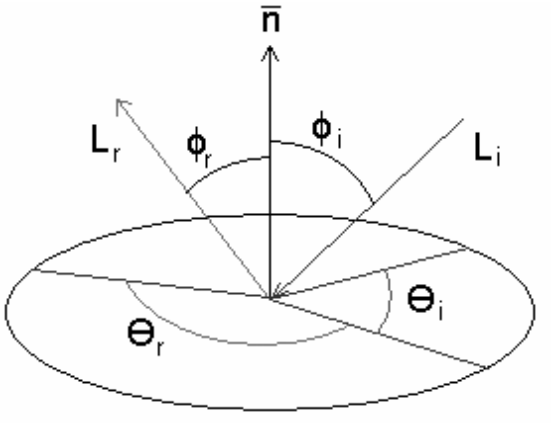
Vzorec pro výpočet BRDF je následující:

$$f(P,\vec{\omega}_i,\vec{\omega}_r) = \frac{L_i(P,\vec{\omega}_i)}{L_r(P,\vec{\omega}_r).\cos\Phi_i.d\omega_i}$$

$L_i(P,\vec{\omega}_i)$ - ozáření

$L_r(P,\vec{\omega}_r)$ - odražená zářivost

$L_e(P,\vec{\omega})$ - vlastní zářivost



Obrázek 1.1: Zářivost BRDF

BRDF se používá například pro zobrazovací rovnici radiozity. Radiozita je metoda globální iluminace scény. Obsáhnout téma radiozity na několika řádcích by bylo velmi obtížné. A proto jen zmíním, že se jedná o metodu vycházející ze zákonu zachování energie. Neumí pracovat s průhlednými objekty. Každou scénu před zpracování rozdělí na malé plošky a spočítá jejich vzájemné vlivy. Výpočet radiozity se provádí v několika iteračních krocích. Velkou výhodou této metody je, že se scéna nemusí přepočítávat při každé změně polohy kamery. Pro zobrazení výsledků radiozity můžeme použít metodu ray-tracing. Tímto způsobem se do scény přidají zrcadlové odrazy objektů a vlastnosti povrchů.

Shrnutí vlastností BRDF:

- Výpočet pomocí BRDF dopomáhá k reálnému vzhledu vykreslovaných 3D scén.
- Výpočet BRDF je velmi složitý, průběh je časově náročný a proto je nevhodný pro realtime vykreslování.
- BRDF se používá tam, kde je potřeba se přiblížit fotorealismu.

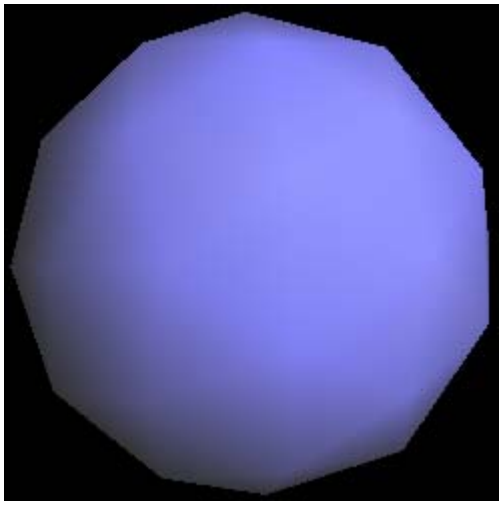
1.1.2 Empirický osvětlovací model

Empirický osvětlovací model je model osvětlení, jehož výpočty byly upraveny s ohledem na rychlost jejich zpracování. Empirické modely se běžně využívají v realtime zobrazení 3D scény, protože jsou na rozdíl od fyzikálních modelů osvětlení schopny rychle zpracovat celou scénu a uložit ji do bufferu grafické karty.

Mezi empirické modely osvětlení patří Lambertův a Phongův osvětlovací model.

V následující části teoreticky popíšu tyto dva modely, přičemž budu věnovat větší pozornost Phongovu osvětlovacímu modelu, neboť jej využívám v implementaci svého výukového programu.

Lambertův osvětlovací model



Obrázek 1.2: Ukázka Lambertova osvětlovacího modelu

Jde o empirický model. Využívá difúzní odraz do všech směrů konstantně. Intenzita difůze závisí na úhlu dopadu světla na povrch. Lambertův model osvětlení je velmi rychlý a nenáročný na výpočet. Často se proto používá pro jednoduché vykreslování objektů ve 3D aplikacích.

Tento model ale nezvládá odlesky vznikající při dopadu světla. Tato vlastnost znemožňuje jeho použití v těch aplikacích, které se snaží vykreslenou scénu co nejvíce přiblížit realitě. Z tohoto důvodu vznikl Phongův osvětlovací model, který vychází z Lambertova a přidává k němu odlesk, neboli reflexi.

Pro tento osvětlovací model platí Lambertovo pravidlo:

$$I' = R.I.\cos(\alpha) + R(1 - I)$$

- Kde I je z intervalu <0,1>a značí intenzitu světelného zdroje
- R je koeficient reflexe materiálu a omezen intervalem <0,1>
- α je úhel, který svírá normála plochy se světelným paprskem.

Tento vzorec je ovšem třeba uplatnit pro každou složku RGB (červená, zelená, modrá).

Toto pravidlo ale platí pouze teoreticky. Proto existuje vztah, jehož součástí není jen intenzita světelného zdroje, reflexe a sklon osvětlované plochy, ale také intenzita okolního světla, optické vlastnosti osvětlené plochy, úhel pozorovatele a vlastnost povrchu.
Tímto nám vzniká vztah:

$$I = K_a \,.\, I_a + K_d \,.\, I_d + K_r \,.\, I_r + K_t \,.\, I_t$$

- Kde je I_a intenzita okolního světla
- I_d je intenzitou difúzního světla
- I_r je intenzitou reflexního světla
- I_t je transparentní intenzita
- K_d je koeficient, který určuje množství odraženého rozptýleného světla
- a konečně K_a je koeficientem určujícím vliv okolního světla

Složka Id je v našem vztahu velmi důležitá, udává totiž podstatnou část intenzity osvětlené plochy. Intenzita difúzního světla je dána vztahem:

$$I_d = \frac{K_d (NL)}{d + d_0} I$$

- Kde je I intenzitou světelného zdroje
- d_0 je minimální vzdálenost. Ta se udává jako ochrana proti dělení nulou.
- d je vzdálenost zdroje světla od dané plochy
- a K_d je koeficientem difúzního odrazu. Ten je určen vlastnostmi povrchu

Vlastnosti povrchu jsou určeny následujícím vztahem.

$$I_r = \frac{K_r}{d + d_0} I.\cos\varphi$$

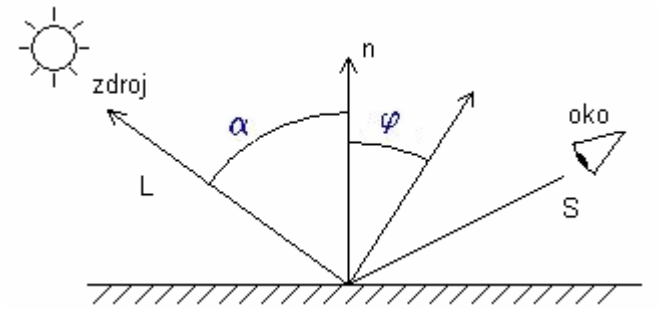
- V tomto vztahu určuje φ úhel, který svírá oko pozorovatele (viz obrázek)
- K_r udává reflexní koeficient.

Pokračujeme určením intenzity I_t. Transparentní intenzita je tak, jak již název napovídá, určena průsvitností osvětlovaného materiálu.
Pro I_t existuje následující vztah:

$$I_t = \frac{K_t (1 - t).I}{d + d_0}$$

Parametr t zde charakterizuje transparentnost. Transparentnost je závislá na vlnové délce světla a nepřímo úměrná průsvitnosti povrchu.

Potřebné úhly jsou k vidění na následujícím obrázku.

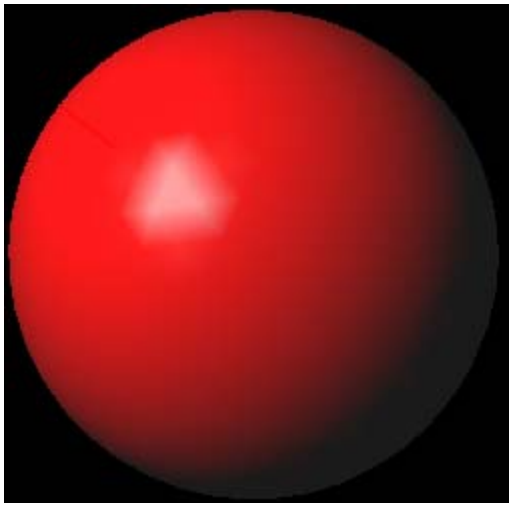


Obrázek 1.3: Vektor tělesa potřebný pro výpočet pomocí Lambertova modelu

Shrnutí vlastností Lambertova osvětlovacího modelu:

- Algoritmus založený na tomto osvětlovacím modelu je velmi rychlý.
- Model nezvládá reflexi objektů.
- Využití modelu pro jednoduché scény u kterých není kladen důraz na realističnost.

Phongův osvětlovací model



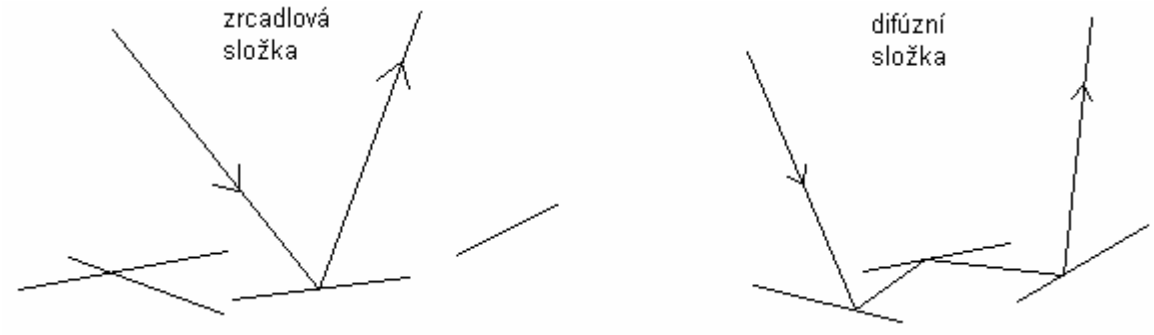
Obrázek 1.4: Ukázka Phongova osvětlovacího modelu

Jde stejně jako u Lambertova modelu o empirický osvětlovací model. Přidává k Lambertovu modelu reflexi. Reflexe je ve směru odrazu symetrická podle normály.

Phongův model osvětlení je v realtime vykreslování nejpoužívanější variantou empirických modelů. Umožňuje totiž rychlé vykreslování realisticky vypadajících scén a to díky tomu, že zvládá pracovat s reflexemi materiálu jednotlivých objektů ve scéně. Tento model osvětlení je použit i v implementaci grafického hardware a využíván knihovnou OpenGL.

Právě z důvodu použití v knihovně OpenGL, je tento modle důležitou součástí výkladu, týkajícího se empirických modelů osvětlení a proto se mu budu věnovat podrobněji.

U Phongova osvětlovacího modelu se dopadající světlo na povrch rozkládá do tří světelných složek. Těmito složkami jsou ambientní, difúzní a složka odlesku. Složku odlesku a difúzní můžete vidět na následujícím obrázku.



Obrázek 1.5: Ukázka zrcadlové a difúzní složky

V Phongově modelu se celková intenzita světla počítá zvlášť pro každou barevnou složku, tedy RGB(červená, zelená, modrá). Vztah uvedený dále se aplikuje na každou barevnou složku RGB zvlášť:

$$I = c_a I_a + c_d I_d (NL) + c_s I_s (VR)^n$$

Tuto rovnici tedy ve skutečnosti počítáme třikrát a to pro $I_{červená}$, $I_{zelená}$, $I_{modrá}$.

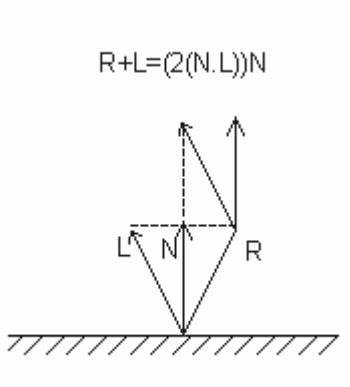
V následující části je uveden podrobný popis všech členů, které se v tomto vztahu nalézají.

- I udává celkovou intenzitu světla. Jak jsem již uvedl, tato hodnota je počítána třikrát, pokaždé pro jednu z barev (červená, zelná a modrá).
- I_a udává intenzitu ambientní složky světla. Ambientní složka světla dopadá na těleso ze všech směrů rovnoměrně. Součinem $c_a.I_a$ je intenzita odraženého ambientního světla. Tato intenzita je nezávislá na poloze zdroje světla, tělesa a poloze pozorovatele.
- I_d je intenzitou difúzní složky světla. Difúzní složku světla popisuje předešlý obrázek. Je to ta část světla, která na těleso dopadá ze světelného zdroje a odráží se do všech směrů rovnoměrně. Součin cd.Id(N.L) v tomto případě udává intenzitu odraženého difúzního světla a je závislá na poloze normál stěn daného tělesa a vektoru dopadu světla L. Hodnota tohoto členu je nastavena při vytvoření zdroje světla.
- I_s je intenzitou světelných odlesků. To je část světla, která dopadá na těleso z určitého světelného zdroje a odráží se v jednom směru podle fyzikálního zákonu o odrazu světelných paprsků. Opět je intenzita závislá na poloze světelného zdroje, tělesa a pozorovatele.

  Hodnota se nastavuje již při vytváření světelného zdroje.
- c_a je koeficientem materiálu ambientní složky světla. Běžná hodnota tohoto koeficientu je v intervalu 0-1. Pokud je hodnota nulová, ambientní světlo se od daného tělesa vůbec

neodráží. Naopak hodnota 1 nastavuje tělesu maximální odrazivost. Hodnota se pro předchozí rovnici nastavuje při definování materiálu.

- c_s udává koeficient odlesků. Každý z koeficientů je nutné nastavit pro každou barevnou složku RGB zvlášť. Zadaná hodnota pro každý materiál pak udává jeho barvu při osvitu bílým světlem. Pro světla jiných barev je nutné mezi sebou pronásobit intenzity jednotlivých složek světelného zdroje a koeficienty použitých materiálů.
- N.L je součinem normály určené k povrchu tělesa N a vektoru světelného zdroje do bodu na povrchu tělesa. Je nutné vědět, že před výpočtem skalárního součinu N.L je třeba tyto vektory převést na vektory normalizované.
- Součin V.R využívá vektoru odraženého paprsku R a vektoru z pozice pozorovatele do bodu na povrchu tělesa V. Na následujícím obrázku je vidět, jak vypočítat vektor R ze znalosti vektorů L a N, tedy vektoru světla a vektoru normály.



Obrázek 1.6: Výpočet vektoru R

- Posledním dosud neznámým členem je n. To je exponent, který určuje míru lesklosti tělesa. Pokud chceme vytvořit například vysoce lesklé těleso, zadáme hodnotu 50. Těleso matné bude mít nulovou hodnotu exponentu.

Shrnutí vlastností Phongova osvětlovacího modelu:

- Algoritmus je schopen počítat s odlesky vykreslovaných těles.
- Model je pomalejší než Lambertův model, ale dokáže vykreslit realističtěji vypadající scény
- Model lze využít pro vykreslování v OpenGL.

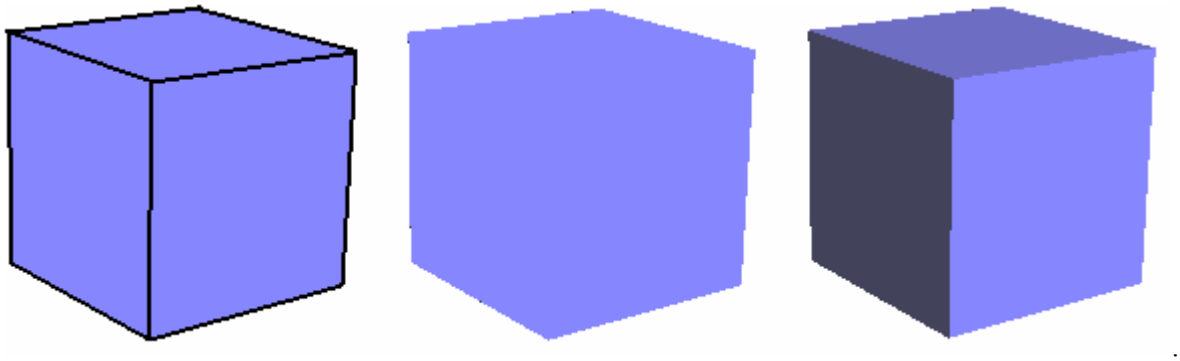
To je tedy vše k Phongovu modelu osvětlení. K tomuto modelu se ovšem ještě vrátíme v kapitole pojednávající o grafické knihovně OpenGL a to z toho důvodu, že jej tato knihovna používá pro vykreslování 3D scén.

Následující kapitola nazvaná Stínování, pojednává o jednotlivých modelech stínování, jejich výhodách, nevýhodách a jejich uplatnění v praxi.

2 Stínování

Jinak též shading, se určuje barvy pixelů obrazu 3D scény. Dodává plastičnost objektům zobrazované scény. Barva se určí jen pro několik bodů pomocí jednoho z osvětlovacích modelů, ostatní body nabývají barev určených pomocí metod stínování.

Na následujícím obrázku vidíme, jak napomáhá stínování reálnému vzhledu dodáním plastičnosti.



Obrázek 2.1: Ukázka těles bez stínování a s použitím stínování

Celá tato kapitola pojednává o jednotlivých metodách stínování. Postupně si teoreticky popíšeme nejjednodušší metodu stínování – konstantní, poté postoupíme dále k početně náročnější metodě – gouraudově a výklad zakončíme metodou nejsložitější – Phongovou. Každá z těchto metod má své uplatnění v praxi, každá z nich má své výhody i nevýhody. V žádném případě nemůžeme jasně určit, která z těchto metod je lepší nebo horší. V následujícím textu však vhodnost použití daného typu těchto metod uvedu.

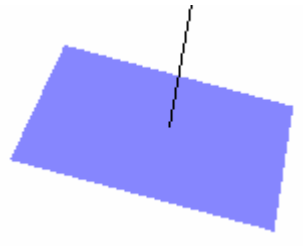
Nyní tedy přistoupíme k první a nejjednodušší metodě.

2.1 Konstantní metoda stínování

Obrázek 2.2: Ukázka konstantní metody stínování

Konstantní metoda stínování je tou nejjednodušší a zároveň nejrychlejší metodou stínováni. Tato metoda se využívá pro stínování rovinných ploch. Metoda je nejvhodnější v případě, pokud chceme velmi rychle vykreslit scénu a nezáleží nám na výsledném vzhledu scény.

Algoritmus funguje na předpokladu, že má každá plocha pouze jednu normálu tak, jak to vidíme na následujícím obrázku.



Obrázek 2.3: Normála použitá při konstantní metodě stínování.

Z obrázku je jasné, že algoritmus počítá s jedním vektorem normály vycházející ze středu plochy. Výsledná barva je poté přiřazena všem pixelům dané plochy.

Velkou nevýhodou tohoto postupu je právě přiřazování jedné barvy celému povrchu dané plochy. Další nevýhodou je nemožnost zobrazení jakýchkoliv zrcadlových odlesků. Nevýhody pak jen doplňuje nevhodnost použití pro obecná tělesa, kde toto stínování ukazuje, že je celé těleso sestaveno z určitého počtu polygonů. Naopak výhodou je rychlost tohoto algoritmu.

V následující části bude v krátkosti popsán algoritmus konstantní metody stínování.

2.1.1 Algoritmus konstantního stínování

1. Vypočítáme normálové vektory pro všechny polygony, ze kterých je daný objekt složen.
2. Pro každý polygon je určena odchylka normálového vektoru a vektoru dopadajících paprsků světelného zdroje. Připomeňme, že čím menší úhel svírá paprsek dopadajícího světla s normálou vycházející s polygonu, tím více je daný polygon osvícen.
3. Podle výsledné odchylky určíme barvu a touto barvou daný polygon vyplníme.

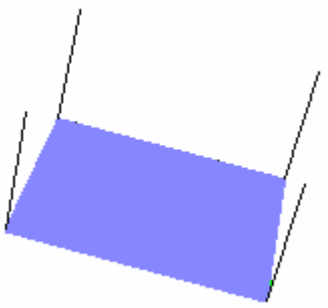
Shrnutí vlastností konstantní metody stínování:

- Algoritmus je schopen pracovat velmi rychle.
- Metoda nedokáže vykreslit vyhlazený povrch objektu.
- Metoda nedokáže pracovat s odlesky.

2.2 Gouraudova metoda stínování

Obrázek 2.4: Ukázka Gouraudovi metody stínování

Metoda vynalezena H. Gouraudem je určena pro výpočet stínování objektů tvořených množinou rovinných ploch. Tato metoda patří mezi pokročilejší metody stínování. Metoda vychází z toho, že známe vektory v každém z vrcholů daného tělesa tak, jak to vidíme na následujícím obrázku.



Obrázek 2.5: Potřebné normály pro Gouraudova metodu stínování

Pro každý z těchto vrcholů určíme barevný odstín RGB a vzájemně je barevně propojíme pomocí interpolace. Tímto úkonem obarvíme jednotlivé hrany daného polygonu. Interpolací pak jednoduše obarvíme celou plochu daného polygonu.

Výhodou tohoto postupu je plynulé stínování zakřivených povrchů tvořených množinou polygonů. Aproximace povrchů polygony pak není viditelná. I přes užití interpolace barev jednotlivých vrcholů však není tato metoda dokonalá. Algoritmus totiž neumožňuje místní změnu jasu na polygonu. Algoritmus taktéž neumožňuje vznik realistických odlesků.

Nyní si v několika krátkých krocích popíšeme postup algoritmu Gouraudova stínování.

2.2.1 Algoritmus Gouraudova stínování

1. Pro všechny polygony, ze kterých je zvolený objekt vytvořen, vypočítáme normálové vektory.
2. Pro každý vrchol spočítáme vlastní normálový vektor. Ten určíme zprůměrňováním normálových vektorů polygonů, které se v daném vrcholu stýkají. Tento vektor nakonec umístíme do zvoleného vrcholu.
3. V dalším kroku určíme odchylku normálových vektorů ve vrcholech a vektoru paprsků světelného zdroje. Stejně jako u konstantní metody stínování platí, že čím menší je úhel svíraný vektorem dopadu a normálou vrcholu, tím větší intenzita záření je na daný vrchol uplatněna.
4. Výslednou odchylkou určíme barvu ve vrcholech.
5. Dále provádíme interpolaci barev mezi jednotlivými vrcholy polygonu. Poté pomocí interpolace obarvíme celou plochu zvoleného polygonu.

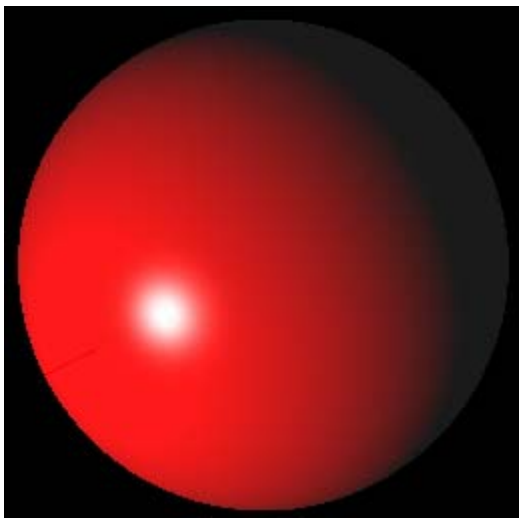
Algoritmus gouraudova stínování je sice znatelně pomalejší než algoritmus používaný u konstantní metody stínování, ale zvládá zobrazit hladce vykreslené objekty. Tato metoda ale nedokáže správně pracovat s odlesky světla vznikajících na povrchu osvětleného tělesa.

Metoda Gouraudova stínování je většinou využívána pro realtime vykreslování. Také grafická knihovna OpenGL s tímto modelem v základu pracuje. I já budu tuto metodu využívat při své práci na výukovém programu pro demonstraci metod osvětlení a stínování 3D objektů.

Shrnutí vlastností Gouraudovy metody stínování:

- Algoritmus je schopen pracovat velmi rychle, přesto je pomalejší než konstantní metoda.
- Metoda dokáže vykreslit vyhlazený povrch objektu.
- Metoda dokáže pracovat s odlesky, ale nedokáže je vykreslit v reálné podobě.

2.3 Phongova metoda stínování



Obrázek 2.6: Ukázka Phongovy metody stínování

Phongova metoda stínování je nejsložitější a početně nejnáročnější metodou.Bui-Tuong Phong vytvořil metodu určenu ke spojitému stínování těles, jejichž povrch je tvořen množinou rovinných ploch.

Phongova metoda je v základu velmi podobná metodě Gouraudova stínování. Také pracuje s normálami v každém z vrcholů daného polygonu. Na rozdíl od Gourauda ale nevyužívá tyto normálové vektory k interpolaci barev, nýbrž k interpolaci normálových vektorů v ostatních bodech polygonu. Tímto určí vlastní normálu pro každý z bodů daného polygonu. Jednotlivé body využijí hodnotu své normály k výpočtu své vlastní barvy.

Phongovo stínování zvládá hladké stínování a na rozdíl od Gourauda dokáže vykreslit realistické odlesky vznikající na povrchu osvětleného tělesa.

Následující úsek textu vysvětlí jednotlivé kroky zmiňovaného algoritmu.

2.3.1 Algoritmus Phongova stínování

1. Pro všechny polygony, ze kterých je zvolený objekt vytvořen, vypočítáme normálové vektory.
2. Pro každý vrchol spočítáme vlastní normálový vektor. Ten určíme zprůměrováním normálových vektorů polygonů, které se v daném vrcholu stýkají. Tento vektor nakonec umístíme do zvoleného vrcholu.
3. Provedeme interpolaci normál pro jednotlivé body polygonu. Průběh je podobný jako u Gouraudova algoritmu stínování.
4. Nyní již máme normály pro každý bod v daném polygonu. Pro tyto normály určíme jejich odchylky od vektoru světelných paprsků dopadajících na stínované těleso.

5. Podle výsledku z předešlého kroku přiřadíme každému bodu příslušnou barvu.

Phongův algoritmus je nejpomalejším z uvedených algoritmů, dokáže ale vykreslit realistický vzhled 3D objektů. Tento algoritmus je možné používat v realtime zobrazení a to například po implementaci do OpenGL za pomoci Shading Language.

Shrnutí vlastností Phongovi metody stínování:

- Algoritmus je velmi náročný.
- Metoda dokáže vykreslit vyhlazený povrch objektu.
- Metoda dokáže pracovat s odlesky a vykreslit je v realistické formě.

2.3.2 Další metody stínování

Metody konstantní, Gouraudova a Phongova jsou nejpoužívanějšími v oblasti 3D grafiky, nejsou ale jediné. Nyní jen pro úplnost uvedu jako příklad další metodu osvětlení. Tuto metodu ale nebudu podrobně rozebírat.

Interpolované stínování (Interpolated Shading)

Tato technika je velmi podobná Gouraudově metodě stínování, která vychází právě z této metody. Metodu je možné uplatnit pouze na trojúhelníky. Stejně jakou u Gouraudovy metody se vypočítají normálové vektory pro každý z vrcholů trojúhelníku a barva mezi nimi je vypočítána interpolací.

3 OpenGL a WxWidgets

Tato kapitola v krátkosti popíše, co je to vlastně knihovna OpenGL a toolkit WxWidgets. Vysvětlí výběr právě této dvojice a popíše jejich použití

3.1 Knihovna OpenGL

Grafický systém OpenGL je softwarové rozhraní pro grafický hardware. OpenGL umožňuje vytvářet interaktivní aplikace sloužící k vizualizaci grafické informace uložené v počítači. OpenGL je vhodným nástrojem ke grafickému zobrazení jak realistických obrazů, tak i takových scén, které se od reality vzdalují imaginativními způsoby. To je důvod, proč byla tato grafická knihovna vybrána pro tvorbu výukového programu pro demonstraci metod osvětlení a stínování ve 3D. V mém programu nechci uživateli ukázat pouze realistické vykreslení scény, chci mu umožnit vykreslit vše, co dokáží základní grafické funkce dnešních počítačů. OpenGL je také vhodné pro výuku díky přehlednému zdrojovému kódu, který využívá. Tento kód je pak uživateli zobrazen pro lepší pochopení všech operací ve 3D.

3.1.1 Použití OpenGL ve výukovém programu

Ve svém programu jsem OpenGL využil k vykreslování celé scény, objektů a světel. OpenGL je věnována zvláštní část okna, ve které se realtime vykresluje scéna podle informací zadaných uživatelem z ovládací panelu. Podrobný popis funkcí OpenGL okna a OpenGL jako celku naleznete v kapitole věnované implementaci programu.

Grafická knihovna OpenGL a její případné rozšíření v podobě Shading Language zvládá prakticky jakýkoliv model osvětlení a stínování. Je dokonce možné zde uplatnit i některý z fyzikálních modelů jako BRDF. Obsáhnout všechny tyto možnosti by ale bylo v krátkém časovém úseku vymezeném na tvorbu bakalářské práce nemožné. Proto bylo třeba vybrat pouze některou součást, na kterou bych se jako programátor zaměřil. V OpenGL proto implementuji pouze základní příkazy týkající se modelu osvětlení, nastavení světelných podmínek a jednoduché volby mezi modelem stínování.

3.1.2 Osvětlovací model OpenGL

V OpenGL se pro výpočet barvy objektů ve scéně používá výše zmíněný Phongův osvětlovací model.

Tento model se snaží co nejvíce zjednodušit fyzikální a optické vlastnosti reálného světla tak, aby výpočty osvětlení byly rychlé a aby se výsledný obrázek dostatečně blížil realitě. V Phongově osvětlovacím modelu se světlo, které na určitý objekt dopadá a následně se od něj odráží, rozkládá na tři světelné složky.

- **Ambientní složka**

  Je složkou určující okolní světlo. Toto světlo není vyzařováno z nějakého určitého světelného zdroje. Světlo si můžeme představit jako světlo, které se několikrát odrazilo od smyšlených stěn místnosti, nelze tedy určit směr záření.

  Phongův osvětlovací model toto světlo považuje za všesměrové. Toto světlo tedy osvětluje objekt ze všech směrů nezávisle na poloze daného zdroje světla.

  Na následujícím obrázku vidíme ukázku působení tohoto světla.

Obrázek 3.1: těleso osvětlené ambientní složkou světla

Z obrázku jasně vidíme, že se objekt jeví jako plošný, ztrácí tedy svou prostorovost. Tu zajistí následující složka.

- **Difúzní složka**

  Difúzní složka vyjadřuje světlo, které na povrch daného tělesa dopadá ze světelného zdroje a od povrchu objektu se odráží do všech směrů se stejnou intenzitou. Difúzní složkou tedy můžeme vytvářet matné materiály.

  Výpočty Phongova osvětlovacího modelu jsou uzpůsobeny tak, že výsledná barva difúzní složky je závislá pouze na orientaci světelného zdroje a objektu. Barvu ani intenzitu difúzní složky neovlivňuje poloha pozorovatele. To značně urychluje průběh vykreslovacích algoritmů.

  Následující obrázek předvádí těleso nasvícené pouze difúzní složkou světla.

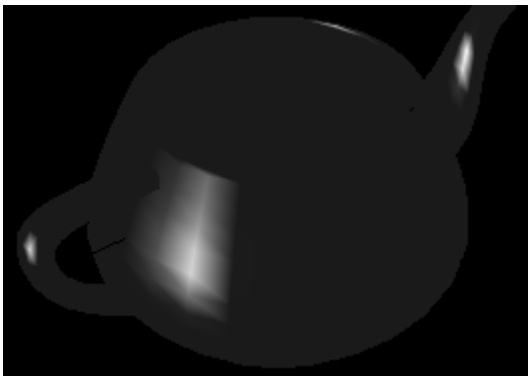
Obrázek 3.2: Těleso osvětlené pouze difúzní složkou světla

- **Specular složka**

  Specular složka udává odlesky tělesa. Ty vznikají v okamžiku dopadu světelného paprsku na povrch tělesa z určitého zdroje světla a jeho odrazu od povrchu, podle zákona odrazu a lomu světla.

  V reálných situacích se ale paprsek nikdy neodrazí ideálně, nastává jisté rozptýlení. Phongův osvětlovací model toto modeluje tak, že určí koeficienty změny intenzity odraženého světla podle úhlu odklonu počítaného a odraženého paprsku od paprsku ideálně odraženého. Součin těchto dvou vektorů se poté umocní zadaným koeficientem.

  Následující obrázek ukazuje těleso se specular složkou.



Obrázek 3.3: Těleso s odlesky

Pokud usilujeme o vykreslení reálných těles, téměř nikdy nepoužíváme jen jednu světlenou složku. U reálných situací složky kombinujeme. OpenGL nám toto umožňuje. Na následujícím obrázku vidíme příklad vykresleného objektu využívajícího všechny tři světelné složky - tedy difúzní, ambientní i odrazovou. Difúzní složka byla v tomto případě nastavena na jasně červenou. Ambientní složka je pro lepší viditelnost objektu nastavena na tmavě šedou (hodnotami 0.2 pro každou z RGB

složek). Specular složka (neboli odlesky) jsou nastaveny na reálné, tedy bílé. Toho je docíleno plným zastoupením všech složek RGB.

Obrázek 3.4: Objekt využívá ambientní, difúzní i specular složku.

3.2 Toolkit WxWidgets

WxWidgets je free software/open source multiplatformní widget toolkit. Je to knihovna základních elementů pro tvorbu grafického uživatelského rozhraní (GUI). WxWidgets umožňuje zkompilovat a spustit program na několika počítačových platformách s minimálními nebo žádnými změnami kódu. To zahrnuje systémy jako Windows, Macintosh, Linux/Unix a další. WxWidgets se tedy tímto stává nejvhodnějším pomocníkem pro tvorbu výukových programů, zvláště pokud náš produkt chceme rozšířit na více platforem. WxWidgets je také velmi vhodný pro svou jednoduchost a rychlost tvorby uživatelských rozhraní. Programátor není nucen dlouhé hodiny vytvářet nástroj, který již existuje v plně funkční verzi a raději se může soustředit na kvalitu výukového programu. Toto je jeden z mnoha důvodu, proč jsem si zvolil tento toolkit pro vytvoření své bakalářské práce.

3.2.1 Použití WxWidgets ve výukovém programu

Jak již bylo řečeno, WxWidgets využívá velmi jednoduchého způsobu tvorby uživatelských rozhraní. Proto jsem jako programátor mohl věnovat větší úsilí logické stavbě programu, než implementaci jednoduchých nástrojů, jako např. editor textu. Popis jednotlivých funkcí, které uživateli za pomoci tohoto toolkitu poskytuji, naleznete opět v následující kapitole.

4 Výukový program

V této části se budeme věnovat problematice tvorby výukových programů. Uvedeme, co by měl dobrý výukový program obsahovat, co by měl nabídnout uživateli a žákovi v jedné osobě, jak by měl látku vysvětlovat a jak by ji měl interaktivně prezentovat.

V následujícím odstavci tedy stručně shrnu, jaké jsou kladeny požadavky na kvalitní výukový program. Splnění všech těchto požadavků je však nemožné, proto také uvedu své vlastní pojetí výukového programu s ohledem na ideální řešení.

4.1 Ideální výukový program

Pro všechny výukové programy lze stanovit zhruba deset základních požadavků. Tyto požadavky si pro větší přehlednost rozdělíme na obecné a pedagogické požadavky.

4.1.1 Obecné požadavky

První čtyři uváděná kritéria jsou v podstatě platná pro všechny typy programů, nejen ty výukové.

- **Dokumentace**
  Každý dobrý program by měl být vybaven kvalitně provedenou dokumentací. Její nejdůležitější součástí je manuál, který krok za krokem popisuje, jak s programem pracovat.
- **Instalace**
  Kvalitní produkt by měl mít vždy vlastní instalační program.
- **Ovládání**
  Uživatelé by neměli být rozptylováni od učení s počítačem tím, že část svého času a úsilí budou muset věnovat snaze přijít na to, jak a z jakého místa jsou dostupné všechny funkce programu a jak zvládnout principy jeho ovládání.
- **Nápověda**
  V případě, že již dojde k situaci, kdy si uživatel opravdu neví rady (buď s ovládáním programu nebo s dalším postupem), přichází na řadu nápověda.

4.1.2 Pedagogické požadavky

Výukové programy musíme posuzovat i z hlediska pedagogicko – psychologického.

- **Výklad látky**
  Úkolem výkladové části výukového programu je zprostředkovat studentovi nové poznatky.
- **Simulace**
  Někdy bývá výkladová část výukového programu obohacena (kromě audio, video ukázek a obrázků týkajících se probíraného tématu), ještě o simulaci probíraných jevů.
- **Procvičování látky**
  Procvičovací část dobrého výukového programu by uživateli měla umožňovat časově neomezený pobyt v této části a volné přecházení do části výkladové a zpět.
- **Zpětná vazba**
  Dobrý výukový program by měl při předání učiva žákovi a následné kontrole získané úrovně znalostí, reagovat i podle výsledků zpětnovazební informace.
- **Testování**
  Součástí výukových programů bývá i testová část, ve které počítač zjišťuje míru získaných znalostí a dovedností žáka.
- **Grafika a zvuk**
  Součástí hodnocení výukových programů je i to, jakým způsobem působí na své uživatele. Toto z velké části ovlivňuje grafické zpracování programu i případná zvuková kulisa.

4.2 Výukový program v praxi

Absolutní splnění všech výše uvedených bodů je však pouze ideálem. Proto jsem se jako budoucí uživatel a tvůrce výukového programu snažil nalézt jistý kompromis. Podrobnějšímu popisu všech funkcí programu jsou věnovány následující kapitoly, nyní pouze ve zkratce uvedu body, které považuji za nejdůležitější a na které jsem se zaměřil.

Ovládaní mého výukového programu klade důraz na maximální přehlednost v příjemném prostředí. Uživatelské rozhraní je vyvedeno v jednobarevném tématu a to svěží modrou barvou. Každý ovládací prvek umožňuje spuštění nápovědy věnované pouze danému prvku. Po každé teoretické části následuje část praktická, ve které si uživatel v praxi vyzkouší všechny funkce, o kterých se v textu objevila zmínka. Po praktické části následuje test vědomostí, který svými otázkami shrnuje teoretickou i praktickou část výuky. Poslední důležitou záležitostí je přístup k nápovědě programu ze kterékoliv obrazovky.

5 Analýza a návrh aplikace

Tato kapitola pojednává o návrhu aplikace, charakterizuje program a popisuje použité vývojové prostředí. Součástí je i grafický návrh provázanosti jednotlivých oken a popis jednotlivých tříd použitých v programu.

5.1 Vývojové prostředky

Výukový program pro demonstraci metod osvětlení a stínování 3D byl napsán v objektově orientovaném jazyce C++. Tento jazyk však není čistě objektově orientovaný, protože je zpětně kompatibilní s procedurálním jazykem C.

Jak již bylo zmíněno dříve, pro program bylo využito toolkitu WxWidgets a to pro tvorbu uživatelského rozhraní.

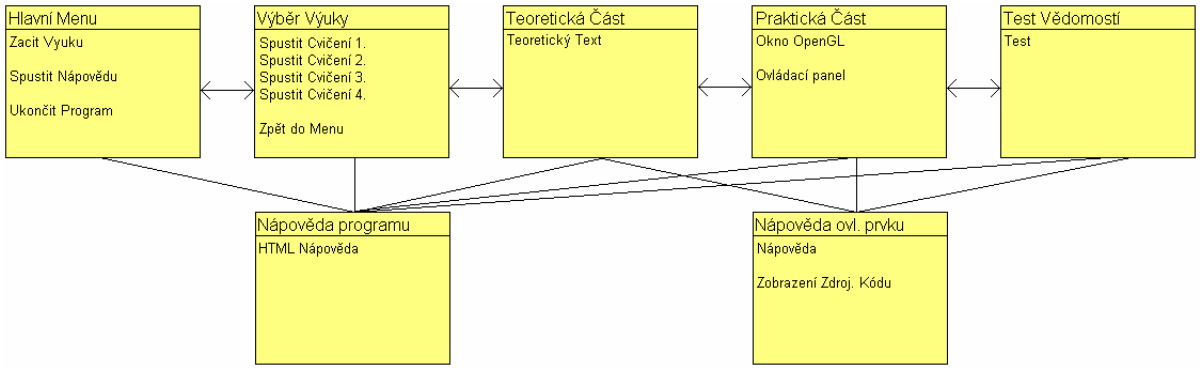
Realtime vykreslování v mém výukovém programu obstarává grafická knihovna OpenGL a její rozšíření GLUT, což je doplněk k této knihovně nazvaný OpenGL Utility Toolkit.

Program by vyvíjen v operačním systému Microsoft Windows XP a přeložen v programu Microsoft Visual Studio 2005.

Přestože byl program vyvíjen v operačním systému Windows, díky toolkitu WxWidgets a multiplatformní knihovně OpenGL, jej není problém přeložit i pro prostředí Linux.

5.2 Návrh aplikace

Před jakýmkoliv psaním zdrojového kódu jsem provedl analýzu zadaného problému a postupně naplánoval další kroky řešení. Na následujícím obrázku (6.1) jsem znázornil jednotlivá okna a jejich logické propojení. Obrázek je pouze zjednodušeným návrhem skutečného programu a jeho jednotlivé části budou popsány později. Je nutné mít na paměti, že okna cvičení jsou tvořeny čtyřikrát pro každý typ probírané látky (normály, světla, model osvětlení a materiál objektů).



Obrázek 5.1: Návrh aplikace

5.2.1 Charakteristika programu

Zadání bakalářské práce zní následovně: „Výukový program pro demonstraci metod osvětlení a stínování 3D objektů“. Problematiku výukových programů jsem rozebíral již v kapitole páté a proto nepovažuji za nutné do této problematiky znovu zasahovat. Úkolem bakalářské práce tedy bylo vytvořit funkční program, který by nejenom demonstroval metody osvětlení a stínovaní, ale který by také uživatele naučil tyto techniky používat v praxi. Po analýze problematiky a nastudování jistých pedagogických zdrojů jsem navrhl aplikaci fungující na několika oknech jednotlivě rozdělených podle jejich funkcí.

5.2.2 Okna aplikace

V této části jsou popsána jednotlivá okna aplikace a jejich požadované funkce. Implementační záležitosti jsou k nalezení v kapitole následující. Ukazatele na všechny okna jsou tvořeny po inicializaci třídy aplikace class MyApp.

- **Okno Hlavní Menu**

  Okno Hlavního Menu nabízí uživateli začít výuku, spustit nápovědu programu, nebo ukončit program. Toto okno je tvořeno třídou `class Frame_main_menu.` Stejně jako pro všechna ostatní okna je součástí Hlavního Menu ovládací panel v horní pravé části obrazovky. Tento panel nabízí uživateli možnost spustit nápovědu určenou pro celý program, minimalizovat okno nebo program ukončit.

- **Okno Výběr Výuky**

  V tomto okně uživatel volí jednu ze čtyř možností výuky. Výuka je rozdělena do logických oddílů takto: Normály, Světelné zdroje, Model osvětlení a Materiál objektů. Výběr je možné opustit a vrátit se tak do hlavního menu. `class Frame_vyber` je třídou tohoto okna.

- **Okno Teoretická Část**

  Teoretická část vysvětluje látku formou technického textu. Součástí výuky jsou četné ilustrace, přehledně uvedené poučky s důležitými informacemi a úseky zdrojových kódů. Z tohoto okna může uživatel pokračovat do následující praktické části výuky, nebo se vrátit zpět na výběr výuky. Třídou tohoto okna je `class Frame_xxx_teorie,` kde xxx udává jméno daného cvičení (`normaly, svetlo, model, material`).

- **Okno Praktická Část**

  Toto okno je nejdůležitějším a nejsložitějším oknem aplikace a to jak pro uživatele, tak pro programátora. Tvoří jej dvě základní třídy `class MyFrame_xxx` a `class GLCanvas_xxx`. První z nich se opět stará o samotné okno a ovládací panel aplikace, druhá, `GLCanvas`, se stará o realtime vykreslování scény.

- **Okno Test Vědomostí**

  Okno testu vědomostí je vytvořeno s ohledem na pedagogický směr aplikace. Nenásilnou formou žádá uživatele o prověření nabytých vědomostí. Třídou tohoto okna je `class Frame_svetlo_test:`

- **Okno Nápověda Programu**

  Do tohoto okna je uživateli umožněn přístup ze kteréhokoliv okna aplikace. Nápověda nabízí pomoc uživateli v orientaci aplikací, ukazuje základní ovládání aplikace i její pokročilé funkce. Vše je prezentováno formou HTML dokumentu a to z důvodu hypertextových odkazů. Třídou okna je `class Frame_html_help:`.

- **Okno Nápověda Ovládacího Prvku**

  Posledním oknem je okno nápovědy jednotlivých prvků ovládacího panelu. Nápověda poskytuje informace o každém prvku jednotlivě, navíc pak uživateli může zobrazit zdrojový kód související s daným ovládacím prvkem. Tento kód je možné zkopírovat do schránky a použít v jakémkoliv textovém editoru. Třídou okna je `class Frame_help_model:`.

6 Implementace

V následující části se budu věnovat popisu implementace. Nebudu však popisovat implementaci celého programu, to by bylo velmi zdlouhavé a také zbytečné a to zejména proto, že je program složen ze čtyř takřka stejných částí (jednotlivá cvičení). Při popisu implementace se budu zabývat pouze důležitými a pro případného čtenáře snad i zajímavými částmi.

6.1 Okna aplikace

Všechna okna aplikace vycházejí ze standardních tříd knihovny WxWidgets. Okna jsou děděna ze třídy `wxFrame`, přitom každé z oken má vlastní canvas, tedy plochu na kterou umisťuji jednotlivé prvky okna. Tato plocha (neboli canvas) vychází ze třídy `wxScrolledWindow` a to proto, aby bylo možné jí případně povolit scrollbar neboli posuvník.

Další společnou vlastností všech oken v mé aplikaci je ovládací panel v horní pravé části obrazovky. Je tvořen třemi prvky typu `wxBitmapButton*`. Tyto prvky umožňují uživateli v kterémkoliv okamžiku přistoupit k nápovědě programu, minimalizovat okno programu nebo program ukončit.

Typ tlačítka `wxBitmapButton*` jsem zvolil z důvodu přiřazení vlastní bitmapy tlačítku. Tím jsem se mohl vyhnout nevzhlednému a zašedlému uživatelskému prostředí, které by podle mého názoru mohlo odradit případné zájemce o studium dané látky za pomoci mnou vytvořené aplikace. Pro zpříjemnění práce v prostředí výukového programu jsem pro každé tlačítko vytvořil odlišnou bitmapu pro tlačítko v normálním i aktivovaném stavu.

Dalším prvkem použitým prakticky ve všech oknech aplikace je `wxBitmap`. Ten vytvoří ukazatel na zvolenou bitmapu načtenou do programu při spuštění pomocí WxWidgets příkazu `image.LoadFile`.

Toto jsou tedy prvky společné všem oknům aplikace výukového programu.

6.1.1 class Frame_main_menu: public wxFrame

Z této třídy vychází okno hlavního menu. Toto okno je velmi jednoduché, obsluhuje pouze tři prvky typu `wxBitmapButton*` a několik `wxBitmap`.

Zdrojový kód tohoto cpp souboru je pro programátora vcelku nezajímavý, povětšinou se v něm objevuje jen volání funkcí na základě tabulky událostí, na jejímž principu funguje toolkit WxWidgets. Tato tabulka ze skládá z identifikátoru události, který se předem přiřadí některému z tlačítek a volanou funkcí dané třídy. Zde je příklad: `EVT_BUTTON(ID_STRANA_FRONT,MyFrame_material::strana_font)`. Tento princip volání

na základě události se uplatňuje v mnou zvoleném toolkitu pokaždé stejným způsobem a proto se tímto již nebudu zabývat.

6.1.2 class Frame_vyber: public wxFrame

Tento Frame je svou funkčností velmi podobný předešlému hlavnímu menu. Opět využívá jednouché bitmap tlačítka a několik obrázků na oživení uživatelského prostředí. Jak bylo dříve zmíněno, funkcí tohoto okna je volba jednoho ze čtyř cvičení. Postoupíme tedy aplikací dále na navazující obrazovku.

6.1.3 class Frame_xxx_teorie: public wxFrame

V tomto okamžiku uživatel vstoupil do výuky jednoho ze cvičení. Teoretická část opět využívá standardní třídu wxFrame spolu s `wxScrolledWindow`, které je zde uplatněno v důsledku velkého rozsahu teoretického výkladu. Výklad je zobrazen formou `wxStaticText *` a často doplňován obrázky pro lepší pochopení látky.

Pro zvýšení přehlednosti textu jsou využívány různé fonty pro úryvky zdrojových kódů a barevné rámečky pro důležité poučky a příkazy. Opět se nebudeme zdržovat nepotřebnými detaily a přistoupíme k následující a nejvíce zajímavé části kterou je …

6.1.4 class MyFrame_xxx: public wxFrame

Nejzajímavější částí programu je skutečně ta praktická. Největším lákadlem pro uživatele bude jistě OpenGL okno vytvořené pomocí třídy `wxGLCanvas`. Ta se stará o všechnu komunikaci mezi WxWidgets a grafickou knihovou OpenGL.

Další věcí, kterou uživatel jistě ocení, je volná rotace renderovaných objektů v okně. Moje aplikace využívá totiž takzvaný trackball (původně implementovaný Gavinem Bellem). Uživatel tak může pouhým drag&drop pomocí myší rotovat objektem po scéně a sledovat tak jakoukoliv změnu, která nastane ve všech směrech.

Implementace trackball je vcelku složitá záležitost a jelikož jsem tuto funkci nepsal já sám, jen zmíním, že se podle souřadnic odeslaných událostí pohybu myši vypočítají jisté čtveřice, podle nichž je sestavena takzvaná rotmatrix (tedy matice rotace) a ta je následně uplatněna na rotaci celého tělesa.

Tak tedy zpět k mému výukovému programu a jeho oknu praktické části.

Ovládaní a nastavování všech atributů zpracovávaných oknem OpenGL je prováděno pomocí ovládacího panelu na levé straně. To je vytvořeno opět pomocí `wxScrolledWindow` a to z důvodu velkého množství ovládacích prvků.

Mezi ovládací prvky patří klasické `wxBitmapButton*` a `wxTextCtrl*`. TextControl slouží k zadávání číselných hodnot. Toho se využívá například pro zadávání abmientní složky světla. Výhodou použití TextControl je možnost zadávat hodnoty ručně pomocí číslic na klávesnici nebo pomocí šipek ovládacího panelu.

6.1.5 InitGL() a GLCanvas::OnPaint

Z důvodu velkého rozsahu je těmto dvěma funkcím věnována samostatná kapitola. Přesto ale bude vše popsáno velmi stručným způsobem z důvodu omezení rozsahu textu. Případným zájemcům doporučuji pročtení programové dokumentace (viz dodatek).

InitGL() je funkce která proběhne před samotným spuštěním smyčky OpenGL. Jak již název napovídá, jde o inicializaci proměnných OpenGL. Do inicializace se obvykle vkládá umístění světel, nastavení jejich barvy (ambientní, difúzní, …), barva a typ materiálu model osvětlení a konečně vypnutí a zapnutí světel. Pro příklad uvádím zjednodušenou ukázku kódu InitGL():

```
// nastavení pozice světla
static const GLfloat light0_pos[4]   = { -50.0f, 50.0f, 0.0f, 0.0f };
// nastavení barvy světla
static const GLfloat light0_color[4] = { 0.6f, 0.6f, 0.6f, 1.0f };
// model stínování
glShadeModel(GL_SMOOTH);
// přiřazení vektorů barev a pozice světlu
glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightfv(GL_LIGHT0, GL_DIFFUSE,  light0_color);
// povolení světel
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
// nastavení materiálu pro objekty ve scéně
glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
```

V tomto ukázkovém kódu vidíme, na kterém místě programu je možné nastavit model stínování, tedy přesněji glShadeModel. V našem případě budeme používat GL_SMOOTH, což je Gouraudovo stínování. Jakoukoliv hodnotu, kterou zadáme ve volání funkce InitGL, je možné změnit v průběhu programu a toho jsem také prakticky pokaždé využil.

GLCanvas::OnPaint je funkce canvasu WxWidgets, která překresluje celé okno po každé události, která nastane. Tedy například změníme hodnotu červené ambientní složky světla a pomocí příkazu Refresh okno OpenGL překreslíme a zobrazíme tím všechny provedené změny. Takto funguje každé odesílání změny přes Widgets. Jako příklad uvedu jednoduchý úkon a to třeba změnu difuzní barvy pro dané světlo. Příklad by byl ovšem stále obsáhlý a proto se omezíme na změnu modré difuzní složky světla ve scéně.

Změna hodnoty difúzní složky

1. myší klikneme na tlačítko šipky nahoru odpovídající nastavení modré difúzní barvy pro zvolené světlo.
2. Widgets rozpozná vniklou událost podle identifikátoru a pomocí tabulky událostí nalezne a spustí správnou funkci (`MyFrame_svetlo::diffuse_up1`).
3. funkce přečte hodnotu z příslušného `wxTextCtrl` , převede tuto hodnotu na float a inkrementuje ji o danou hodnotu
4. vzniklá hodnota je uložena do proměnné `v_svetlo[0].diffuse_b`. v_svetlo je pole struktur světel které obsahuje všechny potřebné hodnoty pro nastavení světla.
5. je vytvořen vektor ze všech hodnot RGB daného světla spolu se změněnou hodnotu B. Tento vektor je následně poslán OpenGL jak je vidět na následujícím příkladu.

```
GLfloat light0_diffuse[4] = {v_svetlo[0].diffuse_r, v_svetlo[0].diffuse_g,
v_svetlo[0].diffuse_b, 1.0f};
glLightfv(GL_LIGHT0, GL_DIFFUSE,  light0_diffuse);
```

6. nakonec zavoláme funkci `Refresh();`. Ta dá povel OpenGL aby překreslil scénu s novým nastavením.

Tímto způsobem jsou OpenGL předávaný veškeré informace programu.

7 Závěr

Hlavním cílem práce bylo vytvoření výukového programu pro demonstraci metod osvětlení a stínování 3D objektů. Tedy aplikace, která by s ohledem na pedagogické zaměření uživateli/studentovi předložila postupně teorii, praktické procvičení a test dané problematiky. Tento cíl byl splněn.

Studiem problematiky osvětlení a stínování se zabývá kapitola první a druhá. Následující, tedy třetí kapitola pojednává o použité knihovně OpenGL a toolkitu WxWidgets. Čtvrtá kapitola se věnuje problematice tvorby výukových programů. Další kapitola s číslem pět se zaměřuje na podrobnou analýzu a návrh jednotlivých částí aplikace. V šesté kapitole popisuji vlastní implementaci programu a poslední kapitola je určena práci s programem.

Výukový program pro demonstraci metod osvětlení a stínování 3D objektů byl vyvíjen pro operační systém Microsoft Windows XP. Na tomto systému byla aplikace také testována, přesněji řečeno na verzi Microsoft Windows XP (SP2 v.2002). Aplikace běžela na tomto systému bez chyb a vykazovala plnou funkčnost odpovídající zadání.

Díky použití toolkitu WxWidgets a grafické knihovny OpenGL je možné tento výukový program bez problémů přenést na jiné platformy. Program byl testován a po lehkých úpravách také spuštěn na systému Linux. V tomto případě šlo o distribuci Kubuntu.

Výukový program byl také testován a s ohledem na názor většiny byl upraven k co největší spokojenosti uživatelů.

Výukový program pro demonstraci metod osvětlení a stínování 3D objektů byl vypracován v rozsahu zadání, dalšími kroky směrem ke zlepšení kvality této aplikace by byla implementace Phongova modelu pomocí OpenGL Shading Language, kde by bylo možné měnit nejen hodnoty vcházející do výpočetních rovnic, ale i samotné rovnice s okamžitým náhledem výsledku.

Dalším rozšířením by mohla být implementace podpory textur a míchání barev. To totiž úzce souvisí s osvětlením a stínováním ve 3D.

Posledním zajímavým rozšířením programu by mohlo být značné zvětšení repertoáru objektů použitelných na simulaci osvětlení, případně implementace rozsáhlých scén interiérů i exteriérů.

Literatura

[1] Žára J., Beneš B., Sochor J., Felkel P., *Moderní počítačová grafika*, Computer Press, 2005.

[2] Shreiner D., Woo M., Neider J., Davis T., OpenGL průvodce programátora, Computer Press, 2006

[3] Wright R., Lipchak B., OpenGL SuperBible, Sams Publishing, 2004

[4] Smart J., Hock K., Cross-Platform GUI Programming with wxWidgets, Pearson Education, 2006

[5] Gouraudovo, Phongovo a konstantní stínování. [online], [cit. 26. března 2008].
URL http://www.futuretech.blinkenlights.nl/gouraud.html

[6] Phongův model osvětlení. [online], [cit. 25. března 2008].
URL http://www.root.cz/clanky/opengl-19-phonguv-osvetlovaci-model/

[7] Lambertův model osvětlení. [online], [cit. 23. března 2008].
URL http://pocitacova-grafika.kvalitne.cz/lambertovo-pravidlo

[8] Problematika tvorby výukových programů. [online], [cit. 22. března 2008].
URL http://www.ceskaskola.cz/ICTveskole/AR.asp?ARI=2895&CAI=2129

Seznam příloh

8 Práce s programem

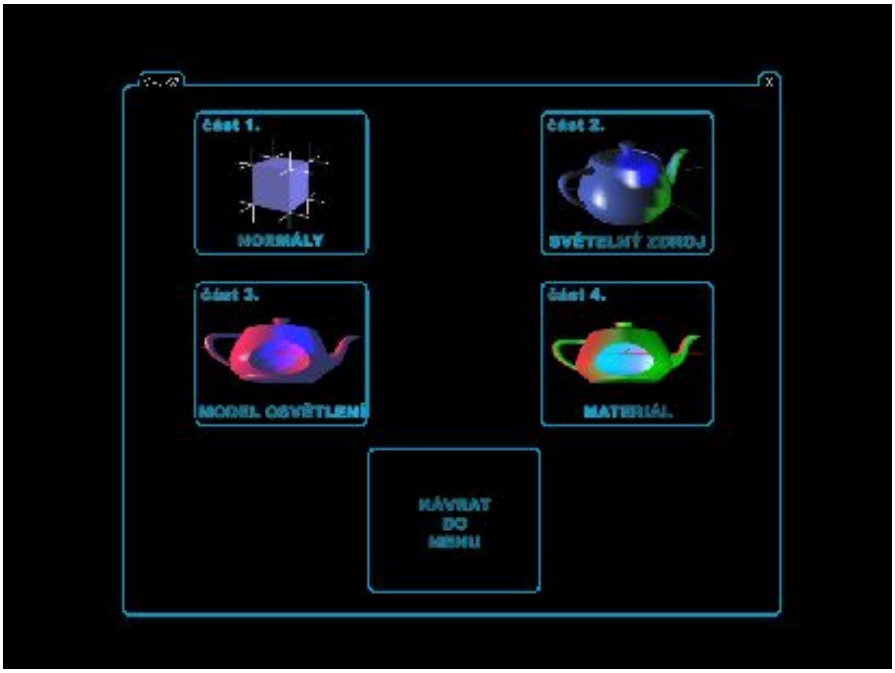
V následující kapitole ve zkratce popíši práci ve „Výukovém programu pro demonstraci metod osvětlení a stínovaní 3D objektů“.

8.1 Okno hlavního menu

Obrázek 8.1: Hlavní menu

Z okna hlavního menu je možné přejít do nabídky jednotlivých kurzů pomocí tlačítka ZAČÍT VÝUKU. Po stisku tohoto tlačítka budete přesunuti do obrazovky výběru cvičení - viz níže. Následující tlačítko O PROGRAMU již znáte - spouští okno nápovědy ve které se právě nalézáte. Posledním tlačítkem UKONČIT PROGRAM se zavře okno hlavního menu a program je ukončen. Součástí každého okna je také trojice ovládacích tlačítek ? _ X.

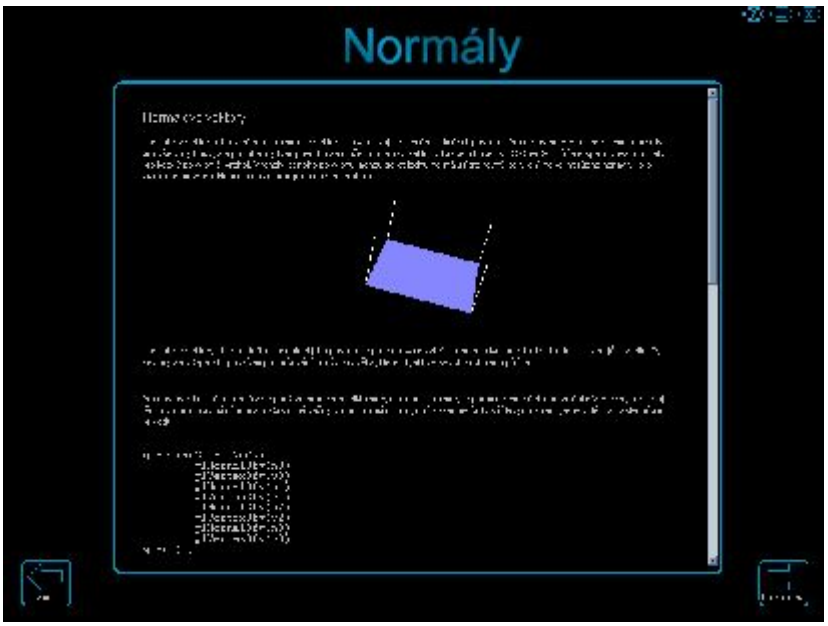
8.2 Okno výběru výuky



Obrázek 8.2: Výběr výuky

V tomto okně si můžete zvolit jedno ze čtyř cvičení a to postupně: Normály, Světelný zdroj, Model osvětlení a Materiál. Posledním tlačítkem NÁVRAT DO MENU se vrátíte na hlavní obrazovku, ze které můžete program ukončit nebo případně spustit nápovědu. Opět připomínám, že je toto možné provést i pomocí ovládacího panelu v horním levém rohu obrazovky.

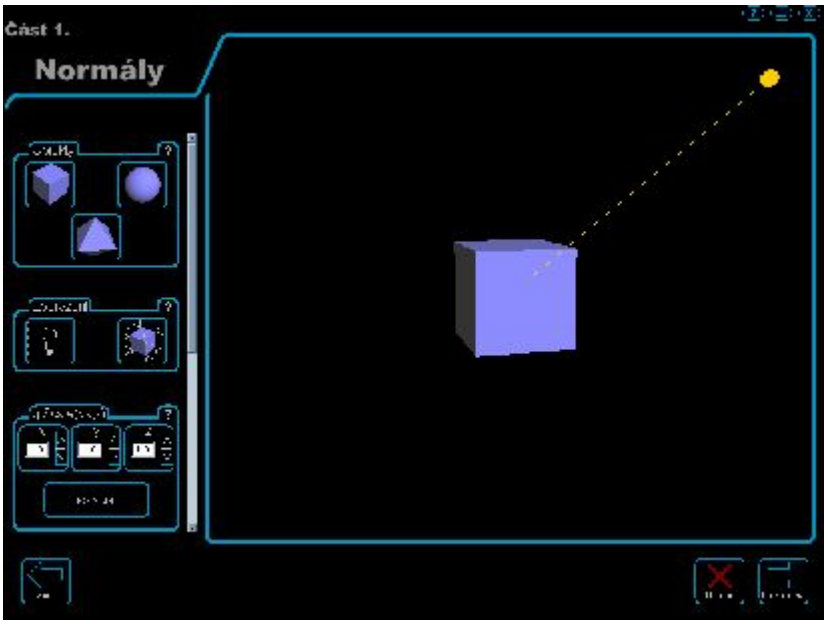
8.3 Okno teoretické výuky



Obrázek 8.3: Teoretická část výuky

Toto je okno teoretické výuky. Nenaleznete zde žádné ovládací (prvky kromě scrollbaru a tlačítek pro posun mezi okny). Pokud se chcete vrátit do okna výběru, klikněte na tlačítko **ZPĚT**. Pro pokračování ve výuce klikněte na **POKRAČOVAT**. V tomto okně jsou shrnuty všechny teoretické vědomosti potřebné pro pochopení látky.

8.4 Okno praktické výuky

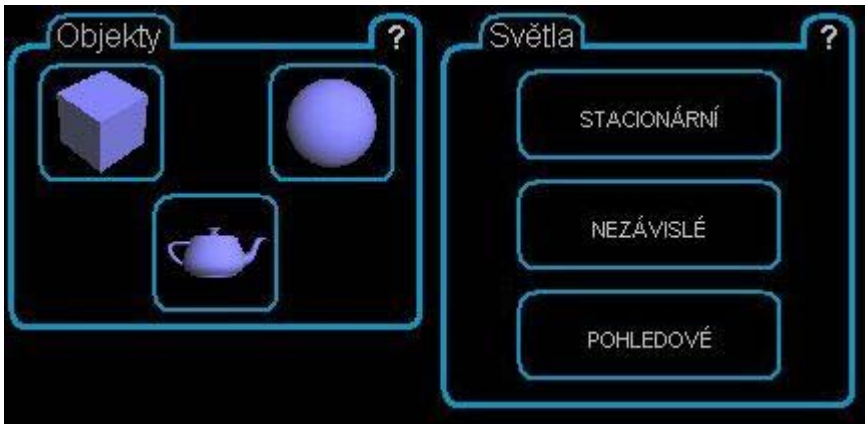


Obrázek 8.4: Praktická část výuky

Toto je okno praktické výuky. Zde si můžete v praxi vyzkoušet aplikaci všech funkcí popsaných v teoretické části. Jednotlivé prvky budou popsány v následující kapitole - Pokročilé funkce programu. Opět se můžete přesunovat mezi jednotlivými okny pomocí tlačítek **ZPĚT** (návrat do teoretické části) a **POKRAČOVAT** (spustí test dané části).

Okno praktické výuky se skládá ze dvou částí. Z ovládacího panelu se scrollbarem a projekčního rámu. V ovládacím panelu si můžete upravovat veškeré hodnoty, které jsou vám nabídnuty a v projekčním rámu můžete sledovat výsledky aplikování těchto hodnot na scénu.

8.4.1 Ovládací panel – tlačítka

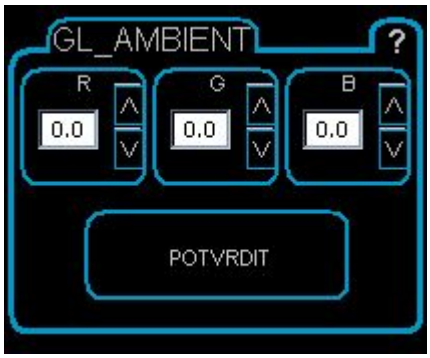


Obrázek 8.5: Tlačítka ovládacího panelu

Na obrázku vidíte část ovládacího panelu. V tomto případě je to volba objektu, který chceme vykreslit a na který chceme aplikovat funkce s námi zvolenými hodnotami, včetně volby typu světla, kterým chceme scénu nasvítit.

Objekt zvolíte jednoduchým kliknutím na tlačítko s obrázkem objektu. Od této chvíle bude v projekčním rámu vykreslován tento objekt. Typ světla zvolíme také jednoduchým kliknutím na daný typ světla. Od této chvíle budeme veškeré atributy nastavovat pro toto světlo. Po kliknutí na další ze světel nastavujeme hodnoty pro právě zvolené světlo.

8.4.2 Ovládací panel - textová pole



Obrázek 8.6: Ovládací panel

Na obrázku vidíte ovládací panel pro nastavení ambientní složky osvětlení. Tento panel umožňuje pohodlně nastavit hodnoty RGB pro dané světlo a ihned je aplikovat na vykreslovanou scénu. Ovládaní je možné provádět dvěma způsoby.

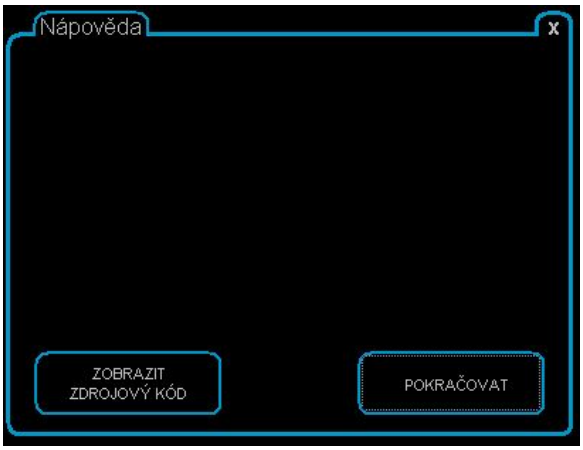
1. Pomocí šipek po stranách každého textového pole. Šipka nahoru hodnotu v textovém poli zvýší o určitou hodnotu, šipka dolů hodnotu sníží. Rozsah hodnot je omezen rozsahem povolených hodnot pro danou funkci OpenGL.

2. Pomocí ručního vložení číselné hodnoty do textového pole a potvrzení této hodnoty tlačítkem POTVRDIT. Hodnota se poté aplikuje na vykreslovanou scénu.

8.4.3 Tlačítko pro spuštění nápovědy

Toto je jedno z nejdůležitějších tlačítek programu. Umožňuje totiž pro každý prvek ovládacího panelu vyvolat podrobnou nápovědu. Kliknutím na toto tlačítko se zablokuje hlavní, okno ve kterém jste dosud pracovali a otevře se malé okno s nápovědou - viz níže.

8.4.4 Okno nápovědy pro prvky ovládacího panelu

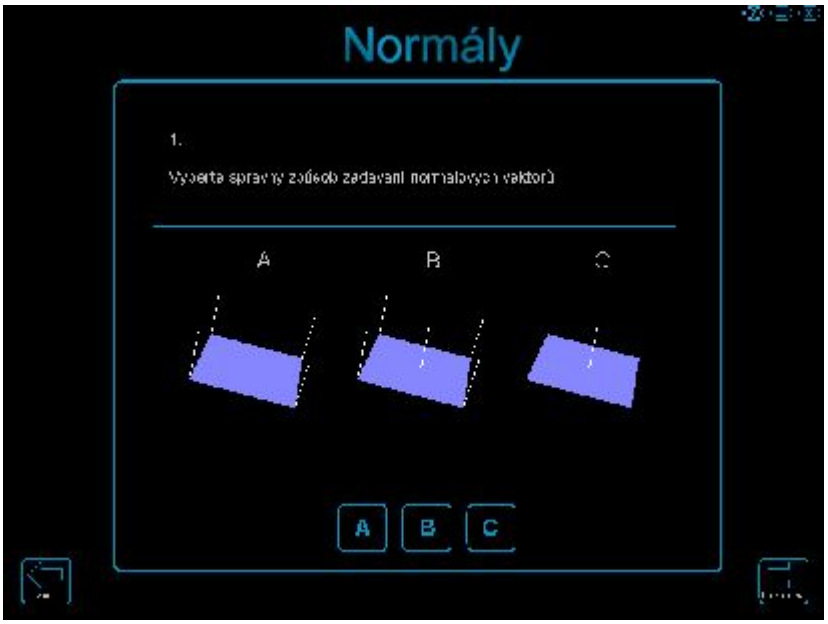


Obrázek 8.7: Nápověda

Toto je okno nápovědy pro jednotlivé prvky ovládacího panelu. V tomto okně je podrobně popsáno, co vlastně daným prvkem nastavujeme, jakou funkci voláme a v jakém tvaru. K naprostému pochopení všech cvičení je nutné prostudovat i tyto nápovědy jednotlivě ke každému objektu.

V okně nápovědy naleznete tlačítko **ZOBRAZIT ZDROJOVÝ KÓD**. Tímto tlačítkem vyvoláte textový editor, ze kterého je možné si zkopírovat přesné volání funkce, tedy stejné, které je voláno v našem programu. Nápovědu opustíte tlačítkem **POKRAČOVAT**.

8.5 Okno testu



Obrázek 8.8: Test znalostí

V okně testu jste zkoušeni z vědomostí nabytých v předešlých dvou částech daného cvičení. Na odpovědi odpovídejte pomocí tlačítek A B a C v dolní části obrazovky. Test není povinný, proto jej můžete kdykoliv přeskočit pomocí tlačítka POKRAČOVAT.