Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Informatika a výpočetní technika

BAKALÁŘSKÁ PRÁCE

Optimalizace trojúhelníkových sítí v E^3 a jejich zobrazování

1999 / 2000

Jan Doubek

Obsah

1 Úvod

S tím, jak se zvyšuje výkonnost současných výpočetních systémů, zvyšují se i nároky na kvalitu prostorových modelů tvořených trojúhelníkovými sítěmi. Stejně tak se zvyšují i nároky na kvalitu a rychlost zobrazení těchto modelů. Jelikož tyto modely pochází z různých oblastí, jako jsou 3D scannery, CAD systémy nebo různé modelovací programy, ne vždy jsou tyto modely v optimálním stavu, co se týče konzistence. Proto jsou navrhovány nejrůznější optimalizační algoritmy, které optimalizují trojúhelníkové modely pro různá kritéria. Prvním úkolem této práce je implementovat a vyhodnotit optimalizační algoritmus, který optimalizuje trojúhelníkovou síť tak, aby měla co nejmenší plochu. Tento algoritmus musí být použitelný nejen z hlediska variability modelů, ale i z hlediska časové a paměťové náročnosti.

I když budeme mít optimalizované trojúhelníkové modely, pokud je nebudeme schopni zobrazit v co nejkratším čase a v co nevětší kvalitě, budou nám k ničemu. Proto existují metody stínování, které snižují výpočetní náročnost i pro velké modely na minimum, a zároveň dosahují dobré kvality zobrazení těchto modelů. Již po několik desítek let se používají dobře známe algoritmy stínování, jako jsou konstantní stínování, Gouraudovo stínování a Phongovo stínování. I tyto metody mají však své nedostatky a při aplikaci v některých oblastech (např. ve zdravotnictví) mohou přinášet nekorektní výsledky. Z tohoto důvodu jsou navrhovány algoritmy nové, které by byly vhodnější pro některé specifické oblasti využití. Jednou z těchto nových metod je i metoda S-shading. Druhým úkolem této práce je právě implementace metody S-shading a srovnání jejích vlastností s vlastnostmi "konkurenční" Gouraudovo metody.

2 Trojúhelníková reprezentace

Tato kapitola obsahuje teoretický úvod do použití trojúhelníkové reprezentace v počítačové grafice, představuje výhody a nevýhody použití trojúhelníkových sítí při aplikaci v různých oblastech počítačové grafiky a dále se věnuje otázce implementace trojúhelníkových modelů. Druhá část této kapitoly se zabývá minimalizací povrchů trojúhelníkových sítí a podává krátký teoretický úvod do problému implementace minimalizace povrchů.

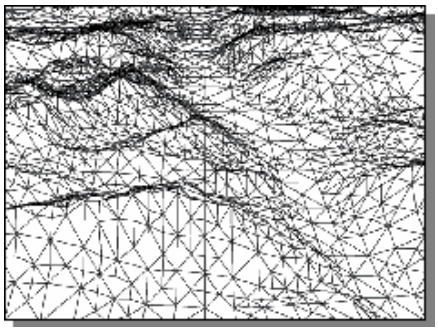
2.1 Použití trojúhelníkových sítí

Trojúhelníková reprezentace je nejpoužívanější reprezentací povrchů v počítačové grafice. Obecně nemusí být reprezentace povrchů pouze trojúhelníková, ale může být libovolná polygonální. V případě polygonů o čtyřech a více vrcholech je však nutné určovat jejich konvexitu [Zar98], což představuje, oproti trojúhelníkům, určitou výpočetní náročnost navíc. Proto zůstává trojúhelník základním prvkem většiny používaných grafických programů. Většina standardů používá trojúhelník jako základní prvek pro vytváření modelů, ať už ve formě jednotlivých trojúhelníků, nebo v optimalizovaných souborech trojúhelníků, jako jsou trojúhelníkové vějíře (fans) nebo pruhy (strips). Zásadní výhodou trojúhelníkové reprezentace je však podpora technickým vybavením počítačů. V poslední době dochází k velkému rozmachu grafických akcelerátorů, jejichž jedním z hlavních parametrů výkonnosti je počet vykreslených trojúhelníků za vteřinu. Tyto akcelerátory mají též hardwarově implementovánu lineární interpolaci, což je důsledek většiny operací nad objekty reprezentovanými trojúhelníkovou sítí.

Reprezentace pomocí trojúhelníkových sítí tedy nachází své uplatnění převážně v časově kritických aplikacích, jako jsou:

- Architektura – modely měst, budov, mostů, …
- CAD aplikace
- Moderní počítačové hry
- Simulace – létání, řízení, procesy
- Virtuální realita – výcvik, terapie, virtuální hry
- Vědecká vizualizace – modely polí, proudění, turbulencí

Jedním z typických příkladů použití trojúhelníkové reprezentace je i modelování terénů, kterého se využívá např. v leteckých simulátorech. Na obr. 2.1 je příklad terénu tvořeného trojúhelníkovou sítí. Obr. 2.2 poté ukazuje stejný model terénu, tentokráte však již potažený texturami.



Obrázek 2.1: Trojúhelníkový model terénu

Obrázek 2.2: Model terénu potažený texturami

Méně vhodná je trojúhelníková reprezentace pro modelování. Z tohoto důvodu modelovací programy používají raději reprezentaci jinou (např. NURBS křivky). Sítě trojúhelníků využívají až při generování svých výstupů. Typickým příkladem tohoto postupu jsou trojrozměrné modely používané ve virtuální realitě. Většina takových modelů je vytvářena pomocí NURBS křivek a síť trojúhelníků je z nich vygenerována až ve finální fázi. Tímto postupem lze z jednoho modelu získat více variant lišících se svou složitostí, a tedy i velikostí detailů.

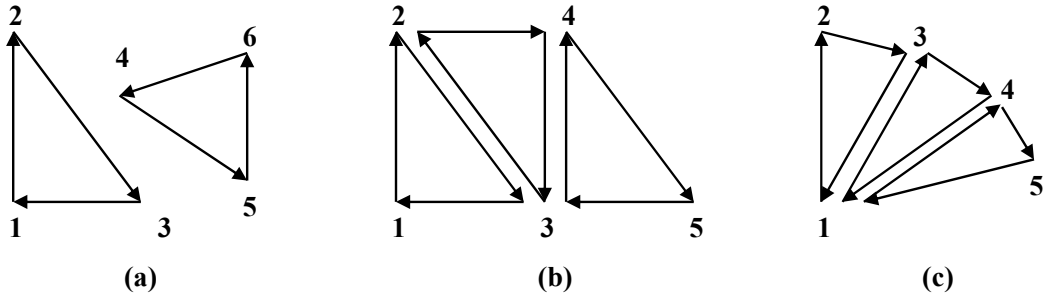
2.2 Implementace trojúhelníkových sítí

Trojúhelník bývá nejčastěji v datech reprezentován pouze svými vrcholy. Jejich pořadí, a tudíž i pořadí jeho stran, pak určuje orientaci celého polygonu. Vrchol s sebou může dále nést i informace o normálovém vektoru nebo další nezbytné informace.

V paměti počítače bývá trojúhelníková síť nejčastěji uložena jako pole vrcholů a trojúhelníků. Pole vrcholů obsahuje prostorové souřadnice všech vrcholů ve scéně. Pole trojúhelníků je tvořeno indexy do pole vrcholů, kde každý index představuje jeden z vrcholů trojúhelníku. Tato implementace je poměrně úsporná, neboť vrcholy,

které incidují s více trojúhelníky, jsou uloženy v paměti počítače pouze jednou. Na druhou stranu je tato úspornost kompenzována menší výhodností při zobrazování.

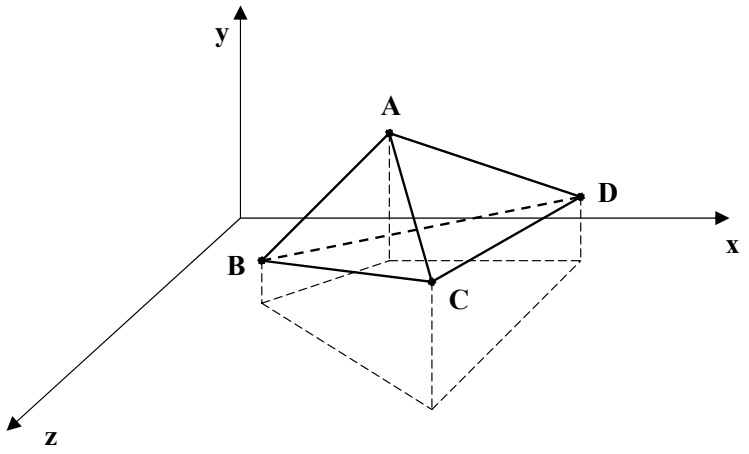
Hlavním cílem implementace trojúhelníků by měla být minimalizace toku dat v rámci počítače. Algoritmus, který bude vykreslovat každý trojúhelník zvlášť, nebude příliš efektivní, neboť společné vrcholy budou přenášeny opakovaně. Z tohoto důvodu jsou hojně využívány optimalizované soubory trojúhelníků. Nejběžnější dva typy jsou trojúhelníkový vějíř (triangle fan) a pruh trojúhelníků (triangle strip). Obr. 2.3 ukazuje srovnání interpretace jednotlivých trojúhelníků, pruhu trojúhelníků a vějíře trojúhelníků tak, jak je implementuje OpenGL.



Obrázek 2.3: (a) Jednotlivé trojúhelníky, (b) Pruh trojúhelníků (strip), (c) Vějíř trojúhelníků (fan)

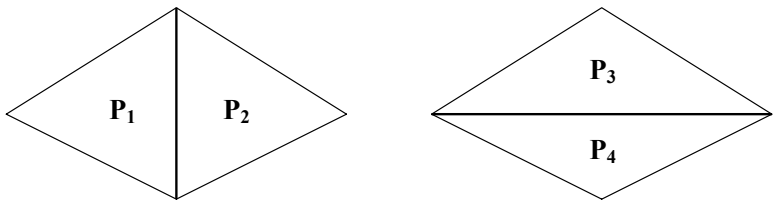
2.3 Minimalizace povrchu trojúhelníkových sítí

V současné době se pozornost obrací také k optimalizaci trojúhelníkových sítí. Trojúhelníkové reprezentace modelů nebývají vždy dokonalé, mohou např. obsahovat příliš mnoho trojúhelníků. Z tohoto důvodu se používají různé optimalizační algoritmy, které optimalizují danou síť pro nejrůznější potřeby.



Obrázek 2.4: Prostorový model čtyřúhelníku tvořeného dvěma sousedními trojúhelníky

Jednou z těchto optimalizací je i minimalizace plochy trojúhelníkové sítě. Její cíl je jasný již z názvu – dosažení co nejmenšího povrchu aproximovaného modelu. Toho lze docílit srovnáváním plochy čtyřúhelníků tvořených dvěma sousedním trojúhelníky. Na obr. 2.4 je příklad takového čtyřúhelníku. Tento čtyřúhelník je tvořen trojúhelníky ACB a ADC, přičemž jejich původní společnou hranu tvoří úsečka AC. Vezmeme-li tento čtyřúhelník a prohodíme-li jeho úhlopříčky, tj. místo úsečky AC bude nyní úhlopříčku tvořit úsečka BD (na obr. 2.4 zobrazená čárkovaně), dostaneme dva nové trojúhelníky ADB a BDC, jejichž společnou hranu bude tvořit úsečka BD. Je-li součet ploch nově vytvořených trojúhelníků menší než součet ploch původních trojúhelníků, nové trojúhelníky nahradí trojúhelníky původní. Konkrétní příklad této operace je na obr. 2.5. Je-li $P_3 + P_4 < P_1 + P_2$, pak proveď výměnu.



Obrázek 2.5: Je-li $P_3 + P_4 < P_1 + P_2$, pak proveď výměnu úhlopříček

V nejjednodušším případě provedeme pro každý trojúhelník ve scéně takovýto test pro všechny jeho sousedy a u páru s nejmenší plochou provedeme prohození trojúhelníků. Konkrétní implementace těchto průchodů trojúhelníkovou sítí je popsána v kapitole 5.3.

Výsledkem této optimalizace je zmenšení celkového povrchu trojúhelníkové sítě. Kritériem úspěšnosti optimalizace je pak počet prohozených trojúhelníků a plocha získaná touto optimalizací. Velikost získané plochy však může být porovnávána pouze v rámci jednoho modelu, neboť čím více trojúhelníků se v modelu vyskytuje, tím menší je plocha jednotlivých trojúhelníků a tím menší bude tedy i získaná plocha. Hlavním kritériem zisku optimalizace je tedy celkový počet vyměněných trojúhelníků.

Optimalizaci lze provádět opakovaně. Smysl má však pouze, dokud je počet prohozených trojúhelníků v rámci jednoho průchodu nenulový (u modelů s menším počtem trojúhelníků – řádově desítky tisíc trojúhelníků) nebo dokud je tento počet významný vůči celkovému počtu trojúhelníků ve scéně (u modelů s velkým počtem trojúhelníků – řádově stovky tisíc až miliony trojúhelníků).

3 Stínování scény

Tato kapitola nás nejprve seznámí s pojmem stínování a vysvětlí, co znamená stínování modelu a jak toto stínování funguje. V druhé části kapitoly jsou popsány dvě nejběžnější metody stínování – konstantní a Gouraudovo stínování.

3.1 Pojem stínování scény

Stínování je jedním z významných prvků v počítačové grafice. Nezabývá se však nalezením vržených stínů, jak by se zdálo podle pojmu *stínování*, ale spíše nalezením odstínů (z anglického shade = odstín, tón) barev na povrchu modelů.

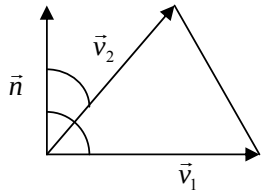
Ve chvíli, kdy máme model objektu (tvořený libovolnou polygonální sítí), známe jeho materiálové konstanty a máme dáno i osvětlení scény, jsme schopni určit barvu v libovolném bodě této osvětlené plochy. Určování barvy bodu v každém bodě obrazovky by však bylo zdlouhavé a neefektivní, a proto se využívá několika metod, u kterých se vyhodnotí osvětlovací model pouze v několika bodech na povrchu plochy, a pomocí algoritmu zvolené metody jsou určeny i barvy ostatních bodů zobrazované plochy.

Tyto metody se souhrnně nazývají stínování (shading). V případě trojúhelníkových sítí, kdy je model aproximován souborem trojúhelníků, se některými metodami úspěšně daří vyhladit hraniční zlomy mezi trojúhelníky tak, že lze docílit přirozeného zaoblení ploch a výsledný trojúhelníkový model se vzhledem přibližuje svému originálu.

Mezi nejpoužívanější metody stínování patří konstantní stínování, Gouraudovo stínování a Phongovo stínování. První dvě jmenované metody jsou zároveň podporovány většinou standardních grafických rozhraní, jako jsou např. OpenGL nebo DirectX. Phongovo stínování je výpočetně nejnáročnější, a proto má využití hlavně tam, kde je kvalita zobrazení důležitější než rychlost. V dalších částech této kapitoly budou popsány obecné principy činnosti prvních dvou uvedených metod a jejich výhody a nevýhody.

3.2 Konstantní stínování (Flat shading)

Jednou z nejjednodušších a také nejrychlejších metod stínování je konstantní stínování. Vychází z předpokladu, že každý trojúhelník má pouze jednu normálu. Pokud tato normála není obsažena v zadaných datech, je možné ji určit jako výsledek znormovaného vektorového součinu dvou sousedních hran, které jsou orientovány proti směru hodinových ručiček (viz obr. 3.1).



$$\vec{n} = \vec{v}_1 \times \vec{v}_2$$

$$\vec{n} = \frac{\vec{n}}{\sqrt{n_x^2 + n_y^2 + n_z^2}}$$

Obrázek 3.1: Výpočet normály trojúhelníku

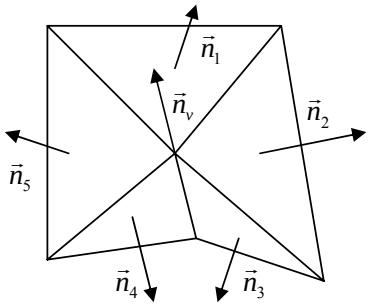
Na základě znalosti těchto normál je vypočítán jeden barevný odstín, který je přiřazen všem bodům trojúhelníku. Jelikož osvětlovací model je vyhodnocován pouze v jednom bodě, je třeba pro každý trojúhelník vybrat jeho reprezentativní bod, ve kterém bude zjišťován barevný odstín trojúhelníku. Nejlépe by k tomuto účelu vyhovovalo těžiště trojúhelníku, ale vzhledem k faktu, že tato metoda je používána hlavně kvůli své vysoké rychlosti a počítání těžišť všech trojúhelníků by výpočet zbytečně zpomalovalo, používá se jeden z vrcholů trojúhelníku. OpenGL např. vždy používá pro umístění normály bod zadaný jako třetí v pořadí.

Největší výhodou této metody je určitě její vysoká rychlost. Na druhou stranu je ale tato rychlost kompenzována nízkou kvalitou zobrazení. Metoda je vhodná např. pro zobrazení mnohostěnů, neboť úspěšně ukazuje umístění a natočení těchto těles v prostoru. Méně vhodná je již pro složitější modely, kde je trojúhelníková síť použita jen jako aproximace originálu. Zobrazení tímto druhem stínování totiž ještě zdůrazňuje fakt, že objekt není oblý, ale že je pouze aproximován souborem plošek. Obecně platí, že čím více trojúhelníky je model tvořen a čím menší tyto trojúhelníky jsou, tím méně je u těchto modelů poznat nevhodnost konstantního stínování.

3.3 Gouraudovo stínování (Gouraud shading)

Metoda navržená Henri Gouraudem [Gour71]. V současné době asi nejpoužívanější metoda stínování v počítačové grafice. Na rozdíl od konstantního stínování, kde byl celému trojúhelníku přiřazen jeden barevný odstín, využívá Gouraudova metoda spojitého barevného stínování.

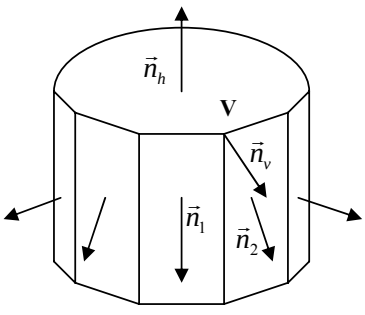
Pro činnost algoritmu je důležitá znalost barev všech vrcholů trojúhelníku. Ze znalosti normál, vyhodnocením osvětlovacího modelu získáme požadované barvy ve vrcholech. Pokud nejsou normály ve vrcholech známy, lze je opět dopočítat. Normála ve vrcholu se určí jako aritmetický průměr normál plošek, které obsahují daný vrchol (viz obr. 3.2).



$$\vec{n}_v = \frac{\vec{n}_1 + \vec{n}_2 + \vec{n}_3 + \vec{n}_4 + \vec{n}_5}{5}$$

Obrázek 3.2: Výpočet normály ve vrcholu

V případě existence hran je nutné použít k výpočtu normály vrcholu pouze normály trojúhelníků aproximujících příslušnou plochu (viz obr. 3.3). V případě použití normály trojúhelníku, který je "přes" hranu, by výsledná barva vrcholu byla nesprávná, stínování všech trojúhelníků obsahujících daný vrchol by bylo nekorektní a hranový přechod by byl vystínován, čímž by vznikl dojem, že hrana je zaoblená.

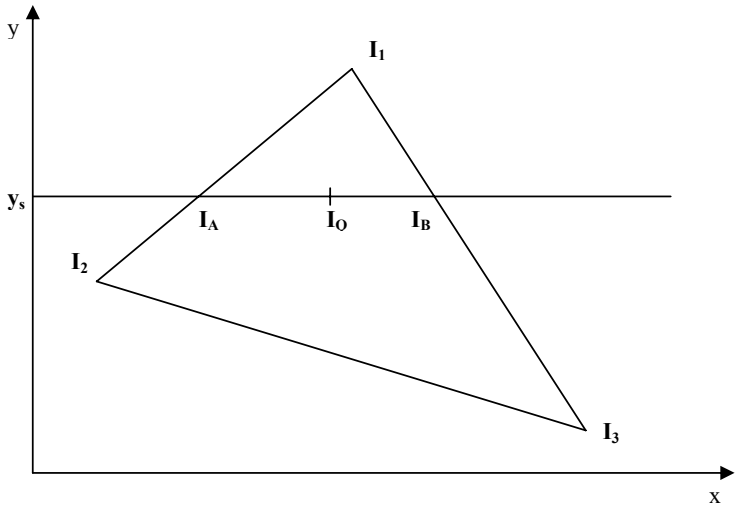


Pro normálu vrcholu **V** platí:

$$\vec{n}_v = \frac{\vec{n}_1 + \vec{n}_2}{2}$$

Obrázek 3.3: Výpočet normály ve vrcholu v případě existence hran

Ve chvíli, kdy známe barvy vrcholů trojúhelníku, můžeme provést jeho vystínování pomocí bilineární interpolace. V případě Gouraudova stínování interpolujeme barvy ve vrcholech tak, jak je to vyobrazeno na obr. 3.4.



Obrázek 3.4: bilineární interpolace barevné intenzity

$$I_A = I_1 + (I_2 - I_1) \cdot \frac{y_s - y_1}{y_2 - y_1} \qquad I_B = I_1 + (I_3 - I_1) \cdot \frac{y_s - y_1}{y_3 - y_1}$$

$$I_Q = I_A + (I_B - I_A) \cdot \frac{x_Q - x_A}{x_B - x_A}$$

Tyto vztahy je nutné aplikovat na každou složku barevné intenzity, tzn. že pokud používáme pro míchání barev barevný systém RGB, je nutné aplikovat vzorce zvlášť pro červenou, zelenou a modrou složku barevného systému.

Vzhledem k tomu, že tato metoda vcelku účinně vyhlazuje barevné rozdíly v místech přechodů mezi trojúhelníky a že je dostatečně rychlá, je v současné době asi nejpoužívanější metodou pro stínování polygonálních modelů. Je implementována ve valné většině grafických rozhranní a pro maximální urychlení se implementuje i přímo do grafických akcelerátorů. Na druhou stranu i tato metoda nepodává vždy věrný obraz skutečných objektů, např. v případě rovinných plošek kolmých na dopadající světlo, kdy tato metoda nedokáže věrně zachytit místní zvýšení jasu na tomto typu plošek. Stejně tak pomocí této metody nelze vytvořit odlesky způsobené odraženým světlem.

4 S - shading

Tato kapitola se věnuje nové metodě stínování, která se nazývá S-shading (zkratka výrazu Smooth shading). První část kapitoly objasňuje důvody, proč tento nový druh stínování vznikl, a vyslovuje požadavky na tuto metodu. V druhé části kapitoly je dopodrobna rozebrána samotná metoda S-shading a její implementace.

4.1 Vznik metody S - shading

S-shading je novou metodou stínování, navrženou prof. Skalou na Západočeské univerzitě v Plzni [Skal00].

Metody stínování, uvedené v předchozí kapitole, i přes svoji vysokou používanost nejsou ve všech případech schopné zobrazit povrch objektu korektně. Zvláště v případech, kdy není dán osvětlovací model, ale přímo barvy jednotlivých trojúhelníků (v reprezentativních bodech), mohou Gouraudova i Phongova metoda vést k nesprávným až zavádějícím výsledkům. Z tohoto důvodu byl navržen S-shading, který kombinuje uspokojivou kvalitu zobrazení, co se týče hladkosti a barevných přechodů, s korektností zobrazení daného povrchu.

4.2 Požadavky na metodu S - shading

Aby se metoda S-shading vyvarovala nedostatků popsaných v úvodu této kapitoly a zároveň byla schopna konkurovat standardním metodám stínování, co se týče rychlosti zobrazení, musí být splněny následující požadavky:

- V případě, že jsou dány barvy trojúhelníků v jejich reprezentativních bodech, musí být zobrazeny ve finálním obraze.
- Stínování musí být plynulé, tj. přechody mezi jednotlivými trojúhelníky musí být vyhlazeny.
- Algoritmus musí používat v maximální míře akcelerační možnosti rozhraní OpenGL.

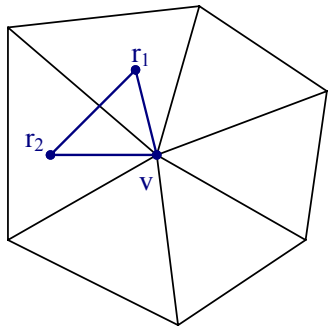
Pro splnění těchto požadavků je nutné definovat předpoklady, za kterých může být S-shading použit:

- Je dána trojúhelníková síť, tvořená jednotlivými vrcholy.
- Pro každý vrchol jsou známy všechny trojúhelníky, které incidují s tímto vrcholem.
- Každý trojúhelník je reprezentován jedním reprezentativním bodem. Známe buď přímo barvy těchto bodů, nebo jejich normály.

4.3 Popis metody S - shading

Comment [JD1]:

Základním požadavkem na metodu S-shading je korektní zobrazení barev v reprezentativních bodech. Toho lze dosáhnout tak, že tyto body položí základ nové trojúhelníkové sítě, kde budou reprezentativní body dvou sousedních trojúhelníků spolu s jedním vrcholem společným pro oba původní trojúhelníky tvořit trojúhelník nový (viz obr. 4.1).



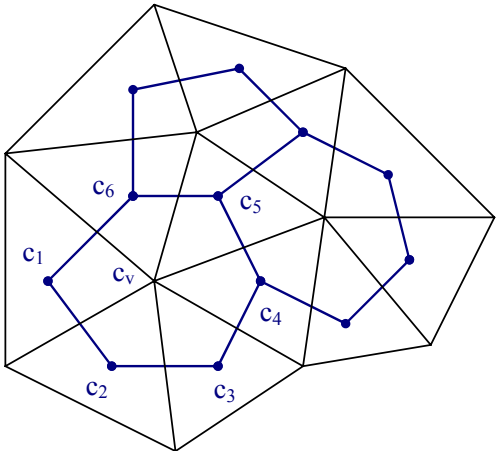
Obrázek 4.1: Vytvoření nového trojúhelníku

Jelikož reprezentativní body budou tvořit vrcholy nového trojúhelníku, je zajištěno, že barvy v reprezentativních bodech budou zachovány.

Jsou-li známy pouze barvy v reprezentativních bodech, je nutné pro každý nový trojúhelník zjistit barvu třetího vrcholu, tj. vrcholu původní trojúhelníkové sítě. Jeho barvu lze dopočíst ze znalosti barev reprezentativních bodů trojúhelníků obsahujících daný vrchol. V případě, že vrchol, jehož barvu chceme zjistit, inciduje s n trojúhelníky, jejichž reprezentativní body mají barvy c_1, c_2, …, c_n, pro barvu vrcholu c_v platí:

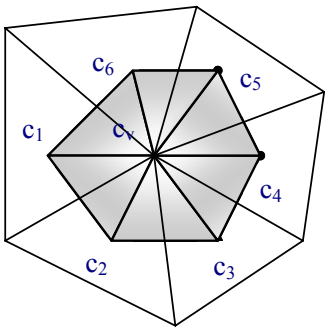
$$c_v = \frac{1}{n} \cdot \sum_{i=1}^{n} c_i$$

Vypočtenou barvu můžeme použít pro všechny trojúhelníky, které vzniknou “okolo“ vrcholu. Obr. 4.2 ukazuje, ze kterých reprezentativních bodů obklopujících daný vrchol se počítá barva tohoto vrcholu.



Obrázek 4.2: Výpočet barvy vrcholu

Aby byl povrch modelu hladký, jsou všechny nově vytvořené trojúhelníky vystínovány pomocí standardního Gouraudova stínování. Kolem každého trojúhelníku tedy vykreslujeme stejný počet nových trojúhelníků, jako je počet původních trojúhelníků obsahujících daný vrchol (viz obr. 4.3).



Obrázek 4.3: Stínování okolo vnitřního vrcholu

Každý původní trojúhelník je rozdělen na tři nové trojúhelníky, je tedy nutné vykreslovat třikrát více trojúhelníku, než bylo v původní scéně. Tento fakt by v případě použití normálního typu vykreslování trojúhelníků pomocí GL_TRIANGLES módu v OpenGL vedl k ztrojnásobení času, potřebného

k vykreslení scény. Vykreslované trojúhelníky obklopující stejný vrchol však tvoří jakýsi "vějíř", u kterého je možné použít účinnějšího módu GL_TRIANGLE_FAN, který umožňuje významné urychlení vykreslování scény v porovnání s klasickým GL_TRIANGLES módem. Pokud původní model obsahuje *n* trojúhelníků, použitím klasického GL_TRIANGLES módu by bylo nutné zpracovat *3*n* vrcholů. V GL_TRIANGLE_FAN módu musí OpenGL zpracovat *n+1* trojúhelníků pro vějíř o *n* stranách, což znamená nutnost zpracovat *3*(n+1)* vrcholů. Porovnáním těchto dvou faktů lze tedy určit relativní rychlost vykreslování *v* podle vzorce:

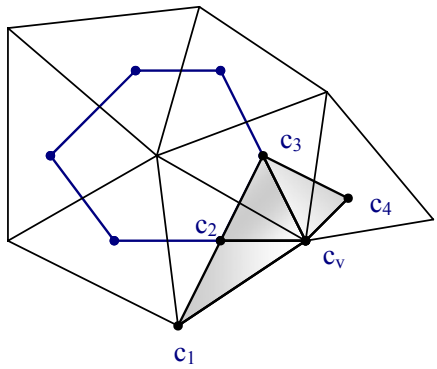
$$v = \frac{n}{n+1}$$

Za předpokladu, že každý vrchol obklopuje v průměru 6 trojúhelníků, bude relativní rychlost vykreslování *v* rovna:

$$v = \frac{n}{n+1} = \frac{6}{7} = 0.857 = 85.7\%$$

V běžných scénách by tedy relativní rychlost vykreslování měla dosáhnout přibližně 85,7 % původní vykreslovací rychlosti.

Ve scénách však nejsou pouze "ideální" vrcholy, které lze obklopit trojúhelníky tvořícími uzavřený disk, ale i okrajové vrcholy. I v těchto případech je možné použít GL_TRIANGLE_FAN mód. Na rozdíl od ideálních vrcholů, kde vějíř trojúhelníků začínal i končil v reprezentativním bodě jednoho z přilehlých trojúhelníků, u okrajových vrcholů začíná vějíř u jednoho ze sousedních vrcholů. Další vrcholy vějíře poté tvoří reprezentativní body obklopující daný hraniční vrchol. Vějíř končí opět reprezentativním bodem, není ale žádným způsobem uzavřen (viz obr. 4.4).



Obrázek 4.4: Stínování okolo hraničního vrcholu

5 Vyhodnocení výsledků

Tato kapitola prezentuje výsledky dosažené programem, který byl vytvořen na základě teoretických poznatků uvedených v předchozích kapitolách. Hlavním obsahem první části této kapitoly je popis implementace algoritmu pro minimalizaci trojúhelníkové sítě, zhodnocení výsledků této implementace a její vizuální aspekty. Ve druhé části této kapitoly provedeme zhodnocení výsledků implementace metody S-shading a porovnání jejich vlastností s vlastnostmi metody Gouraud shading.

Pro účely testování byly použity dva počítače. Model "Turbine Blade" byl testován na počítači s procesorem Intel Pentium III 500MHz s 1GB RAM a grafickou kartou NVIDIA Riva TNT2 s 32MB grafické paměti. Všechny ostatní modely byly testovány na stroji s procesorem Intel Celeron 525MHz se 128MB RAM a grafickou kartou NVIDIA Riva TNT2 Ultra s 32MB grafické paměti. Program byl na obou počítačích testován v prostředí Windows NT 4.0.

5.1 Paměťové nároky

Jak bylo řečeno v předchozích kapitolách, trojúhelníkové modely mohou sestávat z milionů trojúhelníků. Z tohoto důvodu je důležitá efektivní implementace trojúhelníkových sítí tak, aby paměťové nároky této implementace odpovídaly současným hardwarovým možnostem počítačů. Nároky na paměť vyhodnocovaného programu ukazují tab. 5.1 a 5.2. Tab. 5.1 obsahuje všechny položky v paměti, které se vztahují k vrcholům, tab. 5.2 obsahuje všechny položky v paměti, které se vztahují k trojúhelníkům.

Položka	Reprezentace	Velikost (B)
Souřadnice vrcholu	3 * double	24
Souřadnice normály vrcholu	3 * double	24
Počet incidujících trojúhelníků	ulong	4
Indexy incidujících trojúhelníků	ulong + i * ulong	4 + i * 4
Parametry vrcholu	byte	1
Celkem		57 + i * 4

Tabulka 5.1: Obsazenost paměti položkami vztahujícími se k vrcholům

Položka	Reprezentace	Velikost (B)
Indexy vrcholů	3 * ulong	12
Souřadnice normály trojúhelníku	3 * double	24
Souřadnice reprezentativního bodu (těžiště)	3 * double	24
Indexy sousedících trojúhelníků	3 * ulong	12
Typy hran trojúhelníku	3 * byte	3
Parametry trojúhelníku	byte	1
Celkem		76

Tabulka 5.2: Obsazenost paměti položkami vztahujícími se k trojúhelníkům

Parametr *i* v tabulce 5.1 představuje počet trojúhelníků incidujících s každým vrcholem. Tato hodnota se může lišit vrchol od vrcholu a také se mění s každým prohozením trojúhelníků při minimalizaci trojúhelníkové sítě. Průměrná hodnota *i* se pohybuje okolo 6 trojúhelníků na vrchol. Např. pro největší testovaný model "Turbine Blade", který sestává z 882 954 vrcholů a 1 782 518 trojúhelníků, budou nároky na paměť rovny:

$$M = 882\,954 \cdot (57 + 6 \cdot 4) + 1\,782\,518 \cdot 76 = 71\,519\,274 + 135\,471\,368 = 206\,990\,642 \text{ bytů}$$

Program se tedy bude snažit alokovat přibližně 207MB paměti jen pro data trojúhelníkové sítě. Toto je však extrémní případ a pro průměrný model s 50 000 vrcholy a 100 000 trojúhelníky se paměťové nároky pohybují okolo již přijatelných 12MB.

5.2 Přehled testovaných modelů

Většina testovaných modelů pochází ze sbírky Georgia Institute of Technology (http://www.cc.gatech.edu/projects/large_models). Tabulka 5.3 obsahuje seznam testovaných modelů spolu se základními informacemi o nich. Náhledy na testované modely jsou uvedeny v příloze A.

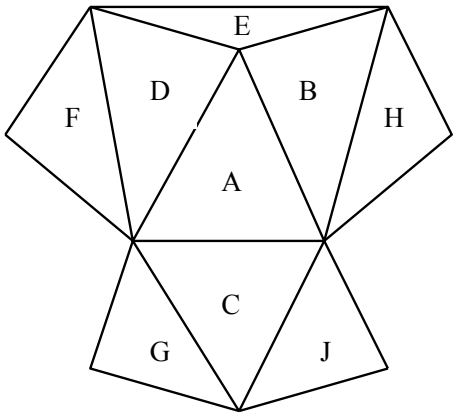
5.3 Implementace algoritmu minimalizace sítě

Navržený algoritmus pro minimalizaci trojúhelníkové sítě pracuje ve 2 základních režimech: "Simple pass" (jednoduchý průchod) a "Through pass" (průchod napříč). Jednoduchý průchod projde všechny trojúhelníky ve scéně a na základě tabulky sousedů trojúhelníků porovná plochy všech čtyřúhelníků, které může

Název souboru	Název modelu	Počet vrcholů	Počet trojúhelníků
hole.tri	Hole	13	16
sphere.tri	sphere	146	288
defkoule.tri	defkoule	314	624
cow.tri	The canonical cow	2 905	5 804
whead.tri	womans head	4 098	8 120
demi.tri	demi	9 138	17 506
teeth.tri	teeth	29 166	58 328
bunny.tri	Stanford Bunny	35 947	69 451
horse.tri	horse	48 485	96 966
bone.tri	bone	68 537	137 072
bone2.tri	Visible Man Bone2	177 907	355 511
bell.tri	bell	213 373	426 572
hand.tri	Skeleton Hand	327 323	654 666
dragon.tri	dragon	437 645	871 414
ok_blade.tri	Turbine Blade	882 954	1 782 518

Tabulka 5.3: Seznam testovaných modelů

daný trojúhelník se svými sousedy sestavit. Je-li plocha jednoho či více z čtyřúhelníků vytvořených prohozením úhlopříček menší než plocha původního čtyřúhelníku, jsou trojúhelníky tvořící tento čtyřúhelník prohozeny. Na obr. 5.1 je jednoduchý příklad trojúhelníkové sítě a tab. 5.4 obsahuje část tabulky sousedů trojúhelníků z obr. 5.1. V tomto konkrétním případě by se při optimalizaci jednoduchým průchodem pro trojúhelník *A* srovnávaly plochy čtyřúhelníků tvořených trojúhelníkovými páry *AD, AB, AC.*



Obrázek 5.1: Modelová trojúhel. síť

Trojúh.	Sousedé trojúh.		
A	D	B	C
B	A	E	H
C	A	J	G
D	A	F	E

Tabulka 5.4: Tabulka sousedů trojúhelníků z obr. 5.1

Na rozdíl od metody jednoduchého průchodu metoda průchodu napříč neporovnává pouze velikosti čtyřúhelníků tvořených sousedy právě testovaného trojúhelníku, ale testuje plochy všech kombinací čtyřúhelníků, které může testovaný trojúhelník a všichni jeho sousedé v danou chvíli vytvořit. Tj. pokud testujeme

trojúhelník *A*, pak porovnáváme úbytky ploch čtyřúhelníků *BA, BE, BH, CA, CJ, CG, DA, DF, DE*. Tato metoda se může na první pohled zdát příliš překomplikovaná a zdlouhavá, dosahuje však lepších výsledků než metoda jednoduchého průchodu.

Algoritmus ovlivňují ještě další dva přepínače. Přepínač "Backward tracing" (zpětné sledování) říká algoritmu, zda má porovnávat pro daný trojúhelník i jeho sousedy, kteří mají index menší než testovaný trojúhelník. Tj. pokud testujeme trojúhelník *D* a přepínač "Backward tracing" není zvolen, pro minimalizaci se budou zjišťovat plochy pouze u čtyřúhelníků *DF a DE*, protože trojúhelník *A* má menší index než testovaný trojúhelník *D*.

Je-li zvolen přepínač "Optimize over obtuse edges" (optimalizovat přes tupé hrany), algoritmus bere v úvahu i čtyřúhelníky, které jsou tvořeny trojúhelníky svírajícími spolu úhel větší než 90°. Tato varianta je v programu uvedena pouze z experimentálních důvodů – ukazuje, že největší zisky v optimalizaci jsou dosahovány právě u čtyřúhelníků s tupým úhlem. Čím blíže je úhel mezi trojúhelníky tvořícími daný čtyřúhelník k hodnotě 180°, tím větší je úbytek plochy (tím větší ovšem také vznikají vizuální chyby v optimalizovaném modelu).

5.4 Vyhodnocení naměřených hodnot

Kompletní test všech druhů optimalizace byl proveden na dvou modelech. Model krávy (The canonical cow) byl vybrán proto, že kromě dobré minimalizovatelnosti jeho plochy je u něj možné i vizuálně srovnat modely před a po optimalizaci, neboť model sestává pouze z 5 804 trojúhelníků. Model lopatky turbíny (Turbine Blade) je nejsložitějším testovaným modelem (1 782 518 trojúhelníků). Je tedy vhodný pro testování, neboť důkladně prověří rychlost a kvalitu implementace algoritmu. Všechny přiložené tabulky obsahují tři důležitá kritéria hodnocení výsledků optimalizace – počet prohozených trojúhelníků, rozdíl v ploše, který vytvořily provedené změny, a délka běhu optimalizace. Rozdíl v ploše závisí na celkové ploše modelu. Nelze tedy srovnávat tyto rozdíly v rámci obou testovaných modelů, neboť celková plocha modelu lopatky turbíny je několikanásobně větší než plocha modelu krávy.

Tabulky 5.5 – 5.8 obsahují hodnoty naměřené pro model krávy. Grafy připojené v příloze B zobrazují srovnání některých kritérií optimalizace. Následující srovnání budou prováděna pro algoritmus bez optimalizace přes tupé hrany (tj. tab. 5.5 a 5.6).

Srovnáním výsledků z tab. 5.7 a 5.8 bychom došli ke zhruba stejným výsledkům, díky optimalizaci přes tupé hrany by však naměřené hodnoty byly vyšší.

Počty výměn jsou u všech čtyř testů zhruba stejné, což ukazuje na fakt, že existuje určité omezené množství trojúhelníků, které jsou vhodné pro optimalizaci. Rozdíl v dosažených počtech výměn se pohybuje v řádech jednotek až stovek prakticky u všech testovaných modelů. V této kategorii tedy "vítězný" druh optimalizace určit nelze. Jiná je již situace u naměřených rozdílů (differences), kde s jasnou převahou dosáhl největšího rozdílu algoritmus "Through pass". Ostatní tři naměřené hodnoty jsou zhruba na stejné úrovni. To poukazuje na fakt, že získaný rozdíl závisí na pořadí, v jakém jsou trojúhelníky prohazovány, tj. že rozdíl závisí na zvoleném druhu optimalizace. Nejlepší druh optimalizace (v rámci rozdílového kritéria) však určit nelze, neboť jak ukáže testovaný model lopatky turbíny, nejlepší druh optimalizace se mění model od modelu. Posledním testovaným kritériem je dosažený čas. U tohoto kritéria vychází nejlépe metoda "Simple pass" (s malým nárůstem času pak v kombinaci se zpětným sledováním). Toto je trend, který je stejný u všech modelů, neboť metoda "Through pass" porovnává 3x více čtyřúhelníků než metoda "Simple pass". Zajímavější z hlediska použitelnosti je ale počet N průchodů, které bylo nutné provést k dosažení „dokonalé" optimalizace – tj. stavu, kdy již nelze zoptimalizovat žádný pár trojúhelníků. Stejně jako u modelu krávy, i u všech ostatních modelů je nejrychlejší z hlediska počtu průchodů algoritmus "Through pass" se zapnutým zpětným sledováním. U drtivé většiny modelů je tento druh algoritmu schopen provést úplnou minimalizaci pouze ve dvou průchodech a i když doba trvání jednoho průchodu je druhou nejhorší, jeho nespornou výhodou zůstává fakt, že na rozdíl od ostatních algoritmů, kde dopředu nevíme, kolik průchodů bude třeba pro úplnou minimalizaci, algoritmus T.P. + B.T. provede úplnou minimalizaci u naprosté většiny případů pouze ve dvou průchodech.

Druhý testovaný model (tab. 5.9 – 5.12) poukazuje na stejné závěry, jako byly vyvozeny u prvního testovaného modelu. Počet výměn je zhruba stejný u všech metod a nejdelší naměřený čas byl opět zjištěn u metody "Through pass". Na rozdíl od prvního testovaného modelu však u modelu lopatky turbíny dosáhl největšího rozdílu v ploše algoritmus "Simple pass". Tento maximální rozdíl ale nepřevyšuje rozdíly u ostatních metod takovým způsobem, jako tomu bylo u modelu krávy. Stejně tak i v počtu průchodů zůstává nejvýhodnějším algoritmus "Through pass" se zapnutým

zpětným sledováním. Tato metoda dosahuje úplné minimalizace již ve dvou průchodech, a to i přesto, že testovaný model obsahuje skoro 1,8 milionu trojúhelníků.

N	Simple pass			Simple pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	1440	0.0025246308	45.29820	1555	0.0033294779	65.20296
2	158	0.0007262147	29.67024	71	0.0005202181	46.64759
3	23	0.0004268170	26.85843	6	0.0000651766	45.63093
4	10	0.0001550312	26.71847	2	0.0000002507	45.35016
5	2	0.0000584702	26.45196	0	0.0000000000	-
6	1	0.0000001548	26.43017	0	0.0000000000	-
Celkem	1634	0.0038913188	181.42747	1634	0.0039151233	202.83164

Tabulka 5.5: Naměřené hodnoty pro model "Cow", bez optimalizace přes tupé hrany – 1.část

N	Through pass			Through pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	1479	0.0032701519	81.63968	1626	0.0038921987	135.51830
2	150	0.0008840091	64.22071	11	0.0000252259	114.08653
3	9	0.0000237713	62.21264	0	0.0000000000	-
4	3	0.0000073057	61.53546	0	0.0000000000	-
5	1	0.0000000959	61.45668	0	0.0000000000	-
6	0	0.0000000000	-	0	0.0000000000	-
Celkem	1642	0.0041853339	331.06517	1637	0.0039174246	249.60483

Tabulka 5.6: Naměřené hodnoty pro model "Cow", bez optimalizace přes tupé hrany – 2.část

N	Simple pass			Simple pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	1452	0.0032888247	45.85386	1577	0.0043417568	65.72509
2	177	0.0011409360	29.70796	89	0.0006340245	46.55367
3	31	0.0005237117	27.11992	7	0.0000651766	44.32601
4	11	0.0001755105	27.60434	2	0.0000002507	44.09721
5	2	0.0000584702	26.59611	0	0.0000000000	-
6	1	0.0000001548	26.54582	0	0.0000000000	-
Celkem	1674	0.0051876080	183.42801	1675	0.0050412085	200.70198

Tabulka 5.7: Naměřené hodnoty pro model "Cow", s optimalizací přes tupé hrany – 1.část

N	Through pass			Through pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	1512	0.0044114189	81.22734	1666	0.0051884879	136.94642
2	160	0.0010482506	65.59100	12	0.0000252259	115.95213
3	13	0.0000251222	61.93690	0	0.0000000000	-
4	3	0.0000073057	61.88159	0	0.0000000000	-
5	1	0.0000000959	61.55054	0	0.0000000000	-
6	0	0.0000000000	-	0	0.0000000000	-
Celkem	1689	0.0054921932	332.18737	1678	0.0052137138	252.89855

Tabulka 5.8: Naměřené hodnoty pro model "Cow", s optimalizací přes tupé hrany – 2.část

N	Simple pass			Simple pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	334147	130.23457	11197.03312	349971	168.81614	15364.54765
2	25518	49.86143	6231.50086	9901	15.48116	10843.70308
3	189	4.37743	5784.41941	44	0.43496	10635.53865
4	27	0.57055	5780.77180	19	0.07590	10691.29488
Celkem	359881	185.04399	28993.72519	359935	184.80818	47535.08426

Tabulka 5.9: Naměřené hodnoty pro model "Turbine Blade", bez optimalizace přes tupé hrany – 1.část

N	Through pass			Through pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	355131	175.53912	21977.13210	358753	184.39986	35792.15884
2	3613	8.81198	15841.88743	60	0.08250	30647.39526
3	54	0.36259	15724.83685	0	0.00000	-
4	29	0.03374	15725.10526	0	0.00000	-
Celkem	358827	184.74743	69268.96164	358813	184.48236	66439.55410

Tabulka 5.10: Naměřené hodnoty pro model "Turbine Blade", bez optimalizace přes tupé hrany – 2.část

N	Simple pass			Simple pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	334387	137.37871	10861.75256	350362	177.64555	15496.71012
2	25710	53.99092	6247.41421	10191	19.46943	10851.76801
3	334	5.86801	5800.98792	293	1.67958	10659.12816
4	156	1.84136	5794.36851	263	0.07590	10726.91461
Celkem	360587	199.07900	28704.52320	361109	198.87047	47734.52090

Tabulka 5.11: Naměřené hodnoty pro model "Turbine Blade", s optimalizací přes tupé hrany – 1.část

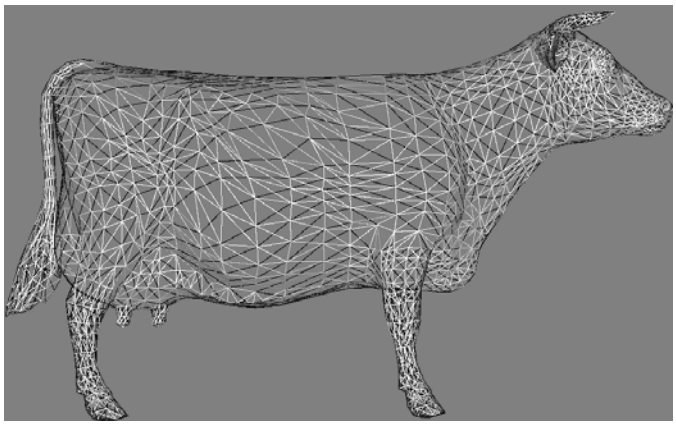
N	Through pass			Through pass + Backward tracing		
	Počet výměn	Rozdíl	Čas	Počet výměn	Rozdíl	Čas
1	355625	186.75973	20942.89629	359599	197.42506	36229.98729
2	4022	10.55998	15891.00913	783	1.23106	31566.66167
3	442	1.56775	15777.25880	0	0.00000	-
4	422	0.03374	15870.68166	0	0.00000	-
Celkem	360511	198.92119	68481.84588	360382	198.65612	67796.64896

Tabulka 5.12: Naměřené hodnoty pro model "Turbine Blade", s optimalizací přes tupé hrany – 2.část

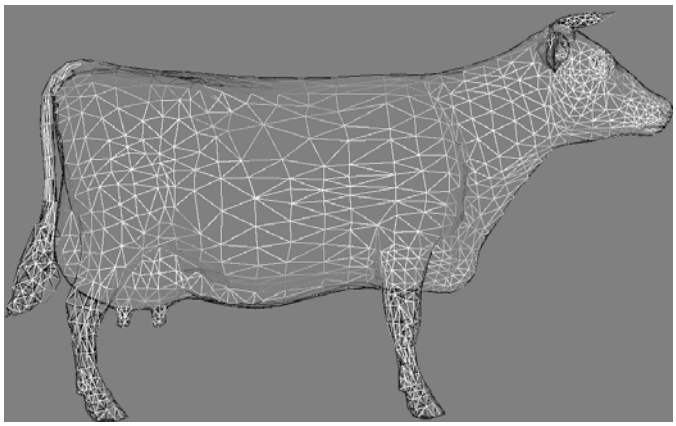
5.5 Vizuální aspekty minimalizace

Minimalizace povrchu trojúhelníkové sítě funguje na principu výměny trojúhelníků. Z tohoto důvodu dochází ke změnám (ať už žádoucím, nebo nežádoucím) vzhledu optimalizovaného modelu.

Ve většině případů dochází k výměně trojúhelníků, které spolu svírají jen velmi malý úhel. V tomto případě zůstává vzhled modelu skoro stejný. Obrázek 5.2 ukazuje drátěný model krávy před optimalizací. Během optimalizace dojde k výměně několika



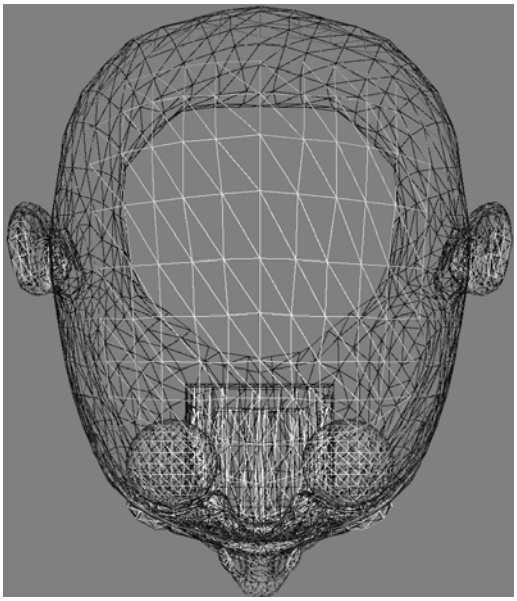
Obrázek 5.2: Drátěný model krávy před optimalizací



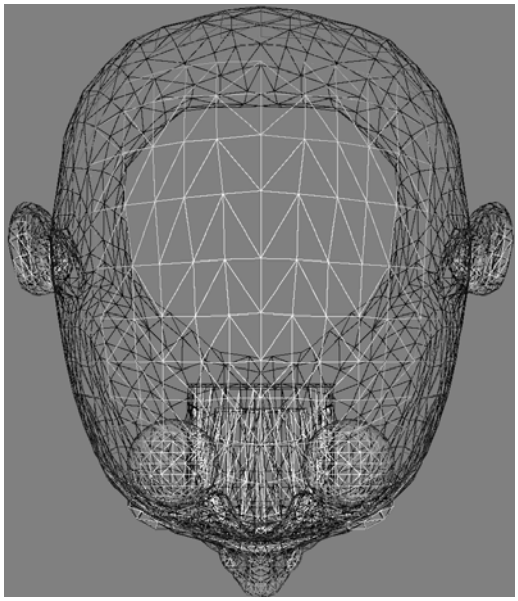
Obrázek 5.3: Drátěný model krávy po optimalizací

set trojúhelníků, což je patrné z obrázku 5.3, kde je stejný drátěný model krávy, ale již po optimalizaci. Při porovnání obou obrázků je vidět na přední straně modelu krávy mnoho prohozených trojúhelníků. Ještě více jsou tyto změny vidět při srovnání

obrázků 5.4 a 5.5, které obsahují model ženské hlavy před a po optimalizaci. Na horní straně lebky jsou jasně patrné všechny výměny úhlopříček.



Obrázek 5.4: Drátěný model ženské hlavy před optimalizací

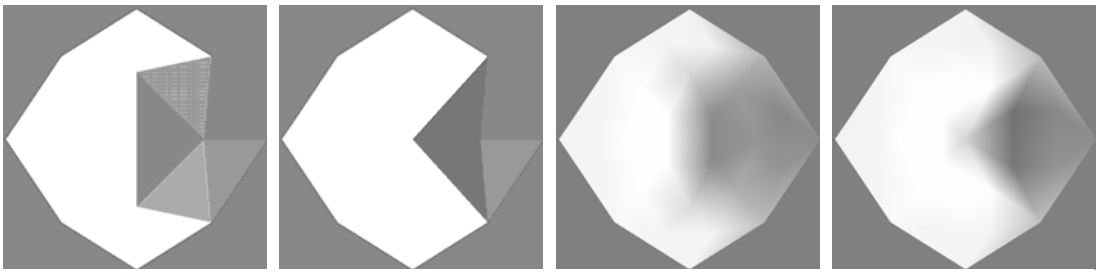


Obrázek 5.5: Drátěný model ženské hlavy po optimalizací

Na těchto modelech se ukazuje i další vlastnost minimalizace trojúhelníkové sítě, a tou je symetrizace modelů. U modelu krávy před optimalizací jsou mezi trojúhelníky tvořícími přední stranu modelu vykresleny tmavou barvou i hrany trojúhelníků tvořících zadní stranu modelu. Když se ale podíváme na model po optimalizaci, žádné tmavé čáry se na břiše krávy nevyskytují. Pokud bychom optimalizovaný model pootočili o pár stupňů kolem libovolné z os, ze světlých čar tvořících přední stranu modelu by se "vynořily" i čáry tmavé, které byly před natočením díky symetričnosti modelu skryty za čarami světlými. Podobný efekt je vidět i na modelu ženské hlavy. U optimalizovaného modelu je většina hran horní části lebky symetrická přes pomyslnou čáru uprostřed lebky.

V případech, kdy úhel, který spolu svírají dva vyměňované trojúhelníky, je větší než pouhých pár stupňů, dochází ke změně vzhledu trojúhelníkového páru. Při každé změně se přepočítávají jak normály trojúhelníků, tak i normály vrcholů. Tyto normály poté určují, jak bude daný trojúhelník vystínován. Pokud se tedy markantně změní směr normály, změní se viditelným způsobem i vystínování daného trojúhelníku. Tento vedlejší efekt je patrný z obr. 5.6, který obsahuje modely díry

před a po optimalizaci, vystínované konstantním stínováním a stínováním S-shading. Díra je patrná tmavým stínováním vpravo uprostřed plochy. Optimalizace prohodí 2 páry trojúhelníků. Jeden z nich je pár, který spojuje nová hrana, která vede z bodu uprostřed plochy do bodu vpravo mezi body nejvíce nahoře a nejvíce vpravo. Srovnáním oblasti okolo této nové hrany na prvním a druhém obrázku je jasné, že se vzhled v této části modelu dosti výrazně změnil. Stejně tak lze změnu pozorovat i u hladkého stínování S-shading, kde se změnila jak velikost, tak i barva tmavé plochy.

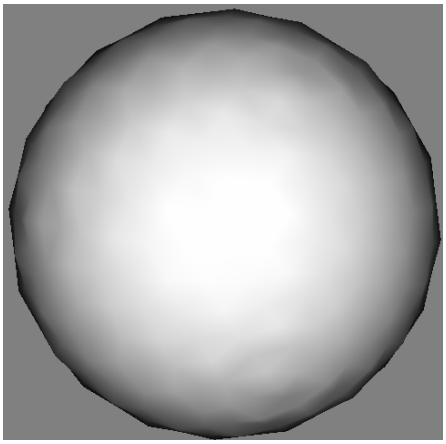


Obrázek 5.6: Vystínování modelu plochy s dírou vpravo uprostřed. První dva obrázky zobrazují model vystínovaný konstantním stínováním před a po optimalizaci, druhé dva obrázky zobrazují model vystínovaný stínováním S-shading také před a po optimalizaci

5.6 Vyhodnocení implementace metody S-shading

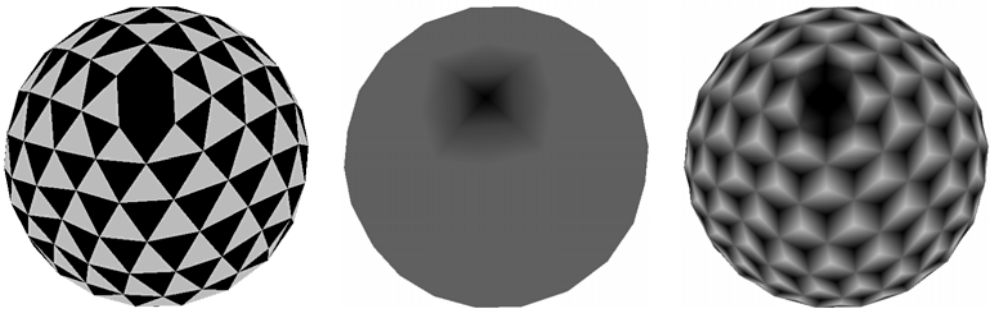
Základním požadavkem na metodu S-shading bylo vytvoření metody stínování, která by vyhlazovala přechody mezi ploškami v trojúhelníkové síti, ale zároveň aby toto stínování neztrácelo některé důležité detaily, které ztrácí metoda Gouraud shading.

První prezentovaný model je klasický model koule složený z 288 trojúhelníků. Na první pohled jsou na modelu na obr. 5.7 vidět dvě základní vlastnosti metody S-shading, které se projevují i u ostatních modelů. 1) Stínování S-shading je hladké, tj. přechody mezi barvami jsou plynulé. 2) Metoda S-shading nevyhlazuje úplně přechody mezi trojúhelníky, ale u modelů s nízkým počtem trojúhelníků vytváří jakousi zdánlivou plastičnost modelu. Např. model koule pak vypadá trochu jako golfový míček.



Obrázek 5.7: Model koule vystínovaný metodou S-shading

Další požadovanou vlastností bylo zachování barvy bodů v reprezentativních bodech trojúhelníků. Pro demonstraci tohoto problému byl vybrán opět model koule, tentokráte však speciálně upravený tak, že stínování nebylo prováděno vyhodnocováním osvětlovacího modelu, ale byly dány přímo barvy trojúhelníků ve svých reprezentativních bodech tak, aby trojúhelníková síť měla podobu šachovnice. Jedna barva šachovnice je černá, druhá je světle šedá. Pouze na obou pólech je ponecháno více černých trojúhelníků dohromady, aby se na modelu nacházely nějaké styčné plochy. Obrázek 5.8 obsahuje tento model vystínovaný všemi třemi implementovanými druhy stínování. Vlevo je konstantní stínování. Pro tento



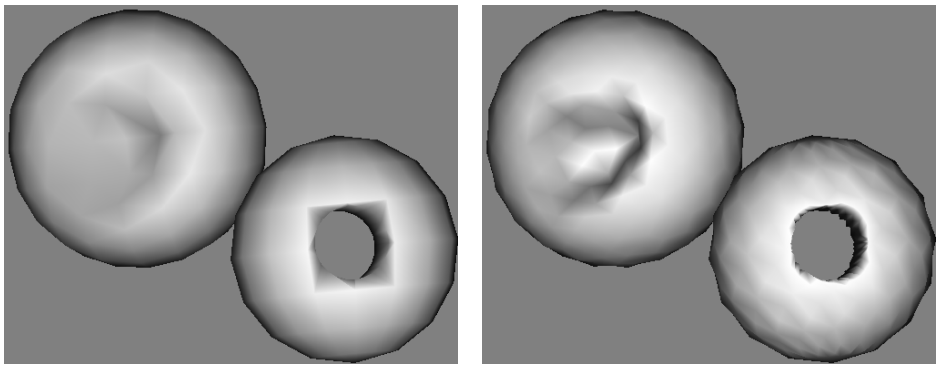
Obrázek 5.8: Vystínovaný speciální model koule - zleva: konstantním stínováním, Gouraudovo stínováním a stínováním S-shading.

konkrétní model je toto stínování z hlediska korektnosti zobrazení nejvhodnější, neboť přesně zachovává požadované barvy trojúhelníků. Konstantní stínování nám ale neposkytuje požadovanou plynulost barevných přechodů.

Prostřední model je vystínován Gouraudovým stínováním. Zde se projevuje jedna ze slabostí Gouraudova stínování. Z obrázku lze vidět, že prakticky celý model je vystínován jedinou barvou, a tou je průměrná barva po sečtení barvy černé a světle šedé, tj. tmavě šedá. Pouze na pólech, kde se vyskytují úmyslné nepravidelnosti, je stínování tmavší. Toto je velice závažný nedostatek, neboť v případě, že máme spočteny barvy trojúhelníků, chceme, aby tyto barvy zůstaly na vystínovaném modelu obsaženy. Gouraudovo stínování ale původní barvy "slije" dohromady a zobrazí jejich průměr, což je nesprávný a nežádoucí výsledek.

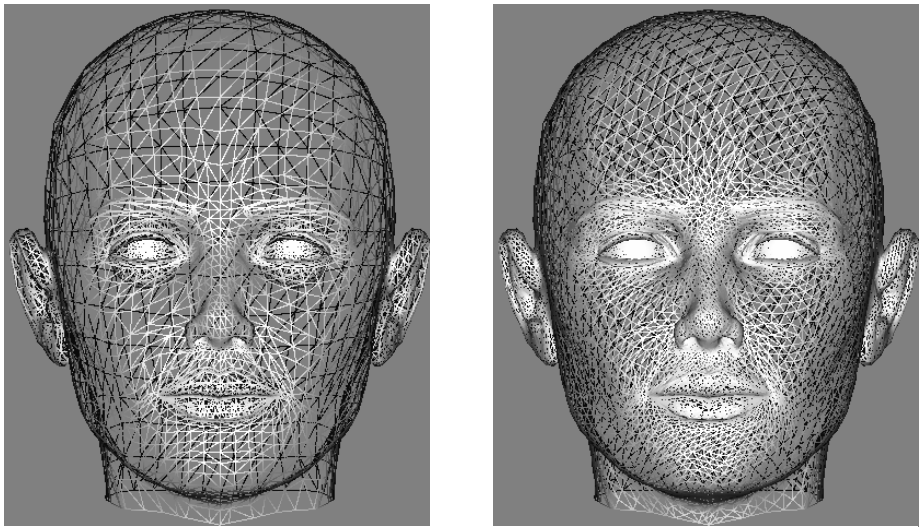
Předkládaná metoda S-shading (na obr. 5.8 vpravo), na rozdíl od dvou předchozích metod, splňuje oba dva požadavky, tj. přechody mezi barvami jsou plynulé a barvy bodů v reprezentativních bodech zůstávají zachovány. Je zajímavé, že i u tohoto modelu vzniká dojem jakési plastičnosti, který se ještě prohlubuje ve chvíli, kdy bychom modelem otáčeli.

Další rozdíl mezi metodou Gouraud a S-shading je vidět na obr. 5.9. Tento obrázek zobrazuje model deformovaných koulí. Vlevo je model stínovaný metodou Gouraud, vpravo metodou S-shading. Jedna z koulí je "vmáčknuta", druhá má po celém svém průměru válcovitý otvor. Rozdíl mezi metodou Gouraud a S-shading je ve způsobu zaoblení hran. U levé koule stínování S-shading lépe zachycuje značnou nerovnost povrchu tím, že oblast spádu zobrazí velmi tmavou barvou. Naopak metoda Gouraud shading zde vytvoří jen velmi malou barevnou změnu, a pokud bychom neměli možnost modelem otáčet, určitě bychom z tohoto vyobrazení nepoznali, že deformace koule je poměrně hluboká a její stěny jsou strmé. U pravé koule je naopak



Obrázek 5.9: Model deformovaných koulí. Vlevo je použito Gouraudovo stínování, vpravo stínování S-shading.

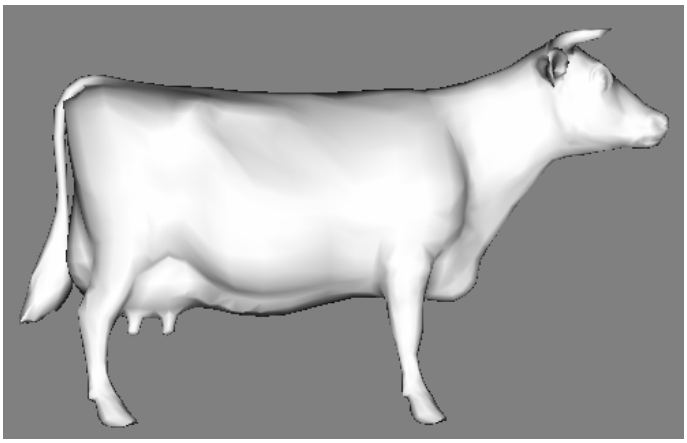
vystínování u Gourauda rozsáhlejší než u metody S-shading. V tomto případě je to však opět jev nežádoucí, neboť se jedná o hranu, která by měla být ostrá, bez stínování. Menší plocha stínování okraje válcového otvoru u metody S-shading je dána tím, že metoda S-shading fakticky rozděluje každý původní trojúhelník na 3 menší, které pak zobrazuje pomocí OpenGL (viz obr. 5.10). Toto zjemnění trojúhelníkové sítě má v místech, jako je otvor u modelu deformované koule za následek lepší vystínování hranových přechodů.



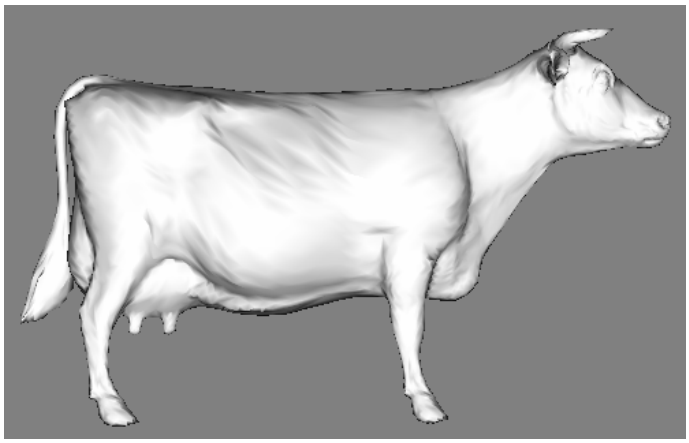
Obrázek 5.10: Trojúhelníková síť modelu ženské hlavy. Vlevo je původní síť metody Gouraud shading, vpravo je 3x zjemněná síť metody S-shading.

Posledním zde diskutovaným rozdílem mezi oběma porovnávanými metodami je schopnost zdůrazňovat drobné a větší nerovnosti v modelech reprezentovaných trojúhelníkovou sítí. Jak už bylo řečeno dříve, velkou nevýhodou metody Gouraud je

v některých případech až přílišné vyhlazování přechodů mezi trojúhelníky a tím i některých důležitých detailů. Obr. 5.11 obsahuje model krávy vystínovaný Gouraudovým stínováním, obr. 5.12 zobrazuje stejný model, tentokráte však vystínovaný pomocí stínování S-shading. Na obr. 5.11 je opravdu vidět, že metoda Gouraud vyhlazuje skoro všechny nerovnosti. Naopak metoda S-shading správně vystínuje většinu drobných nerovností tak, že nijak moc neruší vzhled zobrazovaného modelu, ale zároveň jsou dobře patrné. Fakt, že metoda S-shading zdůrazňuje drobné nerovnosti, nemusí být vždy výhodou. V některých případech jistě platí, že čím hladší je povrch, tím lépe. V těchto případech je dozajista výhodnější použít Gouraudovo stínování. V jiných případech, kde jsou ale drobné detaily kritickou součástí modelů, může opomenutí zobrazení takovýchto drobných detailů vést k nebezpečným omylům.



Obrázek 5.11: Model krávy – Gouraudovo stínování



Obrázek 5.12: Model krávy – stínování S-shading

6 Závěrečné zhodnocení

Bakalářská práce sestávala ze dvou hlavních úkolů: 1) implementovat algoritmus pro optimalizaci trojúhelníkové sítě pro kritérium minimální plochy a vyhodnotit výsledky této metody, 2) implementovat novou metodu stínování S-shading a porovnat její vlastnosti s metodou Gouraud shading. Implementaci obou algoritmů předcházelo seznámení se s modely reprezentovanými trojúhelníkovými sítěmi a naprogramování modulu pro načítání TRI souborů. Všechny tyto tři základní úkoly byly splněny. Aplikace, dodávaná s touto prací, obsahuje modul pro načítání TRI souborů a jejich předzpracování (počítání těžišť trojúhelníků a normál trojúhelníků a vrcholů, zjišťování sousedů trojúhelníků pro algoritmus minimalizace a řazení trojúhelníků, incidujících s určitým vrcholem, pro algoritmus stínování pomocí metody S-shading), modul pro minimalizaci trojúhelníkové sítě a modul pro vykreslování trojúhelníkových modelů pomocí metod Flat, Gouraud a S-shading.

Algoritmus minimalizace trojúhelníkové sítě byl realizován ve čtyřech základních variantách, lišících se hlavně rychlostí a počtem průchodů nutných pro úplnou minimalizaci povrchů modelů. Ostatní testované parametry nabývaly u většiny modelů pro všechny čtyři varianty algoritmu velice podobných hodnot. Algoritmus minimalizace sítě byl primárně optimalizován na rychlost, tudíž i pro největší testovaný model, obsahující 1.8 milionu trojúhelníků, byla maximální doba jednoho průchodu rovna 36 sekundám, což je, vzhledem ke komplexnosti modelu, velice dobrý výsledek.

Zobrazování modelů pomocí implementovaných metod stínování fungovalo díky použití OpenGL prakticky bez problémů. Jediný zádrhel nastal u velkých modelů, kde OpenGL při vytváření call listu potřebuje velké množství paměti, a pokud tuto paměť nemá k dispozici, program skončí s chybou. Proto musely být velké modely (od 150 tisíc trojúhelníků výš) testovány a screenovány na jiném počítači než modely menší.

Literatura

[Gour71] Gouraud, H.: Continuous shading of curved surfaces, IEEE trans. on Computers, 1971

[Pol80] Polák, J.: Přehled středoškolské matematiky, SPN, 1980

[Skal92] Skala, V.: Algoritmy počítačové grafiky I-III, skripta ZČU, 1992

[Skal00] Skala, V.: Fast Smooth Shading Algorithm, draft version, 2000

[Zar98] Žára J., Beneš B., Felkel P.: Moderní počítačová grafika, Computer Press, 1998