MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Aplikace metod raytracingu na molekulární systémy

DIPLOMOVÁ PRÁCE

Bc. František Dvořáček

Brno, podzim 2013

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Bc. František Dvořáček

Vedoucí práce: RNDr. Barbora Kozlíková, Ph.D.

Poděkování

Rád bych na tomto místě poděkoval vedoucí své diplomové práce RNDr. Barboře Kozlíkové, Ph.D. a Mgr. Vilému Šustrovi za jejich trpělivost, cenné rady a pomoc při tvorbě této práce.

Shrnutí

Cílem této práce je popsat implementaci raytraceru, který umožní vytvořit rendery molekul proteinů a jejich tunelů ve vysokém rozlišení. Implementace podporuje vykreslení molekul a jejich tunelů pomocí nejrozšířenějších a nejčastěji používaných metod zobrazení. Z důvodů snížení časové náročnosti algoritmu je pro organizaci scény použita třídimenzionální mřížka. Vytvořený raytracer byl integrován do aplikace CAVER Analyst.

Klíčová slova

raytracing, vrhání paprsku, molekula, protein, tunel, objemová data, trojúhelník, koule, válec, kvádr

Obsah

1 **Úvod** . . . 3
2 **Metody zobrazení proteinů** . . . 5
2.1 Typy zobrazení molekul . . . 5
2.1.1 Van der Waals (VdW) . . . 5
2.1.2 Tyčinky a koule . . . 6
2.1.3 Tyčinky . . . 6
2.1.4 Povrch molekuly . . . 7
2.2 Typy zobrazení tunelů . . . 7
2.2.1 Středová osa (Centerline) . . . 8
2.2.2 Koule . . . 9
2.2.3 Povrch . . . 9
3 **Raytracing** . . . 10
3.1 Algoritmus . . . 10
3.2 Implementace raytraceru . . . 12
3.2.1 Struktura raytraceru . . . 13
4 **Vrhání paprsků** . . . 15
4.1 Paprsek . . . 15
4.2 Kamera . . . 15
4.3 Primární paprsky . . . 17
4.4 Supersampling . . . 19
4.5 Adaptivní vzorkování . . . 21
4.6 Adaptivní supersampling . . . 22
5 **Nalezení průsečíku** . . . 24
5.1 Scéna . . . 24
5.2 Trojúhelník . . . 24
5.2.1 Definice trojúhelníku . . . 24
5.2.2 Barycentrické souřadnice . . . 25
5.2.3 Nalezení průsečíku . . . 25
5.2.4 Implementace . . . 28
5.2.5 Výpočet normály . . . 29
5.3 Koule . . . 30
5.3.1 Definice koule . . . 30
5.3.2 Geometrické řešení . . . 30
5.3.3 Výpočet normály . . . 34
5.3.4 Implementace . . . 34

5.4 Válec 35
5.4.1 Nalezení průsečíku 35
5.4.2 Výpočet normály 38
5.5 Osově orientovaný kvádr 38
5.5.1 Nalezení průsečíku 38
5.5.2 Implementace 41
6 **Výpočet barvy** 43
6.1 Phongův osvětlovací model 43
6.1.1 Materiál 43
6.1.2 Ambientní složka 44
6.1.3 Difúzní složka 44
6.1.4 Spekulární složka 46
6.2 Výpočet stínů 47
7 **Zobrazení objemových dat** 48
7.1 Konstrukce mřížky 48
7.2 Procházení mřížkou 49
7.2.1 Algoritmus 3D-DDA 49
8 **Urychlující struktury** 55
8.1 Vytvoření mřížky 55
8.2 Vložení objektů 56
8.3 Nalezení nejbližšího průsečíku 57
8.4 Porovnání rychlosti 58
9 **Závěr** 59
10 **Příloha** 64

1 Úvod

Struktury proteinů jsou základním stavebním kamenem všech žijících organizmů. Najdeme je v každé z buněk, z nichž se tyto organizmy skládají. Výzkumu zaměřenému na proteiny se tedy věnuje značná pozornost.

Proteiny jsou struktury, představované molekulami, které se mohou skládat z tisíců jednotlivých atomů propojených kovalentními vazbami. Pro správnou představu o struktuře proteinů je tedy potřeba nalézt vhodné nástroje pro jejich zobrazení.

V současné době se pro zobrazení molekul proteinů používá kombinace dvou technik. Pro vykreslování v reálném čase se používají techniky využívající rasterizaci (například OpenGL). Pro vytvoření statických renderů ve vyšší kvalitě a rozlišení se pak využívá raytracing. Vyšší kvalita zobrazení je požadována například při prezentaci a vkládání renderů proteinů do vědeckých prací.

Cílem mé diplomové práce bylo vytvořit implementaci raytracingu, která by umožňovala vytvoření renderů ve vysokém rozlišení pro jednotlivé metody zobrazení používané v aplikaci CAVER Analyst.

Textová část práce se zaměřuje na jednotlivé techniky použité při této implementaci.

Kapitola 2 popisuje metody zobrazení podporované vytvořeným raytracerem. Text této kapitoly je doplněn o rendery proteinů vytvořené pomocí implementovaného algoritmu.

V první části kapitoly 3 představuji základní algoritmus pro zobrazení pomocí raytracingu. V části druhé popíši strukturu implementovaného raytraceru postaveného na tomto algoritmu.

Kapitola 4 se věnuje různým metodám vzorkování scény. Na použité vzorkovací metodě závisí kvalita zobrazení a doba potřebná pro vykreslení scény.

Základní kámen raytracingu, nalezení průsečíků paprsků s objekty ve scéně, je popsán v kapitole 5. V této kapitole se věnuji detailnímu popisu technik pro nalezení průsečíku paprsku s jednotlivými základními objekty, používanými pro vytvoření komplexních scén.

Kapitola 6 uvádí využití Phongova osvětlovacího modelu a stínových paprsků pro výpočet obarvení objektů ve scéně.

Raytracing lze využít i pro zobrazení objemových dat. Této tématice se věnuje kapitola 7.

Kapitola 8 se zaměřuje na vytvoření urychlující struktury pro organizaci scény. S její pomocí je dosaženo snížení časové náročnosti algoritmu.

V závěru práce hodnotím dosažené výsledky a navrhuji možná rozšíření raytraceru.

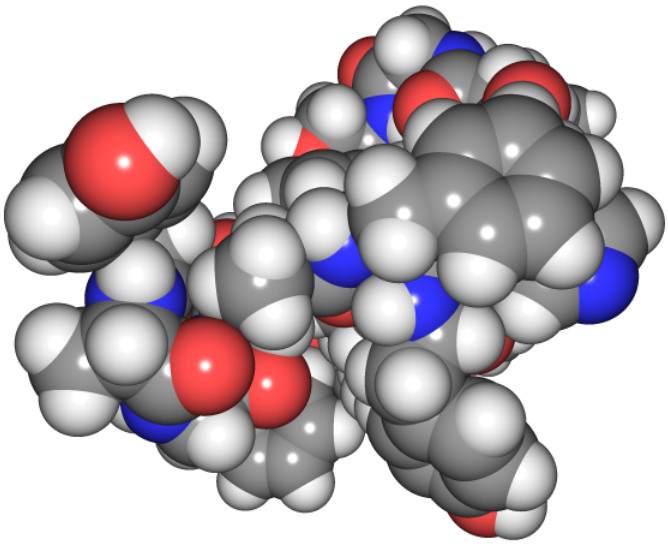
2 Metody zobrazení proteinů

Základní stavební kameny každé molekuly jsou atomy propojené kovalentními vazbami. Existuje několik způsobů jak zobrazovat proteiny. Níže uvedené metody byly implementovány v rámci mé diplomové práce. Všechny obrázky použité v této kapitole byly vyrenderovány pomocí vytvořeného raytraceru.

2.1 Typy zobrazení molekul

2.1.1 Van der Waals (VdW)

Při VdW zobrazení jsou atomy vykresleny jako koule, obrázek 2.1. Jejich velikost je dána van der Waalsovým poloměrem [6]. Vzdálenost mezi atomy, propojenými kovalentními vazbami, je menší než součet jejich poloměrů. Proto se koule použité při VdW zobrazení navzájem pronikají a vazby mezi nimi nejsou viditelné. Barvy atomů reprezentují chemický prvek, který dané atomy představují. Například pro zobrazení atomů kyslíku se používá červená barva. VdW zobrazení se hodí pro představu o celkovém objemu molekuly.

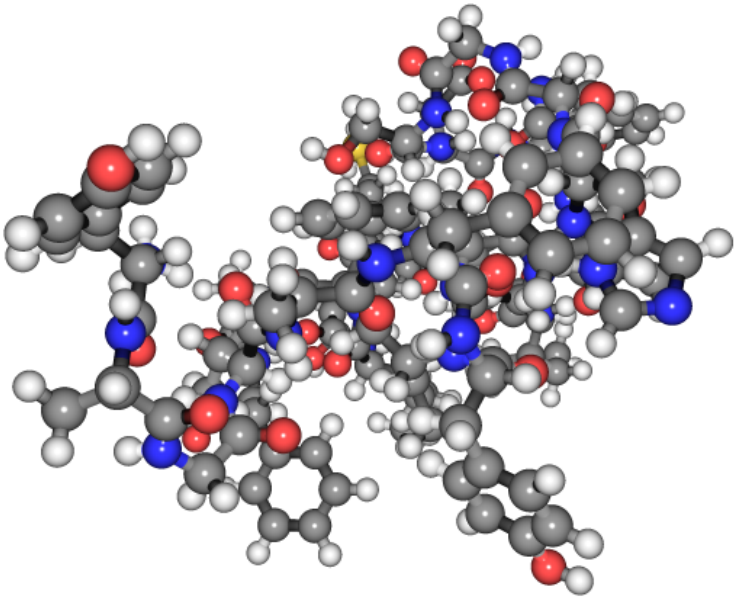


Obrázek 2.1: Molekula 1edw, zobrazena pomocí VdW

2.1.2 Tyčinky a koule

Na obrázku 2.2 jsou vykresleny jednotlivé atomy a vazby mezi nimi. Tomuto zobrazení se říká *tyčinky a koule*. Atomy jsou reprezentovány pomocí koulí. Jejich velikost odpovídá jedné třetině *van der Waalsova poloměru*.

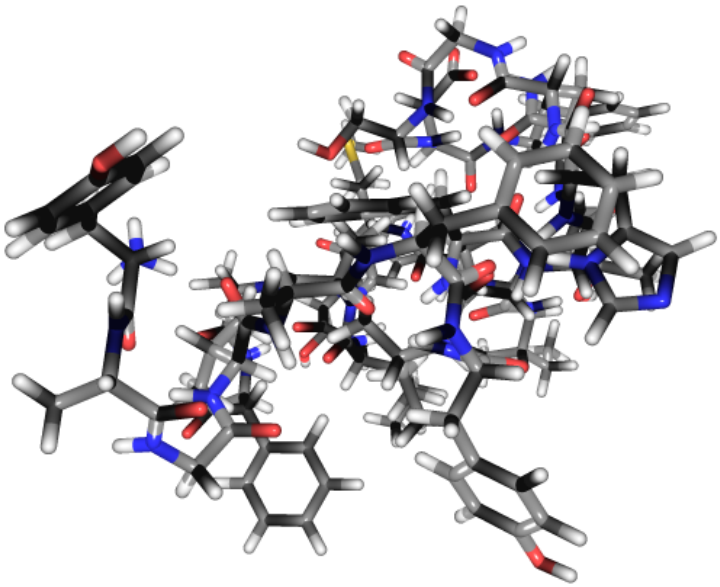
Kovalentní vazby jsou vykresleny pomocí válců s malým průměrem. Jejich jednotlivé poloviny jsou obarveny podle koncových atomů, které propojují.



Obrázek 2.2: Molekula 1edw, zobrazena pomocí tyčinek a koulí.

2.1.3 Tyčinky

Snadnější pochopení celkové struktury proteinu umožňuje zobrazení molekuly jen jako množiny vazeb mezi jednotlivými atomy, obrázek 2.3. Tyto vazby jsou vykresleny pomocí válců s malým poloměrem. Barva jednotlivých konců vazeb odpovídá barvě atomů, které propojují, stejně jako je tomu u předešlého zobrazení.



Obrázek 2.3: Molekula 1edw, zobrazena pomocí tyčinek.

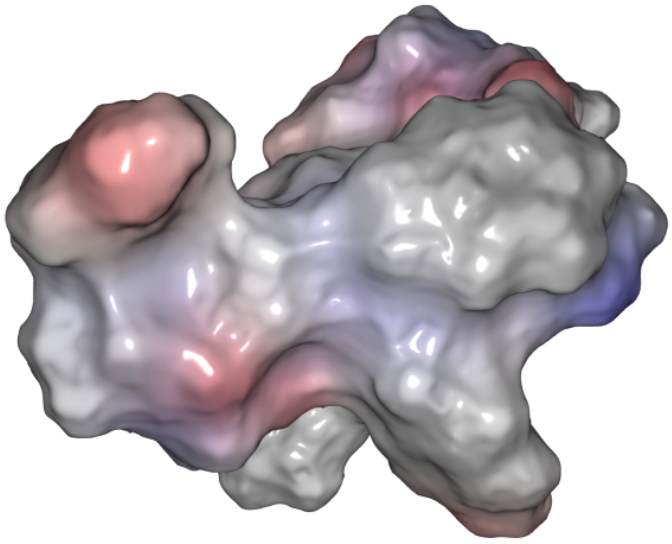
2.1.4 Povrch molekuly

Proteiny samy o sobě nemají fyzický povrch. Umělé vytvoření jejich povrchu usnadní vizuální představu vlastností proteinů. Jednou z důležitých vlastností je členitost povrchu, která může být známkou, do jaké míry je protein schopen reagovat se svým okolím. Více členitý povrch může obsahovat trhliny, tzv. *tunely*, vedoucí do nitra molekuly. Ukázka povrchu molekuly je zobrazena na obrázku 2.4.

2.2 Typy zobrazení tunelů

Tunel představuje cestu vedoucí z povrchu proteinu do jeho středu, tzv. *aktivního místa proteinu*. Cílem výpočtu tunelů je nalézt takové tunely, které by umožnily transport malých molekul (například molekul vody) do aktivního místa proteinu, a tím umožnili jejich vzájemnou reakci.

Detailní popis jednotlivých technik pro nalezení tunelů a výpočet dat pro jejich zobrazení může být nalezen v [6] [1].



Obrázek 2.4: Povrch molekuly 1edw

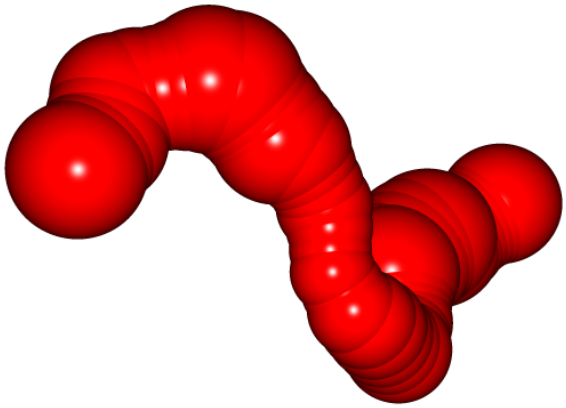
2.2.1 Středová osa (Centerline)

Středová osa spojuje body, které leží ve středech jednotlivých segmentů tunelu, obrázek 2.5. Může být zobrazena pomocí válců s malým průměrem, spojujících sousední body. Toto zobrazení je vhodné pro základní představu o tvaru a délce tunelu.

Obrázek 2.5: Tunel zobrazený pomocí středové linie.

2.2.2 Koule

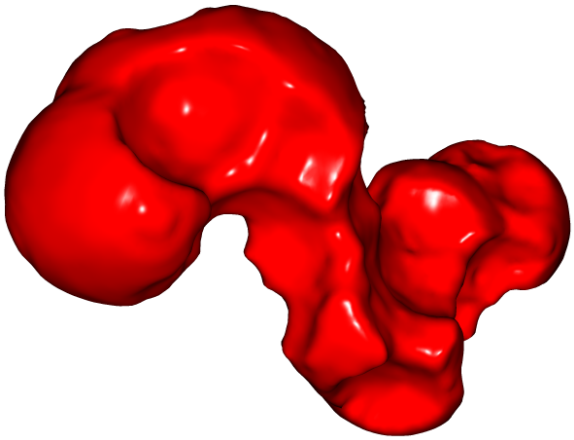
Zobrazení pomocí koulí použité na obrázku 2.6 vznikne sestrojením koulí v jednotlivých bodech středové linie. Poloměr koulí určuje maximální poloměr tunelu v daném místě proteinu. Dostáváme tak představu o tvaru a šířce tunelu.



Obrázek 2.6: Tunel zobrazený pomocí koulí, představující jeho šířku v daném bodě středové linie

2.2.3 Povrch

Na obrázku 2.7 je povrch tunelu vykreslen pomocí sítě trojúhelníků. Takové zobrazení umožňuje přesnou představu o tvaru tunelu.



Obrázek 2.7: Detailní povrch tunelu

3 Raytracing

V první části této kapitoly popíši princip algoritmu publikovaném v roce 1980 Turnerem Whittedem [10]. Druhá část kapitoly se bude zabývat popisem implementace raytraceru vzniklého na základě tohoto algoritmu.

3.1 Algoritmus

Algoritmus 3.1 popisuje základní kroky při výpočtu zobrazení scény pomocí raytracingu. Algoritmus můžeme rozdělit do několika částí.

Algoritmus 3.1 Raytracing

1. Sestroj paprsek skrz každý pixel obrazovky.
2. Nalezni nejbližší průsečík ve scéně. Pokud neexistuje, vrať barvu pozadí.
3. Sestroj stínové paprsky. Viditelné světelné zdroje zahrň do výpočtu osvětlení.
4. Vypočti osvětlení.
5. Sestroj odražený paprsek a lomený paprsek a rekurzivně opakuj kroky 2 až 5 dokud paprsek neopustí scénu nebo není dosaženo hloubky rekurze.
6. Do pixelu obrazovky ulož výslednou barvu.

První částí je *vrhání paprsků*, kdy z pozice pozorovatele vystřelíme do scény primární paprsky skrz všechny pixely obrazovky (krok 1).

V druhé části, *nalezení průsečíků*, zjišťujeme, zda ve scéně existuje objekt, který námi vržený paprsek protíná. Pokud ano, je vypočtena přesná poloha průsečíku s paprskem. V případě, že paprsek protíná více objektů, zajímá nás jen průsečík ležící nejblíže k rovině promítání. Pokud paprsek neprotne žádný objekt, výpočet skončí a do barvy paprsku je přiřazena barva pozadí (krok 2).

Poslední částí je *výpočet barvy* v bodě nalezeného průsečíku. Barva v daném bodě je dána součtem barevných příspěvků osvětlení a barevných příspěvků *sekundárních paprsků*. Na osvětlení se podílí pouze zdroje světla viditelné z daného bodu. Viditelnost světelných zdrojů zjistíme pomocí takzvaných *stínových paprsků* (krok 3). Pro vlastní výpočet osvětlení poté můžeme použít například Phongův osvětlovací model (krok 4).

Existují dva druhy *sekundárních paprsků*, odražené a lomené. *Odražený paprsek* slouží pro výpočet zrcadlových odrazů. Jeho počátek leží v bodě průniku a směr je dán odrazem přicházejícího paprsku od povrchu objektu. *Lomený paprsek* slouží pro výpočet lomů světla v poloprůhledných objektech. Počátek je opět v bodě průniku. Paprsek poté směřuje skrz objekt (krok 5).

Pro dosažení vícenásobných světelných odrazů a lomů opakujeme kroky 2 až 5.

V případě, že paprsek opustí scénu nebo je dosažena předem daná hloubka rekurze, vytváření nových paprsků se ukončí a do obrazové paměti je na pozici pixelu uložena barevná hodnota primárního paprsku (krok 6).

Na obrázku 3.1 jsou znázorněny paprsky vytvořené ve scéně pro výpočet barvy pixelu na pozici (x, y). Scéna obsahuje polopropustné zrcadlo Z, matnou kouli K a dva světelné zdroje S_1 a S_2.

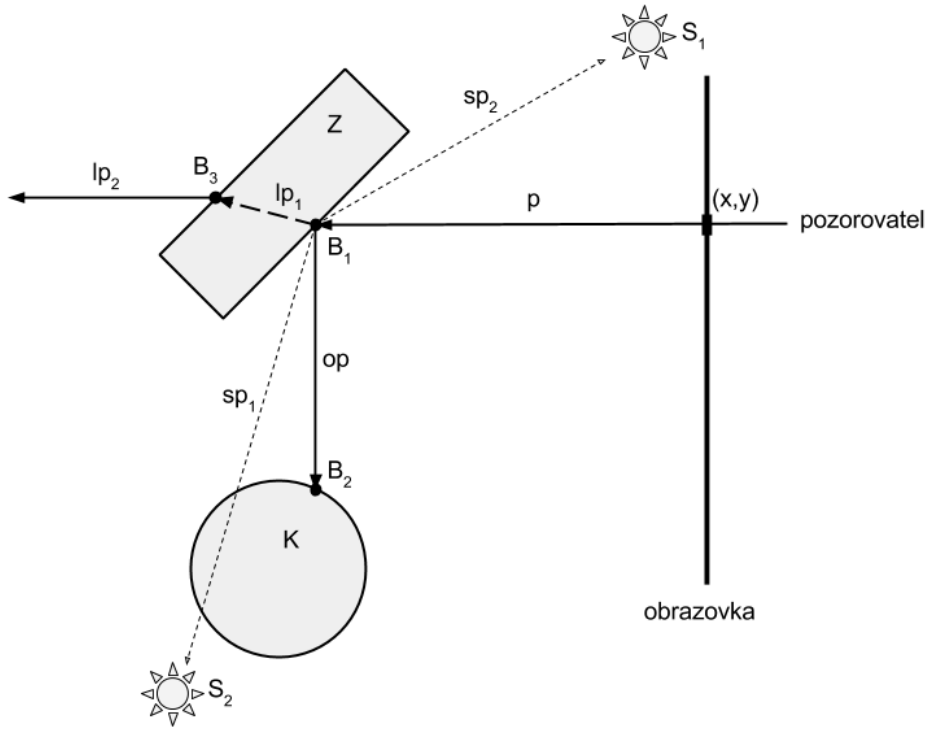
Paprsek p je primární paprsek vystřelený z pozice kamery směrem daným pozicí pixelu. Při průchodu scénou zasáhne zrcadlo Z v bodě B_1. Jedná se o jediný průsečík paprsku s objekty, proto je vypočítána jeho barva, která je později uložena do pixelu (x, y). Pro výpočet osvětlení bodu B_1 sestrojíme stínové paprsky sp_1 a sp_2 ke zdrojům světla S_1 a S_2. Paprsek sp_2 protíná kouli K, proto světelný zdroj S_2 nebude zahrnut do výpočtu osvětlení v bodě B_1.

Pro výpočet celkové barvy bodu B_1 musíme vytvořit odražený paprsek op a lomený paprsek lp_1. Paprsek op dále putuje scénou, kde zasáhne kouli K v bodě B_2. Pro barvu v tomto bodě stačí spočítat osvětlení, protože koule K je matná a nepropustná, takže sekundární paprsky nebudou vystřeleny.

Lomený paprsek $lp1$ projde tělesem a opustí jej v bodě B_3 jako paprsek lp_2. Tento paprsek nezasáhne žádné těleso ve scéně. Výpočet

je ukončen a barva paprsku odpovídá barvě pozadí.

Výsledná barva v bodě B_1 se tedy skládá z osvětlení pomocí světelného zdroje BS_1, barevného přírůstku odraženého paprsku op vypočteného v bodě B_2 a barevného přírůstku lomeného paprsku lp_3.



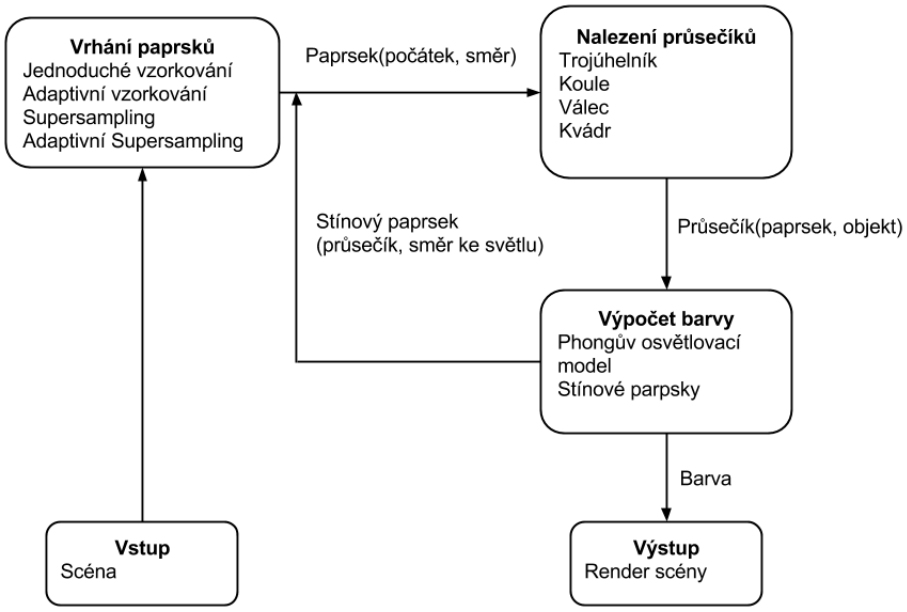
Obrázek 3.1: Výpočet barvy pixelu (x,y)

3.2 Implementace raytraceru

Tato kapitola se věnuje popisu implementace raytraceru, který byl vytvořen podle algoritmu představeném v předchozí kapitole. V následujícím textu se zaměřím na jednotlivé části algoritmu a jejich možná rozšíření.

3.2.1 Struktura raytraceru

Na obrázku 3.2 je znázorněna struktura raytraceru vytvořeného v rámci mé diplomové práce. Jednotlivé stavy odpovídají hlavním částem algoritmu 3.1 Přechody mezi stavy jsou nadepsány názvy objektů, které jsou vytvořeny v jednotlivých částech.



Obrázek 3.2: Obecná struktura raytraceru

Vstupem algoritmu je scéna, kterou si přejeme zobrazit. Scéna se skládá z těles, světel a kamery. Algoritmus umožňuje vytvářet různý počet primárních paprsků procházejících jedním pixelem. Od jednoduchého vzorkování, kdy jedním pixelem prochází jen jeden paprsek, po vzorkování s vyšší frekvencí (*tzv. supersampling*), kdy pixelem prochází paprsků několik. Počet primárních paprsků je možné redukovat pomocí *adaptivního vzorkování*, které mění hustotu primárních paprsků v závislosti na barevných změnách v obraze.

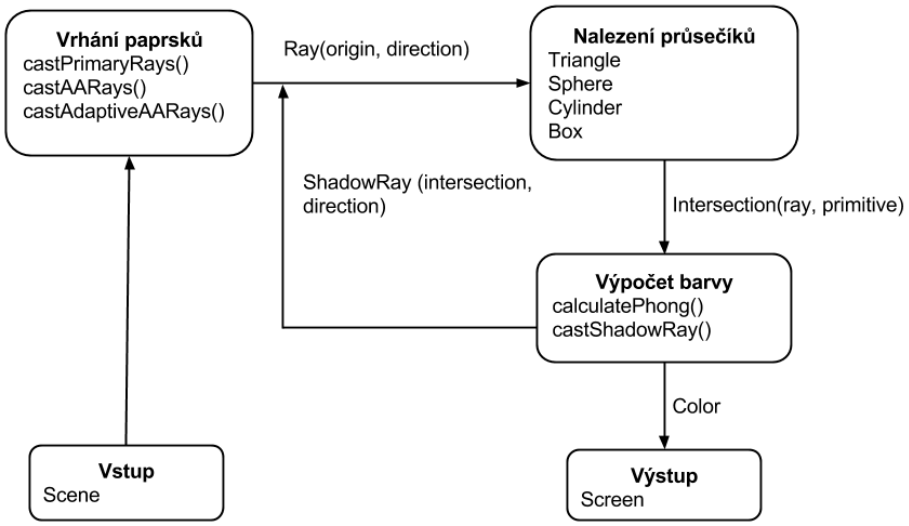
Raytracer umožňuje nalezení průsečíku paprsku se čtyřmi základními objekty Těmi jsou trojúhelníky, koule, válce a kvádry. V části nalezení průsečíku algoritmus projde všechny objekty ve scéně a vrátí průsečík s objektem ležícím nejblíže počátku paprsku.

Po nalezení průsečíku je potřeba vypočítat jeho barvu. Toho je

dosaženo pomocí stínových paprsků a Phongova osvětlovacího modelu. Vícenásobné světelné odrazy a lomy nejsou při zobrazování molekul potřeba a proto nebyl jejich výpočet implementován.

Po vypočtení barev všech pixelů je výsledný render uložen do souboru a zobrazen na obrazovce.

Na obrázku 3.3 je znázorněna struktura algoritmu obsahující názvy hlavních metod jednotlivých částí, popřípadě názvy tříd reprezentujících jednotlivá tělesa.



Obrázek 3.3: Struktura raytraceru - třídy a metody

4 Vrhání paprsků

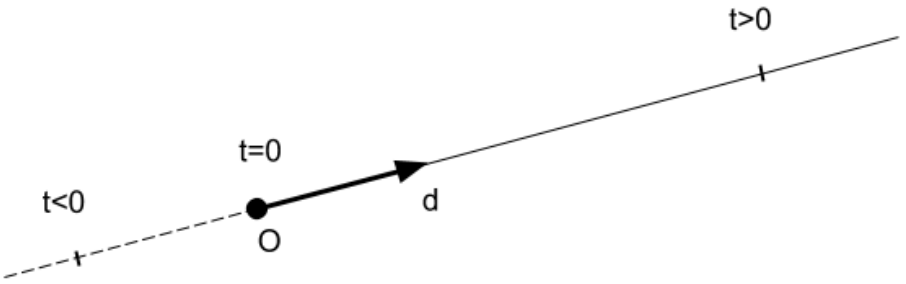
Následující kapitola se věnuje první části algoritmu 3.1. Nejdříve budou definovány pojmy paprsek a kamera, které později použiji při popisu jednotlivých metod vrhání primárních paprsků.

4.1 Paprsek

Paprsek je polopřímka v prostoru definovaná svým počátkem a směrem. Kterýkoliv bod *P* ležící na této přímce je popsán rovnicí 4.1.

$$P = O + t * d \tag{4.1}$$

Trojrozměrný vektor O představuje počátek paprsku, vektor d jeho směr. Pokud je směr paprsku normalizovaný, parametr *t* udává vzdálenost bodu *P* od počátku paprsku.



Obrázek 4.1: Poloha bodu na přímce v závislosti na parametru *t*

Jak lze vidět na obrázku 4.1, bod *P* leží na paprsku, pokud je hodnota parametru $t \geq 0$. Pro $t = 0$ platí, že bod leží přímo v počátku paprsku. Pokud je $t < 0$, bod leží za počátkem a neleží tedy na paprsku.

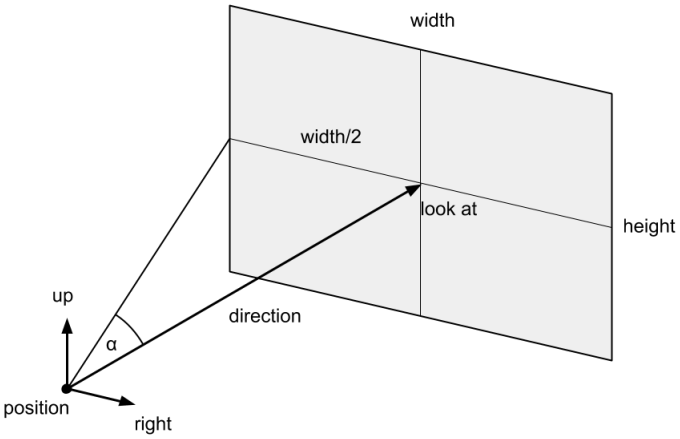
4.2 Kamera

Pro přesné zobrazení scény je potřeba definovat kameru, její pozici a bod, na který je kamera zaměřena. Na obrázku 4.2 vidíme nákres

takové kamery. První dva parametry určují již zmíněnou pozici kamery v prostoru, představovanou vektorem *position*, a souřadnice bodu *look at*, na který je kamera zaměřena. Tento bod je středem promítacího plátna o šířce *width* a výšce *height*.

Směr, kterým je kamera namířena, popisuje vektor *direction*. Tento vektor je definován jako rozdíl vektorů *position* a *look at*.

Vektor *direction* také představuje optickou osu kamery. Natočení kamery je určeno dvěma jednotkovými vektory *up* a *right*. Oba tyto vektory jsou kolmé na osu kamery. Vektor *up* určuje směr vzhůru a vektor *right* ukazuje vpravo od osy kamery.



Obrázek 4.2: Popis parametrů kamery.

Velikost promítacího plátna se vypočítá z úhlu záběru kamery (*field of view*) a vzdálenosti plátna od kamery [7]. Úhel α na obrázku představuje polovinu úhlu záběru kamery. Šířku promítacího plátna vypočteme pomocí rovnice 4.2. Parametr l představuje délku vektoru *direction*, tedy vzdálenost kamery od promítacího plátna.

$$width = 2 * l * \tan \alpha \tag{4.2}$$

Rovnice 4.3 slouží pro výpočet výšky promítacího plátna pomocí šířky plátna a parametru *aspect*, který udává poměr šířky a výšky

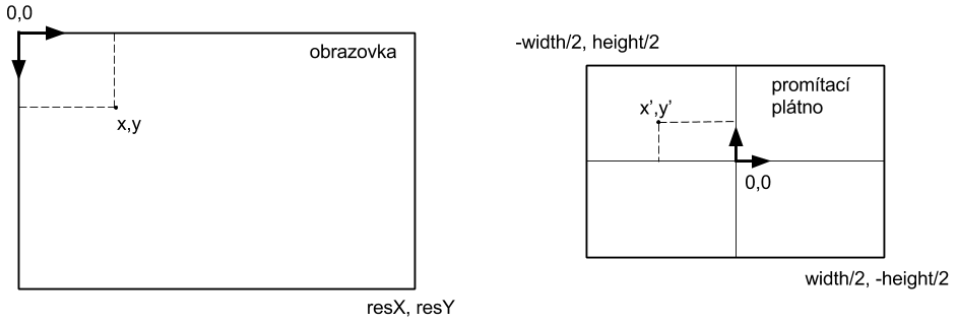
promítacího plátna. Tento parametr může být definován uživatelem nebo odvozen z rozlišení, ve kterém si přejeme scénu renderovat.

$$height = aspect/width \tag{4.3}$$

4.3 Primární paprsky

Při jednoduchém vzorkování prochází každým pixelem obrazovky právě jeden primární paprsek. Jeho počátek je shodný s pozicí kamery a směr je vypočten ze směru kamery a pozice pixelu na obrazovce, kterým daný paprsek prochází.

Obrazovka a promítací plátno nejsou totožné. Liší se velikostí a umístěním počátků svých lokálních souřadných systémů. Pro přesný výpočet směru je tedy potřeba převést souřadnice obrazovky (x,y) na souřadnice promítacího plátna (x',y'). Rozdíly mezi obrazovkou a promítacím plátnem je znázorněn na obrázku 4.3.



Obrázek 4.3: Vztah mezi obrazovkou a plátnem

Nejdříve převedeme velikost obrazovky na velikost plátna. Cílem je, aby plátno mělo stejný počet pixelů jako obrazovka a lišila se jejich velikost. Pixely na obrazovce jsou čtvercové. Pokud předpokládáme, že poměry stran obrazovky a plátna jsou stejné, budou čtvercové i pixely plátna. Pro výpočet velikosti pixelu plátna nám tedy stačí zjistit poměr šířky promítacího plátna a obrazovky podle rovnice 4.4.

$$velikostPixelu = width/resX \tag{4.4}$$

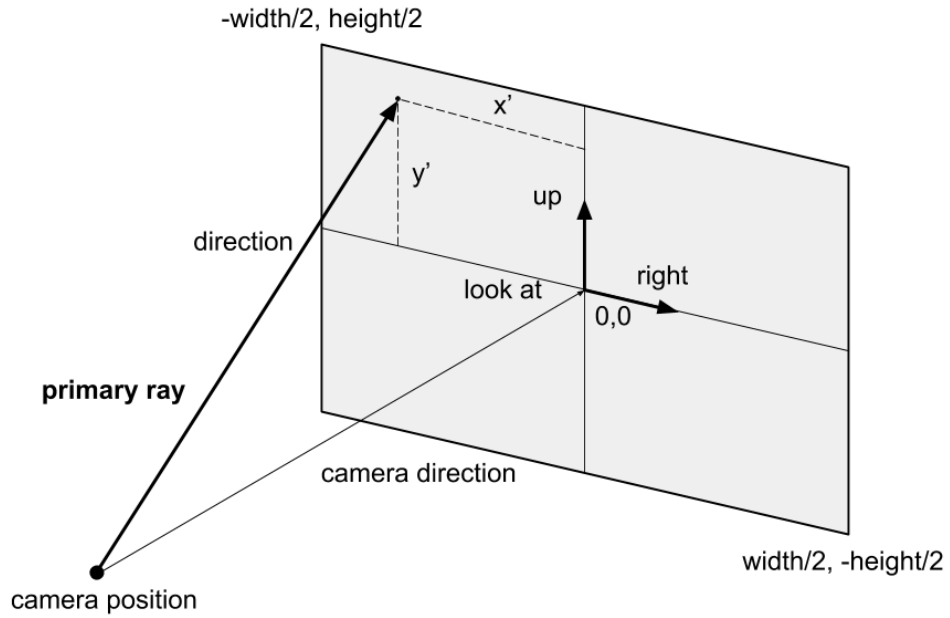
Dalším krokem při výpočtu souřadnice (x',y') je úprava souřadného systému. Zatímco souřadný systém obrazovky začíná v levém horním rohu a hodnoty rostou směrem k pravému dolnímu rohu, souřadný systém promítací roviny je umístěn do jejího středu s hodnotami rostoucími směrem k pravému hornímu rohu. Z toho důvodu při výpočtu x' od souřadnice obrazovky x odečteme hodnotu $width/2$. Protože souřadnice y' roste v opačném směru než y, při jejím výpočtu hodnotu y odečteme od $height/2$.

Převod souřadnice obrazovky (x,y) na souřadnici plátna (x',y') je uveden v rovnicích 4.5 a 4.6.

$$x' = -width/2 + x * velikostPixelu \tag{4.5}$$

$$y' = height/2 - y * velikostPixelu \tag{4.6}$$

Jakmile známe souřadnici (x',y'), zbývá vypočítat směr primárního paprsku. Sestrojení primárního paprsku s počátkem *camera position* a směrem *direction* je zobrazeno na obrázku 4.4.



Obrázek 4.4: Výpočet směru primárního paprsku

Směr paprsku se podle rovnice 4.7 vypočte přičtením vektorů *up* a *right* ke směru kamery *camera direction*. Oba tyto vektory jsou jednotkové. Vynásobením každé z jejich složek hodnotami x′ a y′ změníme jejich velikost, která poté odpovídá poloze pixelu v souřadném systému promítací roviny.

$$direction = cameraDirection + right * (x') + up * (y') \tag{4.7}$$

Všechny kroky potřebné pro vytvoření primárních paprsků jsou shrnuty v algoritmu 4.1.

Algoritmus 4.1 Sestroj primární paprsky

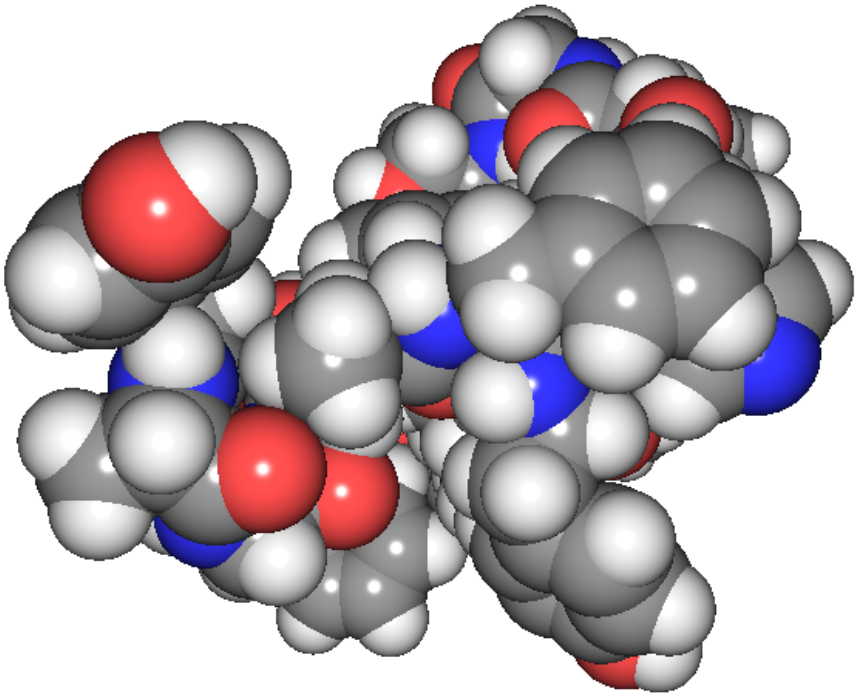
1. Velikost pixelu = šířka promítací plochy / šířka obrazovky
2. Pro každý pixel obrazovky (x,y)
3. x′ = - 1/2 šířky promítací plochy + x * velikost pixelu
4. y′ = 1/2 šířky promítací plochy - y * velikost pixelu
5. Směr = směr kamery + (right vektor * x′) + (up vektor * y′)
6. Sestroj paprsek(pozice kamery, směr)

V prvním kroku je vypočtena velikost pixelu plátna. Tato velikost je stejná pro všechny pixely a její výpočet je proto proveden mimo hlavní cyklus. V cyklu samotném projdeme všechny pixely obrazovky. V krocích 2 a 3 převedeme souřadnice pixelu plátna na souřadnice pixelu obrazovky. Jakmile známe tyto souřadnice, můžeme v kroku 5 vypočítat směr paprsku, který v posledním kroku algoritmu sestrojíme.

4.4 Supersampling

Na obrázku 4.5 je zobrazena molekula 1edw vyrenderována pomocí jednoduchého vzorkování popsaného v kapitole 4.3. Použité rozli-

šení obrazovky je 640 * 480 pixelů a jedním pixelem prochází právě jeden primární paprsek.



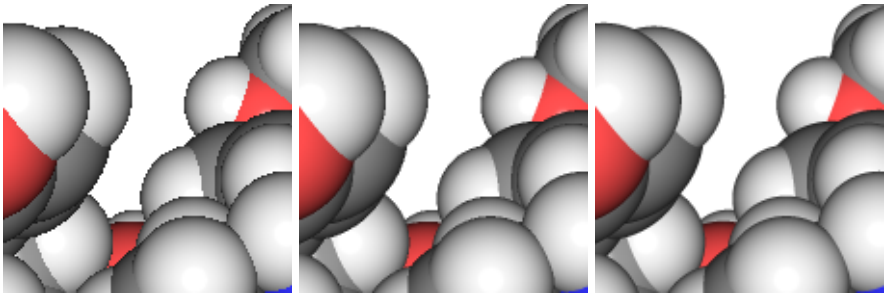
Obrázek 4.5: Molekula 1edw renderována jedním paprskem na pixel

Na okrajích koulí reprezentujících jednotlivé atomy molekuly lze vidět, že hrany v takto vytvořeném obraze jsou zubaté. Tento jev, tzv. *alias*, vznikl díky nízké hustotě primární paprsků, kterými byl obraz vytvořen. Alias může být potlačen pomocí vzorkování s vyšší frekvencí, tzv. *supersampling*, kdy je jedním pixelem sestrojeno větší množství primárních paprsků [12]. Výsledná barva pixelu se potom vypočte jako průměr barev získaných z jednotlivých primárních paprsků procházejících tímto pixelem.

Díky průměrování barevných hodnot v jednotlivých pixelů jsou hrany objektů rozmazané. Takové zobrazení je pro lidské oko přijatelnější a výsledný obraz tak působí lepším dojmem.

Na obrázku 4.6 je zvětšen detail molekuly 1edw. Vlevo je obrázek renderován jedním paprskem skrz pixel, s jasně patrnými zubatými hranami. Uprostřed byly sestrojeny čtyři paprsky skrz každý

pixel. Hrany se začínají jevit hladší, místy však lze stále vidět zuby. Poslední obrázek byl renderován šestnácti paprsky skrz pixel. Díky většímu množství paprsků se hrany jeví téměř hladké.



(a) 1 paprsek (b) 4 paprsky (c) 16 paprsků

Obrázek 4.6: Supersampling - ukázka

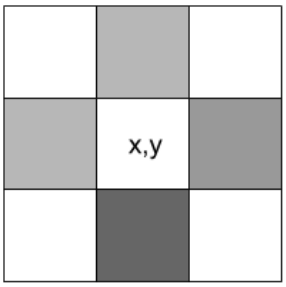
Je patrné, že s vyšším počtem paprsků roste i kvalita zobrazení. Úměrně tomu však roste i čas potřebný k vykreslení scény.

4.5 Adaptivní vzorkování

Jedním ze způsobů, jak snížit dobu potřebnou na vykreslení scény, je snížení počtu primárních paprsků. Důležité však je, aby byla zachována stejná kvalita zobrazení. Toho je možné dosáhnout pomocí *adaptivního vzorkování* [12].

Adaptivní vzorkování analyzuje obraz pomocí primárních paprsků vyslaných s nižší hustotou vzorkování. Na základě získaných vzorků je rozhodnuto, kde v obraze dochází k barevné změně a kde je tedy potřeba vyslat další primární paprsky pro určení přesné barevné hodnoty. Pokud však ve zpracovávané oblasti obrazu k barevné změně nedochází, barevná hodnota je odvozena z okolních vzorků.

Implementované adaptivní vzorkování se skládá ze dvou kroků. V prvním kroku jsou vrženy paprsky skrz každý druhý pixel (šedé pixely na obrázku 4.7). Tyto paprsky slouží pro získání vzorků, které jsou v druhém kroku analyzovány.



Obrázek 4.7: Vzorky

Ve druhém kroku jsou dopočítány hodnoty pixelů, kterými nebyly sestrojeny primární paprsky (např. pixel (x,y) na obrázku 4.7). Pro každý takový pixel se porovnají barevné hodnoty jeho sousedů. Pokud jsou všechny stejné, lze předpokládat, že v dané oblasti nedochází k barevné změně a barva v pixelu bude stejná jako u jeho sousedů. Pokud se však barvy okolních pixelů liší, je potřeba vypočítat přesnou barvu pixelu pomocí vržení primárního paprsku skrz daný pixel.

Zvýšením vzdálenosti mezi jednotlivými vzorky v prvním kroku, by mohlo být dosaženo většího snížení počtu primárních paprsků. S větší vzdáleností však roste i šance na ztrátu barevné informace. Implementované řešení z toho důvodu pracuje s roztečí paprsků jeden pixel. Minimální množství vystřelených paprsků je tedy poloviční oproti vrhání paprsků skrz každý pixel.

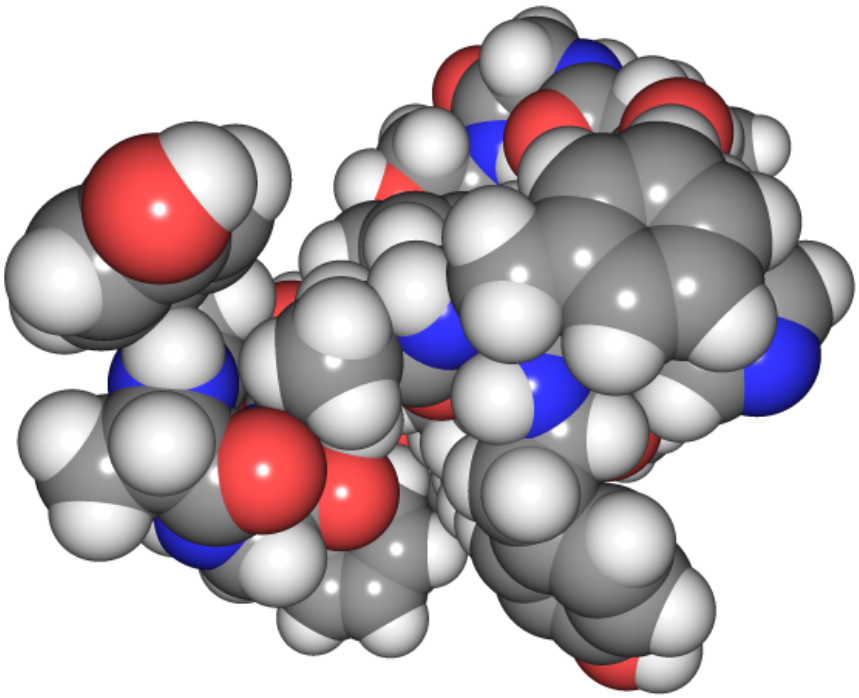
4.6 Adaptivní supersampling

Adaptivní vzorkování může být také kombinováno se vzorkováním s vyšší frekvencí. V principu obraz vytváříme jednoduchým vzorkováním a v případě, že v obraze dojde k barevné změně, použijeme supersampling pro detailnější zobrazení.

Implementace má opět dva kroky. V prvním kroku vytvoříme obraz pomocí jednoduchého nebo adaptivního vzorkování.

V druhém kroku pro každý pixel porovnáme jeho barevnou hodnotu s barvou jeho sousedů. Pokud je hodnota různá, dochází v okolí daného pixelu k barevné změně. Pixelem proto sestrojíme další paprsky a pomocí jejich barevného příspěvku upravíme barevnou hodnotu pixelu.

Na obrázku 4.8 je zobrazena stejná molekula, jako v kapitole 4.4. Obrázek byl vyrenderován ve stejném rozlišení, v tomto případě byl však pro jeho vykreslení použit adaptivní supersampling se 16 paprsky skrz pixel.



Obrázek 4.8: Molekula 1edw renderována 16 paprsky na pixel

5 Nalezení průsečíku

Srdcem každého raytraceru je část zabývající se nalezením průsečíků mezi paprsky a tělesy ve scéně. Jedná se také o výpočetně nejnáročnější krok algoritmu a je tedy potřeba nalézt co nejoptimálnější metody pro výpočet těchto průsečíků.

5.1 Scéna

Všechna tělesa, která je raytracer schopen zobrazit, jsou organizována ve scéně. Kromě nich scéna obsahuje i kolekci světel a definici kamery.

Každé z těles ve scéně musí umožnit nalezení průsečíku s paprskem a pokud se jedná o zobrazitelné těleso, tak i výpočet normály povrchu.

Ve vytvořeném raytraceru existují tři základní druhy zobrazitelných objektů. A to trojúhelníky, koule a válce. Čtvrtým typem objektu je, kvádr, sloužící pro tvorbu obálek urychlujících datových struktur.

V následujících podkapitolách se budu věnovat nalezení průsečíku s každým z těchto těles. Pokud se bude jednat o zobrazitelné těleso, popíši i výpočet normály povrchu.

5.2 Trojúhelník

V počítačové grafice je trojúhelník jedním z nejpoužívanějších objektů. S jeho pomocí lze definovat povrchy složitějších těles, často se skládajících z několik set tisíc trojúhelníků. Z toho důvodu je potřeba, aby algoritmus pro nalezení průsečíku mezi paprskem a trojúhelníkem byl co nejméně časově náročný.

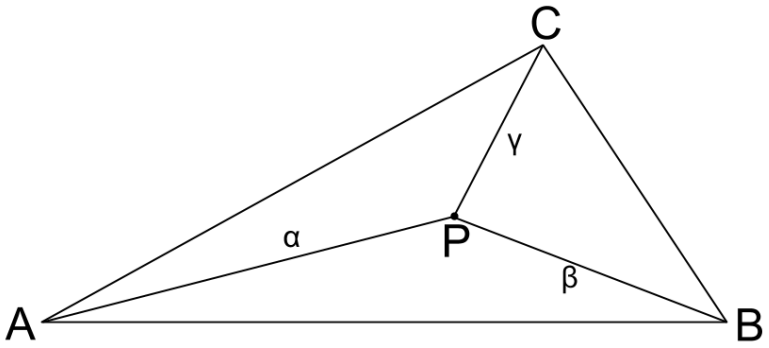
5.2.1 Definice trojúhelníku

Trojúhelník je definován třemi body, svými vrcholy (A, B, C). Podle konvence jsou vrcholy uloženy v pořadí proti směru hodinových ručiček. Pokud body neleží na přímce, definují rovinu. Pozici bodu na

této rovině můžeme vyjádřit pomocí *barycentrických souřadnic*.

5.2.2 Barycentrické souřadnice

Barycentrické souřadnice (α, β, γ), zobrazené na obrázku 5.1, určují vzdálenost bodu v rovině od vrcholů trojúhelníka (A, B, C). Pozice bodu P je popsána pomocí barycentrických souřadnic v rovnici 5.1.



Obrázek 5.1: Barycentrické souřadnice trojúhelníku ABC

$$P(\alpha, \beta, \gamma) = \alpha A + \beta B + \gamma C \tag{5.1}$$

Souřadnice (α, β, γ) musí splňovat podmínku 5.2.

$$\alpha + \beta + \gamma = 1 \tag{5.2}$$

5.2.3 Nalezení průsečíku

Následující postup pro nalezení průsečíku paprsku s trojúhelníkem je převzat z [8].

Rovnice 5.1 platí pro všechny body v rovině, definované body (A, B, C). Naším cílem je však zjistit, zda daný bod leží uvnitř trojúhelníku. Bod P leží uvnitř trojúhelníku, pokud jeho barycentrické souřadnice splňují podmínky 5.3.

$$\begin{aligned} 0 < \alpha < 1 \\ 0 < \beta < 1 \\ 0 < \gamma < 1 \end{aligned} \tag{5.3}$$

Ve skutečnosti k určení polohy bodu nepotřebujeme všechny tři barycentrické souřadnice, stačí nám jen dvě z nich. Vyjádřením α z podmínky 5.2 dostaneme $\alpha = 1 - \beta - \gamma$. Poté substitucí za α do rovnice 5.1 získáme novou rovnici pro výpočet polohy bodu p.

$$P(\beta, \gamma) = A + \beta(B - A) + \gamma(C - A) \tag{5.4}$$

Nesmíme zapomenout upravit i podmínky pro zjištění, zda je bod uvnitř trojúhelníku. Nové podmínky pro β a γ jsou uvedeny v rovnici 5.5.

$$\begin{gathered} 0 < \beta < 1 \\ 0 < \gamma < 1 \\ 0 < \beta + \gamma < 1 \end{gathered} \tag{5.5}$$

Pro nalezení průsečíku trojúhelníku s paprskem dosadíme rovnici paprsku $P = O + td$ do rovnice 5.4.

$$O + td = A + \beta(B - A) + \gamma(C - A) \tag{5.6}$$

Dostáváme lineární rovnici o třech neznámých β, γ a t. Převedením všech neznámých na levou stranu dostáváme rovnici 5.7.

$$\beta(A - B) + \gamma(A - C) + td = A - O \tag{5.7}$$

Všechny známé proměnné A, B, C, d a O jsou vektory. Rovnici 5.7 proto můžeme rozepsat do soustavy tří samostatných rovnic o třech neznámých. Každá z rovnic platí pro jednu z vektorových složek (x, y, z).

$$\begin{gathered} \beta(A_x - B_x) + \gamma(A_x - C_x) + td_x = A_x - O_x \\ \beta(A_y - B_y) + \gamma(A_y - C_y) + td_y = A_y - O_y \\ \beta(A_z - B_z) + \gamma(A_z - C_z) + td_z = A_z - O_z \end{gathered} \tag{5.8}$$

Přepsáním soustavy rovnic 5.8 do maticové formy dostaneme:

$$\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} j \\ k \\ l \end{bmatrix} \tag{5.9}$$

Pro větší přehlednost jsou jednotlivé koeficienty nahrazeny písmeny a - l. Jejich význam je následující:

$$\begin{array}{llll} a = (A_x - B_x), & d = (A_x - C_x), & g = td_x, & j = (A_x - O_x), \\ b = (A_y - B_y), & e = (A_y - C_y), & h = td_y, & k = (A_y - O_y), \\ c = (A_z - B_z), & f = (A_z - C_z), & i = td_x, & l = (A_z - O_z) \end{array} \tag{5.10}$$

Pro vyřešení soustavy lineárních rovnic využijeme Cramerovo pravidlo uvedené v rovnici 5.11.

$$x_i = detM_i/detM \tag{5.11}$$

Podle Cramerova pravidla pro naši soustavu lineárních rovnic 5.9, platí

$$\begin{array}{c} \beta = detM_\beta/M \\ \gamma = detM_\gamma/M \\ t = detM_t/M \end{array} \tag{5.12}$$

Pro výpočet jednotlivých neznámých je nejprve potřeba určit jednotlivé matice. Matice M je matice z levé strany rovnice 5.9.

$$M = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \tag{5.13}$$

Podle Cramerova pravidla je matice M_i sestrojena z matice M s i-tým sloupcem nahrazeným vektorem odpovídajícím pravé straně soustavy lineárních rovnic. V našem případě je to vektor (j, k, l) a matice M_α, M_β a M_t jsou:

$$M_\alpha = \begin{bmatrix} j & d & g \\ k & e & h \\ l & f & i \end{bmatrix} M_\beta = \begin{bmatrix} a & j & g \\ b & k & h \\ c & l & i \end{bmatrix} M_t = \begin{bmatrix} a & d & j \\ b & e & k \\ c & f & l \end{bmatrix} \tag{5.14}$$

Dolní index u matice, značí pro kterou z neznámých je matice vytvořena.

Jakmile máme vyjádřeny všechny matice, je potřeba vypočítat jejich determinanty. Jejich výpočet je popsán v rovnicích 5.15.

$$\begin{array}{rcl} detM & = & a(ei - hf) + b(gf - di) + c(dh - eg) \\ detM_\beta & = & j(ei - hf) + k(gf - di) + l(dh - eg) \\ detM_\gamma & = & i(ak - jb) + h(jc - al) + g(bl - kc) \end{array} \tag{5.15}$$

Když známe všechny determinanty, můžeme podle rovnice 5.12 vypočítat hodnoty pro β a γ. Pokud hodnoty splňují podmínky 5.5, průsečík paprsku a trojúhelníku existuje a vzdálenost průsečíku od počátku paprsku t může být spočítána podle rovnice 5.12.

5.2.4 Implementace

Algoritmus 5.1 Nalezení průsečíku - trojúhelník

1. *při inicializaci nového trojúhelníku*
2. předpočítej proměnné a až g z matice M
3. *v metodě nalezení průsečíku*
4. vypočti zbylé proměnné matice M
5. vypočti pomocné proměnné pro výpočet determinantů
6. spočti determinant matice M
7. spočti $beta$ a ukonči výpočet, pokud nesplní podmínku
8. spočti $gama$ a ukonči výpočet, pokud nesplní podmínku
9. Pokud je $beta + gama < 1$
10. spočti vzdálenost t

Koeficienty a, b, c, d, e, f, g jsou závislé jen na vrcholech trojúhelníku. Můžeme je tedy předpočítat při vytváření trojúhelníků a snížit tak výpočetní náročnost samotného nalezení průsečíku. Zbylé koeficienty se dopočítají v těle metody pro nalezení průsečíku podle rovnice 5.9.

Při pohledu na rovnici 5.15 si všimneme, že rovnice pro výpočet $detM$ a $detM_\beta$ jsou si podobné. Stejně tak si jsou podobné i rovnice M_γ a $detM_t$. Vytvořením pomocných proměnných

$$\begin{array}{ll} beta1 = (ei - hf), & gama1 = (ak - jb), \\ beta2 = (gf - di), & gama2 = (jc - al), \\ beta3 = (dh - eg), & gama3 = (bl - kc), \end{array} \tag{5.16}$$

opět snížíme počet potřebných výpočtů.

Dosazením pomocných proměnných získáme vzorce 5.17 pro výpočet souřadnic $beta$ a $gama$.

$$\begin{aligned} detM &= a * beta1 + b * beta2 + c * beta3 \\ beta &= \frac{j*beta1+k*beta2+l*beta3}{detA} \\ gama &= \frac{i*gama1+h*gama2+g*gama3}{detA} \end{aligned} \tag{5.17}$$

Po spočtení první ze souřadnic, $beta$, testujeme podmínku 5.5. Pokud není splněna můžeme předčasně ukončit výpočet, protože průsečík paprsku a trojúhelníku neexistuje. Pokud splněna je, vypočítáme souřadnici $gama$ a opět testujeme podmínku 5.5.

Pokud jsou všechny podmínky splněny, průsečík P leží v trojúhelníku a zbývá nám jen vypočítat jeho vzdálenost od počátku paprsku podle rovnice 5.18.

$$t = \frac{f * gama1 + e * gama2 + d * gama3}{detM} \tag{5.18}$$

5.2.5 Výpočet normály

Normála je důležitá pro výpočet barvy při použití Phongova osvětlujícího modelu, stejně jako při tvorbě odražených paprsků. Normála trojúhelníku s vrcholy (A, B, C) se vypočítá v době jeho vytváření, jako vektorový součin dvou vektorů představujících jeho strany. Díky uspořádání vrcholů trojúhelníku můžeme tedy normálu vypočítat podle vztahu 5.19.

$$n = (B - A) * (C - A) \tag{5.19}$$

5.3 Koule

Molekula zobrazená v kapitole 4.4 se skládá z 399 atomů. Při použitém VdW zobrazení jsou tyto atomy vykresleny jako koule. K jejich vytvoření bychom mohli použít síť sestavenou z trojúhelníků popsaných v předchozí kapitole. Tato síť by však jen aproximovala tvar koule a na její vytvoření by bylo potřeba mnoho trojúhelníků.

Lepším řešením je implementovat kouli jako jedno se základních těles, s vlastními metodami pro nalezení průsečíku a výpočet normály. Tím snížíme časovou i paměťovou náročnost algoritmu a zvýšíme kvalitu zobrazení.

5.3.1 Definice koule

Koule je jedno ze základních matematických těles. Je definovaná jako množina bodů s konstantní vzdáleností od jednoho bodu, středu. Rovnice 5.20 popisuje matematické vyjádření koule s poloměrem r.

$$x^2 + y^2 + z^2 = r^2 \tag{5.20}$$

Prvním řešením pro nalezení průsečíku koule s paprskem, co nás většinou napadne, je tzv. *algebraické řešení*.

Je založeno na dosazení rovnice paprsku $p = o + td$ do rovnice koule 5.20. Tím získáme kvadratickou rovnici, jenž má dvě řešení v případě, že paprsek kouli protíná, jedno, pokud je paprsek tečnou koule a žádné, pokud paprsek kouli míjí. Více o tomto řešení může být nalezeno v [8].

Nevýhodou algebraického řešení je náročnost výpočtu kvadratické rovnice. Existuje však tzv. *geometrické řešení*, které je o poznání rychlejší [5].

5.3.2 Geometrické řešení

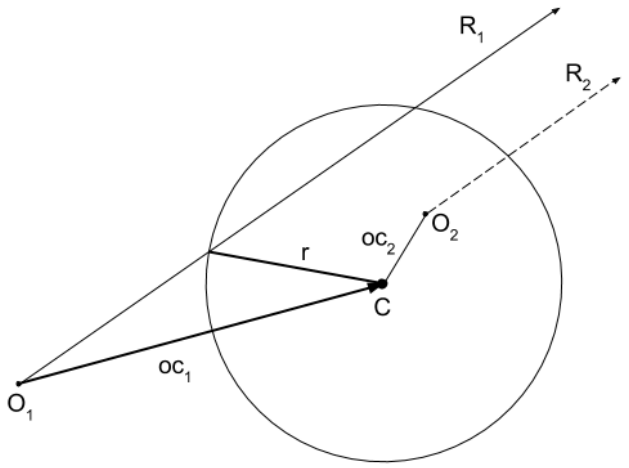
Zatímco algebraické řešení určuje polohu průsečíku numerickým výpočtem rovnice 5.20, geometrické řešení využívá vyjádření paprsku, polohy koule a jejího poloměru pomocí vektorů v prostoru. Porovnáním jejích směrů, velikostí a poloh je možné zjistit, zda paprsek kouli mine nebo ne a určit polohu průsečíku v případě, že takový průsečík existuje.

Vlastní výpočet se skládá ze tří podmínek a, pokud jsou všechny splněny, výpočtu polohy průsečíku. V každé z podmínek testujeme, zda má paprsek šanci trefit kouli. Pokud nemá, průsečík neexistuje a výpočet může být ukončen bez dalších nadbytečných výpočtů. Pokud testovaná podmínka splněna je, hodnoty získané při jejím výpočtu se použijí jako hodnoty pro výpočet podmínky další. Díky tomu nejsou žádné z výpočtů prováděny zbytečně a je tím ušetřen čas potřebný pro nalezení průsečíku.

První podmínka

$$oc^2 \geq r^2 \tag{5.21}$$

Cílem první podmínky je zjistit, zda paprsek začíná uvnitř koule nebo mimo ni. Pokud začíná uvnitř koule, vše, co by bylo zobrazeno, by byla tato koule. Tomu se chceme vyhnout a proto tyto koule nezobrazujeme.



Obrázek 5.2: První podmínka. Výpočet vektoru *oc*.

Na obrázku 5.2 je zobrazena koule a dva paprsky R_1 a R_2, s počátky v bodech O_1 a O_2. Porovnáním velikostí vektorů oc_1 a oc_2 (*origin to center*) s poloměrem koule *r* zjistíme, který z paprsků začíná vně koule (R_1), a který uvnitř (R_2).

Vektor *oc* sestrojíme podle rovnice 5.22. Druhou mocninu jeho velikosti oc^2, vypočteme pomocí skalárního součinu v rovnici 5.23.

$$oc = C - O \tag{5.22}$$

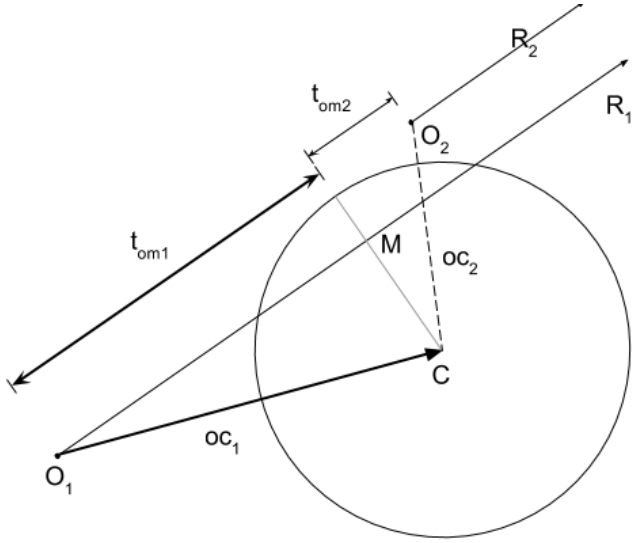
$$oc^2 = oc.oc \tag{5.23}$$

Podmínka 5.21 říká, že pokud je druhá mocnina velikosti vektoru *oc* větší, než druhá mocnina poloměru koule, paprsek začíná mimo kouli a výpočet může pokračovat duhou podmínkou.

Druhá podmínka

$$t_{om} \geq 0 \tag{5.24}$$

Jakmile víme, že paprsek neleží uvnitř koule, je třeba zjistit, zda směřuje směrem ke kouli, nebo od ní. Pokud směřuje od ní, je jasné, že ji nikdy netrefí a výpočet průsečíku se tedy může ukončit.



Obrázek 5.3: Druhá podmínka. Výpočet t_{om}.

To, zda paprsek směřuje ke kouli nebo ne, zjistíme pomocí vzdálenosti počátku paprsku od bodu *M*, ve kterém je paprsek nejblíže středu koule. Výpočet této vzdálenosti (t_{om}) je ukázán na obrázku 5.3.

Vzdálenost (t_{om}) vypočítáme podle vztahu 5.25, jako skalární součin vektoru *oc*, vypočteném v rovnici 5.22, a směru paprsku *d*.

$$t_{om} = oc.d \tag{5.25}$$

Na obrázku 5.3 je opět znázorněna koule a dva paprsky. Vzdálenost t_{om1} je kladná a paprsek R_1 tedy směřuje ke kouli, kterou má šanci protnout. Oproti tomu paprsek R_2 směřuje od koule a vzdálenost t_{om2} vyjde záporná a my tedy můžeme ukončit nalezení průsečíku s tímto paprskem, protože ten kouli jistě neprotne.

Třetí podmínka

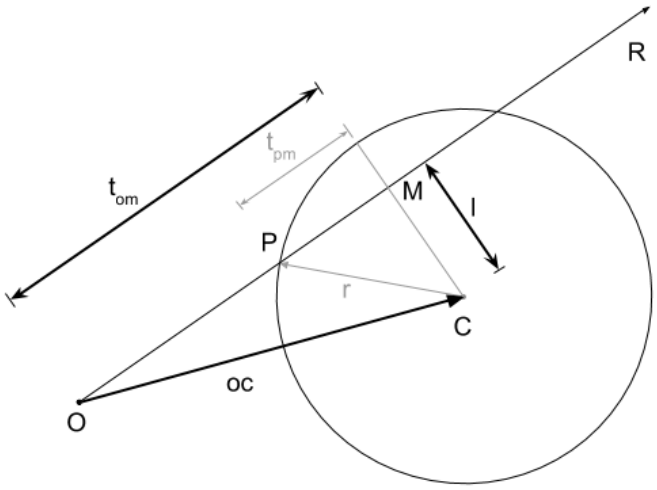
$$t_{pm}^2 \geq 0 \tag{5.26}$$

Doposud víme, že počátek paprsku neleží v kouli, a že paprsek směřuje ke kouli. Poslední co zbývá zjistit je, zda paprsek kouli trefí nebo ne.

To je dosaženo spočtením hodnoty t_{pm}^2, která udává vzdálenost průsečíku P od bodu M. Pokud je $t_{pm}^2 < 0$, průsečík paprsku a koule neexistuje.

Výpočet t_{pm}^2 má dva kroky. V prvním, zobrazeném na obrázku 5.4, vypočteme vzdálenost bodu *M* od středu koule (*l*). Hodnota *l* je vypočítána pomocí Pythagorovy věty, podle vztahu 5.27.

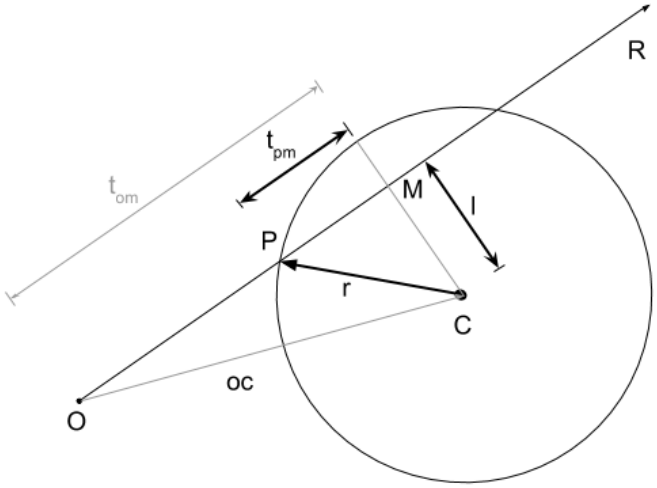
$$l^2 = oc^2 - t_{om}^2 \tag{5.27}$$



Obrázek 5.4: Třetí podmínka. Výpočet d^2.

Ve druhém kroku se vypočítá velikost t_{pm}^2. Výpočet, opět pomocí Pythagorovy věty, je znázorněn na obrázku 5.5.

$$t_{pm}^2 = r^2 - l^2 \tag{5.28}$$



Obrázek 5.5: Třetí podmínka. Výpočet t_{pm}^2.

Pokud je splněna podmínka 5.26, paprsek kouli zasáhne. Poté zbývá už jen vypočítat hodnotu t, udávající vzdálenost nalezeného průsečíku od počátku paprsku. To se provede odečtením hodnoty t_{pm} od t_{om}, podle rovnice 5.29.

$$t = t_{om} - \sqrt{t_{pm}^2} \tag{5.29}$$

5.3.3 Výpočet normály

Normála povrchu koule *n* v bodě *P* odpovídá vektoru sestrojenému ze středu koule *C*, skrz bod *P*. Správnou velikost vektoru zajistíme jeho normalizováním pomocí poloměru koule *r*.

$$n = \frac{P - C}{r} \tag{5.30}$$

5.3.4 Implementace

Implementace nalezení průsečíku s koulí je vcelku přímočará. Postupně kontrolujeme všechny podmínky. Pokud některá z nich není

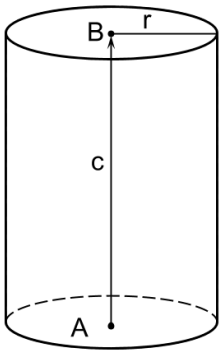
splněna, přerušíme výpočet a oznámíme, že průsečík nebyl nalezen.

Protože hodnoty oc a oc^2, použité při výpočtech ve všech podmínkách, nejsou závislé na parametrech paprsku, můžeme je předpočítat při inicializaci koulí. A dosáhnout tak snížení výpočetní náročnosti.

5.4 Válec

Válec lze definovat dvěma body, A a B, ležícími ve středech jeho podstav. Výška válce je dána vzdáleností těchto bodů. Šířka je dána poloměrem jeho podstav r. Osu válce představuje vektor c. Jeho směr je z bodu A do bodu B.

$$c = B - A \tag{5.31}$$



Obrázek 5.6: Válec definovaný body A a B

5.4.1 Nalezení průsečíku

Pro jakýkoliv bod P, který leží na povrchu válce, platí rovnice 5.32.

$$(v - w)^2 - r^2 = 0 \tag{5.32}$$

Vektor v, zobrazený na obrázku 5.7, směřuje od bodu A do průsečíku P a vypočte se jako:

$$v = P - A \tag{5.33}$$

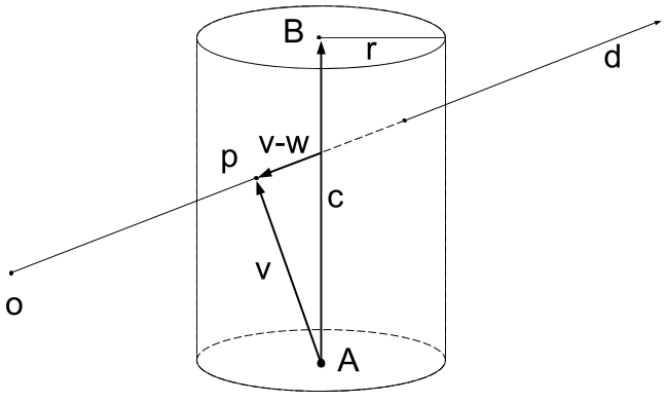
Vektor w odpovídá komponentě vektoru v, rovnoběžné s osou válce c. Vektor w se vypočítá podle vztahu uvedeného v rovnici 5.34.

$$w = \frac{v.c}{c.c}c \tag{5.34}$$

Operátor . představuje skalární součin vektoru. Výsledkem podílu $v.c/c.c$ je tedy skalár, kterým poté násobíme každou ze složek vektoru c.

Rozdíl vektorů v a w znázorněný na obrázku 5.7 je kolmý na osu c a prochází bodem P. Velikost tohoto vektoru představuje vzdálenost bodu P od osy válce. Všechny body ležící na povrchu válce musí mít tuto vzdálenost rovnu r.

Protože skalární součin vektoru se sebou samým se rovná druhé mocnině velikosti tohoto vektoru, v rovnici použijeme r^2 a díky tomu nebudeme muset počítat s odmocninami. Tím dosáhneme snížení náročnosti výpočtu.



Obrázek 5.7: Nalezení průsečíku s válcem.

Pro nalezení průsečíku dosadíme do rovnice 5.32 za bod P rovnici paprsku $P = O + td$. Nyní $v = O + td - A$. Substitucí $m = O - A$ dostáváme nový vztah pro v.

$$v = m + td \tag{5.35}$$

Posloupností matematických úprav rovnice 5.32 dostáváme novou rovnici 5.36 s neznámou t.

$$(d.d - \frac{(d.c)^2}{c.c})t^2 + 2(m.d - \frac{(d.c)(m.c)}{c.c})t + m.m - \frac{(m.c)^2}{c.c} - r^2 = 0 \tag{5.36}$$

Abychom nemuseli počítat s dělením, vynásobíme celou rovnici 5.32 členem $c.c$. Dostaneme upravenou rovnici

$$\begin{gathered}((c.c)(d.d) - (d.c)^2)t^2 + \\ 2((c.c)(m.d) - (d.c)(m.c))t + \\ (c.c)((m.m) - r^2) - (m.c)^2 = 0\end{gathered} \tag{5.37}$$

Získáme tak kvadratickou rovnici pro neznámou t, ve tvaru $at^2 + 2bt + c = 0$, kde jednotlivé členy jsou,

$$\begin{aligned} a &= (c.c)(d.d) - (d.c)^2 \\ b &= (c.c)(m.d) - (d.c)(m.c) \\ c &= (c.c)((m.m) - r^2) - (m.c)^2 \end{aligned} \tag{5.38}$$

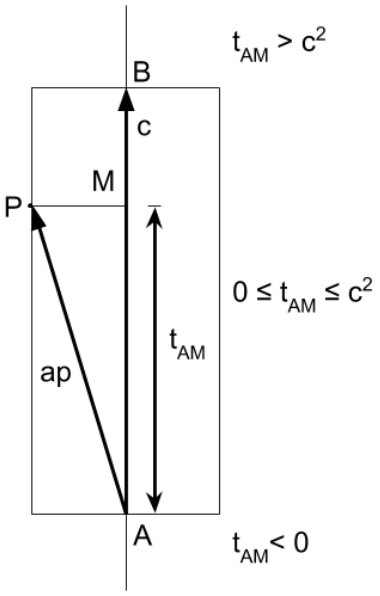
Vzdálenost bližšího z průsečíků t se vypočítá podle vztahu 5.39.

$$t = \frac{-2b - \sqrt{(2b)^2 - 4ac}}{2a} \tag{5.39}$$

Diskriminant rovnice 5.39 ($D = (2b)^2 - 4ac$) určí kolik existuje průsečíků paprsku a válce. Pokud je $D < 0$, paprsek válec míjí. Pro $D = 0$ existuje jeden průsečík, paprsek je tečnou válce. A nakonec $D > 0$ dává dva průsečíky. Menší z hodnot t udává vzdálenost k bližšímu z těchto průsečíků.

Výše uvedené výpočty vedly k nalezení průsečíku s nekonečným válcem. My však chceme, aby byl válec ohraničen rovinami procházejícími body A a B.

Na obrázku 5.8 je zobrazen řez válcem ohraničeným body A a B. Paprsek protíná válec v bodě P. Pro zjištění, zda bod P leží v mezích válce, využijeme stejného postupu jako v druhé podmínce, při hledání průsečíku paprsku s koulí, pomocí geometrického řešení, v kapitole 5.3.2.



Obrázek 5.8: Výpočet vzdálenosti t_{AM}

Tentokrát vypočítáme proměnnou t_{AM}, která udává druhou mocninu vzdálenosti bodu M, ležícího na ose válce c nejblíže k průsečíku P, od dolní hranice válce A. Hodnota t_{AM} se vypočítá jako skalární součin vektorů ap, spojujícího spodní mez válce A s průsečíkem P, a osou válce c.

$$t_{AM}^2 = ap.c \tag{5.40}$$

Pokud je $t_{AM}^2 < 0$, paprsek prochází pod dolní hranicí válce A a válec tedy míjí. Pokud je $t_{AM}^2 > c^2$, paprsek prochází nad horní mezí válce B a válec opět míjí.

5.4.2 Výpočet normály

Normála povrchu válce se sestrojí jako vektor z bodu M, skrz průsečík P. Polohu bodu M vypočítáme pomocí vztahu 5.41.

$$M = A + t_{AM}c \tag{5.41}$$

5.5 Osově orientovaný kvádr

V implementovaném raytraceru neslouží osově orientovaný kvádr jako zobrazitelný objekt. Není totiž vhodný pro vykreslení žádného typu zobrazení molekul. Najde však využití jako obalové těleso, při tvorbě urychlující datové struktury, popsané v kapitole 8.

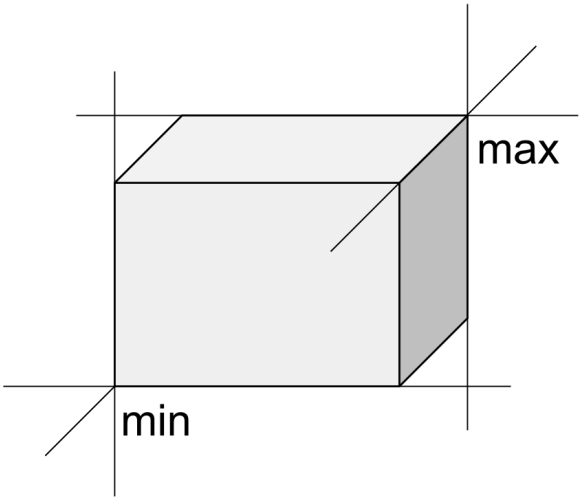
Osově orientovaný kvádr zobrazený na obrázku 5.9 lze definovat dvěma body. Jeho minimálním (min) a maximálním (max) rohem.

5.5.1 Nalezení průsečíku

Metoda nalezení průsečíku paprsku s kvádrem popsaná v [11] rozkládá problém nalezení průsečíku s kvádrem na dvoudimenzionální problém nalezení průsečíků s obdélníky. Tyto obdélníky představují průměty kvádru do rovin rovnoběžných s osami souřadného systému.

Body min a max definují množinu rovin rovnoběžných s osami souřadného systému. Na obrázku 5.9 lze vidět, že v každé z těchto rovin leží průměty jednotlivých stran kvádru. Průměty jsou ohraničené přímkami procházejícími body min a max. Nalezením průsečíků s těmito přímkami v daných rovinách budeme schopni určit polohu průsečíku paprsku s kvádrem v prostoru.

Obecná přímka je definována vztahem 5.42, kde parametr m představuje sklon přímky a B bod, ve kterém přímka protíná osu y.



Obrázek 5.9: Kvádr definovaný rohy $min\ max$

$$y = mx + B \tag{5.42}$$

Přímky definující jednotlivé roviny můžeme vyjádřit pomocí rovnice 5.42. Například přímka procházející bodem min, rovnoběžná s osou y, lze popsat jako:

$$y = min_x \tag{5.43}$$

Pro nalezení průsečíku paprsku s touto přímkou dosadíme do rovnice 5.43 rovnici paprsku pro jeho složku x. Dostaneme vztah:

$$O_x + td_x = min_x$$

Výpočet vzdálenosti průsečíku paprsku s přímkou $y = min_x$ je uveden v rovnici 5.44.

$$tmin_x = \frac{min_x - O_x}{d_x} \tag{5.44}$$

Obdobně můžeme vypočítat vzdálenosti s průsečíky dalších pěti přímek, definovaných body min a max. Výpočet všech vzdáleností je uveden v rovnicích 5.45

$$\begin{array}{rclrcl} tmin_x & = & \frac{min_x - O_x}{d_x} & tmax_x & = & \frac{max_x - O_x}{d_x} \\ tmin_y & = & \frac{min_y - O_y}{d_y} & tmax_y & = & \frac{max_y - O_y}{d_y} \\ tmin_z & = & \frac{min_z - O_z}{d_z} & tmax_z & = & \frac{max_z - O_z}{d_z} \end{array} \tag{5.45}$$

Vypočtené vzdálenosti t použijeme k tomu, abychom zjistili, zda paprsek protíná kvádr.

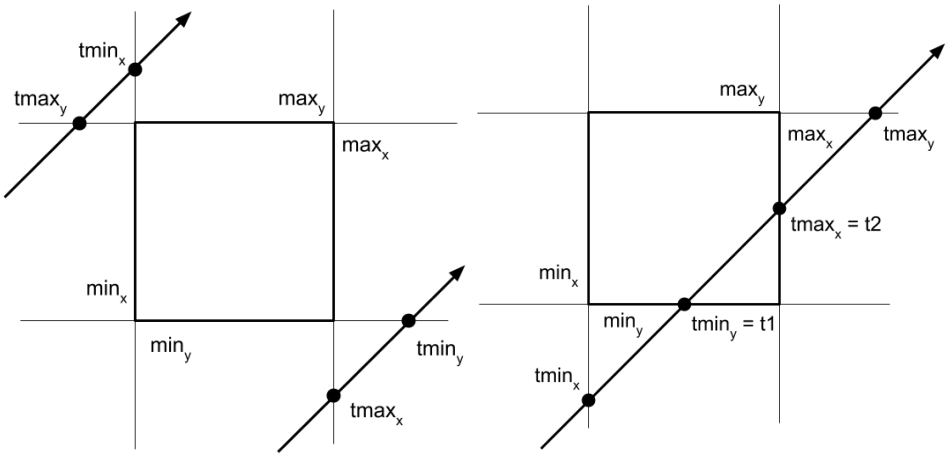
Na obrázku 5.9 je znázorněna situace nalezení průsečíku s přímkami v rovině xy. Levý obrázek 5.10a ukazuje situaci, kdy paprsek průmět strany kvádru míjí. Na pravém obrázku 5.10b je zobrazen paprsek procházející stejným průmětem.

Na levém obrázku 5.10a paprsek mine obdélník ve dvou případech. V prvním z nich protne přímku max_y dříve než přímku min_x. Ve druhém případě protne nejprve přímku max_x a až poté přímku min_y.

To, kterou přímku protne paprsek nejdříve, zjistíme porovnáním vzdáleností t odpovídajícím těmto rovinám.

Paprsek míjí obdélník v rovině, pokud platí podmínka 5.46.

$$tmax_y < tmin_x \vee tmax_x < tmin_y \tag{5.46}$$



(a) Paprsky míjí kvádr (b) Paprsek prochází kvádrem

Obrázek 5.10: Nalezení průsečíku s obdélníkem.

Pokud podmínka 5.46 není splněna, znamená to, že musí existovat alespoň jeden průsečík paprsku s průmětem kvádru.

Na obrázku 5.10b je zobrazen paprsek protínající přímky min_x, min_y, max_x a max_y ve vzdálenostech od počátku paprsku $tmin_x$, $tmin_y$, $tmax_x$ a $tmax_y$.

Vzdálenost bližšího z průsečíků $t1$, ve které paprsek vstupuje do obdélníku, odpovídá větší ze vzdáleností $tmin_x$ a $tmin_y$.

$$t1 = max(tmin_x, tmin_y) \tag{5.47}$$

Naopak vzdálenost $t2$, ve které paprsek opouští obdélník, odpovídá menší ze vzdáleností $tmax_x$ a $tmax_y$.

$$t2 = min(tax_x, tmax_y) \tag{5.48}$$

Nyní jsme nalezli průsečíky paprsku s obdélníkem v rovině xy. Pro rozšíření do trojrozměrného prostoru přidáme podmínku testující vzdálenosti průsečíků $tmin_z$ a $tmax_z$, vypočtené v rovnici 5.45, vůči hodnotám $t1$ a $t2$, které jsme vypočetli v předchozích rovnicích.

$$tmax_z < t1 \vee t2 < tmin_y \tag{5.49}$$

Pokud podmínka není splněna, paprsek opět prochází kvádrem. Vzdálenost průsečíku s rovinou ležící nejblíže počátku paprsku se určí podle vztahu 5.50.

$$t = max(tmin_z, t1) \tag{5.50}$$

Vzdálenost t představuje konečnou vzdálenost průsečíku od počátku paprsku.

5.5.2 Implementace

Postup popsaný v předchozí části úspěšně nalezne průsečík paprsku s kvádrem v případě, že je směr paprsku kladný ve všech svých složkách. Tomu tak však vždy není. Pokud je některá ze složek směru paprsku záporná, musíme upravit výpočty proměnných t_{min} a t_{max} z rovnic 5.45.

Upravíme vždy jen ty proměnné, pro které je složka směru paprsku záporná. Platí tedy

$$\begin{array}{llllll} Pokud & d_x < 0, & tak & tmin_x = \frac{max_x - O_x}{d_x} & a & tmax_x = \frac{min_x - O_x}{d_x} \\ Pokud & d_y < 0, & tak & tmin_y = \frac{max_y - O_y}{d_y} & a & tmax_y = \frac{min_y - O_x}{d_y} \\ Pokud & d_z < 0, & tak & tmin_z = \frac{max_z - O_z}{d_z} & a & tmax_z = \frac{min_z - O_x}{d_z} \end{array} \tag{5.51}$$

V případě, že je směr paprsku rovnoběžný s některou z rovin, v rovnicích 5.45 a 5.51 dojde k dělení nulou. To však není na škodu, protože můžeme využít toho, že kompilátor jako výsledek takového dělení vrátí hodnotu $-\infty$ pro t_{min} a $+\infty$ pro t_{max}. Rozdílné znaménko u ∞ pro hodnoty t_{min} a t_{max} zaručí, že podmínka 5.46 bude platná i pro takovéto případy. Více o tomto problému může být nalezeno v [3].

Z důvodů optimalizace můžeme v rovnicích 5.45 a 5.51 nahradit dělení směrem paprsku d násobením jeho převrácenou hodnotou $\frac{1}{d}$. Tuto hodnotu můžeme předpočítat při vytváření paprsku, protože se během výpočtů nemění.

6 Výpočet barvy

Poté, co jsem nalezli nejbližší průsečík ve scéně, je potřeba vypočítat jeho barvu. Implementovaný raytracer umožňuje výpočet barvy pomocí Phongova osvětlovacího modelu. Tento model umožňuje plynulé stínování ploch spolu s lesklými světelnými odrazy. Neumožňuje však zobrazit stíny. Proto jsou stíny vypočteny zvlášť, pomocí stínových paprsků.

6.1 Phongův osvětlovací model

Phongův osvětlovací model pro výpočet barvy rozkládá světlo odražené od povrchu objektu do tří složek.

I_a : ambientní složka

I_d : difúzní složka

I_s : spekulární složka

Součtem těchto tří složek získáme barvu tělesa v daném bodě.

$$I = I_a + I_d + I_s$$

6.1.1 Materiál

Pro výpočet jednotlivých složek Phongova osvětlovacího modelu je nejprve potřeba definovat materiál povrchu tělesa. Ten se skládá ze tří trojrozměrných barevných vektorů, r_a, r_d a r_s. Jednotlivé složky vektorů odpovídají barevným složkám R, G a B .Tyto vektory se používají pro výpočty jednotlivých barevných složek.

- Vektor r_a představuje ambientní barvu. Ta vyjadřuje schopnost povrchu odrážet ambientní světlo.
- Vektor r_d popisuje difúzní barvu tělesa, která se nejvíce podílí na výsledné barvě tělesa.
- Vektor r_s je koeficient zrcadlového odrazu a pro každou z barevných složek nabývá hodnot $0 \leq r_s \leq 1$.

6.1.2 Ambientní složka

Ambientní složka představuje světlo, které na těleso dopadá ze všech směrů zároveň. Toto světlo má vždy stejnou intenzitu a proto je konstantní pro celou scénu. V reálném světě by tomuto světlu odpovídalo světlo vzniklé mnohonásobnými odrazy od okolních těles.

Použití ambientní složky zabraňuje tomu, aby povrchy těles odvrácené od světelných zdrojů byly vždy černé. Změnou její intenzity lze měnit světlost scény.

Ambientní složka I_a se vypočítá podle vztahu 6.1,

$$I_a = I_A * r_a \tag{6.1}$$

kde koeficient I_A odpovídá množství okolního světla ve scéně a r_a ambientní barvě objektu. Operátor * představuje násobení vektorů po složkách.

V reálné scéně by se měl koeficient I_A s přibývajícím počtem světelných zdrojů zvětšovat. Protože se však s přibývajícími světelnými zdroji zvětšují i hodnoty difúzních a spekulárních složek, docházelo by k přesvětlení scény. Snížením koeficientu I_A v závislosti na počtu světel můžeme světlost scény snížit na požadovanou úroveň [12].

6.1.3 Difúzní složka

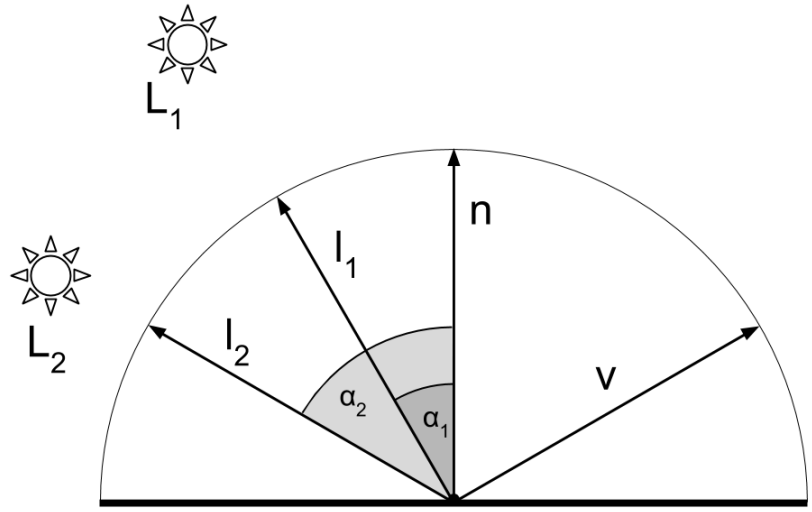
Difúzní složka představuje přímý světelný příspěvek od světelného zdroje umístěného ve scéně. Jejím vypočítáním získáme vystínovanou barvu tělesa, bez odlesků.

Difúzní světlo se na povrchu tělesa odráží rovnoměrně do všech směrů. Jeho intenzita je závislá na směru dopadajícího světelného paprsku l a normále povrchu v bodě výpočtu osvětlení n.

Lambertův zákon difúzního odrazu [12] uvádí, že intenzita difúzního světla je závislá na intenzitě osvětlení a úhlu α, který svírá normála povrchu n se světelným paprskem l. Parametr I_L v rovnici 6.2 představuje intenzitu světelného zdroje.

$$I_d = I_L * cos\alpha \tag{6.2}$$

Na obrázku 6.1 jsou zobrazeny dva světelné paprsky, l_1 a l_2, sestrojené z bodu výpočtu osvětlení, ke dvou světelným zdrojům L_1



Obrázek 6.1: Závislost intenzity difúzního osvětlení na velikosti úhlu α

a L_2. Protože úhel α_1 mezi normálou n a světelným paprskem l_1 je menší než úhel α_2, bude intenzita difúzního odrazu světla pocházejícího ze zdroje L_1 větší než ze zdroje L_2.

Z rovnice 6.2 vyplývá, že intenzita difúzního světla je největší, pokud světlo dopadá kolmo na povrch a klesá s rostoucím úhlem α mezi normálou a světelným paprskem.

Pokud pracujeme s vektory, můžeme výpočet 6.2 nahradit skalárním součinem normály n se světelným vektorem l.

$$I_d = I * (l.n) \tag{6.3}$$

Rovnice 6.3 do difúzní složky barvy bodu započítává jen barevnou složku světla. Pro započítání barevné složky vlastního tělesa vynásobíme rovnici 6.3 vektorem difúzní barvy materiálu tělesa r_d. Výsledný vztah pro výpočet difúzní složky tělesa je uveden v rovnici 6.4.

$$I_d = I_L * r_d * (l.n) \tag{6.4}$$

6.1.4 Spekulární složka

Světelné odlesky povrchu lesklých těles jsou obsaženy ve spekulární složce Phongova osvětlovacího modelu. Intenzita odraženého světla je závislá na poloze pozorovatele, světla a sklonu plochy, na které světlo dopadá. Její výpočet je popsán vztahem 6.5

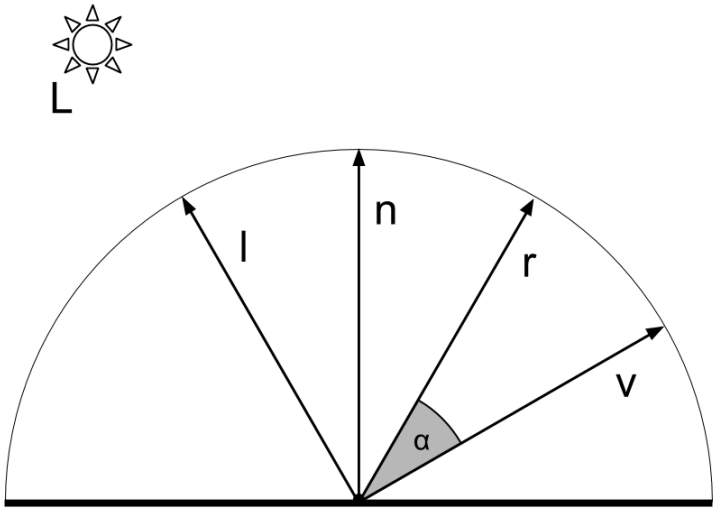
$$I_s = I_L * r_s * (v.r)^h \tag{6.5}$$

parametr I_L představuje intenzitu světelného zdroje a má stejnou hodnotu, jako při výpočtu difúzní složky. Parametr r_s odpovídá koeficientu zrcadlového odrazu materiálu tělesa.

Výpočet členu $(v.r)^h$ je znázorněn na obrázku 6.2. Světelný paprsek l je podle zákonu lomu 6.6 odražen ve směru r.

$$r = 2(l.n)n - l \tag{6.6}$$

Velikost úhlu α se vypočítá obdobně jako v případě difúzního osvětlení. A to pomocí skalárního součinu odraženého vektoru r s vektorem pohledu v. Koeficient h popisuje ostrost lesklého odrazu a je vždy kladný. Čím větší h zvolíme, tím menší a ostřejší budou světelné odrazy na povrchu těles.



Obrázek 6.2: Spekulární složka.

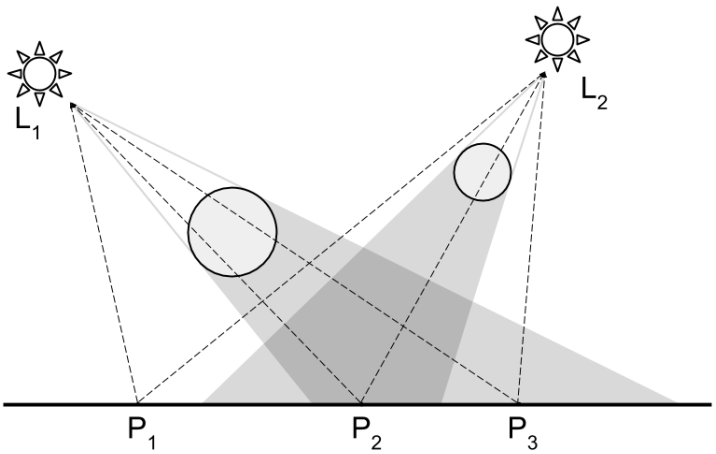
6.2 Výpočet stínů

Ke zjištění, zda bod průsečíku P, pro který počítáme osvětlení, leží ve stínu některého z těles ve scéně, využívá raytracer tzv. *stínové paprsky*. Tyto paprsky mají počátek v průsečíku. Jejich směr je vypočten jako rozdíl polohy průsečíku P a polohy zdroje světla ve scéně L. Rovnice stínového paprsku s je:

$$s = P + t(P - L)$$

Cílem stínového paprsku je zjistit, zda nějaké těleso leží mezi průsečíkem a světelným zdrojem. Pokud existuje průsečík stínového paprsku s některým z těles ve scéně, vrhá toto těleso stín na bod P. Stačí nám nalézt jen jedno stínové těleso. Více těles ležících mezi průsečíkem a zdrojem světla nemá na stín žádný vliv.

V případě, že je ve scéně více světel, sestrojíme stínový paprsek ke každému z nich. Na obrázku 6.3 jsou zobrazeny tři body. Bod P_1 neleží ve stínu žádného z těles a je tedy plně osvícen. Bod P_2 leží ve stínu obou z těles ve scéně a jeho barva tedy bude nejtmavší. A nakonec bod P_3 je osvětlen jen světelným zdrojem L_2.



Obrázek 6.3: Zobrazení stínových paprsků.

Pokud bod leží ve stínu, je do jeho osvětlení započítána jen ambientní a difúzní složka. Získaná hodnota je navíc vynásobena koeficientem, který představuje tmavost stínu. Tento koeficient nabývá hodnot od 0 do 1 a je nepřímo úměrný počtu světel ve scéně.

7 Zobrazení objemových dat

Raytracing je vhodný nástroj pro zobrazovaní objemových dat. Data, která si přejeme zobrazit, jsou uložena v buňkách pravidelné mřížky, tzv. *voxelech*. Ve většině případů se jedná o skalární data, která mohou reprezentovat například hustotu látky v daném bodě, jeho teplotu, nebo třeba stav objektu v čase.

Při výzkumu proteinů lze mřížka využít pro zobrazení tzv. *asynchronních tunelů* [2]. V každé z buněk mřížky je uložena hodnota určující počet snímku, ve kterých měl daný tunel neprázdný průnik s mřížkou. Zobrazení těchto dat nám potom dá představu o pozici tunelu v čase.

Po vystřelení paprsek putuje mřížkou a podle způsobu zobrazení zpracovává data z buněk, kterými právě prochází. Na obrazovku poté zobrazí hodnotu, kterou vypočítal při svém průchodu mřížkou. Data z buněk mohou být zpracována několik způsoby. Mezi základní přístupy zpracování objemových dat patří:

- Zobrazení maximální nebo minimální hodnoty z dat naměřených na dráze paprsku.
- Zobrazení součtu všech hodnot naměřených na dráze paprsku.
- Zobrazení hodnoty získané jako algebraický průměr z dat naměřených na dráze paprsku.

7.1 Konstrukce mřížky

Mřížku představuje trojrozměrné pole dat. Tato data mohou být například naměřené hodnoty, nebo výpočty algoritmů, které si přejeme zobrazit.

Pro procházení mřížkou musíme znát rozměry buňky a rozlišení mřížky. Velikost buňky ve třech směrech představuje vektor $cellSize$. Rozlišení mřížky udává vektor $resGrid$.

Abychom mohli data zobrazit, je potřeba umístit mřížku do prostoru. K tomuto účelu sestrojíme osově orientovaný kvádr, představující obalové těleso pro naše data. Jeho velikost $gridSize$ je dána rozlišením mřížky a velikostí jejích buněk.

$$gridSize = resGrid * cellSize \tag{7.1}$$

Pozice tohoto kvádru může být libovolná. Například pro kvádr o velikosti $gridSize$ se středem v počátku souřadného systému hodnoty minimálního min a maximálního max rohu odpovídají:

$$\begin{aligned} min &= -gridSize/2 \\ max &= gridSize/2 \end{aligned} \tag{7.2}$$

7.2 Procházení mřížkou

Pro procházení mřížkou použijeme algoritmus 3D-DDA (*3D-Digital Differential Analyser*), navržený Akirou Fojimotem [4].

7.2.1 Algoritmus 3D-DDA

3D-DDA algoritmus je rozšíření DDA algoritmu do třetího rozměru. Algoritmus 7.1 se skládá ze dvou fází, *incializační* (kroky 1 - 3) a *inkrementační* (kroky 5-8).

V inicializační fázi vypočteme třísložkový vektor vzdálenosti Δt, které paprsek potřebuje pro přechod do sousední buňky ve směrech x, y a z, vektor t obsahující vzdálenosti pro přechod do následující buňky a souřadnice buňky b, kterou paprsek prochází jako první.

Ve fázi inkrementační nejprve zpracujeme data v aktuální buňce. Poté vypočteme souřadnice buňky b, kterou paprsek projde v příštím kroku. A to tak, že najdeme nejmenší ze složek vektoru t, určující ve kterém ze směrů x, y a z následující buňka leží. Nejmenší ze složek t inkrementujeme o přírůstek Δt a postup opakujeme dokud paprsek neopustí mřížku.

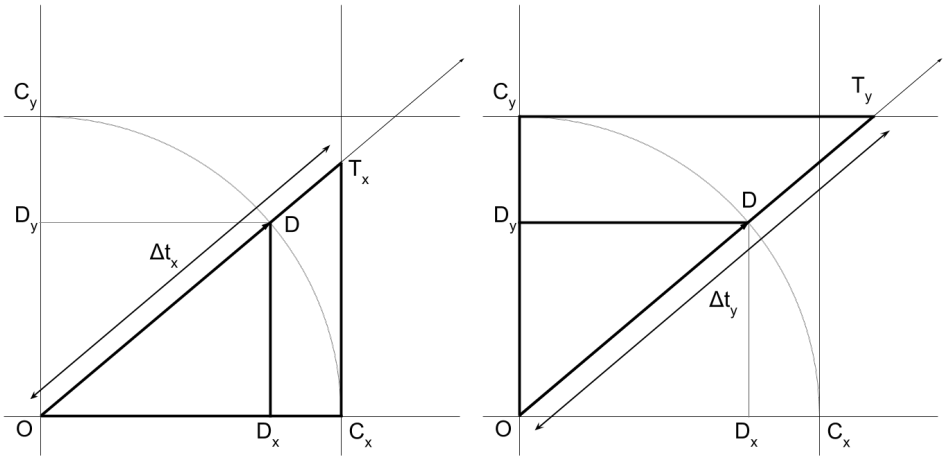
Inicializační část

Pro snadnější pochopení bude výpočet přírůstků vzdáleností vysvětlen nejprve pro dvojrozměrnou mřížku. Buňka má rozměry (C_x, C_y).

Na obrázku 7.1 je znázorněn výpočet vzdáleností Δt_x a Δt_y, ve kterých paprsek protne mřížku ve hranicích buňky rovnoběžných s odpovídajícími osami.

Algoritmus 7.1 3D-DDA algoritmus pro průchod paprsku trojrozměrnou mřížkou

1. Vypočti velikosti přírůstků vzdálenosti Δt ve směrech x,y a z.
2. Zjisti souřadnice počáteční buňky b
3. Vypočti počáteční vzdálenost t, ve které paprsek protne první stěnu buňky.
4. Dokud paprsek neopustí mřížku, opakuj
5. Zpracuj data v aktuální buňce
6. Najdi nejmenší složku t, určující přechod do další buňky
7. Podle nejmenší složky t aktualizuj index aktuální buňky
8. Inkrementuj nejmenší složku t o její hodnotu Δt



(a) Výpočet vzdálenosti Δt_x (b) Výpočet vzdálenosti Δt_y

Obrázek 7.1: Výpočet vzdáleností Δt_x a Δt_x pomocí podobnosti trojúhelníků

Výpočet vzdáleností je proveden pomocí podobnosti trojúhelníků. Pro vzdálenost Δt_x použijeme trojúhelníky OD_xD a OC_xT_x, zobra-

zené na obrázku 7.1a. Bod O odpovídá počátku paprsku v dané buňce a je totožný pro oba trojúhelníky. Hodnota D_x představuje složku x normalizovaného vektoru směru paprsku d. Bod D se nachází na paprsku ve vzdálenosti 1 od počátku paprsku.

Bod C_x, patřící druhému z trojúhelníků, představuje hranici buňky o velikosti C_x. Průsečík paprsku se stranou buňky rovnoběžnou s osou x je označen jako T_x.

Pro podobné trojúhelníky platí, že poměr odpovídajících si stran je pro všechny strany stejný. Tento vztah je pro situaci z obrázku 7.1a vyjádřen rovnicí 7.3.

$$\frac{D_x}{1} = \frac{C_x}{\Delta t_x} \tag{7.3}$$

Vyjádřením neznámé Δt_x z rovnice 7.3 dostaneme

$$\Delta t_x = \frac{C_x}{D_x} \tag{7.4}$$

Výpočet Δt_y je znázorněn na obrázku 7.1b. V tomto případě porovnáváme trojúhelníky ODD_x a OT_xC_x. Pro strany těchto trojúhelníků platí

$$\frac{D_y}{1} = \frac{C_y}{\Delta t_y} \tag{7.5}$$

Neznámá Δt_y se tedy vypočte pomocí vztahu 7.6

$$\Delta t_y = \frac{C_y}{D_y} \tag{7.6}$$

Pro rozšíření do třetího rozměru nám stačí jen vypočítat velikost Δt_z jako

$$\Delta t_z = \frac{C_z}{D_z} \tag{7.7}$$

Proměnné C, D i Δt jsou vektory. Výpočet Δt tedy můžeme zkráceně vyjádřit jako dělení vektorů po složkách

$$\Delta t = \frac{C}{D}$$

Vypočetli jsme vzdálenosti Δt, které paprsek musí urazit, aby překonal hranice buňky rovnoběžné s jednotlivými osami. Díky podobnosti trojúhelníků platí, že vzdálenost mezi průsečíky paprsku s jednotlivými stěnami buněk Δt je pro daný paprsek vždy stejná.

Pro určení souřadnic první buňky, kterou paprsek prochází, musíme nalézt průsečík paprsku s kvádrem obalujícím mřížku P. Vzdálenost

$$Pmin = P - min$$

představuje vzdálenost průsečíku P od minimálního rohu obalového kvádru min.

Její normalizací velikostí buňky $cellSize$ pomocí vztahu 7.8 získáme souřadnice průsečíku $Pcell$ v jednotkách vzdálenosti buňky. Dělení představuje dělení vektorů po složkách.

$$Pcell = \frac{Pmin}{cellSize} \tag{7.8}$$

Celočíselná část jednotlivých složek vektoru $Pcell$ udává souřadnice první buňky b, kterou paprsek v mřížce protne.

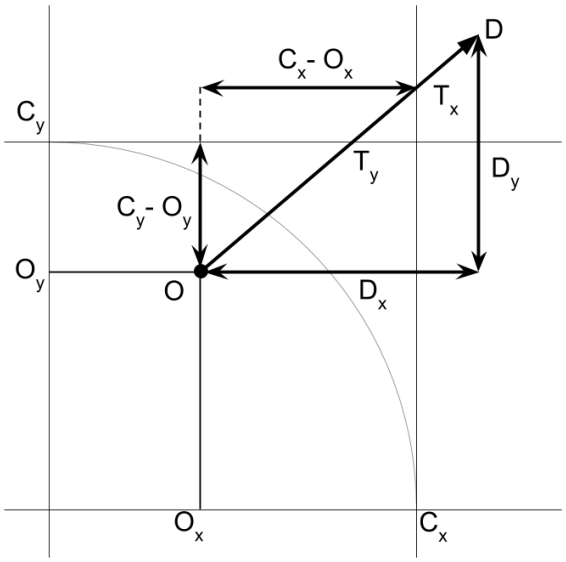
Na obrázku 7.2 je zobrazena situace, kdy počátek paprsku O nezačíná v dolním rohu buňky. V tomto případě se výpočetΔt nemění. Je ale potřeba vypočítat počáteční hodnoty vektoru t, určující vzdálenosti potřebné pro překonání prvních hranic buňky v jednotlivých směrech.

Pro jejich výpočet opět použijeme podobnost trojúhelníků. Dosazením jednotlivých proměnných dostáváme vzdálenosti:

$$t = \frac{C - O}{D}$$

Inkrementační část

Při procházení buňkami ležícími na dráze paprsku hledáme nejbližší průsečíky s hranicemi jednotlivých buněk. Ve vektoru t jsou



Obrázek 7.2: Výpočet počáteční vzdálenosti t

uloženy vzdálenosti k průsečíkům k následujícím nejbližším stranám buněk rovnoběžných s osami souřadného systému. Platí, že strana, kterou paprsek protne nejdříve, je určena nejmenší složkou vektoru t. Pokud je například nejmenší hodnota t_x, paprsek protne stranu buňky rovnoběžnou s osou x.

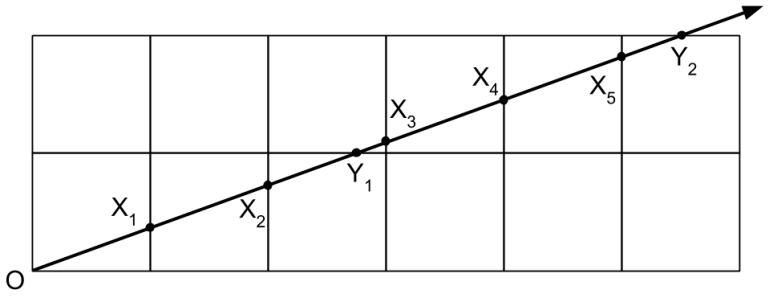
Inkrementační část se skládá z několika opakujících se kroků. Prvním z nich je *zpracování dat* uložených v současné buňce. To může být například akumulace dat z buněk ležících na dráze paprsku.

V druhém kroku nalezneme směr, ve kterém je hodnota t nejmenší. V závislosti na tomto směru inkrementujeme odpovídající index buňky b. Tato buňka bude zpracována v dalším kroku. Nakonec zvětšíme nejmenší vzdálenost t o hodnotu Δt, udávající vzdálenost k dalšímu průsečíku se stranou kolmou na daný směr.

Pokud je například nejmenší vzdálenost t_x, znamená to, že paprsek protne stranu buňky kolmou na osu x. Index buňky b_x zvětšíme o 1 a upravíme hodnotu $t_x = t_x + \Delta t_x$

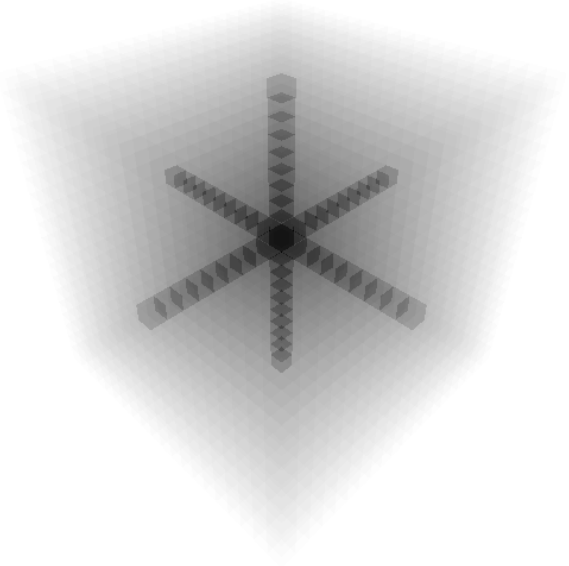
Na obrázku 7.3 je znázorněna situace, kdy paprsek nejdříve dvakrát protne stěnu buněk kolmou na osu x a to v bodech X_1 a X_2. Až ve třetí iteraci algoritmu je hodnota $t_y < t_x$ a paprsek tedy v bodě Y_1

nejdříve protne stěnu buňky kolmou na osu y.



Obrázek 7.3: Procházení paprsku mřížkou.

Na obrázku 7.4 je zobrazena mřížka s rozlišením každé ze stran 20 buněk. Všechny buňky nesou hodnotu 10. Výjimkou jsou buňky, představující osy jednotlivých stran V těchto buňkách byla hodnota nastavena na 100. Obrázek byl vytvořen součtem všech hodnot na dráze paprsku se 16 paprsky skrz pixel.



Obrázek 7.4: Ukázka zobrazení dat z mřížky. Jednotlivé pixely nesou hodnotu o součtu dat z mřížky na dráze paprsku.

8 Urychlující struktury

Detailní scény často obsahují sta tisíce objektů. Provádět test na nalezení průsečíku s každým z nich by bylo velmi nákladné. Je tedy vhodné použít pro organizaci scény některou z urychlujících struktur pro organizaci scény [12].

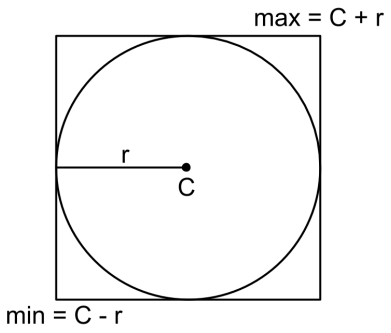
Jednou z urychlujících struktur je trojrozměrná mřížka představená v předchozí kapitole. S tím rozdílem, že místo skalárních dat jsou v jednotlivých buňkách uloženy odkazy na objekty ležící uvnitř těchto buněk.

8.1 Vytvoření mřížky

Pro vytvoření mřížky nejprve potřebujeme znát její celkovou velikost $gridSize$. Ta je dána velikostí obalového těla obsahujícího všechny objekty ve scéně. Toto těleso představuje osově orientovaný kvádr popsaný v kapitole 5.5.

Osově orientovaný kvádr je určen svým minimálním a maximálním rohem. Pro nalezení jejich polohy projdeme všechna tělesa ve scéně a porovnáme pozice jejich obalových těles.

Minimální roh obalového tělesa mřížky $gridMin$ odpovídá nejmenšímu z minimálních rohů všech obalových těles objektů ve scéně. Maximální roh $gridMax$ odpovídá tomu největšímu z maximálních rohů obalových těles.



Obrázek 8.1: Obalové těleso koule.

Na obrázku 8.1 je ukázka obalového tělesa koule. Jeho minimální roh *minCorner* je získán odečtením poloměru koule r od všech složek vektoru C, představujícího pozici středu koule. Maximální roh *maxCorner* se naopak získá přičtením r k C. Ostatní obalová tělesa jsou vytvořena obdobně.

Dalším krokem je určení počtu $resGrid$ a velikosti $cellSize$ buněk v mřížce. Tvar buněk by se měl co nejvíce blížit krychli. Kevin Suffer ve své knize [9] uvádí vztah pro výpočet optimálního rozlišení mřížky, využívané jako urychlující struktury. Tento vztah popsaný v rovnici 8.1 je závislý na velikosti mřížky a počtu objektů ve scéně.

$$\begin{aligned} s &= \sqrt[3]{\frac{gridSize_x * gridSize_y * gridSize_z}{n}} \\ resGrid_x &= floor(\frac{m * gridSize_x}{s}) + 1 \\ resGrid_y &= floor(\frac{m * gridSize_y}{s}) + 1 \\ resGrid_z &= floor(\frac{m * gridSize_z}{s}) + 1 \end{aligned} \tag{8.1}$$

Hodnoty vektoru $gridSize$ představují velikost mřížky v jednotlivých směrech. Parametr m umožňuje regulovat velikost rozlišení mřížky. Pro efektivní rozlišení mřížky leží jeho doporučená velikost mezi 1 a 2. Funkce *floor* vrátí celočíselnou část svého parametru. Získáme tak celočíselné rozlišení mřížky uložené ve vektoru $resGrid$.

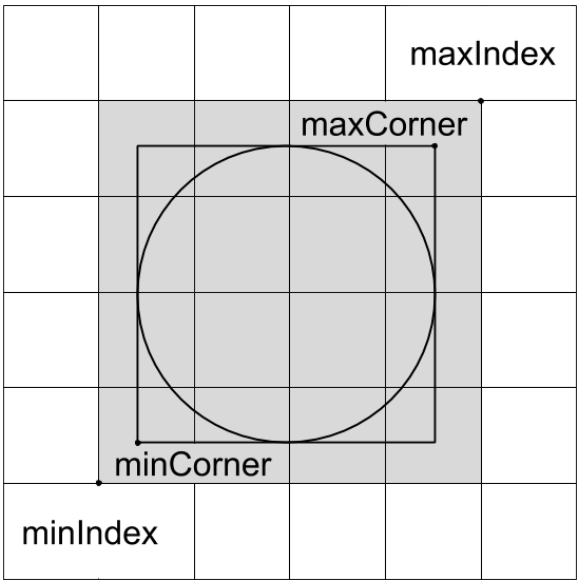
Vektor velikosti buňky $cellSize$ se vypočítá podle vztahu 8.2, kde dělení představuje dělení vektorů po složkách.

$$cellSize = \frac{gridSize}{resGrid} \tag{8.2}$$

8.2 Vložení objektů

Pro vložení objektu do mřížky je potřeba nalézt všechny buňky, které mají s daným objektem neprázdný průnik. Pro jejich nalezení převedeme souřadnice obalového tělesa objektu $minCorner$ a $maxCorner$ do souřadnic buněk mřížky $minIndex$ a $maxIndex$ zobrazených na obrázku 8.2.

$$minIndex = floor(\frac{minCorner - gridMin}{cellSize})$$
$$maxIndex = floor(\frac{maxCorner - gridMin}{cellSize})$$



Obrázek 8.2: Vložení koule do mřížky.

Tím získáme rozsah indexů buněk, do kterých zasahuje obalové těleso objektu. Šedé buňky na obrázku 8.2. Do všech těchto buněk přidáme odkaz na daný objekt.

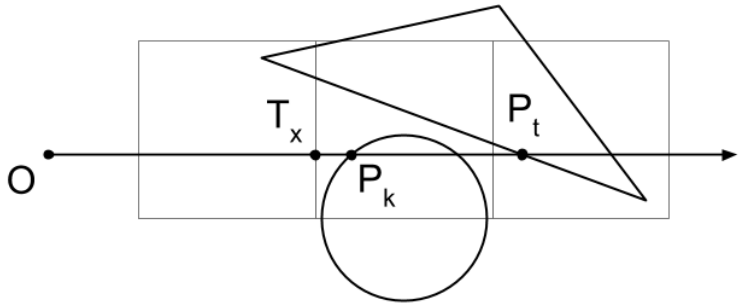
8.3 Nalezení nejbližšího průsečíku

Pro procházení mřížkou použijeme stejný algoritmus jako v kapitole 7. V kroku 5 algoritmu 7.1 však budeme místo zpracování dat hledat průsečík paprsku s objekty uloženými v právě zpracovávané buňce.

Pokud průsečík s objekty v buňce neexistuje, pokračujeme do další buňky. Pokud existuje, musíme zkontrolovat, zda opravdu leží v dané buňce. Průsečík se nachází v buňce, pokud je vzdálenost průsečíku od počátku paprsku menší než vzdálenost hranice s příští buňkou t.

Pokud průsečík paprsku s objektem existuje a leží v právě zpracovávané buňce, máme jistotu, že je to průsečík ležící nejblíže počátku paprsku. Výpočet tedy můžeme ukončit bez nutnosti procházení zbytku mřížky.

Na obrázku 8.3 je znázorněno nalezení průsečíku s objekty leží-



Obrázek 8.3: Určení, zda průsečík leží ve zpracovávané buňce.

cími v buňce nejblíže počátku paprsku O. Buňka obsahuje jediný objekt a to trojúhelník. Vzdálenost průsečíku paprsku s tímto objektem P_t od počátku paprsku je však větší než vzdálenost bodu T_x. Průsečík tedy neleží ve zpracovávané buňce a výpočet se přesune do buňky další, kde je nalezen nejbližší průsečík paprsku s koulí P_k.

8.4 Porovnání rychlosti

Pro porovnání rychlosti renderování s použitím mřížky a bez ní jsem vyrenderoval VdW zobrazení molekuly 1cqw zobrazené v příloze na obrázku 10.1 a její povrch zobrazený v příloze na obrázku 10.2. VdW zobrazení se skládá z 2 754 a povrch z 268 425 trojúhelníků.

Rendery byly vytvořeny v rozlišení 1920 x 1080 pixelů. Bylo použito adaptivní vzorkování popsané v kapitole 4.6. Jedním pixelem byl vržen 1 a 16 paprsků.

Zobrazení	Paprsků	Čas bez	/	S mřížkou	Zrychlení
VdW	1	160 s	/	1,5 s	106x
VdW	16	1300 s	/	22,7 s	57x
Povrch	1	x	/	7,9 s	x
Povrch	16	x	/	85 s	x

Tabulka 8.1: Porovnání rychlosti vykreslování s použitím mřížky a bez ní

Z tabulky 8.1 vyplývá, že využití mřížky značně urychlí vykreslení scény. Zobrazení povrchu molekuly bylo časově náročné a proto výpočet byl po půl hodině ukončen.

9 Závěr

Cílem práce bylo vytvořit raytracer, implementovaný jako modul v aplikaci CAVER Analyst. Tento modul mohou uživatelé použít vždy, když potřebují vytvořit obraz molekuly proteinu, nebo jejího tunelu, ve vysokém rozlišení. Podporované jsou všechny typy zobrazení popsané v kapitole 2.

Modul také umožňuje zobrazení objemových dat vzniklých při analýze asymetrických tunelů.

Pro snížení celkové doby vytvoření renderu je implementována urychlující struktura využívající mřížku pro organizaci scény. Nárůst rychlosti zobrazení je patrný zejména při vykreslování povrchu molekul a tunelů, kdy pracujeme se značným množstvím trojúhelníků.

Jako rozšíření současného řešení bych navrhl jeho další optimalizaci a to jak časovou, tak datovou. Dalším rozšířením může být vytvoření série renderů představující jednotlivé snímky dynamiky molekuly.

Literatura

[1] BENEŠ, P. *Computation and analysis of channels in protein dynamics*. Dizertační práce, Masarykova univerzita, Fakulta informatiky, 2011.

[2] BÝŠKA, J., JURČÍK, A., SOCHOR, J. Geometry-based Algorithm for Detection of Asymmetric Tunnels on Protein Molecules. 2013.

[3] ERICSON, C. *Real-Time Collision Detection*. Elsevier, 2005, ISBN 1-55860-732-3.

[4] FUJIMOTO, A., TANAKA, T., IWATA, K. Tutorial: Computer Graphics; Image Synthesis. 1988.

[5] GLASSNER, A. *An introduction to ray tracing*. Academic Press, 1989, ISBN 0122861604.

[6] KOZLÍKOVÁ, B. *Visualization Techniques for Static and Dynamic Protein Molecules and Their Channels*. Dizertační práce, Masarykova univerzita, Fakulta informatiky, 2011.

[7] SCRATCHPIXEL. Rays, Cameras and Images. 2012. URL <http://www.scratchapixel.com/lessons/3d-basic-lessons/lesson-6-rays-cameras-and-images/>

[8] SHIRLEY, P., MORLEY, K. *Realistic Ray Tracing*. A K Peters Ltd., 2003, ISBN 1-56881-198-5.

[9] SUFFERN, K. *Ray Tracing from the Fround Up*. A K Peters Ltd., 2007, ISBN 978-56881-272-4.

[10] WHITTED, T. An Improved Illumination Model for Shaded Display. *Commun. ACM*, 1980.

[11] WILLIAMS, A., BARRUS, S., MORLEY, R. K., aj. An Efficient and Robust Ray-box Intersection Algorithm. 2005.

[12] ŽÁRA, J., BENEŠ, B., SOCHOR, J., aj. *Moderní počítačová grafika*. Computer Press, 2004, ISBN 80-251-0454-0.

Seznam obrázků

2.1 Molekula 1edw, zobrazena pomocí VdW *5*
2.2 Molekula 1edw, zobrazena pomocí tyčinek a koulí. *6*
2.3 Molekula 1edw, zobrazena pomocí tyčinek. *7*
2.4 Povrch molekuly 1edw *8*
2.5 Tunel zobrazený pomocí středové linie. *8*
2.6 Tunel zobrazený pomocí koulí, představující jeho šířku v daném bodě středové linie *9*
2.7 Detailní povrch tunelu *9*

3.1 Výpočet barvy pixelu (x,y) *12*
3.2 Obecná struktura raytraceru *13*
3.3 Struktura raytraceru - třídy a metody *14*

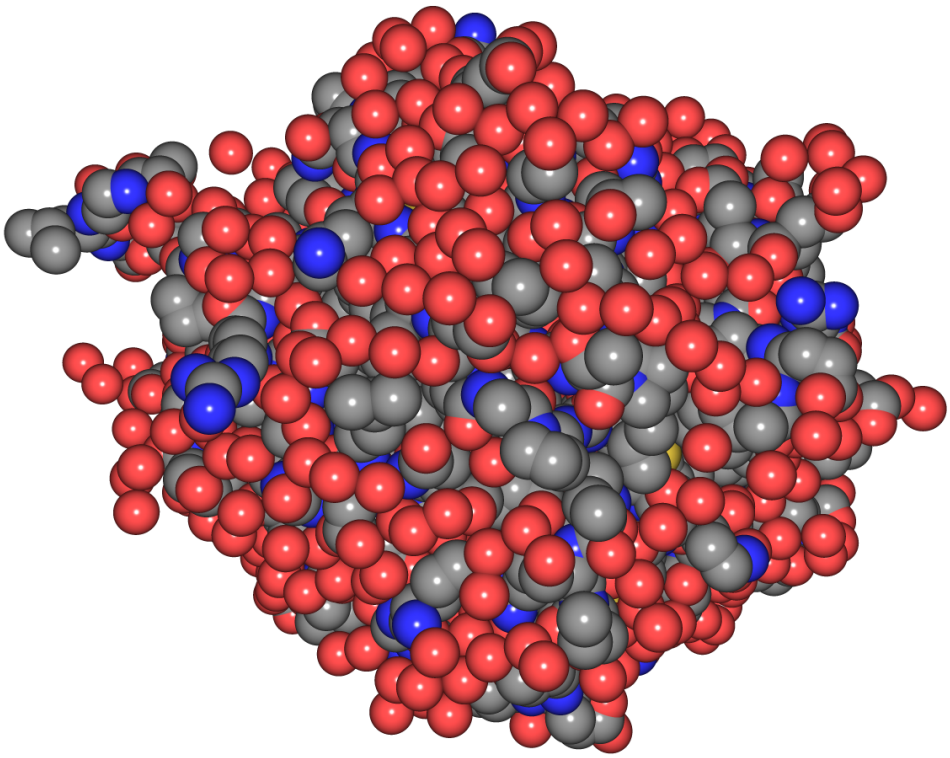
4.1 Poloha bodu na přímce v závislosti na parametru *t* *15*
4.2 Popis parametrů kamery. *16*
4.3 Vztah mezi obrazovkou a plátnem *17*
4.4 Výpočet směru primárního paprsku *18*
4.5 Molekula 1edw renderována jedním paprskem na pixel *20*
4.6 Supersampling - ukázka *21*
4.7 Vzorky *22*
4.8 Molekula 1edw renderována 16 paprsky na pixel *23*

5.1 Barycentrické souřadnice trojúhelníku ABC *25*
5.2 První podmínka. Výpočet vektoru *oc*. *31*
5.3 Druhá podmínka. Výpočet t_{om}. *32*
5.4 Třetí podmínka. Výpočet d^2. *33*
5.5 Třetí podmínka. Výpočet t_{pm}^2. *34*
5.6 Válec definovaný body A a B *35*
5.7 Nalezení průsečíku s válcem. *36*
5.8 Výpočet vzdálenosti t_{AM} *37*
5.9 Kvádr definovaný rohy $min\ max$ *39*
5.10 Nalezení průsečíku s obdélníkem. *40*

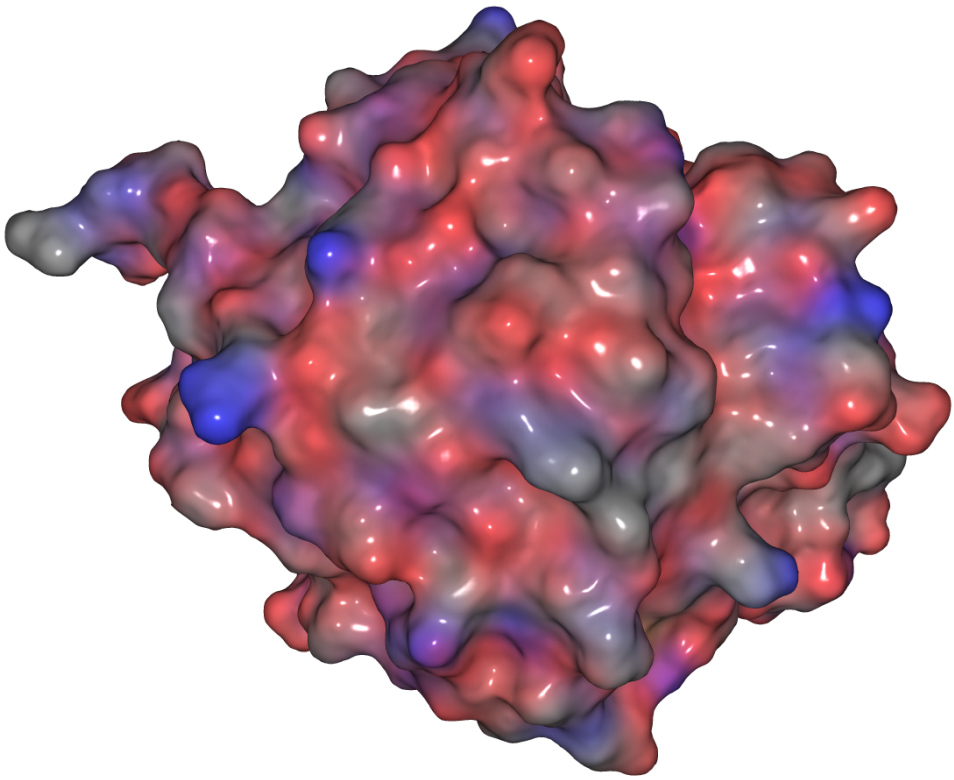
6.1 Závislost intenzity difúzního osvětlení na velikosti úhlu α *45*
6.2 Spekulární složka. *46*
6.3 Zobrazení stínových paprsků. *47*

7.1 Výpočet vzdáleností Δt_x a Δt_x pomocí podobnosti trojúhelníků *50*
7.2 Výpočet počáteční vzdálenosti t *53*
7.3 Procházení paprsku mřížkou. *54*
7.4 Ukázka zobrazení dat z mřížky. Jednotlivé pixely nesou hodnotu o součtu dat z mřížky na dráze paprsku. *54*
8.1 Obalové těleso koule. *55*
8.2 Vložení koule do mřížky. *57*
8.3 Určení, zda průsečík leží ve zpracovávané buňce. *58*
10.1 VdW zobrazení molekuly 1cqw, rederované v rozlišení 1920x1080 pixelů, 16 paprsků skrz pixel *64*
10.2 Povrch molekuly 1cqw, rederovaný v rozlišení 1920x1080 pixelů, 16 paprsků skrz pixel *65*

10 Příloha



Obrázek 10.1: VdW zobrazení molekuly 1cqw, rederované v rozlišení 1920x1080 pixelů, 16 paprsků skrz pixel



Obrázek 10.2: Povrch molekuly 1cqw, rederovaný v rozlišení 1920x1080 pixelů, 16 paprsků skrz pixel