Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2007

Jan Syrovátka

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Modelování slunečního cyklu

Plzeň, 2007 Jan Syrovátka

Abstract

In this thesis we explore the possibilities of the sky and sun modeling. At the start we must create an object that will represent the sky. Our scene will be „packed“ to the sphere with the sky texture. We also present an inexpensive model that approximates the effects of the sunset and sunrise. This is a very important part of this thesis, because the sun tends to become red or orange, especially when is low in the sky in this time we can also watch that the horizon has an another color than rest of the sky. We must analyse the sunpath within a year.

In the second parth of this thesis we will dispute a real-time shadow techniques in a computer graphics. Here will be closely describe several of these methods. Than we used one of them -concretely a technique known as the Shadow maps – for any objects in our scene. These algorithm has two passes: In the first pass whe create a depth map – here is stored how far away every fragment is from the light. In the second pass our scene will be rendered normally and it is now possible to determine whether a fragment can 'see' the light source.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne *29.4.2007 Jan Syrovátka*,

Obsah

1. Úvod

Cílem práce je vytvoření vzájemně nezávislých knihoven: Pro zobrazení bezmračné oblohy a stínů. V první je hlavní zřetel brán na období stmívání a rozednívání, kdy by obloha měla co nejlépe připodobňovat realitu. Dalším důležitým aspektem této části práce je slunce, které se má po obloze pohybovat tak, aby to co nejlépe kopírovalo jeho skutečnou trajektorii v průběhu roku. Se sluncem jsou spjaty stíny, které vrhají objekty, na něž sluneční paprsky dopadají. Vytvořením stínů od zadaného zdroje světla se pak zabývá druhá knihovna. Obě knihovny by měly být připojitelné k vybrané klientské aplikaci.

V teoretické části jsou rozebrány metody umožňující praktickou realizaci jednotlivých součástí aplikace. Jsou zde vždy zhruba nastíněny postupy, které vedou k výsledku. Podrobněji zde jsou vysvětleny metody generování stínů a to včetně problémů spojených s použitými metodami a řešení, která mohou vést k jejich odstranění. Jednotlivé kapitoly realizační části pak blíže specifikují jednotlivé součásti vypracovaného řešení. Pokud bylo v teoretické části označeno více způsobů vedoucích k řešení, je pak v realizační části popsáno to řešení, které je v samotné aplikaci použito a to včetně některých implementačních detailů. Rovněž jsou zde popsány závažnější problémy, které se v průběhu implementování objevily. Pro ně je buď přímo uvedeno, jak byly odstraněny či případně nastíněno řešení, jakým způsobem by se odstranit daly.

2. Teoretická část

2.1 Sféra

Celá scéna musí být umístěna do nějakého objektu, který bude scénu zastřešovat. Možností pro typ tohoto objektu je několik:

1.) Krychle či kvádr – Implementace je nejjednodušší, ale problémem jsou spoje stěn, které by byly hodně patrné a působili rušivě.
2.) Válec – Problém spoje stěn by zde byl částečně vyřešen, kromě horní podstavy, která by i zde byla vidět.
3.) Koule – Nejlepším objektem je koule(sféra). Její vytvoření z trojúhelníků je sice náročnější než u krychle, ale nejlépe modeluje realitu.

2.1.1 Mapování textury na sféru

Proces nanášení textury na povrch těles se nazývá mapování textury a je předurčen třemi důležitými faktory:

1.) Definicí textury, tj. kolika rozměrná textura je
2.) Tvarem tělesa, na které je nanášena
3.) Mapovanou veličinou

Rovinou texturu můžeme definovat jako funkci $\mathrm{T}(\mathrm{u}, \mathrm{v})$, přiřazující bodům v rovině hodnoty mapované veličiny. Abychom mohli texturu aplikovat, musíme zavést funkci $\mathrm{M}(\mathrm{x}, \mathrm{y}, \mathrm{z})$, která každému bodu na povrchu tělesa přiřazuje bod z definičního oboru textury T. Funkce M ve vzorci (2.1.1) se označuje jako *inverzní mapování*.

$$M : D_M \rightarrow D_T, \quad kde...D_M \subset R^3 \; jsou\, body\, na\, povrchu\, koule \quad a\; D_T \subset R^2$$

Vzorec 2.1.1: Inverzní mapovací funkce

Při inverzním mapování kulové plochy se postupuje takto: Sféru o daném poloměru posuneme tak, aby její střed ležel v počátku souřadného systému. Body ležící na povrchu vytvořené sféry můžeme vyjádřit pomocí sférických souřadnic. Tak jak je uvedeno ve vzorci (2.1.2)

$$[\mathrm{x}, \mathrm{y}, \mathrm{z}] = \left[r * \cos\alpha * \cos\beta, r * \sin\alpha * \cos\beta, r * \sin\beta\right], \quad kde\ \alpha \in \langle 0, 2\pi \rangle\ a\ \beta \in \langle -\pi/2, \pi/2 \rangle .$$

Vzorec 2.1.2: Bod sféry převedený do sférických souřadnic

Inverzní mapovací funkce M má pak tedy pro kouli tvar:

$$u = \frac{1}{2\pi} ar\cos\frac{x}{\sqrt{x^2 + y^2}}, pro...y \le 0 \qquad v = 0.5 + \frac{1}{\pi}\arcsin\frac{z}{r}$$

$$u = 1 - \frac{1}{2\pi} ar\cos\frac{x}{\sqrt{x^2 + y^2}}, pro...y \succ 0$$

Vzorec 2.1.3: Inverzní mapovací funkce pro otexturování sféry [Spher05]

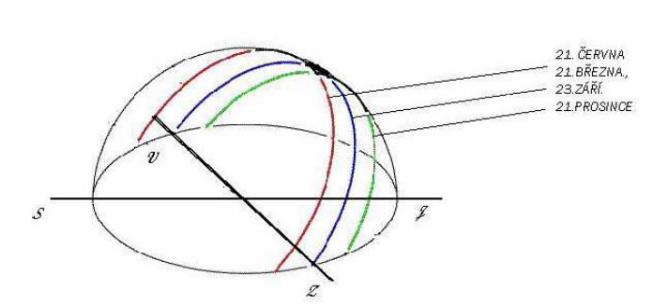
2.2 Slunce

Slunce jako takové je možno vytvořit více způsoby. Možností by bylo vytvořit sféru podobnou té pro zemi, která by byla z vnitřku sféry pro pozorovatele viditelná. Sféra představující zemi by kolem té pro slunce rotovala. Další možností je vytvořit pouze desku, na níž bude nanesena poloprůhledná textura a tato deska se bude pohybovat uvnitř sféry. První možnost lépe kopíruje realitu. Slunce zde je koulí jako ve skutečnosti, její rozměry jsou několikrát větší, než má samotná Země. Dalším aspektem je, že v tomto modelu Země rotuje kolem slunce a ne naopak, což také lépe popisuje skutečný stav věcí. Problémem zde však bude zařídit správné zobrazování reality v závislosti na čase. V realitě totiž Země rotuje kolem slunce v eliptických drahách a navíc rotuje kolem své osy. Obě tyto rotace mají za následek například střídání ročních období, či v případě rotace kolem osy střídání dne a noci. Den bude vždy na té polokouli, které je ke slunci přivrácená.

Světlo od slunce by muselo neustále dopadat na sféru, představující Zemi, tedy by muselo být všesměrové. Alternativou by bylo nepohybovat s ním v závislosti na slunci ale na Zemi. Sféra by si pak vhodnými rotacemi zajistila, že na její vybrané části světlo dopadat bude a na jiné ne, tím by vytvořila iluzi toho, že na některých místech na ní je den, na jiných noc. Složení všech těchto částí dohromady, spolu s dalšími obtížemi činí tento model nepoměrně pracnější a možná ani ne zcela správně funkční. Druhý model není tak podobný realitě - slunce je zde jen deskou rotující po obvodu země. Modelování zmíněných jevů je však při tomto přístupu mnohem méně pracné. Světlo od slunce je vytvořeno jako bodové a putuje se sluncem samotným. Pokud v tomto případě chceme vytvořit noc, slunce prostě zajde za horizont.

2.2.1 Podstata fyzikální dráhy slunce po obloze

Slunce celoročně vychází přibližně na zeměpisném východě a na západě zapadá. Neputuje však přímou trasou z východu na západ po nadhlavníku. Jeho trajektorie je o jistý úhel naklopena vzhledem k zemské ose. Úhel tohoto naklopení zůstává celoročně stejný. To samozřejmě neznamená, že by dráha slunce byla celoročně stejná. Mění se vzdálenost, o níž se východ slunce posouvá. Díky tomuto posunutí urazí slunce pokaždé i jinou vzdálenost a to má vliv na délku dne a noci. Nejdelší trasu musí absolvovat 21. června. Postupně se dráha krátí, nejkratší je pak 21.prosince. Tudíž od jarní rovnodennosti směrem k červnu se dráha pomalu prodlužuje. Od podzimní dále pak zkracuje. Tyto speciální situace ilustruje obr.2.2.1, všechny ostatní dny je dráha někde mezi nimi. Slunce totiž v průběhu roku nevychází ani nezapadá na stále stejném místě. Tato poloha se mění v závislosti na aktuálním ročním období.



Obr.2.2.1: Cesta slunce po obloze během roku

Toto vše je samozřejmě zjednodušené nahlížení, protože slunce samo o sobě se po obloze takto ani nijak jinak nepohybuje. Jeho zdánlivý pohyb během dne i posun dráhy během roku je ve skutečnosti způsoben rotací země kolem své vlastní osy a také rotací kolem slunce.

2.2.2 Barevné prostory a barva slunce

Slunce, ač se do něj ve skutečnosti nemůžeme příliš dlouho dívat, mění v průběhu dne svoji barvu. Tento barevný přechod je možno simulovat použitím různých koncových barev a vzájemným přechodem mezi nimi. K tomu je možné využít různé barevné prostory. Barevných prostorů existuje celá řada. Všechny mají své výhody i nevýhody, což také určuje jejich použití. V dalším textu jsou blíže specifikovány dva z nich:

1.) RGB – jako nejznámější a v počítačové grafice hojně využívaný prostor
2.) HSV – je vybrán pro některé jeho důležité vlastnosti, které usnadňují práci s barevnými přechody v něm

Nejznámějším zástupcem je barevný prostor RGB. Zde různé barvy, které se používají při vytváření obrazu, jsou tvořeny kombinací několika základních barev z barevného spektra. Na barevné obrazovce například vidíme barvu jako výsledek složení tří složek – červené (R, red), zelené (G, green) a modré(B, blue). Složíme-li v tomto barevném prostoru červenou, zelenou a modrou dohromady v plné intenzitě získáme barvu bílou. Podobně získáme stupně šedi skládáním všech tří barev se shodnou, postupně se snižující intenzitou až k černé barvě. Základní vlastností

tohoto barevného prostoru je součtové(aditivní) skládání barev – čím více barev sečteme, tím světlejší bude výsledek.

Dalším prostorem je prostor HSV. Tento prostor taktéž definuje barvu jako trojici složek. Zde však (H, hue) znamená barevný tón, (S, saturation) sytost a (V, value) jasovou hodnotu. Barevný tón označuje převládající spektrální barvu, sytost určuje příměs jiných barev a jas je dán množstvím bílého(bezbarvého) světla. Prostor podobně jako prostor HLS, který je jeho obdobou umožňuje postupně měnit barevné charakteristiky při zachování typických vlastností barvy. Pouhou úpravou složky představující jasovou hodnotu V, lze barvu ztmavit či zesvětlit.

2.2.3 Světelné zdroje

Světlo je elektomagnetické vlnění o vnové délce přibližně 390nm (světlo fialové) až 760 nm (světlo červené). Světlo má dualistický charakter, to znamená, že má částicové i vlnové vlastnosti. Ve vakuu se šíří rychlostí $3 \cdot 10^8 m \cdot s^{-1}$, ve všech ostatních prostředích o něco pomaleji. V počítačové grafice se používá několik druhů světelných zdrojů:

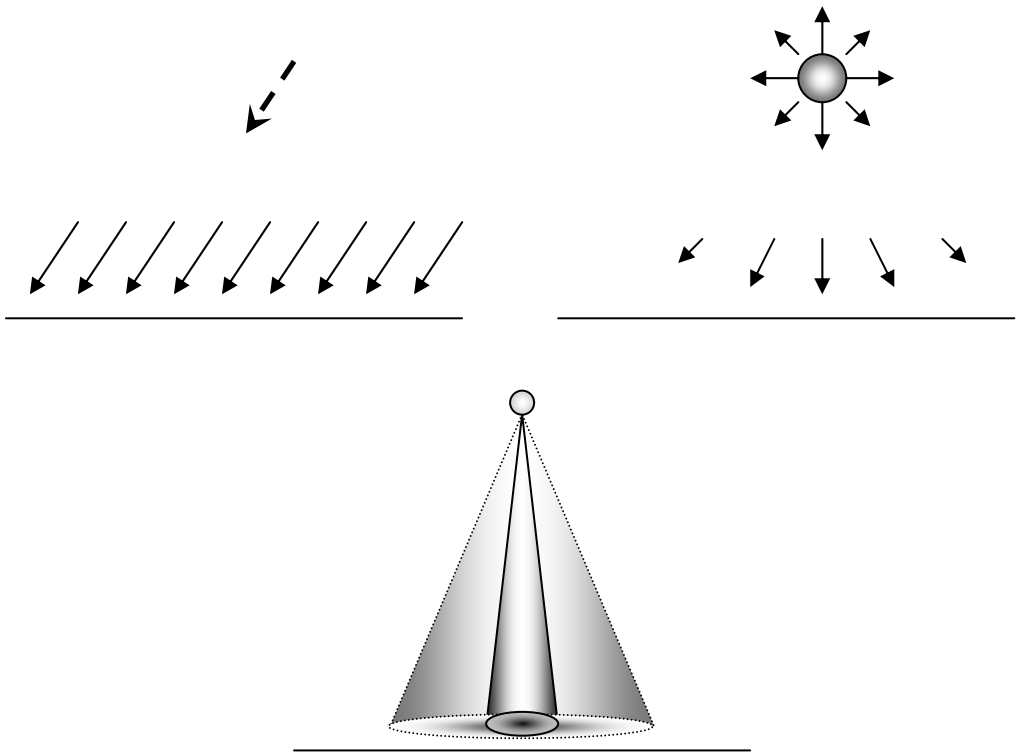
1.) Směrový zdroj světla (Directional light)
2.) Bodový zdroj (Point light)
3.) Reflektor (Spot light)

Směrové světlo je používáno pro simulaci vzdálených světelných zdrojů, jako například slunce, protože jím lze poměrně vhodně aproximovat právě sluneční svit. Je určeno směrem. Světelné paprsky pak dopadají rovnoběžně s tímto směrem.

Bodové světlo reprezentuje bodový zdroj. Na rozdíl od směrového má místo směru určenu pozici. Z této zadané pozice jsou do scény vysílány paprsky všemi směry a mají stejnou intenzitou. Lze jej použít pro simulaci zářivých světelných bodů jako například slunce, měsíč nebo hvězdy.

Reflektor je velmi komplexním zdrojem světla – je určen pozicí světla a směrem. Pouze objekty, které se vejdou do zorného pole jsou světlem ovlivněny. Nejvíce světla je vyzařováno ve směru osy vyzařování. Kolem této osy může být

jasnější oblast. Tento zdroj může sloužit v aplikacích jako baterka, která nejjasněji září uprostřed a směrem ke kraji intenzita světla klesá.



Obr. 2.2.3: Zdroje světla, vlevo nahoře směrový zdroj světla, vpravo nahoře bodový a uprostřed reflektor.

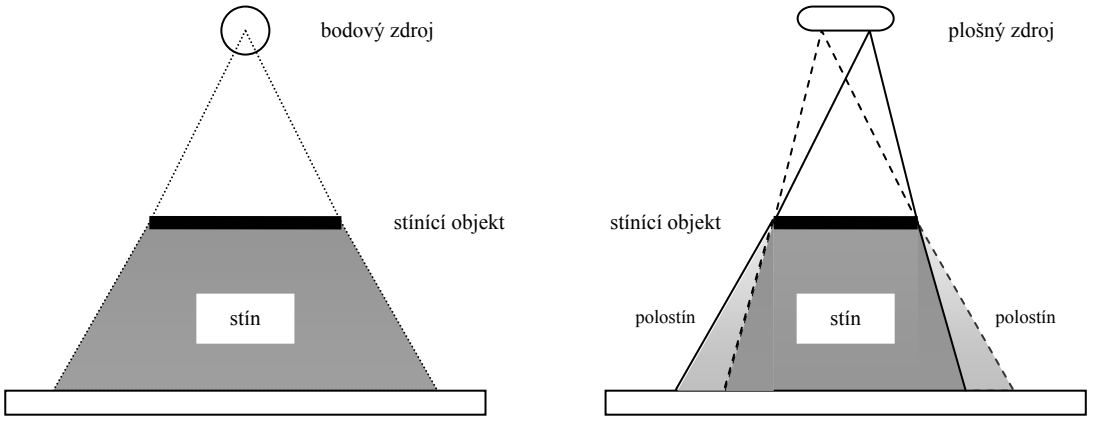
2.3 Stíny

Při prostorovém vnímání scény hrají stíny významnou roli. Slouží pro správné pochopení 3D scény, kterou se snažíme převést do roviny. Bez nich lze těžko odhadnout skutečný tvar nebo rozměr objektů, stejně tak jejich rozložení ve scéně. Rovněž díky stínům lze odvodit polohu a povahu světelných zdrojů. V počítačové grafice se stíny rozdělují na dva druhy:

1.) vržené – stíny, který vrhají tělesa na sebe navzájem
2.) vlastní – stíny, které objekt vrhá sám na sebe

Pozice objektů, které stíny mají vrhat, objektů(ploch), na něž stíny dopadají a světla ovlivňuje tvar a rozměry stínu. Charakter stínu je pak ovlivněn velikostí světla.

V zásadě lze světelné zdroje brát jako bodové či plošné. Bodové zdroje světla dokáží vytvářet ostré stíny. Jelikož jde o bod lze pro tento bod stanovit, zda je z libovolného objektu ve scéně viditelný či nikoliv. Plošné zdroje světla dokáží produkovat měkké stíny. U nich se již nejedná pouze o části plochy, které jsou ve stínu nebo ne. Mezi nimi je přechod, kterému se říká polostín.



Obr.2.3: Vlevo ostrý stín způsobený bodovým zdrojem světla. Vpravo stín s polostínem ze zdroje plošného

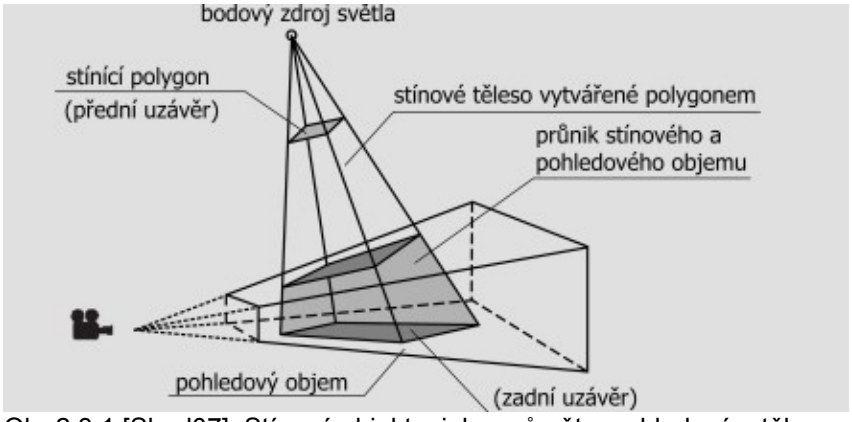
Důležitým faktorem pro stín je jeho barva. Jakou barvou vlastně má být stín ve scéně vykreslen. Bylo by možné brát v úvahu, že objekty ve stínu jsou zcela zastíněny stínícími objekty. Při této úvaze by šlo vykreslovat stín zcela černou barvou. Ve skutečnosti však stíny doopravdy černé nejsou a částečně je pod nimi vidět i povrch objektu, který je zastíněn. V počítačové grafice se proto barva stínů získává pomocí míchání barvy stínu s barvou zastíněného objektu nebo pomocí osvětlovacího modelu, kdy se stíny vykreslují jen při ambientním a částečně difúzním osvětlení bez uvažování zrcadlové složky.

Pro zobrazování stínů v aplikacích, kde záleží na rychlosti zobrazování se pro generování stínů nepoužívají globální zobrazovací metody. Místo toho dostávají přednost rychlejší samostatné metody generování stínů. Jejichž podstatou je většinou řešení viditelnosti. Mezi nejčastěji používané metody generování stínů patří Stínové těleso a Stínová paměť hloubky.

2.3.1 Stínová tělesa

Algoritmus v základní podobě pracuje s polygony a bodovými světly a poskytuje tedy pouze ostré stíny. Základní myšlenkou je vytvoření tzv. stínového tělesa pro každý ze stínících objektů. Jak ukazuje obrázek 2.3.1, stínové těleso (též stínový objem) ohraničuje část prostoru ve scéně, ze kterého není přes stínící objekt (v tomto případě jediný polygon) světelný zdroj vidět, a vymezuje tak zdrojem neosvětlený prostor. Stínové těleso je potřeba vytvořit pro každý ze stínících objektů. Známe-li všechna stínová tělesa, stačí během zobrazování testovat vzájemnou polohu zobrazovaných objektů, resp. jejich povrchových polygonů, s těmito tělesy. Výsledkem testu může být jedna z následujících tří situací:

1.) Polygon leží celý uvnitř stínového tělesa – je tedy ve stínu,
2.) Polygon leží celý vně stínového tělesa – je osvětlen světelným zdrojem, a konečně
3.) Polygon leží částečně ve stínovém tělese – je nutné provést rozdělení polygonu na osvětlenou a neosvětlenou část



Obr.2.3.1 [Shad07]. Stínový objekt a jeho průmět s pohledovým tělesem

Určení stínového tělesa pro jeden samostatný stínící polygon není příliš náročné. U obecného stínícího objektu je třeba nejprve nalézt jeho obrys – ten určuje hranici stínu. Obrys je složen z obrysových hran. Každá tato hrana definuje jednu stěnu stínového tělesa. Stínové těleso lze vytvořit i dalším způsobem: Je možno jej vytvořit zvlášť pro každou plochu daného objektu, na kterou ze světelného zdroje dopadá světlo. Tento způsob se sice může zdát být jednodušším, ale znamená jisté zvýšení počtu stínových těles. Tím se prodlužuje doba, po níž se stíny přepočítávají.

Určení polohy stínového tělesa a polygonu nemusí být zcela triviální záležitostí, pokud například polygon leží jenom částečně ve stínovém tělese (bod 3).

V případě algoritmu stínového tělesa se částečně osvětlené polygony nerozdělují na zcela osvětlené a zastíněné části, ale pracuje se v prostoru rastru. Celý proces má fungovat tak, že jsou vysílány testovací paprsky ze středu promítání každým pixelem (pojem vysvětlen v příloze A - Terminologie) směrem k zobrazovanému povrchu ve scéně. Paprsek při své cestě zřejmě bude protínat tělesa, kterými prochází. Při tom se počítá, kolikrát testovací paprsek vstoupil do nějakého stínového tělesa a kolikrát z něj naopak vystoupil. Na konci se vezmou tyto počty a odečtou se od sebe. Pokud výsledek je roven nule, pak testovací paprsek opustil všechna stínová tělesa, do nichž vstoupil. Z toho plyne, že bod na povrchu tělesa, ke kterému paprsek mířil nebude zastíněn. Pokud naopak výsledek nulový bude znamená to, že paprsek zůstal v některém ze stínových těles, kterými na své cestě prošel. Bod tedy bude ležet ve stínu. Znamená to tedy, že pokud paprsek vstoupí do tělesa a dorazí k objektu je těleso automaticky přesunuto do stínu. Naopak pokud ze stínového tělesa někde vystoupí tento účinek je zrušen a objekt, ke kterému pak paprsek dorazí není zastíněn. K tomu, aby těleso nebo jeho část ležely ve stínu, pak musí platit, že se nacházejí za přivrácenými a současně před odvrácenými plochami stínových těles [Shad07]. Existují dva základní algoritmy implementující výše uvedený postup. V anglické literatuře jsou označovány jako*:*

1.) depth-pass, který je poněkud snadnější naimplementovat a bývá rychlejší. Vznikají zde však častěji problémy s ořezáváním.
2.) depth-fail, který je o něco složitější, pomalejší ale univerzálnější.

V prvně jmenovaném algoritmu každému pixelu přiřadíme čítač, jehož počáteční hodnota bude rovna počtu stínových těles, uvnitř kterých se nachází kamera. Provede se hloubkový test polygonů stínových těles s ostatními objekty ve scéně. Bude-li pro zvolený pixel test úspěšný, tj. polygon stínového tělesa se nachází před všemi objekty ve scéně, pak v případě přivrácené stínové plochy čítač inkrementujeme a v případě odvrácené plochy dekrementujeme. Pokud po zpracování všech ploch stínových těles zůstane v čítači nenulová hodnota, pak pixel leží ve stínu. V druhém algoritmu obrátíme smysl algoritmu depth-pass, tj. budeme měnit hodnotu čítače (iniciálně nastaveného na nulu) jen v případě, že hloubkový test selže (stěna stínového tělesa bude za testovaným pixelem). Budou se tak vlastně počítat navštívená a opuštěná stínová tělesa paprskem, který směřuje z nekonečné

vzdálenosti k zobrazovanému (nejbližšímu) povrchu – tedy přesně opačným směrem, než v depth-pass algoritmu. Obě metody se většinou implementují naráz a teprve během zobrazování se rozhoduje o tom, který z nich se doopravdy použije vzhledem k dané situaci.

Rozšířením algoritmu může být použití i pro tvorbu měkkých stínů [Nish85], kdy se zkonstruují stínová tělesa pro rohy plošných světelných zdrojů, tak jako by v nich byly umístěny bodové zdroje světla. Výsledkem je tolik stínových těles, kolik je rohů plošného zdroje. To je samozřejmě zaplaceno zvýšením výpočetní složitosti.

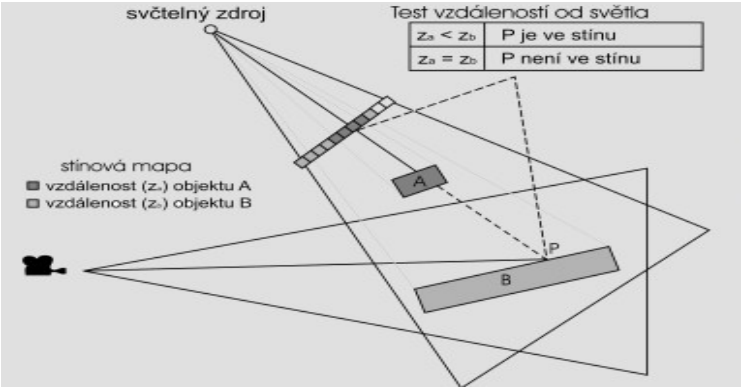
2.3.2 Stínová paměť hloubky

Metoda stínového tělesa počítala stíny v objektovém prostoru a vytvářela proto geometricky přesné stíny. Kladla však jisté nároky na scénu. Scéna musela mít ploškovou reprezentaci a objekty byly rozděleny na ty, které stíny vrhají a na ty, na něž dopadají. Metoda stínové paměti hloubky (shadow depth map) počítá stíny v obrazovém prostoru. Její výhodou je, že dokáže pracovat s libovolnou reprezentací scény a jednoduchost. Implementovat ji lze podle následujícího algoritmu [Will78]:

1.) Zobrazení scény z pohledu světelného zdroje Li.
Hodnoty z paměti hloubky (Z-buffer) jsou uloženy do hloubkové mapy Hi.
2.) Scéna se zobrazí z pohledu kamery pomocí paměti hloubky
3.) Pro všechny pixely [u, v] (s hloubkou w) zobrazené scény se provádí:
- *a.)* Převedení bodu [u, v, w] do soustavy zdroje světla Li a získání jeho nové souřadnice [x, y, z].
- *b.)* A = Hi[x, y]
- *c.)* B = z.
- *d.)* Pokud (A < B), pak je pixel [u, v] ve stínu, jinak je zdrojem světla Li osvětlen.

Je-li více zdrojů světla, vytvoří se hloubková mapa pro každý z nich. Třetí bod se pak opakuje pro každou takto vytvořenou hloubkovou mapu. Mapy lze užívat opakovaně do té doby, dokud se objekty, od nichž se stíny uvažují, nezmění. Rovněž tak po změně polohy světla je potřeba hloubkové mapy vytvořit znovu.

Metoda pracuje ve dvou krocích. Nejdříve se zobrazí scéna z pohledu světla (s řešením viditelnosti pomocí paměti hloubky) a její obsah se uloží do tzv. hloubkové mapy (depth map). V této mapě pixely nenesou barevnou informaci, ale jejich hodnota reprezentuje vzdálenosti od kamery, v tomto případě od světelného zdroje. Ve druhém kroku se scéna zobrazuje z pohledu pozorovatele s řešením viditelnosti algoritmem Z-buffer. Pixely hloubkové mapy tohoto obrazu se transformují do pohledu ze světla a takto přepočtená vzdálenost od světla se porovná se souřadnicí *z* v paměti hloubky (viz obr. 2.3.2) Leží-li bod získaný z pohledu kamery po transformaci dále, nežli při pohledu ze světla, je ve stínu a jeho intenzita se sníží.



Obr.2.3.2 [Shad07]. Stínová paměť hloubky: pohled ze světla a od pozorovatele

Ani metoda stínové paměti hloubky není bezproblémová. Problémy vznikají při použití hloubkových map, na jejichž rozlišení je dosažená kvalita stínů závislá. Při malém rozlišení hloubkové mapy budou mít hrany vytvořených stínů příliš výrazný schodovitý charakter. Rozlišení hloubkové mapy by tedy mělo být dostatečné, v ideálním případě by na každý zobrazovaný pixel měl připadat právě jeden pixel v hloubkové mapě. Toho však v praxi většinou nelze dosáhnout, neboť vytváření hloubkových map s velkým rozlišením by bylo časově příliš náročné. Rozlišení hloubkových map je proto vždy kompromisem mezi kvalitou a rychlostí. Použití diskrétní hloubkové mapy a přístup do ní může mít tedy za následek vznik *aliasingu.* Vznik aliasu (pojem vysvětlen v příloze A - Terminologie) navíc podporuje skutečnost, že každý zobrazovaný pixel reprezentuje určitou plochu z povrchu těles, jejíž projekce do hloubkové mapy může pokrýt hned několik jejích pixelů. Tento problém lze odstranit tak, že se při porovnání vzdáleností v testu neuplatňuje jedna jediná hodnota, ale i několik okolních. Pro každou z nich se pak provede samostatné srovnání s transformovanou souřadnicí. Další možností je použití perspektivní

stínové mapy – *perspektive shadow map* [Stam02], kdy se mapy produkují až po perspektivním promítání. Tímto postupem je pro objekty, které se nacházejí blíže středu promítání, zvětšeno rozlišení hloubkové mapy a pro vzdálenější naopak.

Problém aliasingu je odstranitelný pomocí metody perspektivní stínové mapy. Metoda stínové paměti hloubky v sobě nese další úskalí, a tím je *self-shadow aliasing*. Zde následkem konečné (nedostatečné) přesnosti paměti z-buffer a numerických nepřesností vrhá těleso stín samo na sebe, i když by nemělo. Je to důsledek jevu, kdy pro jeden bod vyjdou rozdílné hodnoty v hloubkové mapě a hodnoty transformované do soustavy souřadnic světla. Problém *self-shadow aliasu* lze elegantně vyřešit, pokud se ve scéně vyskytují jen *manifoldy* (pojem vysvětlen v příloze A - Terminologie). Pokud ano, stačí při vytváření hloubkové mapy vykreslovat pouze odvrácené plochy. Alias ve skutečnosti nezmizí, ale přesune se z přivrácených ploch na odvrácené. Zde však nebude patrný, neboť odvrácené plochy budou zatemněny v procesu stínování. Obecné řešení tohoto problému spočívá v technice, kdy se do hloubkové mapy místo jednoho jediného nejbližšího bodu ukládá průměr z hodnot vzdáleností dvou nejbližších bodů promítnutých do dané pozice v hloubkové mapě [Woo92]. Díky tomu je nejbližší těleso vždy plně osvětleno a tělesa za ním leží ve stínu.

Jak je patrné z obrázku 2.3.2, je tato technika generování stínů velmi efektivní pro směrové světelné zdroje. Hloubková mapa totiž dokáže obsáhnout jen tu část scény, která je pro daný světelný zdroj určena pohledovým tělesem. Pokud má světelný zdroj světlem zasahovat do většího prostoru scény, než je možno postihnout jednou hloubkovou mapou, pak nezbývá nic jiného, než vytvoření více hloubkových map. Naopak nejméně efektivní je pro všesměrové světelné zdroje, kdy je třeba vytvořit až šest hloubkových map. Každá z nich pak reprezentuje pohled přes jednu stěnu pomyslné krychle obklopující světelný zdroj.

Krom výše zmíněných dvou metod existují i další způsob, jakým stíny vytvářet. Například projekční metody, které vypočítávají stíny pomocí vhodně zvoleného zobrazení. S ohledem na polohu světelného zdroje se pro každou plochu, na kterou může dopadat stín, nalezne transformace zobrazující do roviny této plochy libovolný stín - vrhající objekt jako dvourozměrný obrazec. Tato metoda je velmi jednoduchá na aplikaci, ale její jednoduchost je zaplacena velký počtem problémů, která jsou s ní spojena.

3. Realizační část

Téměř celá aplikace je napsána v programovacím jazyce C#. Jedinou výjimku tvoří *shader*(pojem vysvětlen v příloze A - Terminologie), použitý pro vytvoření stínů, který je v jazyce HLSL. Veškeré grafické náležitosti jsou vytvořeny využitím managed DirectX 9. Nástrojem, který byl využit pro naprogramování a odladění celé práce bylo Microsoft Visual Studio 2005.

3.1 Pohyb a rozhlížení

Pohyb pozorovatele po scéně a rozhlížení po ní je umožněn pomocí `Microsoft.DirectX.DirectInput`, jehož instance je vytvořena zvlášť pro klávesnici, sloužící k pohybu a zvlášť pro myš, která slouží k rozhlížení. Pohyb samotný je řešen pomocí pohybového vektoru, ke kterému se při každém kroku přičítá směrový vektor. Směrový vektor určuje, kterým směrem bude další pohyb vykonán. Tento vektor je ještě předtím znormován a nakonec vynásoben, aby bylo dosáhnuto odpovídající rychlosti. Celou tuto funkčnost obstarává metoda `ReadKeyboard()`. Ve scéně nejsou umístěny žádné objekty, s nimiž by bylo třeba řešit kolize. Rozhlížení na myši je řešeno v metodě `ReadMouse()`. V této metodě jsou počítány dva úhly: *azimuth*, pro rozhlížení v horizontálním směru a *zenith*, pro rozhlížení vertikálně. Pro *zenith* platí, že musí ležet v intervalu od $-\pi/2 \; do \; \pi/2$, tím je zajištěno, že se při pozorování scény nemůžeme ve vertikalním směru přetočit. Navíc je zde spočten směr, kam se pozorovatel dívá, ten se pak využívá jinde.

3.2 Zeměkoule

3.2.1 Modelování objektu, v němž je scéna umístěna

Jako objekt pro zastřešení scény byla vybrána koule – sféra. Zeměkoule, obklopující celou scénu je samostatným objektem a je vytvořena pomocí `VertexBufferu`, který obsahuje jednotlivé vrcholy sféry. Jejich počet lze nastavit

přímo z programu, čím více, tím je sféra méně kostrbatá. Sousedy každého vrcholu nastavuje `IndexBuffer`. Pro sféru jsou důležitými parametry počet dílků, z nichž je složena a především její poloměr. Zatímco počet dílů ovlivňuje pouze sféru samotnou a určuje, jak kostrbatá bude, její poloměr ovlivňuje celou vykreslovanou scénu. Poloměru se totiž odvíjí rozměry slunce apod. Závislost FPS (pojem vysvětlen v příloze A - Terminologie) na parametrech sféry je uvedena v příloze C.1. Z testu vyplývá, že uživatel má celkem velkou volnost v nastavení parametrů, při dodržení doporučených rozmezí a postupů v testu uvedených. Po té co je sféra dokončena, je třeba připodobnit ji ke skutečnému vzezření oblohy. Prvním pokusem bylo texurování sféry.

3.2.2 Texturování sféry

Skládání sféry z trojúhelníků je náročnější nežli složení krychle. Stejně tak i texturování sféry představuje větší problém než otexturování krychle.
Pro texturování povrchu sféry je použita dvojrozměrná textura.

Jak vyplývá ze vzorce (2.1.3) pro určitý bod povrchu sféry – konkrétně pokud bude x-ová i y-ová souřadnice nulová, bude textura uložena zde do jediného bodu, tomu v aplikaci odpovídají póly sféry. Textura jako taková je určitým způsobem v jejich okolí taktéž deformovaná.

Kvůli deformaci textury směrem k pólům je textura vybrána jednobarevná a bez jakýchkoliv motivů, které pokud by při texturování „vyšly“ právě na inkriminovaná místa, působily rušivě. Například pokud by se nějaký motiv ocitl v oblasti okolo pólu byl by zdeformován do nepatrného, oválného útvaru.

3.2.3 Přechod mezi texturami

Sféra byla „polepena“ dvěma různými texturami. Jedna představující denní oblohu a druhá noční. Mezi těmito texturami pak byl vytvořen přechod za pomoci dalších dvou textur a změn intenzity a dosahu světla . Tím bylo docíleno toho, že při stmívání se světle modrá obloha postupně zatahuje a tmavne. Nakonec se změní v klasickou noční černomodrou s hvězdami. Ráno, při svítání naopak stejným způsobem denní střídá noční.

Efekt při tomto postupu nebyl špatný, měl pouze dvě nevýhody: První byla jakási nepřesnost v inverzní mapovací funkci (vzorec 2.1.3) pro kouli, kdy přes celou její délku se táhla rýha, v níž byla celá textura uložena znovu. Tento problém byl vyřešen, otočením koule, tak že rýha byla na horizontu. Následně byla podlaha posunuta nad její úroveň. Zeměkoule po celý cyklus rotuje kolem osy y, takže rýha není nikdy viditelná. Druhým problémem byla obtížná simulace efektu při stmívání a rozednívání. Zde je obloha zabarvena v několika vrstvách. Na horizontu při okraji bývá většinou oranžová, žlutá či červená nad ní pak modrá. Tento efekt bylo možno docílit pomocí renderování do textury.

3.2.4 Renderování do textury

Pozorovatel je obklopen pravidelným n-úhelníkem, na jehož stěny je v případě potřeby nanesena textura, odpovídající obsahu samotného renderu. To znamená, že se na texturu renderuje vše, co se děje v samotné scéně. Výjimku tvoří desky, představující podlahu, která se na texturu nenanáší. Důvod je ten, že n-úhelník představuje iluzi oblohy se sluncem. Výhodou tohoto je, že na n-úhelník je možné vykreslit i věci, které ve vlastním renderu nejsou. Například by bylo možné zde v některých chvílích-odpovídajících času východu resp. západu slunce nechat dočasně svítit světlo. Světlo by obarvovalo svou barvou oblohu na textuře. Tím lze dosáhnout například oné barevné stopy na obloze při západu či východu slunce. Není to možnost jediná, dosáhnout by se tohoto efektu dalo za pomoci renderování textury více způsoby. Problémem renderování do textury může být citelné zpomalení celé aplikace.

Z tohoto důvodu se tento postup opravdu ukázal nepříliš vhodným, protože zbytečně aplikaci zpomaloval. Navíc by bylo potřeba v pravidelných časových intervalech renderování opakovat, aby nevznikaly velké rozdíly, v tom co se promítá na danou renderovací plochu a tím, co se doopravdy ve scéně děje. Docílený výsledek navíc nebyl takový, aby se tento postup doopravdy vyplatil.

3.2.5 Obarvování bodů z nichž je sféra složena

Pokus s texturováním sféry a renderováním do textury nepřinesl patřičný výsledek. Nahradit ho mělo postupné obarvování bodů, z nichž se sféra skládá. V konečné fázi mělo obarvování poskytnout realističtější efekt. Především šlo o to, aby při svítání kolem horizontu byly barvy jako žlutá nebo světle oranžová. Ty by měli ve větší výšce přecházet ve světle modrou nebo šedou. Později, když se slunce dostane výš, by tento pruh měl postupně přejít do barvy zbytku oblohy. Od poledne dál by měla být situace obrácená, tedy čím víc se bude slunce přibližovat k západu a rovině podlahy. Přitom by měla obloha tmavnout, později se pod sluncem objeví sytě oranžová až červená skvrna, která se s klesajícím sluncem rozplyne do pruhu přes celý horizont podobně jako při svítání. Nakonec by obloha měla od nachové přejít v tmavě modrou až černou v noci. Šlo by tedy pouze o funkci času, která v závislosti na jeho hodnotě přiřazuje obloze barvu, jakou má mít.

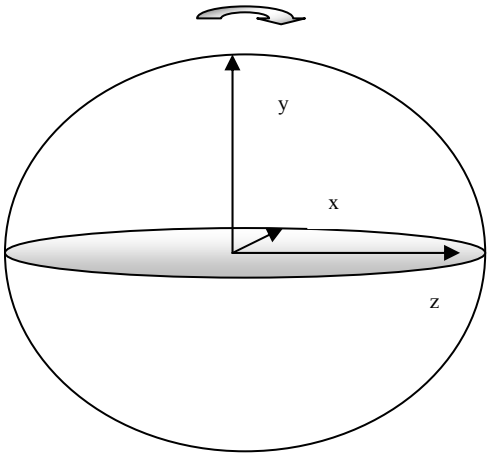
Proto byla aplikace textur dočasně nahrazena obarvováním bodů v závislosti na denním čase. I zde se však objevil problém: v přímém okolí světla se některé zadané barvy měnily na bílou. Šlo i o odstíny modré, které by byly k reálnému modelování oblohy zapotřebí. Tím pádem nešlo docílit požadovaného efektu.

V konečné verzi došlo k návratu k texturování. S tím, že efekt více barevných vrstev současně byl vytvořen pomocí válce, který je umístěn mezi sféru a slunce. Válec se objevuje pouze v době východu a západu slunce, v jiný čas je válec skryt pod podlahou.

3.2.6 Pohyb sféry

Pro napodobení reality, kdy zeměkoule rotuje kolem své osy, i sféra rotuje kolem pozorovatele (resp. kolem osy y - jak ukazuje obr. 3.2.6). Při každém renderu se rotuje o určitý úhel. Velikost tohoto úhlu je určena rychlostí otáčení. Samotný uživatel má pak možnost měnit tuto rychlost na *Panelu nastavení*.

Čím větší rychlost je nastavena, tím rychleji sféra okolo osy y rotuje. Naopak čím menší rychlost bude, tím menší je i úhel rotace oproti předchozímu renderu. Se zmenšujícím se úhlem klesá i rychlost rotování sféry. Rotování lze i vypnout/zapnout, tím se zastaví resp. rozběhne čas (viz kapitola 3.6.1).



Obr. 3.2.6: Sféra s označením os a směrem rotace

3.3 Modeláž horizontu

3.3.1 Vytvoření válce

Jak již bylo řečeno v kapitole: 3.2.3 bylo zapotřebí vytvořit na horizontu při východu a západu slunce pruh, doprovázející rozbřesk a setmění. Po vyzkoušení několika možných přístupů pro vytvoření tohoto pruhu se nejlépe osvědčil válec, umístěný mezi sféru a pohybující se slunce. Válec je vytvořen podobně jako sféra. To jak je hranatý, určuje počet obdélníkových stran, z nichž je vytvořen. Válec, složený z n stran má výšku, která je spočtena z poloměru sféry. Takže pro různá nastavení poloměru sféry si zachovává konstantní poměr mezi jím a velikostí výškou válce.

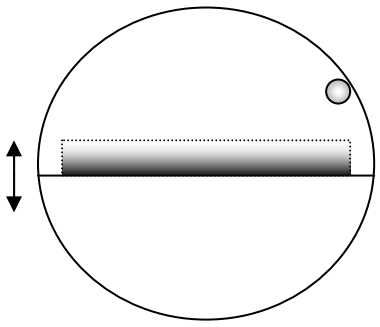
3.3.2 Textura na válci

Aby byl válec k užitku, je na něm rovněž jako na sféře nanesena textura. Avšak textura nanesená na válec není zcela shodná s tou na sféře. Obsahuje barevný přechod od některé z cílených barev pruhu(oranžová, fialová, žlutá) k barvě, kterou má obloha. Důležité je, že horní okraj textury na válci je již v barvě oblohy. Tedy celý přechod mezi barvami je na textuře pro válec. Tím pádem je eliminován nežádoucí efekt, kdy by byl okraj textury válce patrný na sféře. Ten by vzniknul, kdyby byla na válci oranžová nebo jiná tato barva a na sféře modrá.

Odstínů pro textury je více. V závislosti na roce jsou střídány textury s různými odstíny. Například v zimě jsou použity poněkud světlejší odstíny než v létě. Rovněž červená nebo spíše fialová je použita výlučně při západu slunce. Naopak oranžová či žlutá je používána většinou při svítání. Všechny textury jak na sféře, tak odstíny pro jednotlivé textury používané na válci jsou vytvořeny v programu *Gimp 2.2.10.*

3.3.3 Pohyb válce během dne

Protože bylo řečeno, že pruh na obloze je patrný pouze při setmění či při rozbřesku a textura,která je na válci nanesená je stále stejná. Je zřejmě potřeba válec v některých chvílích „schovat“ jindy zase ukázat. Válec se tedy pohybuje nahoru a dolů po ose y a to v závislosti na denním čase. Pokud se blíží východ slunce, začíná válec postupně vyjíždět po obloze vzhůru, pak na určité mezi se zastaví, aby nad okraj podlahy nevyjel příliš a neobjevila se sféra za ním. Jak den pokračuje, začne opět válec sjíždět zpět pod povrch podlahy tak, aby nebyl vidět. Pokud se den schyluje ke konci a slunce klesá k západu, pak začne pruh vyjíždět znovu. Večer má jiný odstín než ráno. Dřív než slunce zajde za obzor pruh zmizí a objeví se opět až ráno(viz. obr. 3.3.3).



Obr.3.3.3: Sféra s válcem a sluncem

3.4 Slunce

3.4.1 Vytvoření slunce

Nejdůležitější částí aplikace je slunce. Slunce je vytvořeno pomocí čtvercové desky. Na tuto desku je aplikována textura, která je uprostřed neprůhledná. Směrem k okraji se průhlednost zvyšuje. Tím je docíleno toho, že přes střed slunce není vidět, zato po krajích prosvítá obloha v pozadí.

Velikost slunce nebo spíše desky, na níž je nanesena textura pro slunce se dynamicky mění v závislosti na tom v jaké výšce se slunce právě nachází. V ranních hodinách, kdy je slunce nízko těsně nad horizontem, je deska se sluncem největší. Jak postupně stoupá k nadhlavníku, deska se zmenšuje až do té doby, kdy je nad hlavou pozorovatele. V tu chvíli je to pouze drobná tečka asi jako slunce v poledne. Od té chvíle, kdy opět slunce začne klesat k obzoru, se opět zvětšuje. Při pohybu přes horizont se spíše rozvine v barevný pruh. Velikost desky je rovněž nezávislá na poloměru sféry - mění se podle jeho nastavené velikosti.

3.4.2 Pohyb slunce

Stejně jako rotuje sféra okolo pozorovatele ani slunce nezůstává staticky na místě. Jeho pohyb se poněkud liší od rotace samotné sféry. V každém renderu je spočítán úhel, který svírá slunce v daném okamžiku s rovinou podlahy. Velikost úhlu podobně jako pro sféru je ovlivněna rychlostí otáčení. Na rozdíl od sféry se u slunce rotuje okolo osy z. Rychlost otáčení v obou případech ovlivňuje velikost rotace stejně, takže čím větší je nastavena, tím rychleji se slunce i koule pohybují. Pohyb slunce však není ani ve skutečnosti takto jednoduchý. A tak je navíc přidána rotace okolo x o úhel *naklopení*, který by měl simulovat úhel, pod kterým sluneční paprsky dopadají na zem. Je to pomyslný úhel mezi sluncem a osou y. Velikost tohoto úhlu udává, jak vysoko nad zemí se bude slunce pohybovat. Důležité je, že tento úhel se v průběhu roku pro slunce nemění a tedy sklon dopadu paprsků je stále stejný. Parametrem, který se však mění je posun od skutečného zeměpisného východu viz 2.2.1.

Tím vším je docíleno toho, že slunce na jednom místě vychází a na opačné straně pak zapadá. Samotné slunce je přitom pouze ploška s texturou. Tato ploška

pak rotuje po obvodu koule dle popsaného způsobu. Přesněji řečeno, nerotuje přímo po něm, ale kousek před ním, aby byla z povrchu země viditelná. Věrněji by realitu simuloval již pouze model, zmíněný v kapitole 2, ve kterém by bylo slunce modelováno jako samostatná sféra. A země by pak kroužila kolem ní a my bychom pak museli stát přímo na povrchu sféry země. Nebo by země na sobě měla průhlednou texturu, přes kterou by bylo vidět a celá výplň okolo obou sfér by měla barvu oblohy. S tímto by pak byla spousta dalších problémů, uvedených již v druhé kapitole, takže použitý model je jednodušší a poskytuje poměrně dobré ztvárnění reality.

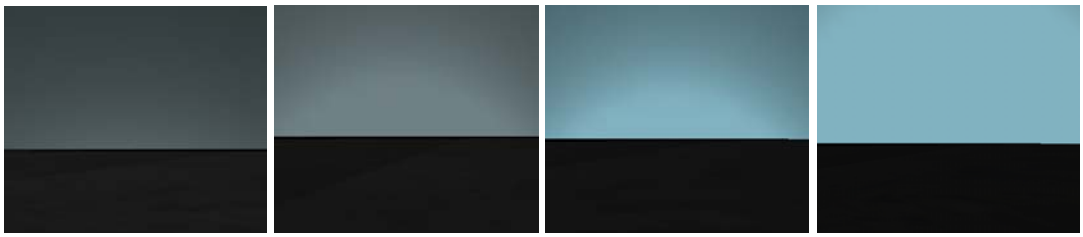
3.4.3 Světlo od slunce

Se sluncem je spojeno bodové světlo, které zajišťuje osvětlení scény ve dne. Jeho pozice je určena z aktuální polohy slunce, takže po celou dobu putuje s ním. Jeho dosah(range) je počítán z poloměru sféry. Dosah slunečního svitu je přesně tak velký, aby slunce od určitého okamžiku – typicky chvíli po východu osvětlovalo celou rovinu podlahy. Pokud slunce zajde za obzor při západu, je ještě po určitou dobu světlo ponecháno, což navozuje realistický efekt. Ten je podobný skutečnosti, kdy po západu zůstává místo, kde slunce zapadlo, po nějakou dobu světlejší než zbytek oblohy. Pak je dosah snížen natolik, aby v noci (kdy je slunce na polokouli pod námi) neosvětlovalo scénu. Jakmile se ráno blíží k východu slunce, je děj reprodukován opačně. Hodnota dosahu se zvedne tak, aby některé místo na obloze při horizontu zesvětlalo, čím signalizuje, že v tomto místě zanedlouho vyjde slunce.

V kapitole 3.2.3 bylo řečeno, že na sféru bylo použito několik textur, které se střídají v závislosti na denním čase. Pro denní oblohu je volena světle modrá, která při stmívání přechází do světle šedé a později tmavošedé. Poslední je typická černá noční obloha. Samotný přechod mezi texturami by sám o sobě asi těžko připomínal reálný výjev, ztmavení oblohy. Šlo by pouze o čtyři textury, které by se mezi sebou po určitých časových odstupech zvoleně střídali. Dokonce ani zmenšování či zvětšování dosahu světla by moc dobře neposloužilo. Bylo by to totiž na obloze patrné, světlo zmenšované tímto způsobem by mělo příliš ostré přechody a působilo spíše rušivě.

Řešením je úprava parametru pro jas světla. V aplikaci je používán spíš pro jeho útlum. Od východu slunce přes poledne až do doby okolo 16-17 hodiny je

hodnota nastavena tak, aby se na výsledku vůbec neprojevoval. Zde má mít totiž světlo takovou intenzitu, aby naplno osvětlovalo celou scénu. Po chvíli začne působit útlum a místa nejvzdálenější od slunce přestanou mít výrazně světlou barvu a začínají tmavnout. Ztmavení se šíří po celé sféře směrem ke slunci. Souběžně s tmavnutím oblohy a skláněním slunce k obzoru vystupuje i pruh v barvách pro západ. Jak stmívání pokračuje, slunce se blíží k horizontu a s ním ustupuje i proužek. Nakonec je slunce obklopeno pouze malým kruhem světla, kam dosáhne jeho intenzita, když se zcela zanoří za obzor část tohoto kruhu, po něm ještě ve scéně zůstane. Postupně však mizí a dojde k výměně textury za tmavší a tmavší. Útlum pro světlo se ještě jednou navýší resp. se sníží intenzita světla a naposled se vymění textura, pak už je naprostá tma. Při východu se situace opakuje, jen jsou textury z tmavých měněny za světlejší a intenzita roste. Stejně jako při západu se nejprve objevuje kruh či elipsa okolo slunce, kam až dosáhne. Ta se postupně rozšiřuje po celé sféře. S tím pak začne i vycházet slunce. Útlum světla je vždy počítán z aktuálního času a rovněž polohy(výšky) slunce. Čas určuje rozmezí, do nějž velikost útlumu spadá. Existují časová pásma a přechody mezi nimi. Přechody mají charakter skoků, jsou mezi sebou posunuty o velikost konstanty, jíž se útlum násobí. To může způsobit, že se tma v některých okamžicích, přesněji na přechodech mezi pásmy prohlubuje rychleji. Naopak poloha slunce přesněji určuje velikost útlumu v rámci intervalů, zajištěných časovými pásmy. Díky tomu je útlum v rámci jednotlivých časových pásem pozvolný. Situace před východem slunce, demonstrující funkci útlumu je zachycena na obr. Nebo v příloze B4 a to pro oba barevné prostory.



Obr. 3.4.4: Situace ve scéně před východem slunce

Tento model má velkou výhodu v tom, že lze snadno použít nejen při západu či východu slunce, ale i při prudké změně počasí, typické pro bouřku a následné vylepšení počasí. Alternativou k tomuto postupu může být například multitexturing,

kdy místo změn intenzit světla je pozvolná samotná výměna textur. V takovém případě by postačily textury dvě. Během dne světlá by s přibývajícím časem přecházela do druhé textury určené pro noc.

3.4.4 Barva slunce

Jak již bylo řečeno, slunce mění v průběhu dne svoji velikost. Ani barva slunce však nezůstává stále stejná, ale mění se dynamicky v průběhu dne. V aplikaci jsou pro nastavení barvy slunce použity dvě zcela odlišné techniky. O obojí se stará metoda `BarvySlunce`, která podle výšky slunce mu přiřazuje barvu jakou má mít.

První metodou je přechod barev modelovaný v barevném prostoru *RGB.* Slunce zde vychází zářivě oranžové. Postupem času je zelená složka světla navyšována, dokud nedosáhne hodnoty, která odpovídá žluté barvě. Jak bylo řečeno, čím více barev sečteme, tím světlejší výsledek. Poté tedy směrem k poledni se přičítají hodnoty k modré složce. Do té doby, dokud se nedosáhne takřka bíle barvy. Směrem od poledne se barva slunce mění přesně naopak. Takže slunce zapadá, v téže barvě jaké vyšlo. Odstíny barev, v nichž slunce vychází nebo zapadá, lze generovat náhodně, jde pouze o vymezení hranic, kterých může barva světla nabývat.

Druhým barevným prostorem je *HSV,* na jehož základech je založena druhá metoda. Zde slunce vychází v oranžovočervené barvě. Z této barvy jsou vzaty jednotlivé HSV složky a upraveny tak, aby nejprve barva přecházela z oranžovočervené do žluté. Po žluté pak přechází takřka do bílé a to v období poledne, kdy je slunce nejvýše. Upravené složky jsou předloženy metodě: `HSBToRGB(int Hue, float Saturation, float Brightness)`, která převede jednotlivé složky zpět do RGB systému. Převod mezi prostory HSV/HLS do prostoru RGB má charakter algoritmu. Výsledná RGB barva je přiřazena materiálu, který je určen pro slunce.

Postup přes HSV podával o něco lepší výsledek, než když byly barevné přechody prováděny přímo v systému RGB. Použity jsou však přesto oba. Důvodem je to, že barvy při v RGB jsou o něco výraznější a pestřejší a lépe se hodí například pro letní měsíce. Naopak přechod nad prostorem HSV je používán pro podzimní a zimní měsíce, kdy slunce bývá poněkud vybledlejší.

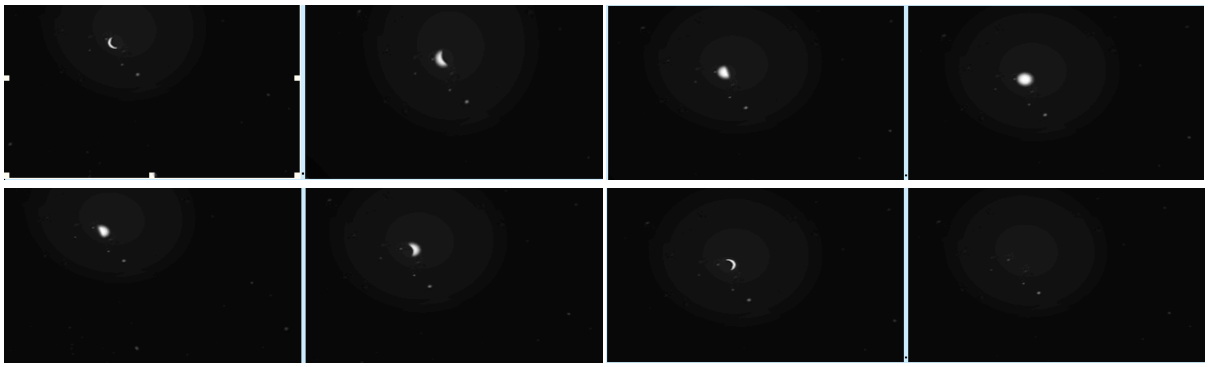
3.4.5 Nezávislost na sféře

Důležité je, že všechny výšše zmíněné atributy pro slunce(jako velikost, změny barev, dosahu světla apod.) nejsou dány jako konstanty, ale jsou přímo přepočítávány z poloměru koule. Takže se nemůže stát, že pokud bude sféra obrovská, bude v ní slunce stejně velké jako, by bylo ve sféře třikrát menší a takřka zanikne. Velikost slunce je vždy určitým poměrem z poloměru sféry. Tedy pro pozorovatele slunce zabírá stále stejně velkou část prostoru, ať už ten stojí ve sféře o poloměru deset či sto. Stejně tak se nestane, aby při pohybu slunce začalo měnit svoji barvu dřív, protože už dosáhlo výšky, ve které by k tomu došlo pří menším poloměru sféry.

3.5 Měsíc

3.5.1 Vytvoření měsíce

Podobným způsobem jako slunce je v aplikaci vytvořen měsíc. Obdobně je na čtvercovou desku nanesena TGA textura (pojem vysvětlen v příloze A - Terminologie). U měsíce na rozdíl od slunce se v průběhu jeho pohybu po obloze nemění rozměry desky, na níž byla textura měsíce aplikována. Sice se nemění velikost desky, ale za to se mění samotná textura, která je použita. To je z toho důvodu, že na rozdíl od slunce, které během dne mění svůj poloměr, ale neustále si zachovává tvar. Měsíc mění podle období svoji fázi, což se projeví na jeho tvaru. Jednotlivé fáze měsíce jsou zajištěny pomocí několika různých textur, které jsou střídány případně překlopeny podle osy. Fáze měsíce obsažené v aplikaci jsou zachyceny na obr.3.5.1 a v příloze B.2



Obr. 3.5.1: Měsíční fáze

3.5.2 Pohyb Měsíce a světlo od něj

Měsíc rotuje okolo osy x a je naklopen pod určitým úhlem podobně jako slunce. Úhel, o nějž se rotuje, musí být i zde počítán z rychlosti rotace sféry, aby měsíc „držel“ rychlost rotace s ostatním objekty. Na rozdíl od slunce se zde nebere v úvahu posunování výchozího bodu měsíce v průběhu roku. Rovněž ani barva měsíce se s průběhem jeho pohybu po obloze nemění. Měsíc si po celou dobu zachovává stříbrnou barvu.

I měsíc vyzařuje určité světlo. Díky tomu taky není v noci naprostá tma. Světlo od měsíce však nemá takovou intenzitu jako denní od slunce. Je však vytvářeno podobně jako světlo pro slunce, když slunce zapadá za horizont a světlo se soustřeďuje jen do oválu kolem něj. Velikost tohoto oválu se mění v průběhu noci, podobně jako u slunce přes den, takže největší světlo vyzařuje, pokud je přibližně v nadhlavníku. Tyto změny však nejsou tolik patrné jako u slunce a mají spíše kosmetický charakter. Díky světlu okolo měsíce je textura v jeho nejbližším okolí poněkud jasnější než zbytek oblohy.

3.5.3 Funkce měsíce

Kromě vizuální stránky má měsíc v aplikaci ještě důležitější roli, jak je řečeno v oddíle 3.7.2**.** Pro vytvoření stínů je vždy nejprve třeba stanovit polohu světelného zdroje. Stín se pak pohybuje a mění tvar v závislosti na poloze zdroje. A protože v době od západu do východu je slunce pod povrchem podlahy, nemůže proto scénu ovlivňovat a stíny by tedy od něj vznikat neměly. Pokud by měsíc v aplikaci nebyl, znamenalo by to, že bude tma a stíny vůbec vznikat nebudou, což není ideální stav.

Proto knihovna SkySun v metodě PolohaIntenzitaSvetla(out Vector3 polohaSvetla, out float intenzita) vrací pozici zdroje světla, který aktuálně může scénu ovlivňovat. V noci je to tedy měsíc a ve dne slunce. Tuto funkci pak využije knihovna pro vytvoření stínů, která si tyto údaje přebere. Musí si totiž světla vytvořit znovu a ve formě v jaké s nimi dokáže pracovat. Společně s polohou světelného zdroje navíc získá z této metody i jeho intenzitu to proto, aby světlo v noci nebylo tak patrné jako ve dne.

3.6 Datum a čas

3.6.1 Čas

Čas vypočítává při každém renderování metoda DenniCas(). Je počítán z úhlu, který svírá slunce s podlahou. Čas se počítá a později vypisuje ve formátu ´ hodiny: minuty´. Rychlost plynutí času je závislá na rychlosti pohybu slunce a rotace sféry. Tato rychlost je samozžejmě měnitelná. Výsledná rychlost rotace sféry se navíc přinásobuje určitým poměrem, který představuje rozdíl dvou po sobě následujících vykreslených snímků. Tím je docíleno toho, že rotace je nazávislá na FPS. V každém renderu se vezme úhel (vypočítaný podle výšše zmíněného stavu), o nějž má být sféra otočená oproti počátečnímu stavu.

Čas jako takový velmi ovlivňuje dění ve scéně. Pokud dostoupí určité hodnoty, objevuje se na horizontu pruh, signalizující východ či západ slunce. Na jeho základě se začíná stmívat či se rozednívá apod. S tím souvisí nepříjemný problém aplikace a to, že určité sekvence akcí jsou naskriptovány k určitým časům. Což je problémem, pokud chceme uvážit, že slunce opravdu každý den nevychází ani nezapadá ve stejnou hodinu. S ohledem na délku dne a noci v různých ročních dobách by mělo slunce v zimě vycházet, co nejpozději a co nejdříve zapadat. Toho by se dalo dosáhnout pomocí konstant, které by se přičítaly k času v každou chvíli dne. Hodnota těchto konstant by závisela na aktuálním dni a měsíci.

3.6.2 Datum

S denním časem velmi úzce souvisí datum. V aplikaci je datum používán především pro rozlišení různých období roku, kdy slunce například v letních měsících vychází na severovýchodě, zapadá na severozápadě a jeho dráha přes oblohu je nejdelší. Naopak v zimě vychází na jihovýchodě a zapadá na jihozápadě a urazí nejmenší trasu. Rovněž barva slunce a pruhu, při jeho východu a západu, nabývá poněkud jiných odstínů v létě na rozdíl od zimy.

Datum při spuštění aplikace začne na nějaké hodnotě. Vždy, když čas dosáhne určité hodnoty(0:00), zvedne se počítadlo prošlých dnů. To je pak přičteno k hodnotě datumu, na níž se začínalo. Program respektuje střídání jednotlivých měsíců i počet dnů, které mají. Stejně jako lze ovlivňovat rychlost běhu času, lze se přesouvat i mezi jednotlivými měsíci, aniž bychom museli čekat, až doopravdy uplyne doba, jež je dělí. Fáze měsíce jsou ovlivněny datumem. Vždy po stanoveném počtu dní se textura vymění, tím je střídání fází zajištěno. Přesouvat se mezi jednotlivými dny je možné v klientské aplikaci pomocí klávesových zkratek - více v uživatelské dokumentaci – příloha D.

3.7 Stíny

3.7.1 Metoda pro generování stínů

Zásadní otázkou pro použití stínů je volba metody pomocí, které budeme stíny zobrazovat. Metoda stínových těles vytváří geometricky přesné stíny a to včetně stínů vlastních. Zároveň umožňuje vrhat stíny pouze u objektů, u kterých si to sami vybereme. Metoda pracuje bezchybně jak s bodovými tak i rovnoběžnými světelnými zdroji. Na rozdíl právě od stínové paměti hloubky není pohledově závislá, takže ji lze snadno aplikovat i pro všesměrové světelné zdroje. Stíny v aplikaci jsou však vytvářeny technikou stínové mapy. Hlavním důvodem, proč je použita tato technika a ne technika stínových těles, je její rychlost a snadnější implementace. Velké časové nároky spojené s vytvářením stínových těles nejsou zdaleka jediné mínus hovořící proti metodě stínových těles. Jde například i o požadavky na reprezentaci scény. Metoda stínových těles se navíc špatně kombinuje s dalšími grafickými efekty.

3.7.2 Vytvoření stínů

V aplikaci je vykreslování stínů zajištěno v samostatné knihovně `shadows`. Tato knihovna lze připojit k takřka libovolné aplikaci, u které bude následně vykreslovat stíny objektů v ní. Lze ji použít i samostatně a pouze do ní vložit objekty, pro něž chceme stíny vykreslovat. Knihovna využívá napsaný shader model [riemer07].

Soubor `shader.fx` je načten do knihovny a ta jej využívá. Nehledě nato, jakým způsobem je daná knihovna využívána, je vždy nejprve třeba stanovit polohu světelného zdroje, který osvětluje scénu a vytváří stíny od daných objektů. Poloha světelného zdroje se samozřejmě může v průběhu aplikace měnit. Což je kupříkladu, pokud jde o slunce produkované knihovnou `SkySun`, přirozený požadavek. Díky tomu se v tomto případě může stín pohybovat v závislosti na tom, jak slunce putuje po obloze. Navíc pokud je poledne, stíny jsou takřka nepatrné jen kolem objektu. Naopak, pokud slunce zapadá či vychází, stíny se protahují. Tvar i velikost stínů je tedy závislá na aktuální poloze světelného zdroje. Protože je v knihovně použit *shader,* není možné pracovat se světly, tak jak je tomu běžné v aplikacích bez něj. Takováto světla by nevykazovala žádnou funkčnost. Světlo musí být vždy vytvořeno v *shaderu* samotném. Výhodou toho je, že v aplikaci, kde bude knihovna použita stačí definovat vždy pouze polohu světla a jeho intenzitu a použít odpovídající metodu z knihovny. Není tedy nutné takovéto světlo doopravdy vytvářet. Tato vlastnost je důležitá, pokud by pro sénu nestačilo jedno bodové světlo, ale byl kupříkladu vyžadován plošný zdroj světla, složený ze spousty dílčích bodových světel. V aplikacích nevyužívajících *shaderu* je počet světel vždy omezen. Naopak pokud je použit, je možné využívat libovolný počet světel, díky nimž lze vytvořit i komplexní zdroj světla.

Poloha světelného zdroje lze nastavit vně samotné knihovny nebo jí nechat nastavit implicitně danou knihovnu. Dalším krokem je vygenerování hloubkové mapy. O to se již stará knihovna samotná, tato funkce není z vnějšku třídy viditelná. Náhled na stínové mapy viz obr. B.3a a B.3b**.** Hloubková mapa tedy odpovídá scéně vygenerované z pohledu od světelného zdroje, přesněji řečeno kamera je posunuta do pozice odpovídající světelnému zdroji. Z tohoto bodu pohledu jediné co je v tuto

chvíli zajímavé, je vzdálenost každého pixelu scény k světelnému zdroji. Tato vzdálenost je pro každý bod ve scéně uložena.

Během další fáze generování stínů bude celá scéna vykreslena z pohledu pozorovatele se skutečnými barvami. Opět pro každý pixel scény je spočtena vzdálenost k světelnému zdroji. Tato vzdálenost je pak následně porovnávána se vzdáleností, která byla v minulém kroku uložena do hloubkové mapy. Celá pointa spočívá v tom, že pro většinu objektů ve scéně jsou tyto vzdálenosti totožné. Tyto body scény tedy neleží ve stínu, ale pro objekty například pro podlahu za objektem, který je do scény umístěn, je takto spočtená vzdálenost větší než ta, která byla uložena do hloubkové mapy. To tedy znamená, že toto místo – tento pixel nemůže být z pohledu od světla viditelný čili leží ve stínu.

Celá scéna, která je vykreslována metodou `Drawscene`, je tedy vykreslována dvakrát. Jednou pro vytvoření hloubkové mapy – zajišťuje metoda `GenerateShadowMap` a podruhé pak při vykreslení scény samotné včetně stínů `RenderShadowedScene`. Do metody `Drawscene` je tedy nezbytné umístit všechny objekty, které mají vrhat stíny. Stejně tak se do ní musí umístit i objekty, na něž mají být stíny vrhány např. podlaha. Samozřejmě, že stíny vrhají i tělesa jedno na druhé, pokud jsou vůči sobě v zákrytu. Naopak všechny objekty u nichž stíny nemají být uvažovány, jsou vykresleny vně této metody. To zajistí, že pokud jsou například obě knihovny použity současně, nevrhají objekty umístěné ve scéně stíny na oblohu, kterou vykresluje druhá knihovna zvlášť.

3.7.3 Alias na hranici stínů

Metoda stínové paměti hloubky aplikovaná v programu, navzdory řadě svých výhod (rychlost, jednoduchost implementace, práce nad libovolnou – nejen ploškovou reprezentací scény), trpí jedním závažným nedostatkem - na hranicích stínů se projevuje výrazný alias, který je popsaný v kapitole 2.3.2**.** Alias byl tím patrnější, čím menší bylo nastaveno rozlišení použité hloubkové mapy. To je velmi patrné především v situacích, kdy kamera detailně zabírá stín vrhaný objektem, který se nachází ve větší vzdálenosti od světelného zdroje. Čím větší je totiž vzdálenost objektu, pro nějž chceme stíny určovat od světelného zdroje, který stín objektu vytváří, tím menší počet pixelů bude zaujímat jeho „obraz“ ve vytvořené hloubkové mapě. Stíny od těchto objektů mají proto velmi zkreslený tvar i přesto, že rozlišení

hloubkové mapy bylo v případě ostatních objektů dostatečné. Nepřesnosti na hranicích stínů opravdu poněkud kalí vizuální dojem z vytvořených stínů. Snahou tedy bylo alias vzniklý na hranicích stínů eliminovat. Jak již bylo řečeno alias na hranicích stínů je způsoben nedostatečným rozlišením vytvořené hloubkové mapy. Přirozeným řešením na jeho odstranění bylo proto zvýšení rozlišení hloubkové mapy. Použití mapy s velkým rozlišením však ještě není zdaleka zárukou, že alias bude snížen pod přijatelnou (vizuální) hodnotu. Ani zvýšení rozlišení nedokázalo v aplikaci zcela alias odstranit.

Řešení, jak potlačit alias na hranici stínů, aniž by bylo nutné zvětšovat rozlišení hloubkové mapy nebo provádět filtraci, spočívají v její vhodné parametrizaci. To umožňuje již v kapitole 2.3.2 zmiňovaná metoda perspektivních stínových map. Hlavní rozdíl oproti původní metodě generování stínových map je v tom, že hloubková mapa je vytvářena až v normalizovaném souřadném systému, tedy až po perspektivním promítání. Díky tomu se dá zajistit, že objekty, které jsou umístěné ve větší blízkosti kamery jsou zobrazovány větší, než objekty od kamery vzdálenější. Rovněž stíny v blízkosti kamery, u kterých je alias nejpatrnější a které by měly být zobrazeny co možná nejpřesněji, vrhají objekty taktéž blízké kameře. Proto, pokud v aplikaci provedeme perspektivní promítání ještě před vytvořením hloubkové mapy, docílíme toho, že ke kameře bližší objekty budou do hloubkové mapy uloženy ve větším rozlišení než objekty vzdálenější. Tím pádem objekty od kamery vzdálenější nemají tak dokonalé stíny, ale to lze zanedbat. Problémem je, že vytvoření hloubkové mapy v normalizovaném prostoru není až tak jednoduché, jak se může zdát. S tím vyvstala spousta problematických situacích, kdy se např. světelný zdroj nachází až za kamerou. Ještě horší pak je, pokud se (společně se světlem) nacházejí za kamerou i objekty, které vrhají do viditelné části scény stín. Pro ně platí, že mají být samozřejmě zahrnuty do hloubkové mapy, avšak perspektivní transformace tyto objekty „ořízne“. Takových komplikací je víc, třeba se v daném směru nachází více objektů, jejich pořadí se v normalizovaném prostoru změní, neboť objekty za kamerou budou vždy zobrazeny dále než objekty před kamerou. Řešení tohoto problému pomocí speciální projekční transformace pracující pouze v normalizovaném souřadném systému lze nalézt v [Kozl04]. Z těchto důvodů nebylo toto vylepšení do aplikace zahrnuto, což sice má za následek, že stíny nejsou tak dokonalé, jak by mohly být, ale přesto jsou celkem věrohodné a zvýšení rozlišení hloubkové mapy je ještě mírně vylepší. K deformaci stínu na jejich hranicích

samozřejmě nebude docházet ve všech scénách, jde o to jak bude daná scéna formulována. Problémy budou působit hlavně venkovní scény s všesměrovými světelnými zdroji viz. 2.3.2.

3.8 Stereo výstup

Program je doplněn podporou pro stereo stěnu (dual-headový výstup). V tom případě budou místo jednoho standardního renderovacího formuláře vytvářeny dva. Jeden bude posunut o určitou konstantu vlevo oproti nedualheadovému výstupu a druhý je o stejnou vzdálenost posunut vpravo. Tím je vytvořen dojem levého a pravého oka. Program musí být samozřejmě možné spouštět i standardním způsobem – pouze na jediný formulář. Dotaz na typ výstupu je první věc, která se objeví hned po spuštění.

3.9 Nezávislost na fps

Veškeré animace a pohyb pozorovatele jsou závislé na reálném čase, nikoliv na snímkovém kmitočtu, tj. změna doby nutné na zobrazení jednoho snímku nezpůsobuje změnu rychlosti pohybu či animace v zobrazovaném světě. Toho je dosaženo tak, že v renderu se vždy počítá rozdíl dvou po sobě následujících vykreslení. Rychlost pohybu pozorovatele, jakož i rychlost otáčení sféry, slunce a rychlost pohybu dalších objektů se tímto rozdílem vždy vynásobí.

4. Závěr

4.1 O programu

Celá aplikace není myšlenkově příliš náročná, obtížnosti se skrývají v implementaci jednotlivých součástí. Největším problémem pak je sladění jednotlivých dílčích částí dohromady s ostatními a jejich účinná součinnost. Vytvořeny byly dvě navzájem na sobě nezávislé knihovny.

V knihovně SkySun je vytvořena sféra, představující Zemi. Na sféru se nanáší několik textur podle denního času. Ve sféře se pozorovatel může volně pohybovat a rozhlížet. Sféra rotuje kolem osy y, rychlost rotace si může uživatel nastavovat na ovládacím panelu. S rychlostí rotace sféry je spojena i rychlost obíhání slunce a měsíce po obloze. Slunce je vytvořeno pomocí poloprůhledné textury nalepené na čtvercovou desku. Rotuje těsně před povrchem sféry okolo osy z. Barva i velikost slunce se dynamicky mění s denním časem. Měsíc rotuje okolo osy x a textura nanesená na desku, představující měsíc se mění podle období v kalendářním měsíci, tím se simuluje střídání měsíčních fází. Mezi sférou a sluncem se pohybuje válec, na němž je nanesena textura simulující východ či západ slunce. Tento pruh se samozřejmě objevuje pouze při rozednívání či stmívání. Při zapadání slunce dochází rovněž k útlumu světla spojeného se sluncem (při východu je to naopak).

Se sluncem, měsícem a vůbec osvětlením scény souvisí stíny o něž, bylo třeba aplikaci doplnit neboť implicitně v directx nejsou. Stíny jsou vytvářeny pomocí knihovny shadows. Ta pracuje na principu metody stínových map. Zde je nejprve generována hloubková mapa s uloženými hodnotami, představujícími vzdálenost daného pixelu od světla. V druhém kroku je scéna vykreslena z pohledu od kamery. Hodnoty jsou spočteny znovu a porovnány s hodnotami, uloženými v hloubkové mapě. Z toho je určeno, jestli je daný pixel ve stínu či nikoli.

Navíc je vytvořena klientská aplikace, která v sobě kombinuje obě dvě knihovny. Nejprve jsou pomocí knihovny pro stíny vykresleny objekty, které mají vrhat stíny nebo na něž jsou stíny vrhány. Poté je použita druhá knihovna, která do dané scény vykreslí všechny zbylé objekty, které stíny vrhat nemají nebo je na nich

nechceme. Jde tedy o oblohu, slunce, měsíc a pruh spojený s východem či západem slunce.

4.2 Shrnutí

Jako u všech aplikací i zde v obou knihovnách by se dalo ještě řadu dalších věci vylepšit. U slunce by mohl být vylepšen efekt, kdy při pohledu do slunce bude pozorovatel oslněn jasným světlem, z něj vycházejícím. To by mu podávalo ještě realističtější vzezření. Na noční obloze, by se měli začít postupně objevovat hvězdy a postupně ráno zase mizet. Textura připadající na noc je sice obsahuje, ale ty se tím pádem objeví vždy v jednu chvíli najednou. Stejně tak i zmizí. Rovněž části oblohy, připadající na okolí hvězd, by měli vytvářet dojem velmi světlých bodů. Stíny samotné působí podle očekávání – pohybují se podle světelného zdroje, podle něhož i mění tvar. Jediným problémem s nimi může být v některých situacích vzniklá nepřesnost na jejích hranicích. Panel nastavení, který ale nepatří do žádné z knihoven, nýbrž je obsluhován z klientské aplikace a do patřičné knihovny pouze nastavuje zvolené hodnoty, by mohl projít změnami grafického vzhledu. Rovněž by mohl obsáhnout nové možnosti, jako například přepínání času na určitou hodnotu. Což by znamenalo, že by se den(slunce či měsíc i obloha) posunula do stavu, určeného daným časem. Rovněž by měl panel být doplněn o některé funkce ovlivňující způsob vykreslování některých částí. Nebo by klientská aplikace obsáhla uživatelskou konzoly do, níž by se psali příkazy, ovlivňující scénu.

Práce splňuje požadovanou funkčnost. V aplikaci se pravidelně střídá den s nocí spolu se všemi věcmi, které k tomu patří. Svit světla, ať už měsíce, slunce nebo uživatelem definovaného zdroje světla způsobuje vznik stínů objektů, které se ve scéně nacházejí. V zadání byly splněny všechny body zadání, některé součásti aplikace jako např. odlesk od slunce či alias stínů by se daly ještě rozpracovat do lepšího výsledku. Naopak v některých aspektech je rámec zadání zjevně překročen, jde zejména o vypracování střídání měsíčních fází v závislosti na aktuálním dni v měsíci. Stejně tak zpracování pruhu na horizontu, jehož přesný odstín stejně tak jako barva u slunce závisí na daném měsíci.

Literatura

[Kasp03] Kasper Fauerby and Carsten Kjær. Real-time Soft Shadows in a Game Engine. Dostupné na: *http://www.peroxide.dk/papers/realtimesoftshadows.pdf, 2003*

[Kozl04] [Kozl04] Simon Kozlov, „Perspective Shadow Maps: Care and Feeding“, GPU Gems, Addison-Wesley Professional, 2004, 6

[Nish85] T. Nishita and E. Nakamae. Continuous Tone Representation of Three-Dimensional ObjectsTaking Account of Shadow and Interreflection. *SIGGRAPH '85 Conference Proceedings,* červenec 1985.

[Riem07] XNA a directx tutoriály, Directx užitím prog. Jazyka c#. Dostupné na: *http://www.riemers.net/chz/tutorials.php*,2007

[Sphe05] Sphere texturing, three part series explaining the addition of textures to DirectX Primitive shapes. Dostupné na: *http://blogs.msdn.com/coding4fun/archive/2006/10/31/912562.aspx*, 2005

[Shad07] David Ambrož Techniky generování stínů v počítačové grafice. Dostupné na: *http://www.shadowstechniques.com/intro_cz.html*, 2007.

[Stam02] M. Stamminger and G. Drettakis. Perspektive shadow maps. *SIGGRAP '02 Conference Proceedings, 2002.*

[Will78] Lee Williams. Casting curved shadow on curved surfaces. *Computer graphics,* 1978

[Woo92] A. Woo. The shadow Depth Map Revisited. *Graphics Gems III, AP Professional*, 338-342, 1992.

Příloha A - Terminologie

Alias

Alias je jedním z nejdůležitějších pojmů počítačové grafiky. Vzniká při rekonstrukci signálu vzorkovaného Nyquistovým limitem a projevuje se jako nová nízkofrekvenční informace, která nebyla v původním signálu přítomna.

V počítačové grafice se setkáváme s aliasem v mnoha situacích, např. při vzorkování textur nebo právě při vytváření stínů technikou hloubkových map. Zde působí na hranách zobrazených stínů ozubení a různě velké nepřesnosti. Příkladem aliasu v praktickém životě je obrazovka počítače či televize, která je natáčena kamerou a následně reprodukována. Zde dochází k interferenci frekvencí s nimiž vysílá obrazovka a s nimiž pracuje kamera a tím následnému aliasu. Ten se projevuje blikáním či pruhy přes obraz.

FPS

Frame per sekond **-** Počet snímků, který se vejde do okénka o velikosti 1s. V aplikaci je snaha, aby veškeré animace a pohyb pozorovatele byly závislé na reálném čase, nikoliv na snímkovém kmitočtu. To znamená, že změna doby nutné na zobrazení jednoho snímku by neměla ovlivnit rychlosti pohybu či animace v dané scéně.

Manifold

Je zavedený pojem, pro takové modely těles, které odpovídají nějakému skutečnému tělesu. Za manifold se z praktického hlediska považuje těleso, pro které platí, že každá jeho hrana inciduje právě se dvěma plochami. Zároveň pro něj platí, že osamocený vrchol nespojuje dvě části tělesa a hrany manifoldu neprotínají jiné plochy. Nonmanifold jak vyplývá z názvu je jeho opakem je to prakticky nevyrobitelné těleso.

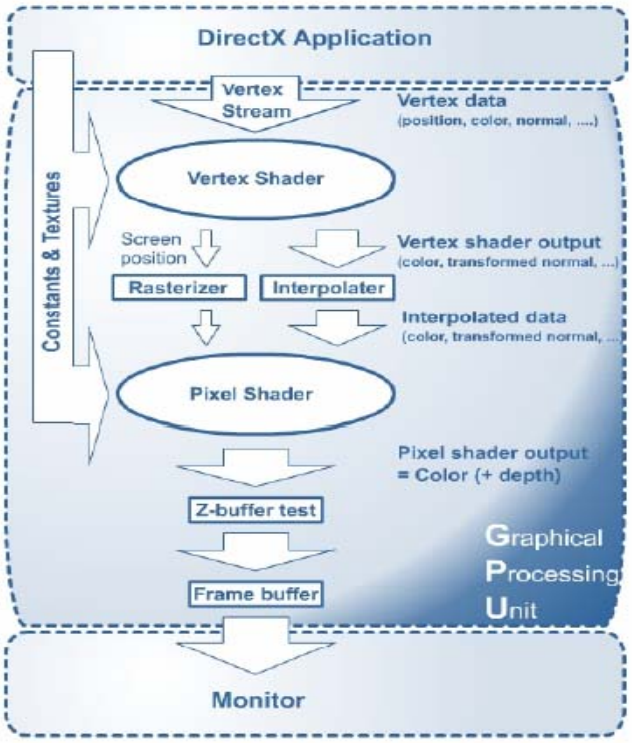
Pixel

Picture element – je jeden obrazový element, jednotlivé obrazové elementy dávají dohromady rastr. V podstatě je to jeden bod obrazu. Tento termín se

v počítačové grafice velmi ujal a rozšířil svoji působnost. Jeden element textury se tak nazývá texel a volume element pro objemy se zkracuje na voxel.

Shader

Napsání vlastního shaderu je prostředek pro zvýšení vizuální kvality finálního obrazu. Jazyk definovaný pro shadery se nazývá HLSL – High Level Shader Language. Použití shaderu vyžaduje manuální nastavování matic pro polohu kamery, souřadnic pro vykreslování I světel. Pořád je však třeba spoustu věcí vytvořit mimo shader a pořád je také třeba doručit do něj potřebné informace z aplikace. Funkčnost shaderu je na obr. A.1.

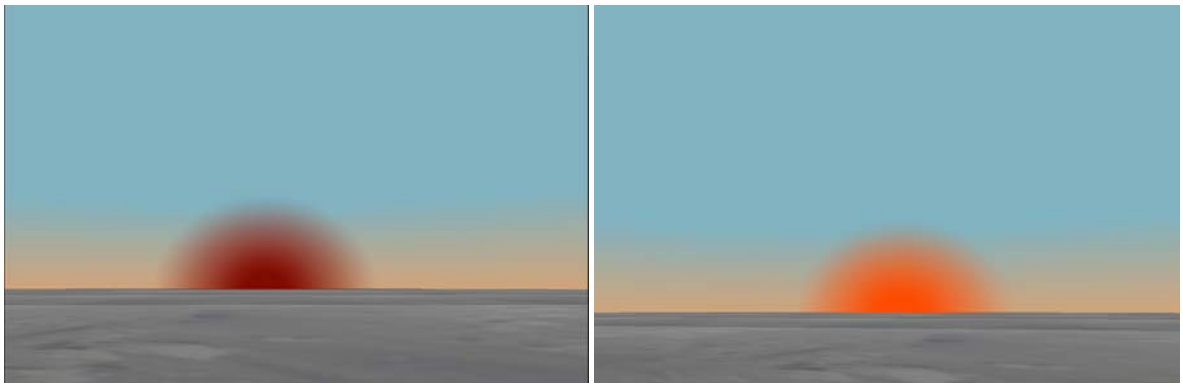


Obr. A.1[Riem07]: Funkčnost shaderu

TGA

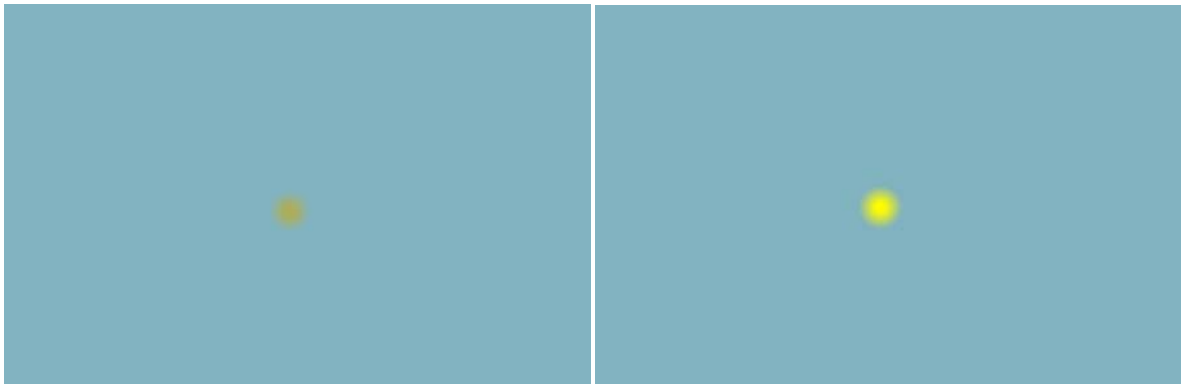
Formát obrázků vyvinutý firmou Truevision. Dokáže uložit obrázky v barevném rozlišení 1, 8, 16, 24 a 32 bitů / pixel. Poslední případ je určen pro barevný model RGBA. Kde krom konstant R, G, B vysvětlených v X.Y. A znamená průhlednost. Tak je možno v tomto formátu vytvořit obrázky s rozdílnou průhledností na různých místech obrázků.

Příloha B - Obrázky

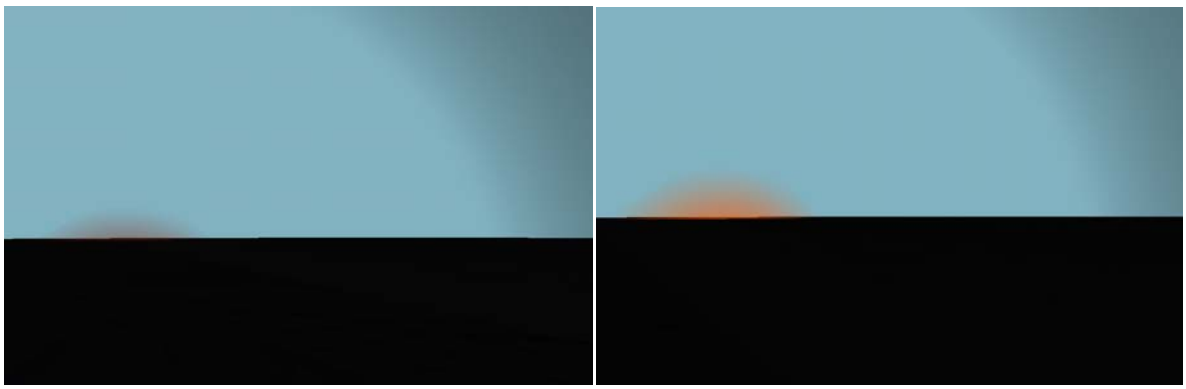


Obrázek B.1a: Východ slunce, vlevo v prostoru HSV vpravo RGB

Obrázek B.1b: Slunce o 2 hodiny později, vlevo v prostoru HSV vpravo RGB

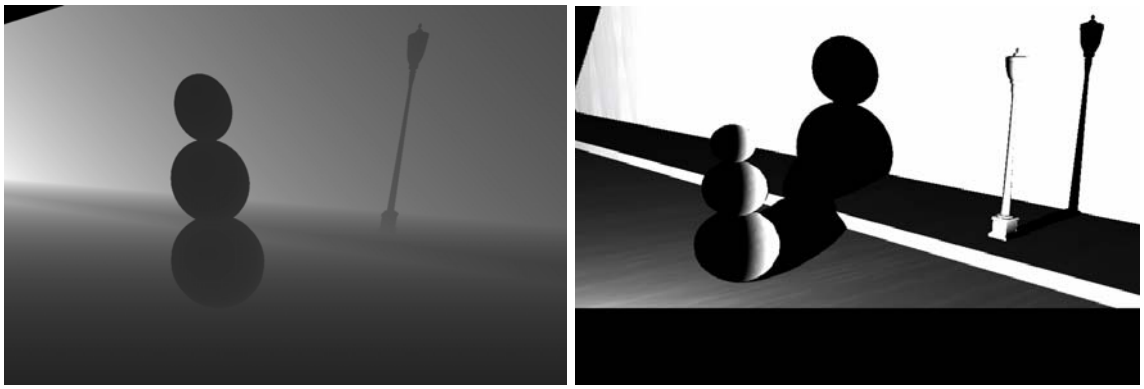


Obrázek B.1c: Slunce o další 2 hodiny poději, vlevo v prostoru HSV vpravo RGB

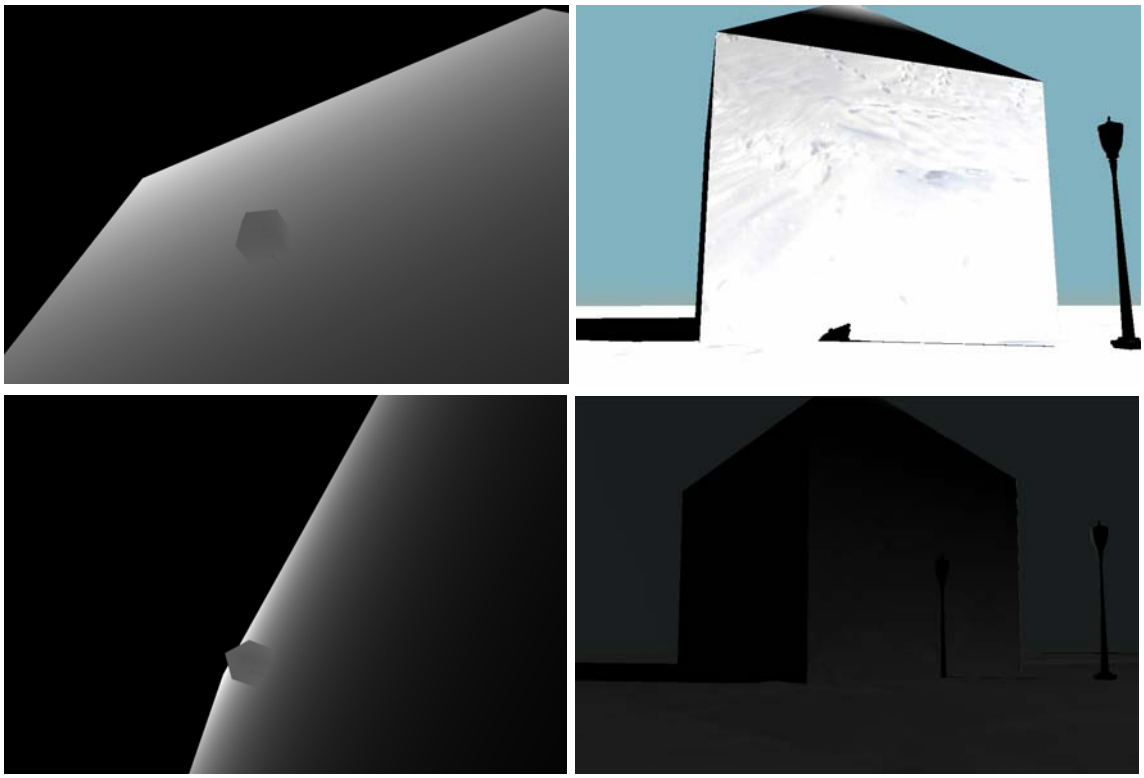


Obrázek B.1d: Slunce při západu, vlevo v prostoru HSV vpravo RGB

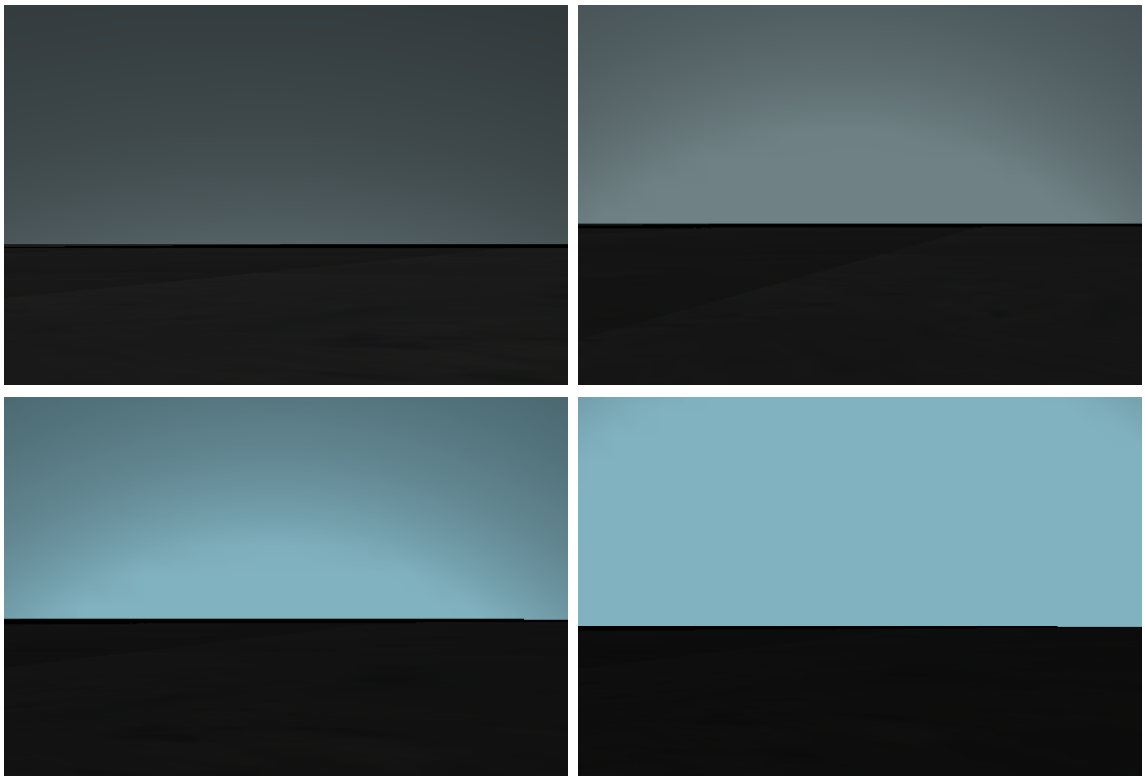
Obrázek B.2: Měsíční fáze



Obr.B.3a: Scéna se dvěma objekty, vlevo náhled na hloubkovou mapu vpravo pak samotná vykreslená scéna



Obr.B.3b: Scéna z knihovny `SkySun`, doplněná o stíny objektů, vlevo náhled na hloubkovou mapu vpravo pak samotná vykreslená scéna. Situace je zachycena v poledne a v podvečer
Na hloubkové mapě, která zachycuje to, co je právě viditelné z pohledu od světla je poznat v jaké poloze, vzhledem ke scéně se světelný zdroj nalézá.



Obr. B4: Scéna před východem slunce

Příloha C - Testy

Všechny testy byly prováděny na dvou rúzných sestavách:

1.) AMD Athlon 2000XP, 1GB RAM, grafická karta NVIDIA GeForce 5600FX. – 128MBRAM

2.) AMD Athlon 3600+, 1GB RAM, grafická karta ATI Radeon X1600 PRO – 256 MBRAM

První část testů - C.1 se vztahuje pouze ke knihovně `SkySun`. Je zde demonstrováno jak rostoucí počet trojúhelníků, z nichž jsou použité objekty (sféra, slunce, horizont, měsíc) složeny, snižuje FPS. Trojúhelníky, z nichž je sféra, měsíc i slunce složena jsou vykreslovány jako `triangle strip`. Horizont, lépe řečeno plocha, na níž je nanášena patřičná textura pro modelář horizontu (kapitola 3.3), je složena z trojúhelníků vykreslovaných jako `triangle list`.

Knihovna zde byla použita samostatně z klientské aplikace – ta prováděla pouze potřebnou inicializaci a vykreslení samotné scény z knihovny. Vykreslována byla nejen samotná sféra, ale kompletní obsah dané knihovny (obloha, slunce, měsíc, horizont a další). Důležité však je, že do testu nebylo zahrnuto vykreslení podlahy pro tuto scénu. Knihovna tuto možnost sice nabízí, avšak při použití většího poloměru sféry snímkový kmitočet klesá nejvíce díky vykreslené podlaze. Sféra je totiž při všech nastaveních poloměru složena ze stejného počtu trojúhelníků. Počet trojúhelníků, z nichž je sféra složena se rovná dvojnásobku součinu počtů horizontálních a vertikálních dílků. Podlaha se přirozeně roztahuje s poloměrem sféry a to tak, že na každé políčko o velikosti 1x1 připadnou dva trojúhelníky. Což má za následek, že čím větší je poloměr, tím více trojúhelníku potřebujeme – neboť je více plošek, potřebných na poskládání podlahy. Do testu podlaha zahrnuta není, protože není předmětem aplikace a především pokud uživateli její vyobrazení nevyhovuje má možnost si vytvořit vlastní a přidat k jí ke zbytku scény.

Program zapisoval hodnotu FPS ve vybraných hodinách (ráno od 6 do 12 a večer od 18 do půlnoci), při pohybu a rozhlížení pozorovatele. Tyto časy byly vybrány proto, aby postihly všechny situace, které knihovna poskytuje (svítání, západ, poledne a noc) Pohyb a rozhlížení bylo třeba provádět, aby byl vidět dopad tohoto na

snímkový kmitočet. Pro každou hodnotu času bylo zaznamenáno 10 hodnot, aby se u nich neprojevovaly výkyvy, způsobené pohybem. Pro každé nastavení rozlišení počtu vertikálních a horizontálních dílků bylo tak vzato 12 hodnot FPS v průběhu dne. Ty byly vždy zprůměrňovány a určily tak výslednou hodnotu pro dané nastavení. Rozlišení byly brány od 20x10 (vertikální x horizontální) po 500x250. Krok byl většinou roven dvojnásobku předešlé hodnoty.

Jak ukazuje graf, hodnoty FPS logicky klesají se zvýšením daného počtu, ne však zcela lineárně. Je nutno si uvědomit, že k danému počtu trojúhelníků je třeba ještě přičíst další aspekty vyskytující se ve scéně např. při východu slunce či při stmívání je na horizontu pruh s texturou. Tento pruh je rovněž složen z určitého počtu trojúhelníků a to konkrétně 2xnastavená hodnota, pro počet jeho stran. Tento počet stran ovlivňuje hranatost tohoto útvaru. U tohoto parametru stačí hodnota nastavená okolo 50. Tato hodnota již „zakulacení“ ´tvaru do podoby válce. Z toho vyplývá, že FPS není tímto údajem příliš zatížena, protože při počtu 50 stran je výsledkem 100 trojúhelníků, což je zanedbatelná hodnota vzhledem k počtu trojúhelníků, nichž je vytvořena sféra.

Výsledkem testu by mělo být určení optimálního nastavení parametrů pro sféru. S hodnotami pro počet horizontálních a vertikálních částí určujících sféru jako i s poloměrem sféry je možno libovolně hýbat mimo samotnou knihovnu. Způsob je uveden v příloze D – uživatelská dokumentace. Pro počet dílků na sféru je optimální nastavení okolo 50x50 do 200x200. Snížení počtu pod tuto mez má za následek zhranatění sféry. Naopak jejich zvýšení o moc lepší výsledek nepřinese, naopak klesne FPS.. S poloměrem je to problematičtější. S rostoucím poloměrem roste i velikost všech objektů ve scéně. Tento fakt nemusí působit problémy, pokud se nepoužije podlaha, použitá v samotné knihovně. Pokud si uživatel bude přát vytvořit podlahu vlastní půjde v první řadě o to, jakou scénu je třeba vytvořit. Pokud stačí menší scéna, kde nepůjde opustit ohraničenou mez uprostřed není problém použít menší poloměr. V tom případě pokud je navíc použito optimální rozlišení horizontálních a vertikálních dílků, pak je garantováno, že samotné vykreslení oblohy s jejími náležitostmi příliš neovlivní FPS. Pokud však má být scéna rozsáhlejší, typicky se spoustou objektů, rozmístěných po ní je toto obtížnější. Při použití podlahy z knihovny, je již při poloměru nad 80 FPS okolo 20 snímků / sec. Tato rychlost zobrazování většinou znamená předěl v tom, co lze považovat za „naprosto plynulé“ a „příležitostně se sekající“. Při poloměru okolo 200 pak FPS klesá na 5 snímků za

sec, což obecně představuje spodní mez únosnosti. Pokud je použita vlastní může být tato situace o něco lepší, ale i horší v závislosti na tom, jak je vytvořená podlaha náročná zobrazit.

FPS je samozřejmě parametr čiře závislý na Hardwarovém vybavení, takže pro superrychlý počítač nemusí být situace při takto vysokých poloměrech až tak závažná. Přesto je třeba si uvědomit, že tato knihovna vytváří pouze oblohu slunce a s tím spojené věci, takže musí zbýt ještě nějaký prostor pro vykreslení samotných objektů či dalších grafických prvků. Poloměr by proto neměl překročit hodnotu 100.

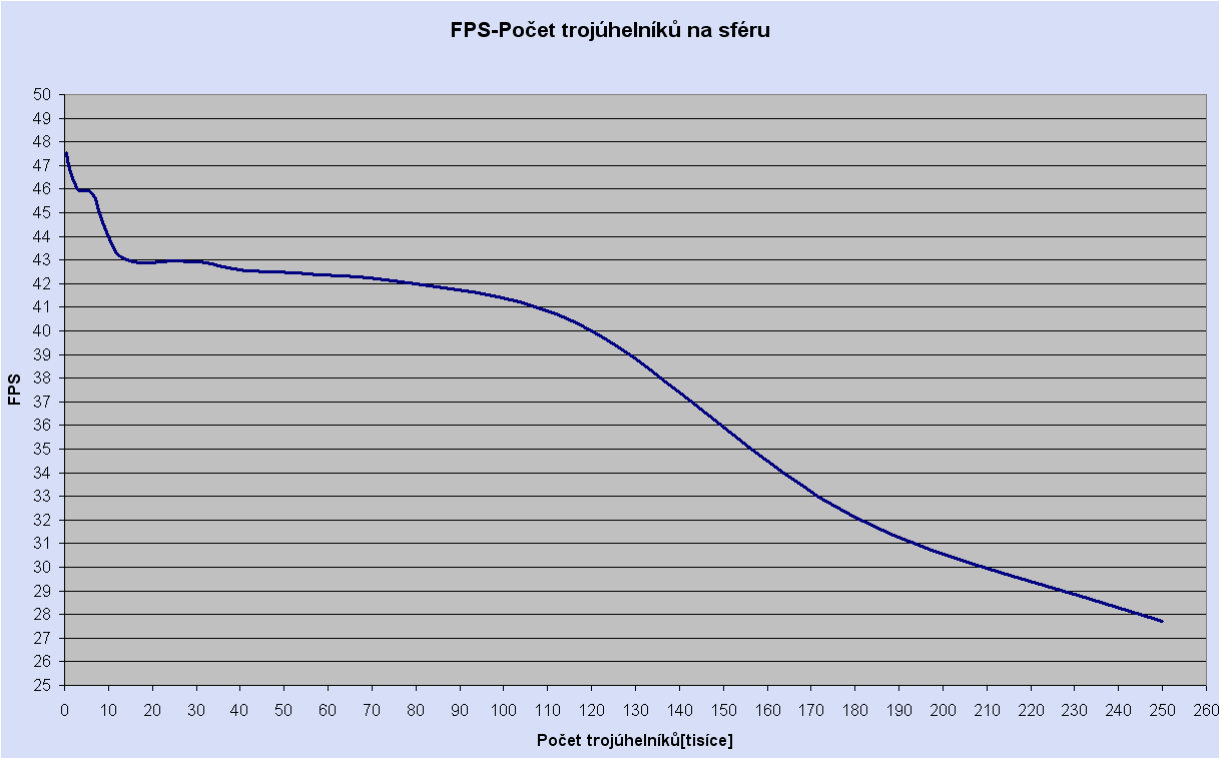
Naměřené hodnoty pro první sestavu

rozlišení:	20x10															průměr
hodiny:		6	7	8	9	10	11	12	18	19	20	21	22	23	24	
FPS:		47,29	44,29	44,89	48,13	47,19	48,83	46,5	47,2	46,15	49,3	49,5	50,23	48,3	47,8	**47,54286**
rozlišení:	20x20															
		46,31	46,13	44,29	44,56	48,23	47,19	48,32	44,2	46,78	48,9	49,7	49,5	48,35	45,6	**47,00429**
rozlišení:	40x20															
		45,28	44,9	44,33	46,56	46,38	48,23	47,96	43,39	49,6	50,13	47,62	47,78	46,24	43,85	**46,58929**
rozlišení:	40x40															
		44,29	43,39	44,16	44,35	47,23	47,16	47,36	42,68	48,26	48,96	48,42	47,92	45,36	43,33	**45,91929**
rozlišení:	80x40															
		44,53	44,26	45,13	44,78	46,98	47,23	46,64	45,31	46,35	47,26	46,08	46,61	45,99	44,13	**45,80571**
rozlišení:	80x80															
		43,13	42,91	43,3	41,89	39,68	42,91	42,92	43,65	44,9	46,15	43,35	44,82	42,26	41,84	**43,12214**
rozlišení:	160x80															
		44,29	43,31	42,02	43,35	41,09	42,61	41,98	42,94	44,25	43,36	44,21	42,23	43,06	42,76	**42,96143**
rozlišení:	180x90															
		44,92	43,36	43,58	43,21	42,29	42,12	42,06	43,13	42,36	41,96	43,36	42,12	42,66	43,33	**42,89**
rozlišení:	200x100															
		44,35	44,28	42,96	43,17	43,36	41,2	39,03	41,68	43,33	43,27	42,65	42,28	41,74	42,98	**42,59143**
rozlišení:	200x200															
		44,16	42,36	44,81	40,28	43,32	42,74	40,69	42,14	42,36	39,01	39,78	40,65	41,5	44,17	**41,99786**
rozlišení:	300x200															
		40,35	40,43	39,52	40,43	40,35	40,43	39,12	39,12	40,43	39,12	40,56	40,43	40,31	39,5	**40,00714**
rozlišení:	300x300															
		32,73	33,65	31,69	31,36	32,68	30,28	32,39	31,14	33,33	32,23	32,29	32,91	30,31	32,96	**32,13929**
rozlišení:	500x250															
		28,76	27,71	28,13	27,64	27,56	27,69	28,13	26,41	27,26	28,09	28,13	27,26	28,24	27,14	**27,725**
	400	**800**	**1600**	**3200**	**6400**	**12800**	**25600**	**32400**	**40000**	**80000**	**120000**	**180000**	**250000**			
	47,54286	**47,00429**	**46,58929**	**45,91929**	**45,80571**	**43,12214**	**42,96143**	**42,89**	**42,59143**	**41,99786**	**40,00714**	**32,13929**	**27,725**			

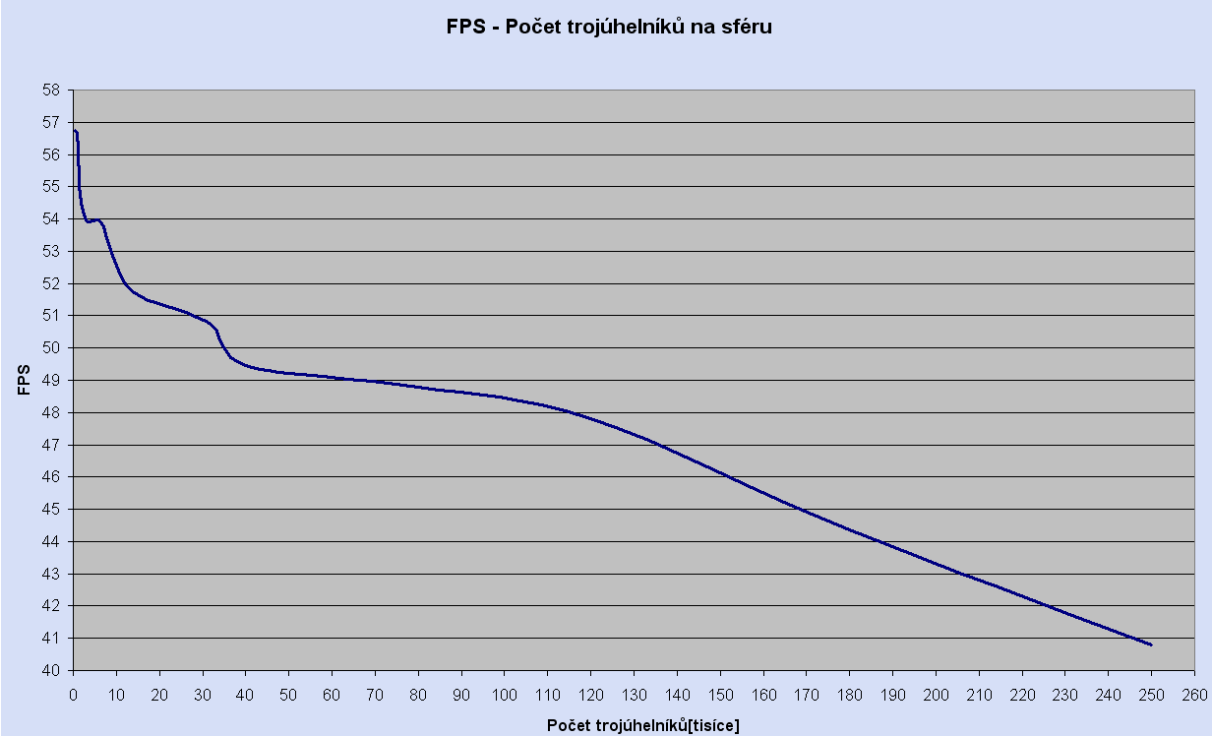
Naměřené hodnoty pro druhou sestavu

400	800	1600	3200	6400	12800	25600	32400	40000	80000	120000	180000	250000
56,74632	56,68463	54,79561	53,92687	53,90012	51,87678	51,13546	50,67891	49,46805	48,78952	47,79892	44,3502	40,79511

Graf pro první sestavu



Graf pro druhou sestavu



Test C.2 testuje knihovnu Shadows. Ověřuje závislost snímkové frekvence na počtu objektů, které jsou umístěny v jedné scéně.

Knihovna zde byla použita podobně jako SkySun v minulém testu. Byla samostatně spuštěna z klientské aplikace – v ní se prováděla pouze potřebná inicializace a vykreslení samotné scény z knihovny. Do scény byly postupně přidávány po určité době nové objekty. Tyto objekty (aby se jich vešlo do scény co nejvíce) byly krychle o délce hrany 2 a skládány byly vedle sebe do několika řad – u poslední hodnoty pak nad sebe. Všechny krychle byly tvořeny pouze ze čtyř stran, takže jsou bez spodní a horní podstavy. Pro vytvoření každé strany stačí dva trojúhelníky, takže každá krychle je tvořena z osmi trojúhelníků. V průběhu testu byly vždy nejprve do textového souboru zapsány čtyři hodnoty z přímého pohledu. Po té byl program nastaven tak, aby provedl několik posunů pozice světla do základních stran. Během těchto posunů jsou zapsány vždy dvě hodnoty. Jedna když se světelný zdroj posouvá tam druhá zpátky. Tyto posuny samozřejmě znamenaly pokles snímkové frekvence. Ten u menšího počtu objektů nebyl výrazný. Pro velký počet objektů však znamenal patrný pokles FPS.

Na grafech je patrný přibližně lineární pokles FPS na počtu objektů. Tento test byl do značné míry simultánní, protože takovéto umístění objektů nemá přílišné uplatnění. Přesto je pokles u tohoto pokusu transparentní. Pokud by byly objekty rozmisťovány po scéně spíše náhodně (což je představa modelující reálné aplikace), pak by k poklesu s podobnými proporcemi došlo také. Z testu však přesto vyplývá, že i při zatížení 50 – 80 objektů, pro něž chceme vrhat stín neklesá pod mez při níž by se obraz trhal a to i v případě náhodného rozmístění objektů. Drobný problém může přinést fakt, že do aplikace je vždy nutné vnést nejen objekty, stín vrhající ale i t, na něž stíny jsou vrhány. Takže pokud chceme stíny na podlaze potřebujeme ji přenést do vykreslení v dané knihovně. Test samozřejmě nepříliš přesně simuluje reálné podmínky pro nasazení aplikace, protože všechny objekty jsou složeny ze stejného (stejně malého) počtu trojúhelníků, což v praxi nenastane.

Srovnáním grafů z obou sestav je rovněž patrné, že pokles FPS v závislosti na počtu objektů (trojúhelníků, z nichž jsou složeny) je velmi podobný. Oba dva grafy vykazují lineární průběh. S lepším Hardware samozřejmě lze dosáhnou vyššího počt snímků.

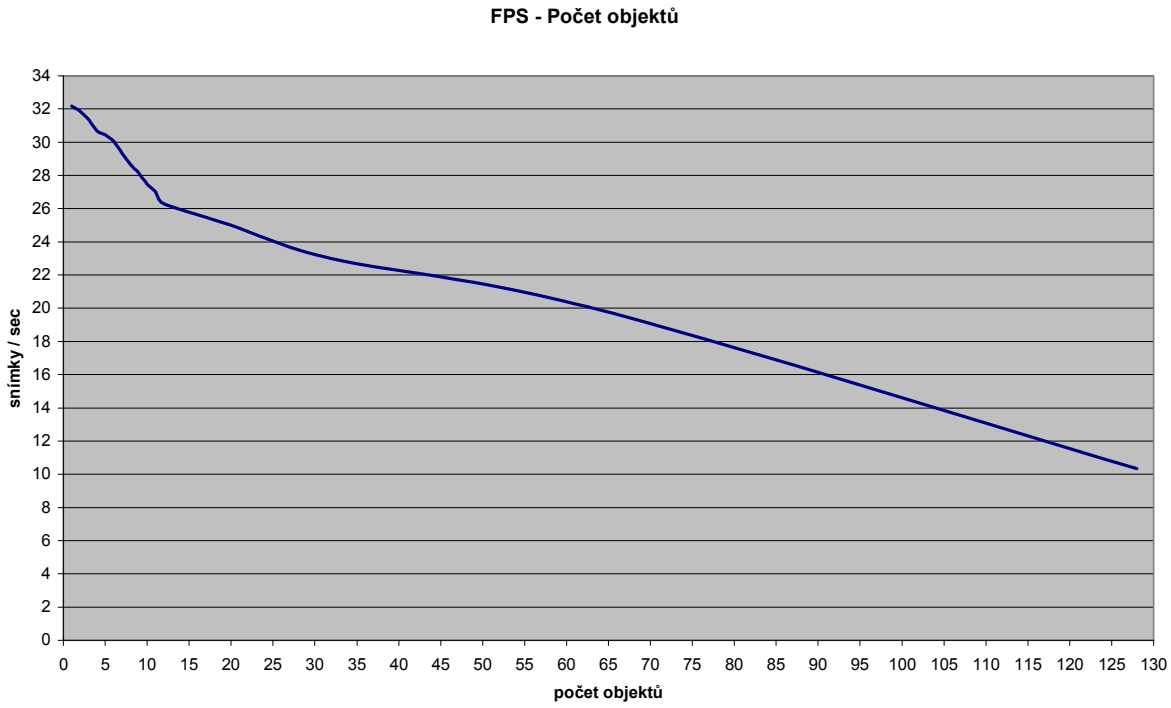
Naměřené hodnoty pro první sestavu

počet objektů	1	2	3	4	5	6	7	8	9	10	11	12	20	32	64	128
počet trojúhelníků na objektech	8	16	24	32	40	48	56	64	72	80	88	96	160	256	512	1024
							FPS									
přímý pohled	32,01	31,5	31,5	30,51	30,07	30,07	29,65	29,1	28,54	27,7	27,16	26,57	25,22	23,28	20,06	12,24
	31,98	31,53	31,04	31,04	30,54	30,51	29,56	28,54	28,56	27,73	27,13	26,6	24,96	22,96	20,37	11,13
	31,52	32,01	31,01	30,54	30,54	30,04	29,1	29,1	28,13	27,79	27,21	26,57	25,21	23,24	20,06	11,96
	32,01	31,53	31,5	30,51	30,51	30,54	30,01	28,54	28,54	28,53	26,68	26,8	25,12	22,96	20,31	11,13
zleva	31,5	31,5	31,04	30,51	30,07	30,04	29,53	28,54	27,56	27,73	27,13	26,2	24,83	22,92	19,56	10,54
	31,96	31,5	31,5	31,01	30,21	29,56	29,1	28,46	27,98	27,14	26,54	25,99	24,76	22,94	19,84	10,16
zprava	32,56	32,51	31,5	31,04	30,54	30,04	29,56	29,1	28,13	27,53	27,13	26,2	25,12	23,12	20,06	10,54
	32,98	32,48	31,98	30,54	31,01	30,07	29,1	28,54	28,56	27,73	27,16	26,57	25,21	23,24	19,86	11,13
zhora	31,5	31,5	31,04	31,01	30,07	30,51	29,51	29,1	28,54	27,14	27,21	26,57	24,96	22,92	20,31	9,98
	32,48	32,01	31,53	30,51	31,01	30,04	29,56	28,54	28,16	27,53	27,13	26,6	24,83	22,86	20,01	10,16
zdola	32,01	31,5	31,56	30,51	30,54	30,04	29,2	28,46	27,56	27,73	26,84	26,2	25,21	22,94	19,56	9,6
	32,51	31,98	31,04	30,41	30,07	30,07	29,21	28,51	28,54	27,21	27,01	25,9	24,96	22,92	19,9	10,21
průměr	32,085	31,79583	31,35333	30,67833	30,43167	30,1275	29,42417	28,71083	28,23333	27,62417	27,0275	26,3975	25,0325	23,025	19,99167	10,73167
	1	2	3	4	5	6	7	8	9	10	11	12	20	32	64	128
	8	16	24	32	40	48	56	64	72	80	88	96	160	256	512	1024
	,17611	31,86398	31,3937	30,69093	30,43907	30,05528	29,35491	28,66231	28,14037	27,48491	27,01972	26,29194	24,99028	22,98722	19,89907	10,33907

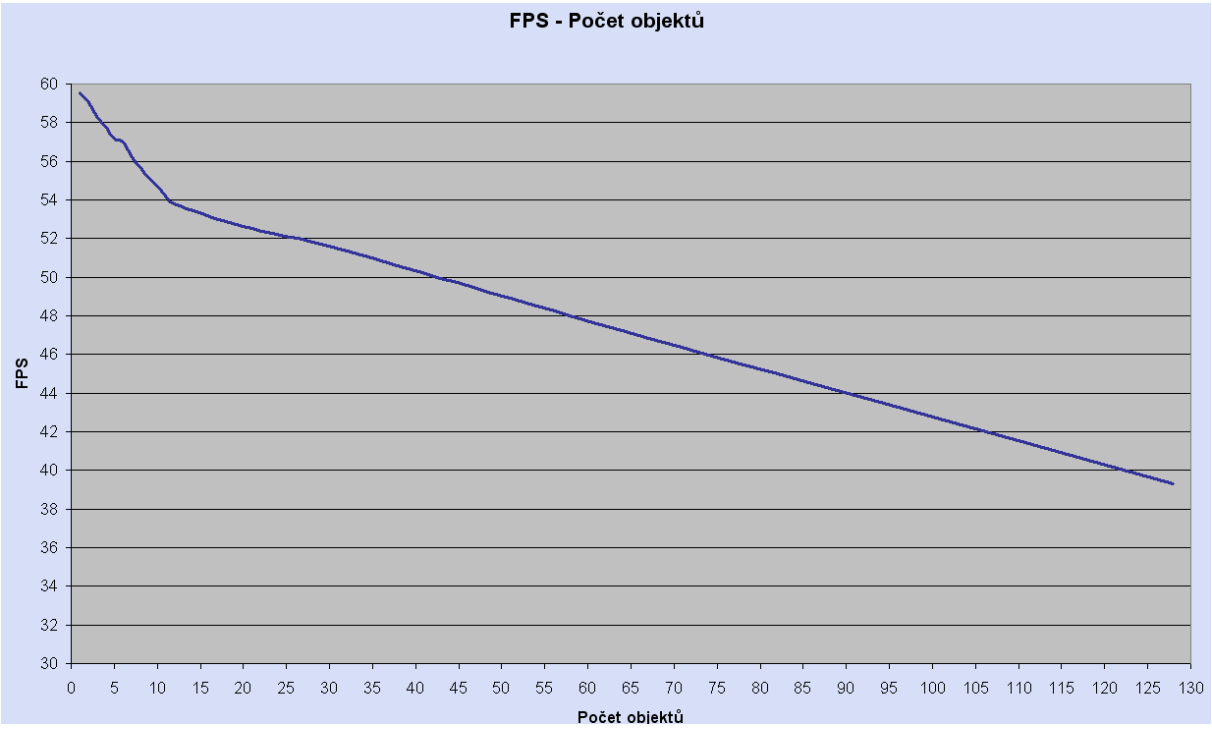
Naměřené hodnoty pro druhou sestavu

1	2	3	4	5	6	7	8	9	10	11	12	20	32	64	128
8	16	24	32	40	48	56	64	72	80	88	96	160	256	512	1024
59,54673	59,03547	58,3257	57,78951	57,13576	56,98775	56,26571	55,75215	55,12135	54,68742	54,21346	53,79984	52,64615	51,3378	47,23461	39,32448

Graf pro první sestavu



Graf pro druhou sestavu



Poslední test C.3 testuje rovněž knihovnu `Shadows`. Využívá částečně předchozího testu této knihovny. Bere jeden počet objektů a pro ten ověřuje závislost snímkové frekvence na počtu světel, které danou scénu ovlivňují.

Pro 80 objektů přidaných do vybrané scény je při každém spuštění aplikace, pracující s knihovnou přidáno jedno světlo. Důvodem, proč je aplikace spuštěna vždy znovu je nutná úprava *shaderu* pro zvolený počet světel.

Z údajů je patrný velmi zřejmý pokles FPS s přibývajícími světly. I při nízkém rozdílu 1 - 6 světel marginálně FPS klesá. Vysvětlením je, že pro každé světlo musí být vytvářena jiná hloubková mapa. Mohlo by se zdát, že počet šesti bodových světel je dostatečný, což je ale mylná představa pokud budeme chtít do aplikace uvažovat nějaký komplexní zdroj světla, složený z více zdrojů bodových. Pokud takových komplexních zdrojů bude víc nastane pro pomalejší stroje, na nichž by aplikace běžela rapidní úbytek FPS, klesající hodně pod únosnou mez.

Naměřené hodnoty

počet objektů	**80**					
počet trojúhelníků	**640**					
počet světel	**1**	**2**	**3**	**4**	**5**	**6**
			FPS			
přímý pohled	*17,43*	*15,36*	*14,86*	*13,45*	*12,06*	*11,31*
	17,68	*15,24*	*14,84*	*13,76*	*11,97*	*11,27*
	17,96	*15,36*	*14,78*	*13,45*	*12,04*	*11,33*
	18,01	*15,24*	*14,89*	*13,45*	*11,97*	*11,27*
zleva	*16,76*	*14,97*	*13,98*	*12,91*	*11,81*	*10,54*
	16,93	*15,98*	*13,96*	*13,02*	*11,79*	*10,95*
zprava	*17,21*	*15,21*	*14,26*	*13,41*	*11,64*	*10,59*
	17,33	*15,16*	*14,54*	*13,22*	*12,01*	*11,18*
zhora	*17,23*	*14,48*	*14,62*	*13,17*	*11,98*	*11,32*
	17,33	*15,21*	*13,92*	*13,21*	*11,84*	*10,26*
zdola	*16,94*	*15,32*	*14,31*	*12,84*	*12,02*	*11,03*
	17,16	*14,98*	*14,78*	*13,09*	*11,86*	*10,98*
průměr	***17,33083***	***15,20917***	***14,47833***	***13,24833***	***11,91583***	***11,0025***

Příloha D – Uživatelská dokumentace

Neboť výsledným produktem jsou dvě nezávislé knihovny je nejprve důležité je k nově vytvářené aplikaci připojit. Připojení knihovny se realizuje takto:

1.) Přidání reference na danou knihovnu
Provede se klepnutím pravého tlačítka na záložku *References* v pravém panelu vývojového prostředí. Po té je nutné nalézt hledaný .dll soubor. Pro knihovnu se stíny je to `Shadows.dll`, pro oblohu `SkySun.dll`. Tyto soubory se nalézají implicitně ve: *složka knihovny// bin// debut*

2.) Je nutné naimportovat daný namespace knihovny. Typicky na počátku vašeho kódu v části věnované importu knihoven přidání:
using + jméno knihovny.

Tím končí analogie mezi knihovnami další práce je poněkud odlišná.
S knihovnou pro oblohu je práce možná dvěma způsoby. V obou je na počátku nutné vytvořit novou instanci od příslušného objektu. To zajistí příkaz `Renderform název = new Renderform()`. Pokud máte aplikaci, která nevytváří vlastní formulář. Pak stačí zavolat funkci z dané knihovny `D3DInit`, která provede kompletní inicializaci. Pokud ji chcete pouze přidat ke grafické aplikaci, v níž již je inicializace vytvořena, pak stačí v samotné inicializaci, kterou již máte vytvořenu zavolat metodu knihovny `LoadTextureAndMaterials`, která iniciuje pouze věci, které užívá(textury, objekty). Podobným způsobem se pracuje s vykreslováním samotným. Lze jej zavolat kompletně včetně chození, panelu nastavení, klávesového ovládání apod. – to zajistí metoda z této knihovny `Render`. Pokud již máte samotný pohyb nebo jinou část aplikace vytvořenu či pouze chcete vykreslit oblohu se sluncem a zbytkem do vaší již existující scény použijete metodu `VykresleniSceny`. Rovněž je možno nastavovat některé důležité parametry, kupř. poloměr sféry. V tom případě je nutné nastavit mu hodnotu po vytvoření instance a před iniciováním komponent. Používat se dají i další metody, které již většinou mají nějaký specifikovaný účel a aplikaci lze obsluhovat bez nich.

Práce s knihovnou pro stíny není o mnoho náročnější. V prvním kroku opět nutné vytvořit instanci od příslušného objektu: `ShadForm název = new ShadForm()`. Pak stačí iniciovat náležitosti, v ní používané. To vyřizuje její metoda: `InitializeDevice`. Zde už však není možné použžít variantu, která incializuje vše sama. To v sobě nese skryté nebezpečí, protože klientská aplikace by měla obsahovat test na zařízení počítače, jímž musí být grafická karta s technologií shader 2.0 a vyšší. Pokud test bude chybět a grafika nebude dostatečná aplikace se zhroutí!. Jakmile je inicializováno nastává opět více možností jak knihovnu využít. To zajišťuje přetížená metoda `OnRun`, která může vykreslovat implicitní scénu, do níž lze umisťovat objekty. Být připojena ke knihovně `SkySun` nebo po několika úpravách k vaší aplikaci. I zde je několik parametrů, vyvedených jako vlastnosti, se kterými je možno nakládat. Jako vlastnosti jsou provedeny i některé proměnné, ovlivňující způsob vykreslování. Pro stíny je např. možné zapnout hloubkovou mapu, pro sféru vypnout čas apod.

K vytvořené klientské aplikaci je přidán Panel nastavení. Ten je samostatným objektem vytvářeným ve třídě `Nastaveni`. Vykresluje se jako rámec s ukazateli. První z nich slouží pro změnu rychlosti běhu času. Ukazatelé kontrolují na zda do nich nebylo kliknuto, pokud ano přepnou se na tuto mez. Zároveň s tím se mění rychlost rotace sféry a slunce. V klientské aplikaci je ovládání nastaveno takto:

W – pohyb dopředu
S – pohyb dozadu
A – vlevo
D – vpravo
Šipky – nastavování pozice světla, pokud je použita implicitní scéna v knihovně `Shadows`.
N – zapne / vypne vykreslení panelu nastavení
H - zapne / vypne vykreslení hloubkové mapy
1(alfanum. klávesnice) – přidání dne pro scénu s oblohou
2(alfanum. klávesnice) –odebrání pro scénu s oblohou
3(alfanum. klávesnice) – přidání měsíce pro scénu s oblohou
4(alfanum. klávesnice) –odebrání měsíce pro scénu s oblohou
0(num. klávesnice) - zapne / vypne vykreslení hud pro scénu s oblohou
Mezerník – zapne / vypne pohyb sféry
Escape – ukončí aplikaci

Min. konfigurace: Procesor 1GHz a vyšší 512 MBRAM, 20 MB volného prostoru na disku. Grafická karta – pro stíny musí být navíc vybaveny *Pixel* a *Vertex Shaderem 2.0* a výš.