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

Bakalářská práce

Rychlé zobrazení husté vegetace

Plzeň, 2008 Jan Plas

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky
Akademický rok: **2007/2008**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jan PLAS**

Studijní program: **B3902 Inženýrská informatika**

Studijní obor: **Informatika**

Název tématu: **Rychlé zobrazení husté vegetace**

Zásady pro vypracování:

1. Definujte pole včetně způsobu jeho zadávání a specifikujte datový formát.
2. Definujte jevy, které budou způsobovat animaci rostlin a specifikujte datový formát.
3. Zvolte rostlinu a s přihlédnutím k předchozímu bodu navrhněte možné animace, které bude vykonávat. Navrhněte strukturu a formu rostliny, kterou budete osazovat do pole.
4. Navrhněte způsob zobrazování rostliny s důrazem na úsporu času.
5. Proveďte implementaci knihovny tak, aby bylo možné knihovnu pro zobrazení připojit k libovolné aplikaci. Ověření správnosti proveďte implementací testovací aplikace. Součástí předaného výsledku bude minimálně jedno demonstrační pole rostlin.
6. Všechny kroky a sestudované materiály řádně zdokumentujte.

Prohlášení

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

V Plzni 16.5.2008 .....

Jan Plas

Poděkování

Je mojí milou povinností poděkovat všem, kteří mi pomohli při vzniku této práce. Zejména bych rád poděkoval svému hlavnímu konzultantovi práce Ing. Ivu Hanákovi za jeho užitečné rady a obětavý přístup. Dále bych rád poděkoval Ing. Petru Lobazovi, za to, že si mě vzal pod svá křídla, stal se vedoucím mé práce, a poskytl mi také velice užitečné informace. Díky patří také v neposlední řadě mé přítelkyni a rodině za to, že mi poskytli maximální podporu a vytvořili výborné zázemí k mému studiu.

Abstract

The purpose of my work was to achieve fast rendering of dense vegetation and the creation of a dynamic library for this purpose. This library could be used as a part of a greater program such as game aplication or virtual reality when we need to render this type of vegetation. The final work was precede by intense study of materials about the creation of vegetation and in detail study of Level Of Details (LOD) methods. Sunflowers were chosen for purposes of this work. The work is divided into two sections, the first one is the output of my theoretical studies, the second one is mainly about the aplication of these theoretical knowledge.

The first part is divided into three chapters. The creation of plants using L-systems and models of 3Dstudio Max is described in the chapter Modelování rostlin. The next one is LOD and finally the chapter called Návrh animace which contains the theoretical knowledge of possibilities of animation of plants. The possibilities are the direct and the inverse kinematics and the animation of the chosen model of plant.

In two chapters of the second part: Výsledné řešení and Měření a výsledky contains the summary of solutions and problems which accompanied this work.

Obsah:

1. Úvod

Rychlé zobrazení husté vegetace je velmi potřebné pro dnešní počítačový svět, ať již se jedná o počítačové hry, či virtuální realitu. Tento úkol však není úplně triviální a vyžaduje podrobnější přístup. Dříve stačilo pozorovateli vidět uprostřed scenérie jen statické obrázky, ale dnes je široká veřejnost mnohem náročnější a chce, aby se dané objekty co nejvíce podobaly realitě. To znamená, aby byly detailně zpracovány, a hlavně aby reagovaly na okolní vlivy jako například na poryvy větru.

Cílem mé bakalářské práce je dosáhnout rychlého vykreslování husté vegetace, proto musím nastudovat a zpracovat existující materiály týkající se postupů tvorby rostlin. Dále je mým úkolem seznámit se s problematikou animace přirozeného pohybu těchto rostlin a osvojit si a následně použít metody LOD. Poté je nutno si vše ověřit vytvořením knihovny, která bude připojena k demonstrativní aplikaci, v níž bude umožněn průlet nad polem slunečnic v reálném čase.

Obr. 1:Ukázka z programu

V první řadě je však nutné se seznámit s prostředím, které má být simulováno, jak se rostliny chovají a jaké faktory mají vliv na jejich pohyb. Zda se hýbají všechny stejně, nebo jestli jsou mezi nimi nějaké zásadní rozdíly. Je také nutno si zvolit takovou rostlinku, aby byla pro účely této práce vyhovující.

Po první číslované kapitole, kterou je *Úvod*, následují teoretickými poznatky, ve kterých jsou zmíněny metody pro *modelování rostlin*, tvorbu *LOD* a v neposlední řadě také *Návrhy animace*. Postupy a principy implementace jsou pak k dispozici pod stejnojmenným názvem, tedy *Implementace*. V poslední číslované kapitole *Závěr* najdeme zhodnocení mého úsilí, nakolik bylo dosaženo vytyčených cílu a je zde zrekapitulováno vše, co se čtenář mohl díky této práci dozvědět, stejně jako další možná vylepšení této práce.

2. Modelování rostlin

Základními stavebními kameny v počítačové grafice jsou trojúhelníky. Ty se spojují do trojúhelníkových sítí. Čím více trojúhelníků síť má, tím je větší pravděpodobnost, že bude model detailnější. Je mnoho způsobů, jak modely vytvořit. Například použitím nástrojů 3D studia Max, MAYA a dalších programů, které jsou určené na tvorbu modelů nebo se můžou vytvořit přímo v aplikaci.

Vytváření modelů se také liší tím, jaké jsou použity postupy. Seznámil jsem se s následujícími postupy:

- L-Systémy
- modelace rostlinky jako jednolité trojúhelníkové sítě
- modelace pomocí kostí
- modelace rostlinky po částech

Je také nutné si uvědomit, že při modelaci rostliny musíme brát v úvahu, jaké síly budou na rostlinku působit. Již při prvním pohledu je zjevné, že ne každá rostlina se pohybuje stejným způsobem. Závisí to na výšce a tuhosti stonku, povrchu rostliny, apod. Například tráva se pohybuje samozřejmě odlišným způsobem nežli smrk a dokonce i dvě stejné rostliny, v tomto případě slunečnice, se mohou vlivem těchto rozdílných vlastností pohybovat odlišně. Přesto i tyto dvě květiny mají v pohybu mnoho společného. Důležité je si tohoto faktu všimnout a pokusit se ho využít. Jako vhodnou rostlinu pro mé účely, jak jsem již nastínil, jsem zvolil slunečnici. Ta disponuje nejen dobrým vzhledem, ale je také rostlinou, pro níž je pole přirozeným prostředím.

Další věcí, která by měla být zjištěna je, jaké faktory působí na pohyb rostlin. Jako hlavní lze označit působení větru. Nesmíme však opomenout ani vliv slunce a přítomnost okolních rostlin. Pro naše účely zde však budeme brát v úvahu pouze vliv větru. Slunce ovlivňuje pohyb pouze minimálním způsobem, proto jej můžeme vypustit.

2.1. L-Systémy

Lindenmayerovi systémy, neboli L-Systémy [1], byly vytvořeny jako matematická teorie na vývoj rostlin. Původně neobsahovaly dostatečné množství detailů, aby bylo možno

je používat na komplexní tvorbu vyšších rostlin. Důraz byl kladen na vztah mezi sousedními prvky rostliny, jakými jsou buňky nebo větší rostlinné části. Jejich prostorové uspořádání bylo základem teorie L-Systémů.

Ve své nejjednodušší formě je L-Systém vykonávání série příkazu zapsané pomocí určitých bezkontextových gramatik a přepisovacích pravidel. Po několika přepsáních se získá posloupnost znaků, která se následně geometricky interpretuje. Každý znak má příslušný význam, jako například transformaci, rotaci, generování nového znaku, či provedení nějaké akce.

K interpretaci znaků se používá tzv. želví grafika [3], ve které želva představuje pomyslné kreslící zařízení. Ve své základní podobě se želva pohybuje pouze v rovině, ale lze rozšířit i na prostor. Zde však bude uveden příklad pro pohyb v rovině. Aby želva věděla, kam se má vydat, musí být definována svým stavem a tabulkou akcí. Její stav se skládá z její polohy a její orientace, přičemž orientace se v trojrozměrném prostoru, podle [1], skládá ze tří vektorů. Jsou jimi:

- $\vec{H} - (heading)$ – směr, kterým se želva pohybuje vpřed
- $\vec{U} - (up)$ – směr želvího krunýře
- $\vec{L} - (left)$ – strana, na které má levou přední i zadní nožičku

Význam dále použitých symbolů je takovýto:

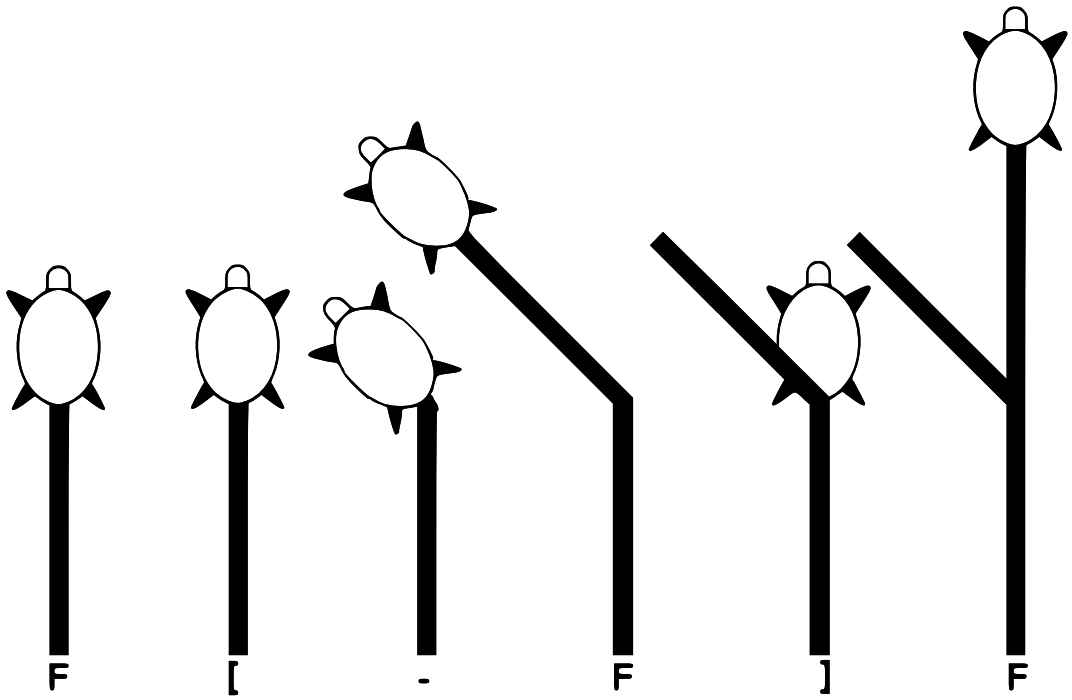
- *F* – pohyb želvy o jeden krok délky *d* a současné namalování čárky.
- *f* – pohyb želvy o jeden krok délky *d* bez kreslení čárky
- + a – – pootočení želvy o 60° doprava, resp. doleva

Při trojrozměrné modelaci se navíc používají symboly:

- & a ^ – rotace nahoru a dolů podle vektoru $\vec{L}$
- / a \ – rotace doleva a doprava podle vektoru $\vec{H}$

Dále je možné použít závorkové notace použitím *[* a *]*. Tím se dá docílit toho, že je želva obohacena zásobníkovou pamětí. To znamená, že při nalezení znaku *[* si želva zapamatuje svůj stav, tj. svou polohu a natočení, a pokračuje v pohybu podle znaků

uvedených dále a to do doby, než narazí na symbol *]*. Tento symbol říká, že se má želva vrátit do polohy, kterou si zapamatovala při nalezení znaku *[*. Takto lze vytvořit různé větvení (vyobrazené na Obr.2).



Obr.2: interpretace řetězce F[-F]F

Pomocí L-systémů se závorkami lze tvořit složité větvení a struktury jakými jsou větve, strom, keře apod. Dále viz [4]. Pro účely této práce však není jejich použití ideální, protože slunečnice je ve své podstatě velmi jednoduchá rostlina, která se skládá pouze ze stonku, listů a okvětí, které je kompaktní. Proto se zaměřím na ruční modelování rostlin v programu k tomu určenému bez pomoci L-systémů.

2.2. Rostlinka jako jednolitá trojúhelníková síť

V programech jakými jsou například MAYA nebo, v mém případě, 3D Studio Max lze vytvořit nespočet 3D modelů [7, 8, 9]. Všechny tyto modely jsou tvořeny trojúhelníky, které se spojují do trojúhelníkové sítě. Vytvořeným modelům je možné přidělit materiály a následně lze celý model vyexportovat v podobě trojúhelníkové sítě.

Tento způsob tvorby rostlinky je výborný, protože můžeme vytvořit model přesně tak detailní, jak potřebujeme. Pokud však chceme uvést rostlinku do pohybu, není tento způsob modelování nejlepším řešením. Z tohoto důvodu jej lze rozšířit o použití kostí.

Objekt s kostmi se jeví jako jeden z nejlepších, jelikož je předurčen přímo k tomu, aby se dal velice snadno animovat. Všechny kosti jsou spojeny pivoty (z angličtiny čep, kloub), které obsahují vlastnosti jejich spojení jako je například tuhost. Na vytvořenou trojúhelníkovou síť se pak výsledná kostra připojí pomocí různých nástrojů. Jednotlivé body sítě jsou následně ovlivňovány kostmi a jejich vzájemným působením dochází k deformaci této sítě, což nám vyhovuje.

Jejich použití je však v tomto případě zbytečné, protože se jedná o velice jednoduchý model. Proto je lepší využít následujícího řešení.

2.3. Rostlinka jako soubor jednotlivých částí

Tento způsob je velmi podobný postupu, který je zmíněn výše. Rozdíl je v tom, že v tomto případě nebude model exportován jako jednolitá trojúhelníková síť, ale že se vyexportují jednotlivé části zvlášť (viz obr. 3). To znamená, že se například pro rostlinu může vyexportovat stonek, list nebo okvětní lístky zvlášť. Vznikne nám tak místo jednoho souboru, který obsahuje údaje o trojúhelníkové síti, souborů více. Jejich počet je pak roven počtu vyexportovaných částí.

Každá jednotlivá část se však musí posunout tak, aby její kořen byl v počátku soustavy souřadnic. Na své pravé místo se pak dostane až v aplikaci. V té se vypočítá její skutečná poloha, do které je následně přemístěna, viz níže rovnice (1) v kapitole 4.2.2. Tímto se dá nahradit i funkce kostí.

Další výhodou je, že se může kdykoliv vyměnit jakákoliv část modelu za jinou nebo pootočit střední část modelu, aniž bych musel tvořit celý nový model, což u modelu v kapitole 2.2 nebylo možné.

Obr. 3:(a) horní část, (b) prostřední část, (c) dolní část modelu rostliny

3. LOD (Level Of Details)

Prvním, kdo vyslovil myšlenku Level Of Details, nebo-li úrovně detailů, byl v roce 1976 James H. Clark, první průkopník 3D počítačové grafiky. Ten si uvědomil, že na větší vzdálenost se nemusí kreslit věci tak detailně, protože lidské oko stejně vidí například místo očí jen tmavé skvrny. Proto se například na 20 pixelech nemusíme snažit zobrazit 500 trojúhelníků ale například jen 50. [6, 2]

Čím více trojúhelníků model má, tím se dá předpokládat, že bude kvalitnější a dá se zobrazit více detailů na jeho povrchu. To však znamená problém, protože čím má objekt více trojúhelníků, tím je vykreslování náročnější. Proto je nutné se s tím vypořádat a to použitím úrovně detailů. Tento pojem je chápán jako možnost odebírat nebo přidávat určitému objektu jeho detaily. Nejvyšší úroveň reprezentuje objekt s nejvyššími detaily a naopak nejnižší úroveň znamená co nejvíc zjednodušená reprezentace objektu, kterou jsme ochotni akceptovat.

Při používání LOD se setkáváme s problémem, který se nazývá „vyskakování“ jednotlivých částí objektů. To se projevuje tak, že při jednotlivém střídání úrovní zmizí nějaké trojúhelníky. Toto chování je nežádoucí a lze ho odstranit použitím technik jakými jsou například morfing geometrie jednoho stupně na druhý, nebo že v předstihu vytvoříme snímek s modelem, který následuje po změně LOD, a poté interpolujeme mezi tímto novým modelem a původním modelem. Jedná se o tzv. prolínání.

Při použití metod LOD se často vyskytuje problém problikávání mezi jednotlivými úrovněmi LOD. Tento efekt se dá redukovat tím, že se zavede takzvaná Hysterezí smyčka. To znamená, že se přidá opoždění k přepnutí mezi jednotlivými úrovněmi LOD. Konkrétně se přechod na nižší úroveň LOD uskuteční později a naopak, přechod na vyšší LOD model o něco dříve.

3.1 Výběr úrovně LOD

Jak již naznačil Clark [2], úroveň detailů je závislá na měnících se podmínkách. V jeho případě na vzdálenosti od pozorovatele, což je nejčastěji používané kritérium. To však není jediná možnost, jak zjistit, zda se má přepínat mezi různými úrovněmi. Těchto možností je podle [1] více:

- Vzdálenost objektu od pozorovatele

- Velikost průmětu objektu na obrazovku (počet pixelů, v nichž je objekt vidět)
- Velikost prostorového úhlu, v němž je objekt či jeho obálka vidět z pozice pozorovatele
- Celkový počet polygonů v pohledovém objemu
- Způsob lidského vnímání – detailněji zobrazovat objekty ve směru pohledu (střed obrazovky) a k krajům obrazovky kvalitu snižovat
- Měření a uvažování chyby ve výsledném obraze, kterou LOD způsobí

3.2 Způsoby reprezentace LOD

Jednotlivé úrovně detailů je možno tvořit různými způsoby. Jsou jimi například:

- Spojité stupně detailů
- Diskrétní stupně detailů
- Zjednodušení pomocí jednotlivých obrázků a bodů

3.2.1 Spojité stupně detailů

Metodu, která je známa pod názvem *progressive meshes*, vytvořil v roce 1996 Hugeus Hoppe [11]. Ta zachycuje jednotlivé kroky pro vytvoření zjednodušených sítí. Ze zjednodušeného modelu se dá tedy dostat na sousední úrovně detailů.

U modelů využívajících této a jí podobných metod se stupně detailů mění v jednotlivých krocích, ve kterých se provede úprava v trojúhelníkové síti. Ve většině případů se model zjednodušuje pouze o jeden či dva trojúhelníky a to tak, že ho buď přidají či odeberou. Vycházejí z decimačních algoritmů [2]. Tyto se zaměřují na trojúhelníky, které podle algoritmu nemají větší význam. Vybrané trojúhelníky následně odstraní a tento nový model uloží do odpovídající struktury. Tento postup by se však měl provádět již ve fázi předzpracování, aby nebylo nutné při vykreslování provádět složité geometrické výpočty.

Použití jemné, spojité změny úrovně detailů je náročné na datovou strukturu a vyžaduje průběžné vyhodnocování kvality modelu. Proto je zde jednodušší řešení.

3.2.2 Diskrétní stupně detailů

V tomto případě se pracuje na úrovni celých objektů, pro které se před samotným spuštěním aplikace vytvoří několik reprezentací (většinou tři až pět) s klesající úrovní detailu. Čím vyšší je úroveň detailu, tím je objekt detailnější. Pro vytvoření těchto úrovní se může

použít rovněž decimačních algoritmů, ale vhodnější je udělat to ručně v nástroji na to určeném, protože návrhář sám nejlépe ví, které detaily se mohou vynechat.

Každé úrovni je přiřazeno číslo představující hodnotu, podle které pak dochází, při zobrazování, k přepnutí na další model. Hlavní nevýhodou je skokové přecházení mezi jednotlivými úrovněmi. Naproti tomu je výhodou snadná implementace nízký počet porovnávání potřebný k rozhodování, kterou úroveň zrovna použít.

Využil jsem tohoto způsobu reprezentace úrovně detailů. Byly vytvořeny tři modely slunečnic (obr. 4) s různými stupni detailů. Změny v detailech se týkaly především okvětních lístků. Pro zjištění, kdy se mají stupně přepínat jsem zvolil vzdálenost od pozorovatele. Modely se přepínají:

- *obr. 4a)* a *obr. 4b)* ve vzdálenosti 6 metrů
- *obr. 4b)* a *obr. 4c)* ve vzdálenosti 10 metrů

Obr.4: Jednotlivé modely s úrovněmi LOD: (a)1.st – 4030 trojúhelníků, (b) 2.st – 2083 troj. (c) 3.st – 1145 troj.

3.2.3 Rovinné representace modelů

Ve větší vzdálenosti je zbytečné zobrazovat 3D model. Stačí ho nahradit vhodnou reprezentací ve 2D. Objekt s velkým množstvím trojúhelníků se přemění na mrak barevných bodů nebo na jednoduchý nosný polygon, nejčastěji obdélník. Ten si lze představit jako

skleněnou tabuli, na které je vyobrazen nahrazovaný objekt. Tyto náhražky (nosné polygony) se liší svým vypracováním a používají se pro ně tyto názvy:

- Impostory
- Billboardy
- Sprity
- Point clouds

Impostor

Na nosném obdélníku, který se staticky umístí do scény, se může zobrazit i více objektů najednou. Tento nosný obdélník je tvořen polygony, které dovolují jeho snadnou deformaci. Impostory se ve většině případů nevytváří předem, ale vypočítávají se ze skutečných modelů přímo v aplikacích. To však vede k problému. Pokud se použije například v lesním porostu, kde zastupuje řadu stromů, tak pozorovatel při pohybu na různé strany bude očekávat, že se i pohled změní. Toto však statický Impostor neumí a snižuje tak kvalitu vjemu. Působí pouze jako obraz na stěně. Proto se musí vytvářet z předchozích snímků.

Billboard

Tato reprezentace se hodí, pokud se má použít na jeden objekt, kterým může být například strom. Billboard je velice podobný Impostoru, ale na rozdíl od něj nestojí staticky ve scéně, ale natáčí se pokud možno k pozorovateli. Díky tomu se zamezí, aby viděl pozorovatel objekt z boku s velmi zkreslenou texturou.

Billboard je vhodné použít například v případě, že se již nedá, kvůli velké vzdálenosti, či přílišnému množství rostlinek v poli, rozpoznat o jakou rostlinu vlastně jde.

Sprite

Sprite je velmi podobný Impostoru, ale na rozdíl od něj je jeho nosný obdélník obvykle tvořen pouze dvěma trojúhelníky. Navíc se často používá rozšířená varianta, která spočívá v tom, že se vytvoří Sprite kombinací několika dalších, navzájem se protínajících, nosných obdélníků. Na ně je opět nanesen příslušný obraz tak, aby co nejvíce odpovídal předchozímu modelu. Díky tomu vznikne pro pozorovatele dojem trojrozměrného obrazu. Pokud by však

byl Sprite tvořen pouze jedním nosným obdélníkem, mohlo by dojít k tomu, že pozorovatel ho zahlédne z boku, a protože se na rozdíl od Billboardu nenatáčí směrem k pozorovateli, viděl by pouze tenkou placku.

Point clouds nebo-li mraky bodů

Dalším způsobem, jak urychlit scénu je přeměnit model z trojúhelníkové sítě na směs barevných bodů. Tento způsob representace tedy není 2D zobrazení jako předešlé možnosti, ale naopak zůstává 3D. Mraky bodů pak mají stejnou barvu, jako model ve stejných místech. Například když se část nahradí bodem v oblasti okvětního lístku žluté barvy, tak tento bod v těchto místech obarví také žlutou barvou. Naopak bod v oblasti stonku zelené barvy se nahradí také bodem stejné barvy. Přesto se tato representace vykresluje rychleji, neboť nedochází k vykreslování všech plošek modelu, ale jen jeho některých jeho bodů. Nemusí se tedy vykreslovat celá jeho plocha. Více viz [1, 2].

3.3 Mé návrhy LOD a dodatečná urychlení

3.3.1 Zmenšování počtu slunečnic v trsu

Jednotlivé úrovně detailů by bylo možné tvořit také změnou počtu rostlinek v jednom trsu (obr. 5). Jedná se vlastně o analogický postup jako v podsekci 3.2.2, kdy by se jednotlivé úrovně přepínaly v závislosti na vzdálenosti od pozorovatele. Tento postup však není nejvhodnější, protože účelem LOD není snižovat počet rozpoznatelných objektů, nýbrž ušetřit čas při jejich vykreslení. Navíc slunečnice nerostou v trsech a nejednalo by se o universální řešení. Dále by byl přechod mezi jednotlivými úrovněmi detailů velmi znát, proto jsem od tohoto způsobu upustil.

Obr. 5: Jednotlivé změny počtu rostlinek v trsu

3.3.2 Výpočet úrovně detailů

Vypočítávat, zda se má provést změna mezi jednotlivými úrovněmi detailů, je časově náročné. Obzvláště pokud by se měl výpočet provádět pro každou rostlinku zvlášť. Aplikaci by to zbytečně zatěžovalo. Proto se tato operace může provádět například jen každých čtvrt vteřiny. I tak bude přechod mezi úrovněmi dostatečně kvalitní. Pokud by však byl interval mezi výpočty ještě větší, mohl by být přechod mezi úrovněmi příliš nápadný.

Aplikace se dá ještě více urychlit v případě, že se tento výpočet nebude provádět pro každou rostlinku zvlášť, ale pouze pro vybrané „jedince“. Takovým „jedincem“ může být například rostlinka nacházející se v samém středu mezi devíti květinami (obr. 6). Ta se jeví jako perfektní adept na výpočet úrovně detailů. Ve chvíli, kdy se změní úroveň tohoto „jedince“, změní se i úroveň okolních rostlinek. Tento postup by neměl být implementován, pokud zde není dostatečný počet rostlinek. V tom případě by mohlo nastat, že by se všechny přeměňovali naráz, což je pro naše účely nevhodné.

Obr. 6: Rozdělení pole na jednotlivé podčásti (označeny modře). Červeně zakroužkované rostliny jsou našimi „jedinci“

Pokud nemají rostlinky potřebný počet kolem sebe, jako je tomu na obr.6, přivlastní si alespoň ty rostlinky, které jsou kolem dostupné. Pokud takové nejsou žádné, zůstane rostlinka osamocena.

4. Návrh animace

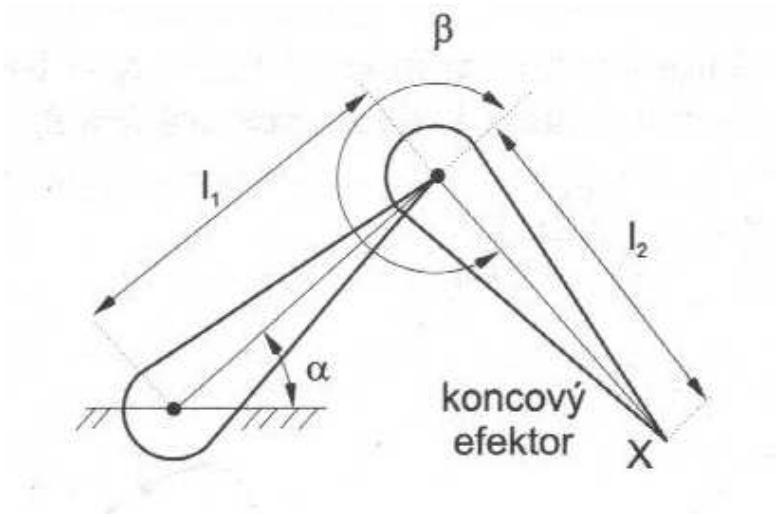
Základem animace je takzvané klíčování. Tuto techniku použili jako první v dílnách Walta Disneye. Hlavní osobou při výrobě animovaných filmů byl vrchní animátor, který vymýšlel jednotlivé postavy a maloval jejich pohyb. Ale protože malování jednotlivých přechodů a jejich pohybu je stereotypní práce, kreslil pouze klíčové snímky a pomocní animátoři pak dokreslovali mezistavy.

S příchodem výkonnějších počítačů byla snaha kreslení mezisnímků zautomatizovat. Postupem času se začal tento postup, kdy se zadávají pouze klíčové snímky, používat i v trojrozměrné animaci [1, 9]. Klíčování je zástupce nízkoúrovňové animace.

4.1 Kost a kostra

4.1.1 Segmentová struktura

Záměrem tohoto druhu animace je převážně animovat lidský pohyb. K tomu je zapotřebí segmentová struktura [1]. Objekty takto postavené jsou tvořeny pevnými částmi, které jsou k sobě připojeny. V každém tomto spojení lze s jednotlivými segmenty pohybovat. Tato struktura bývá většinou na jednom konci pevně ukotvena. Pokud je opačný konec volný jedná se o otevřenou segmentovou strukturu a bod, který je na tomto konci je koncový efektor.



Obr. 7: Otevřená segmentová struktura se dvěma segmenty (podle [1])

Dále k určení přesné polohy tělesa v prostoru je potřeba zvolit souřadnicový systém a celkem šest čísel. První tři z nich určují polohu v prostoru a další tři natočení tělesa vůči

jednotlivým osám. Tyto veličiny, které jednoznačně charakterizují jakýkoliv systém, se označují jako stupně volnosti (DOF- Degree Of Freedom). Jejich počet se může měnit a to buď přidáním dalšího segmentu, nebo pokud se některé části pevně svážou. Na obr. 7 je znázorněna segmentová struktura (podle [1]), která se skládá ze dvou segmentů. První je pevně připevněn k podložce a druhý je volný. První se tedy může pohybovat pouze kolem jedné osy a druhý se může otáčet jen ve spojovacím kloubu rovněž pouze kolem jedné osy. Z toho vyplývá, že tato struktura má pouze dva stupně volnosti, protože je plně popsána pomocí dvou úhlů α a β.

Všechny stavy, do kterých se mohou segmenty dostat, tvoří stavový prostor této struktury. Okamžitý stav segmentu může být popsán stavovým vektorem Θ, který má stejnou délku jako je počet stupňů volnosti. Struktura na obr. 7 je tedy popsána stavovým vektorem $\Theta = (\alpha, \beta)$.

4.1.2 Aplikace kostry

V počítačové grafice je snaha zjednodušit reprezentaci objektu, tak aby obsahovala co možná nejméně parametrů a přitom umožňovala snadnou manipulaci. K tomu se používá v počítačové animaci tzv. DH-notace pojmenovaná podle Denavita a Hartenbergra. Ti ji v roce 1955 popsali pro použití v robotice.

V této notaci se každý kloub reprezentuje vlastní souřadným systém a čtyři parametry, které určují, pomocí jaké transformace se dostaneme k dalšímu segmentu [1, 2]. Pomocí těchto parametrů lze sestavit transformační matice.

4.1.3 Kinematika

Kinematika je odvětvím fyziky studující pohyb nezávisle na silách, které tento pohyb způsobují. Jsou zde dva druhy a to *přímá* a *inverzní* kinematika. Oba dva druhy se zabývají zjišťováním polohy koncového efektoru a hodnot stavových vektorů.

Přímá kinematika

Princip určování polohy spočívá v tom, že se jednotlivé stavy určují pro všechny segmenty struktury postupně. Začne se od prvního segmentu a pokračuje se až ke koncovému efektoru. Například, pokud chceme pohnout okvětními lístky květiny, musíme nejdříve pohnout všemi segmenty stonku. Pokud ovšem bude rostlina působit nepřirozeně a bude se

muset změnit například úhel natočení u kořene – počátku trojúhelníkové sítě, změní se tím i celý tvar rostliny a budeme muset postupovat opět od kořene. Toto je asi hlavní nevýhodou přímé kinematiky.

Poloha koncového efektoru *X* lze tedy zapsat jako

$$X = f(\Theta).$$

To tedy znamená, že je určena pomocí transformace *f*, která vznikla pomocí nové hodnoty Θ. Do transformačních matic pro jednotlivé klouby jsou dosazeny konkrétní hodnoty parametrů DH-notace a tím je určena výsledná poloha efektoru.

Tento druh kinematiky je vhodný, pokud nepotřebujeme interaktivní vytváření pohybu. Je neintuitivní a ve většině případů zdlouhavé.

Inverzní kinematika

Zde se používá opačný postup výpočtu než je uveden u přímé kinematiky. Cílem je nalézt stavový vektor Θ podle informací o poloze koncového efektoru. Proto se inverzní kinematice také jinak říká *cílem řízený pohyb*. Formálně se dá tento postup zapsat jako:

$$\Theta = f^{-1}(X).$$

Je zde však několik významných problémů. Hlavními jsou:

- $f^{-1}()$ nemusí vůbec existovat
- f() je nelineární a velmi komplexní

Řešením těchto problémů je například inverze Jakobiánu (více viz [1, 2]).

4.2 Další možnosti animace rostliny

Animace jednotlivých modelů je velmi závislá na tom, jak je model vytvořen. Zmíním zde možnosti pro dva různé druhy modelů:

- Jednolitá trojúhelníková síť s kostmi
- Model složený z jednotlivých trojúhelníkových sítí

4.2.1 Jednolitá trojúhelníková síť

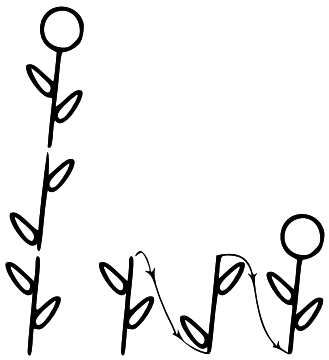
Při animaci tohoto modelu je hlavní slabina v tom, že rostlinka je schopna se pohybovat jen jako ručička na hodinách. Budu-li tedy chtít ohnout rostlinku uprostřed, její konstrukce mi to nedovolí. Z tohoto hlediska se tento postup jeví jako nevyhovující.

Další možností je vybavit trojúhelníkovou síť kostmi. Pak je model vytvořen přímo k tomu, aby byl animován. V tom případě se musí zvolit příslušný exportér, který dokáže vyexportovat trojúhelníkovou síť s kostmi ve formátu, který je podporován zvoleným programovacím jazykem. Jedná se o soubor, ve kterém jsou obsaženy veškeré informace o modelu. Existuje však další varianta, jak přistoupit k animaci trojúhelníkové sítě.

4.2.2 Model složen z jednotlivých částí

Při použití modelu složeného z jednotlivých částí se dá v aplikaci vytvořit spojení, které se mírně podobá kostem a kloubům. Na toto spojení však není možné použít předchozí postupy, protože zde fyzicky žádné kosti s klouby neexistují. Přemístěním jednotlivých částí na svá místa se vytvoří celistvý model rostliny, přičemž se dá s jednotlivými částmi dobře pohybovat. Musí se pouze dbát na to, aby jednotlivá část končila v místech, kde druhá začíná.

To se dá zajistit tím, že každou část (viz obr.3), při každé změně polohy, posuneme na své místo tak, že pomocí úhlu rotace pro jednotlivé části vypočítáme velikost posunu a tu pak aplikujeme na koncovou část rostlinky (viz obr 8). Výsledně tedy dostaneme kýžený efekt i bez použití kostí. Například pokud začne působit vítr, první část, která se začne pohybovat, je dolní část rostlinky. Nejprve se přesune do počátku souřadné soustavy, natočí se podle jednotlivých os a poté se přesune do příslušné polohy. Následně se bude pohybovat stejným způsobem prostřední část a jako poslední vrchní část s okvětím.



Obr 8:Výsledná rostlinka složená ze tří částí, kde se každá z nich nejprve natočí a pak teprve přesune na své místo

5. Výsledné řešení

V této kapitole se budeme zabývat implementací výše zmíněných poznatků. K tomu jsem použil nástrojů programu Visual Studio a Direct3D.

5.1 Vytvoření vyšších stupňů LOD

Z informací, které jsem uvedl v kapitole 3.2.3, je poměrně dobře pochopitelné, jak by se měly jednotlivé vyšší stupně LOD tvořit. V mém případě se tedy jedná o Billboardy a Sprity. Oba tyto stupně mají mnoho společného. Co se týče pohybu, liší se pouze v tom, že jeden se natáčí v závislosti na pozorovateli a druhý zůstává neměnný. Mezi společné vlastnosti Billboardů a Spritů patří způsob jejich tvorby a minimální nároky na počet trojúhelníků. Na základě těchto vlastností jsem konstruoval nosný obdélník pouze ze dvou trojúhelníků.

Na nosné obdélníky se dále musí namapovat textura. K tomu, aby však mohl tento obdélník vzniknout, musíme nejdříve vytvořit jednotlivé body, které uložíme do VertexBufferu a v této struktuře dochází k jejich propojení. Body musí být za sebou ve správném pořadí, protože pro vykreslení VB je použito triangle stripu. Ta je určena pro vykreslování pásu trojúhelníků, kde první tři body tvoří trojúhelník a další vznikne pomocí nového bodu a posledních dvou bodů předchozího trojúhelníku. Vytvoříme tedy dva VB, kdy jeden je pro Billboard a druhý pro Sprite.

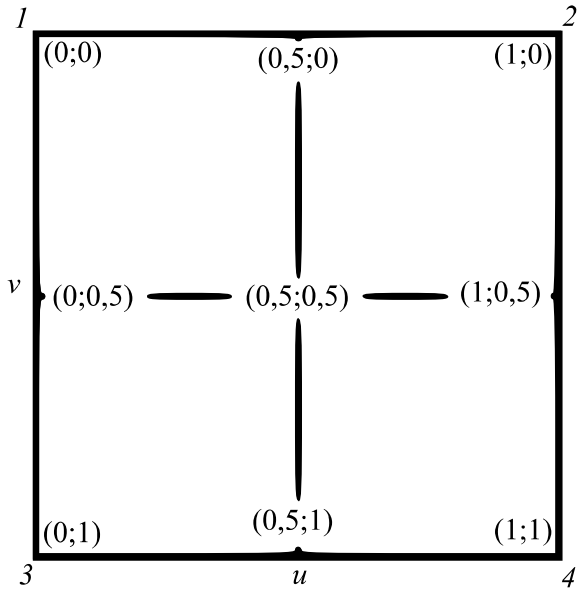
Každý trojúhelník se, jak známo, skládá ze tří bodů. Trojúhelníky, které tvoří nosný obdélník, mají společné právě dva z těchto bodů. Z toho plyne, že budeme potřebovat vytvořit pouze čtyři body. V tomto případě se jedná o PositionNormalTextured. Pro každý bod se zde musí samostatně nastavit:

- Pozice bodu – *x, y, z* pozice
- Směr normály – udává se opět v *x, y* a *z* smyslu
- Souřadnice textury vzhledem k bodu

Normála je vektor kolmý k povrchu, který se používá k vypočítání výsledného odraženého světla v daném bodě [2, 5].

Co se týká souřadnice textury vzhledem k bodu, je zde dvojice čísel, většinou označovaných *u* a *v*, která jsou k tomu přímo určená. Jedná se o nezbytnou informaci, chceme-li na námi vytvořený nosný obdélník nanášet texturu, což je hlavní důvod, proč

vznikl. Jedná se v podstatě o to, jak velkou část textury chceme zobrazit, při čemž se zde pohybujeme v rozmezí 0 až 1. Souřadnice *u* nám určuje horizontální rozsah a *v* rozsah vertikální. Chceme-li tedy, jak je tomu v tomto případě, nanést texturu na celou plochu, nastavíme u horních vrcholů souřadnici $v = 0$ a u spodních $v = 1$. U vrcholu vlevo zadáme $u = 0$ a vpravo $u = 1$ (viz obr. 9). Chceme-li přichytit bod k textuře přímo uprostřed, nastaví se jak *u*, tak i $v = 0{,}5$. Avšak v případě, že máme v jednom obrázku uloženy vedle sebe například 2 textury a chceme přichytit k bodu pravou krajní, změní se nám souřadnice *u* tak, že na vrcholech vlevo bude $u = 0$ nahrazeno $u = 0{,}5$. Navíc je vhodné, jak pro rychlejší vykreslování, tak pro snazší navázání textury, volit její rozměry $n \times n$.

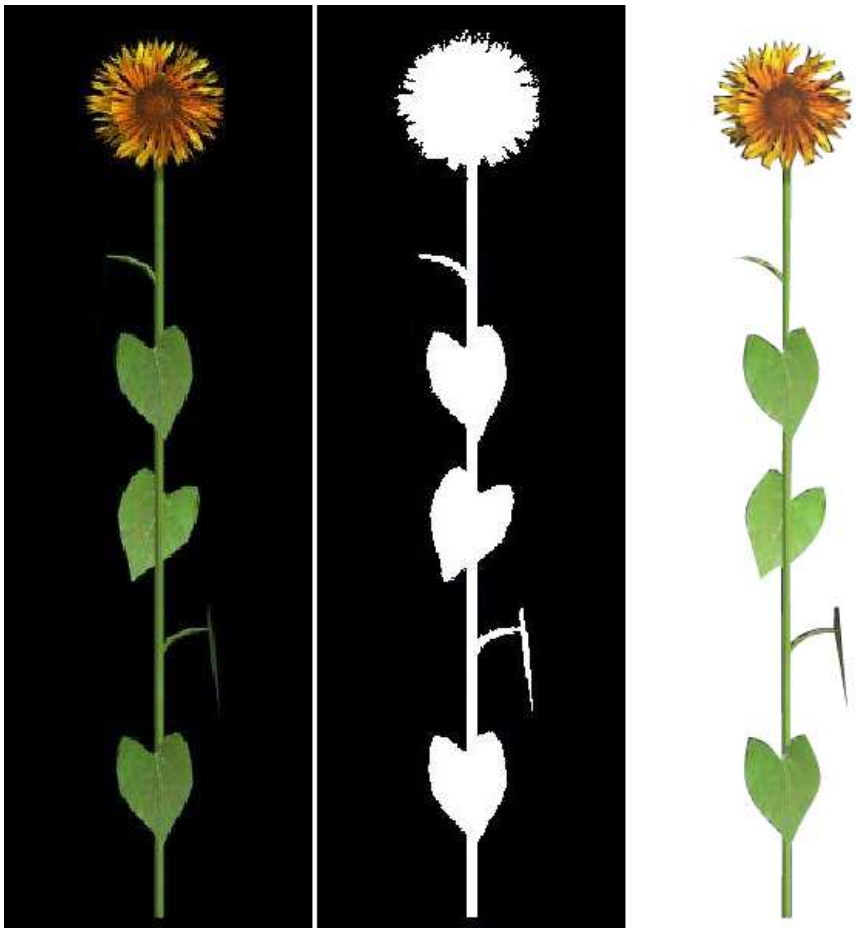


Obr. 9: Souřadnice (u,v) při nanášení textury na body

5.1.1 Textura

Součástí vyšších i nižších stupňů LOD je textura. Ta nám udává, jak ten, či onen model nebo Billboard bude vypadat. Z tohoto důvodu se jí budeme nyní blíže věnovat. Dříve než se rozhodneme, že nějaký objekt, v tomto případě nosný obdélník, bude mít určitou texturu, musíme si v první řadě zvolit vhodný obrázek. Pro náš účel byly vytvořeny obrázky odvozením přímo z modelů v 3D Studiu MAX. Díky tomu je přechod mezi modelem a Spritem velice plynulý.

Tento obrázek však není úplný, musíme ho dále upravit, přizpůsobit jej potřebám Billboardu. To znamená, že musíme změnit rozměr obrázku. Dále pak jen sloučit vzniklé obrázky do jednoho jediného a to je vše, jelikož 3DStudio MAX umí exportovat obrázek ve formátu PNG, který je podporován Direct3D, i s jeho průhledností, takzvaným *alfa kanálem* (viz Obr. 10). Alfa kanál je zejména důležitý pro textury Billboardů a Spritů, jelikož určuje, která místa budou průhledná a která nikoliv (tzv. částečně průhledná textura). Vznikne nám tak tedy výsledný obrázek (obr. 10c), který se skládá z původního obrázku a jeho alfa kanálu (obr 10a, 10b), s potřebnou velikostí stran a průhledností, čehož jsme se snažili dosáhnout.



a) *b)* *c)*

Obr. 10: a) Původní obrázek, b) alfa kanál , c) výsledný obrázek

5.1.2 Billboard

Jak již bylo řečeno, co se týče pohybu, máme dvě kategorie. Billboard patří do kategorie první, charakterizujeme ho tím, že se natáčí vždy čelem k pozorovateli. K natočení je vhodné použít úhlů *yaw* a *pitch*, které určují směr pohledu pozorovatele. Pro naše účely stačí

aplikovat jen úhel *yaw*, jenž určuje natočení nosného obdélníku kolem osy y. K vytvoření nosného obdélníku potřebujeme tedy celkem čtyři body, které se uloží do VB.

5.1.3 Sprite

Jak bylo popsáno v sekci 3.2.3, Sprite jsou nehybné a obvykle se skládají z více nosných obdélníků. Tohoto faktu jsme zde využili a na vytvoření Spritu jsme použili jen dva nosné obdélníky, protože se nám objevují až ve větší vzdálenosti (viz níže sekce 5.2).

K tomu opět použijeme stejný postup jako u Billboardů, ale horní dva body mírně nakloněny dopředu. Je to z toho důvodu, že slunečnice není jednoduchého tvaru a u svého květu je mírně nakloněná dopředu a při pohledu ze strany na Sprite by mohlo dojít k tomu, že by se příslušná místa na okvětí slunečnice navzájem přímo nekryla. Z tohoto důvodu muselo dojít k tomu, že se přizpůsobí poloha horních bodů a naklonit se mírně vpřed.

5.2 Určení LOD

Určování jednotlivých úrovní je základem pro rychlejší vykreslování. Pokud by se stalo, že rostlinka je špatně zařazena, může dojít k tomu, že je vykreslována buď příliš kvalitně, což může aplikaci zpomalit, nebo naopak nedostatečně kvalitně. Proto jsem musel experimentálně vyzkoušet několik variant. Všechny byly založeny na vzdálenosti rostlinky od pozorovatele. Jelikož je zde přesně pět úrovní detailů, tj. tři různě kvalitní modely, Sprity a Billboardy, musel jsem určit, kdy vykreslovat jaký druh representace slunečnice. Nejdříve jsem se rozhodl provádět výpočet podle (1), ale vzhledem k tomu, že provádění odmocniny je poměrně náročné, rozhodl jsem se od tohoto upustit, pokračovat dále bez odmocniny a srovnávat vzdálenosti v neodmocněné formě.

$$d'_{i,j} = \sqrt{\left(\frac{j}{2} - x\right)^2 + \left(\frac{i}{2} - z\right)^2} \qquad (1)$$

Toto je ale pouze matematické vyjádření a protože nepotřebujeme přesnou hodnotu, ale potřebujeme pouze zachovat pořadí, že je zbytečné provádět odmocninu. Nakonec má tedy rovnice tvar

$$d_{i,j} = \left(\frac{j}{2} - x\right)\left(\frac{j}{2} - x\right) + \left(\frac{i}{2} - z\right)\left(\frac{i}{2} - z\right), \quad (2)$$

kde $d_{i,j}$ představuje pomocnou vzdálenost pozorovatele od rostlinky, j a i jsou indexy polohy slunečnice v poli. Dále z a x je poloha pozorovatele.

Výsledně je ale potřeba nalézt přesnou hodnotu $d_{i,j}$ pro přepínání jednotlivých representací. Velikost $d_{i,j}$ je udávaná v metrech, ale přesná vzdálenost od rostlinky nám vznikne až když tuto hodnotu odmocníme. Po několika pokusech jsem dospěl k tomu, že jako nejlepší hodnoty pro jednotlivé druhy jsou tyto:

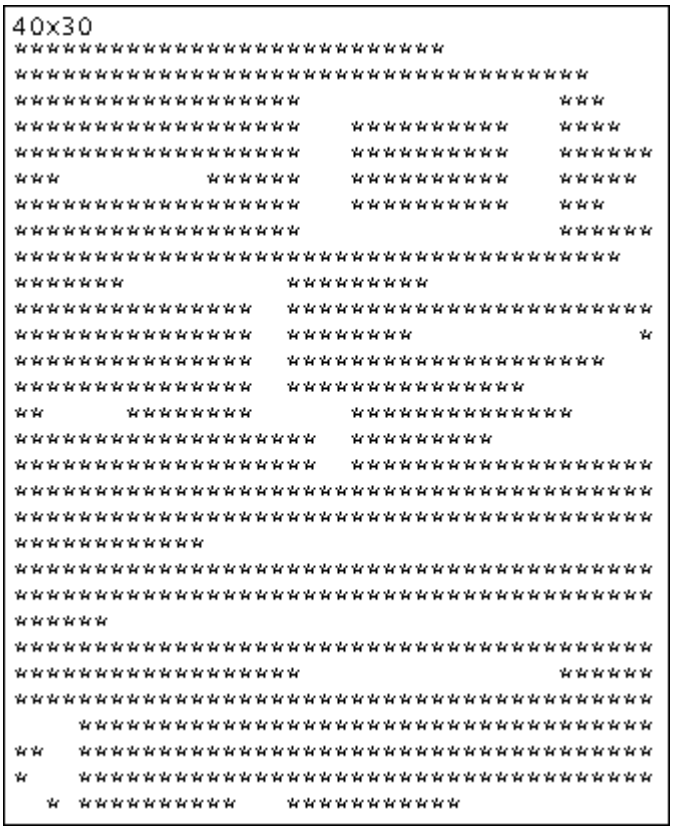
Přepnutí mezi	hodnota $d_{i,j}$	Počet trojúhelníků	
		první	následující
1. a 2. úroveň modelu	36	4030	2083
2. a 3. úroveň modelu	100	2083	1145
3. úroveň modelu a Sprite	400	1145	4
Sprite a Billboard	625	4	2

Tab. 1: Vzdálenosti pro přepnutí mezi jednotlivými úrovněmi LOD

5.3 Vytvoření pole

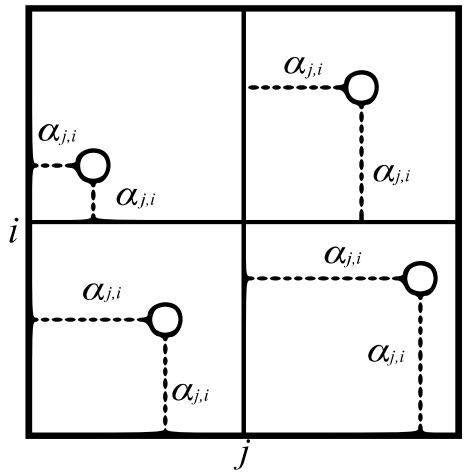
Když jsme si již vysvětlili, jak se určuje LOD, je také důležité nadefinovat, jak bude pole vlastně vypadat. K tomu nám dobře poslouží obrázek, který je representovaný jako textový soubor. Na první řádek uvedeme rozměry matice rozložení slunečnic na poli a na další řádky se vloží '*' do míst, kde má být později umístěna slunečnice. V místech, kde nemá být nic, se nachází mezera. Rozměry pole jsou však ve skutečnosti poloviční, jak bude vysvětleno později. Příklad takového souboru je na obr. 11.

Ve zde uvedeném případě se jedná o slunečnicové pole o rozměrech 20×15 metrů. Výhodou této representace je, že mohu později doplnit i nějaké další „věci" do pole, jako je například velký kámen nebo cokoliv jiného. Stačí jen dodefinovat nový znak a přiřadit mu příslušnou hodnotu při vytváření pole v programu. Další výhodou je, že toto pole si může později každý jakýmkoliv způsobem upravit sám pomocí jednoduchého textového editoru. Nevýhodou ale zůstává, že musíme zadávat každou slunečnici zvlášť.



Obr.11: Textový soubor s rozložením pole

Výsledné pole bude mít rozměry jen 20×15 metrů, i když v mapovém souboru je 40×30 znaků. Je to tím, že výsledné pole má jako základní jednotku čtverec o rozměrech 0.5×0.5 metrů. V tomto případě se jedná o pole obdélníkového půdorysu, kde se na každém čtverci nacházejí právě čtyři rostlinky, které jsou pak v tomto čtverci rozmístěny přesně podle určitého klíče (viz obr. 12) Ten se skládá z toho, že se rostlince přiřadí pozice, která jí podle mapy náleží a přičte se k ní náhodná hodnota α, která se k tomuto účelu před spuštěním programu vygeneruje pro každou rostlinku zvlášť.



Obr. 12: Náhodná poloha rostlin ve čtverci pole

Náhodná hodnota α se generuje v rozpětí 0 - 0,5 metrů. To zaručuje, že budou všechny čtyři rostlinky umístěny v jednom čtverci, protože jeho hrana měří jeden metr. Každá rostlinka se při vykreslování posune přesně na svou pozici (viz výp. 1). Pokud by tato proměnná neexistovala, byly by rostlinky umístěny v řadě za sebou, což zaprvé nepůsobí příliš estetickým dojmem a zadruhé se v takovémto rozložení slunečnice na reálném poli příliš nevyskytují.

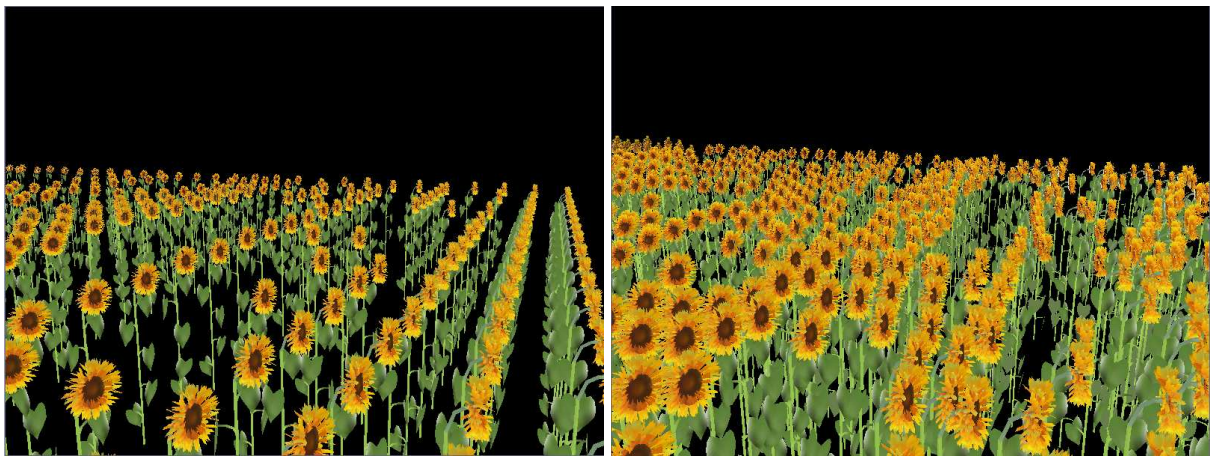
```
for (int i = 0; i < rozmer_i; i++)   // Prochazeni celeho pole slunecnic
{
        for (int j = 0; j < rozmer_j; j++)
        {
        x = (j + alpha[o]) / 2;              // X-ova pozice rostlinky
        z = (i + alpha[o]) / 2;              // Z-ova pozice rostlinky
        o++;
    }
}
```

Výp. 1: Určení polohy rostlinky v poli

Dále se dá použít α také k tomu, aby nebyly všechny slunečnice stejně vysoké. Budeme ji tedy aplikovat na zvětšení střední části rostlinky a to tak, že se její nová výška *s'* vypočítá podle

$$s' = s + \frac{\alpha}{10} \qquad (3),$$

kde *s* je původní velikost části rostliny, *s'* je nová velikost rostliny a α je náhodná veličina. Velikosti ve směru os *x* a *z* se nezmění, protože by se nám mohla zdeformovat celá část modelu. Hodnota α/10 je však natolik malá, že i když při zvětšování podél osy *y* dochází k lehké deformaci listů, není tato deformace nijak výrazná, ale na zvětšení rostlinek je dostačující. Dále pak musíme posunout i vrchní část rostliny, květ, který je nejdůležitější částí naší květiny, ostatně stejně jako je tomu u květin ostatních, a to opět o stejnou hodnotu jako se předtím zvětšila část prostřední. Použití náhodné hodnoty *α* je ukázáno na obr. 13. Na obr. 13a je ukázáno, jak vypadá pole pokud není použito náhodné polohy rostlin v poli. Toto uspořádání nevypadá hezky a navíc působí rušivým dojmem. Naopak na obr. 13b je ukázáno pole, na kterém je použito náhodné polohy. Je zde vidět i použití této hodnoty na velikost slunečnice.



Obr. 13: Zobrazení pole (a) bez použití a (b) s použitím náhodné polohy rostlinky

5.4 Pohyb rostlin

Další důležitou součástí mé práce je, aby se rostlinky hýbaly podle síly větru. Ten je stabilní, je určen směrovým vektorem a velikostí síly, která je v ms^{-1}. Na síle větru závisí, jak moc a jak rychle se rostlinka ohne. Existuje více možností, jak pohyb realizovat, jednou z nich je dokonalý fyzikální model, ten by byl však příliš komplikovaný na to, aby mohl být v rozumném čase vypočítán a realizován. Proto jsem se rozhodl pro způsob, který sice nebude naprosto dokonale simulovat pohyb slunečnic, ale bude co nejvěrněji reagovat na poryvy větru.

Pohyb rostlin je závislý na síle větru, která určuje, jak moc a kterým směrem se rostlina ohne. V kapitole 4.2.2 je nastíněno, jak se zachází s jednotlivými částmi rostliny (viz obr. 8). Konkrétní posun jednotlivých částí je

$$
\begin{aligned}
m_1 &= \sin\left(\frac{p_w}{d_1}\right), \quad m_2 = \sin\left(\frac{p_w}{d_2}\right) \\
y_1 &= \cos\left(\frac{p_w}{d_1}\right), \quad y_2 = \cos\left(\frac{p_w}{d_2}\right) \\
X_1 &= X_1' - w_z(m_1 + m_2), \quad X_2 = X_2' - w_z m_2 \\
Z_1 &= Z_1' + w_x(m_1 + m_2), \quad Z_2 = Z_2' + w_x m_2
\end{aligned}
\qquad (4)
$$

pro osu X a Z, kde X_1', X_2', Z_1' a Z_2' jsou původní polohy konce rostlinky, m_1 a m_2 jsou konstanty závislé na síle větru p a určují náklon jednotlivé části, stejně jako y_1 a y_2, které udávají posun konce rostliny podél osy y. Dále pak konstanty d_1, d_2 udávají tuhost rostlinky. Ty určují, jak moc bude rostlinka schopná se ohnout. Čím vyšší číslo, tím se rostlinka méně

ohne. Během několika pokusů jsem zjistil, že ideální hodnoty jsou pro d_1 = 1,5 a pro d_2 = 3. V neposlední řadě jsou zde konstanty w_z a w_x, které určují směr větru.

Nejprve byl pohyb poměrně zajímavý, ale bohužel se všechny rostlinky hýbaly naprosto stejně a navíc lineárně, což je v porovnání se skutečným polem nemyslitelné. Proto jsem se rozhodl, že se pokusím změnit lineární pohyb na nelineární a to konkrétně na pohyb podle některé z goniometrických funkcí. To nám zaručí, že rostlinka bude mít určitý nástup a doběh v průběhu pohybu. Výsledná síla ohybu p je tedy

$$p = p_w h \sin \varphi , \qquad (5)$$

kde p_w je síla větru v rozmezí od 0 do 100, h maximální velikost ohybu a φ je úhel, který ovlivňuje již zmíněný nástup a doběh pohybu. Hodnota 1.0 u proměnné p_w představuje přibližně 1 m/s.

Úhel φ se obměňuje v závislosti na čase, viz (6). Výsledná síla ohybu se pak následně předá dalším rostlinkám s určitým zpožděním. To způsobuje, že se každá rostlinka hýbe jinak. Pohyb se zároveň velmi podobá opravdovému pohybu rostliny. Ale stále to neodpovídalo mé představě. Proto jsem se rozhodl pro další vylepšení a přidal novou pomocnou proměnnou b. Je to jistá složka náhody, která způsobí, že pohyb rostlin bude ještě více podobný realitě.

Proměnná b se pravidelně obměňuje každé dvě vteřiny. Pokud by byl tento interval menší, docházelo by k jejímu přepínání příliš často a byl-li by větší, přepínala by se jen zřídkakdy. Je zde 50% šance, že se rostlinka začne pohybovat opačným směrem než doposud. Dříve bylo také nutno zjišťovat, zda rostlinka již nedosáhla „kritické hranice“. Tou bylo myšleno, že se rostlinka dostala do míst, odkud by se měla začít vracet. Ta byla určena tím, že bylo dosaženo maximální velikosti ohybu h, pro kterou bylo experimentálně zjištěno, že ideální velikostí je $\pi/12$. Při přechodu na nelineární pohyb jsem zjistil, že již není nutností proměnnou h přímo kontrolovat. Rostlina se pohybuje po sinusoidě, což znamená, že se pohybuje jak vpřed, tak i vzad v závislosti na velikosti úhlu φ.

Náhodná proměnná b se změní také v okamžiku, kdy by se kvůli ní dostal úhel φ do záporné hodnoty. Pokud tedy chci, aby se slunečnice pohybovala ve směru větru, musím nastavit proměnnou b na hodnotu 1, která určuje, že se bude úhel φ zvětšovat, případně hodnotu –1, aby se úhel zmenšoval.

Výpočet úhlu φ je také závislý na velikosti p_w. Jsou dvě možnosti, kterých může p_w nabývat. Hodnota φ se výsledně vypočítá

$$\varphi(t) = \begin{cases} tbp_w\pi & \text{pro } 0 \le p_w \le 1 \\ tb\left(\frac{p_w}{20}+1\right)\pi & \text{pro } p_w > 1 \end{cases}, \quad (6)$$

kde t je časová konstanta. Pokud je $\varphi < 0$, což znamená, že proměnná $b = -1$, nastaví se proměnná b zpět na hodnota. Při použití této rovnice nám ale vzniká riziko, že se bude hodnota φ zvětšovat do nekonečna. Tento problém byl v aplikaci vyřešen tak, že pokud dosáhne hodnoty 2π, což je fázový posun funkce sinus, odečte se tato od úhlu φ.

5.5 Alfa průhlednost

Jelikož jsou v mé práci vyšší úrovně LOD tvořeny Billboardy a Sprity, tedy pomocí průhledných textur, je zde nutné použít alfa průhlednost a nastavit správnou prahovou hodnotu.

Otázka tvorby alfa kanálu a jeho podoby je již vysvětleno v kapitole 5.1.1. Průhlednost naší textury je určena přímo alfa kanálem. Jedná se tedy v podstatě o nový obrázek (viz obr. 10b), který je ve stupních šedi. V místech kde je na obrázku bílá barva, je výsledná textura zcela neprůhledná. Naopak v místech kde je barva černá, je textura naprosto průhledná. Pokud jsou hodnoty barev mezi těmito hraničními barvami, je textura více či méně průhledná v závislosti na světlosti tohoto odstínu. Prahová hodnota tento interval dále zmenšuje, protože specifikuje referenční hodnotu alfa barvy, vůči které se provádí testování jednotlivých pixelů, zda má být průhledný, takzvaný alfatest. Tato hodnota je automaticky nastavována na nulu. To znamená, že pouze naprosto černý pixel v alfa kanálu, který má prahovou hodnotu rovnu právě nule, odpovídá úplné průhlednosti v textuře. Pokud však nepoužíváme řazení vykreslovaných objektů, což by program mohlo zpomalovat, prolínají se v místech, kde jsou jejich pixely poloprůhledné, pixely z pozadí. To může být například další slunečnice, nebo barva z pozadí.

Problematika nastavení prahové hodnoty je vysvětlena v [12]. Proto jsem zde použil stejného postupu, protože řazení by nám program opravdu jen zpomalilo. Na zobrazení, kde se na okrajích neprůhledných částí textury bude vyskytovat co nejméně chyb. Podle [12] je ideální prahová hodnota, na kterou se testuje daný pixel textury, mezi 180 a 200. Z toho plyne, že úplně průhledná je textura již tam, kde alfa kanál má hodnotu mezi 180 a 200 (viz obr. 14). Jinak řečeno, průhlednou se stává již hodnota, která má v masce světle šedou barvu. Tím sice přestanou být neprůhledné okraje textury tak vyhlazené, ale vzhledem k tomu, že vykreslujeme Sprity a Billboardy ve větší vzdálenosti, tak nám to nevadí. Navíc na

nich nevznikají tak výrazné zobrazovací chyby. Přesto je však poznat, že zde tyto chyby vznikají v závislosti na nastavení prahové hodnoty (viz obr.15).

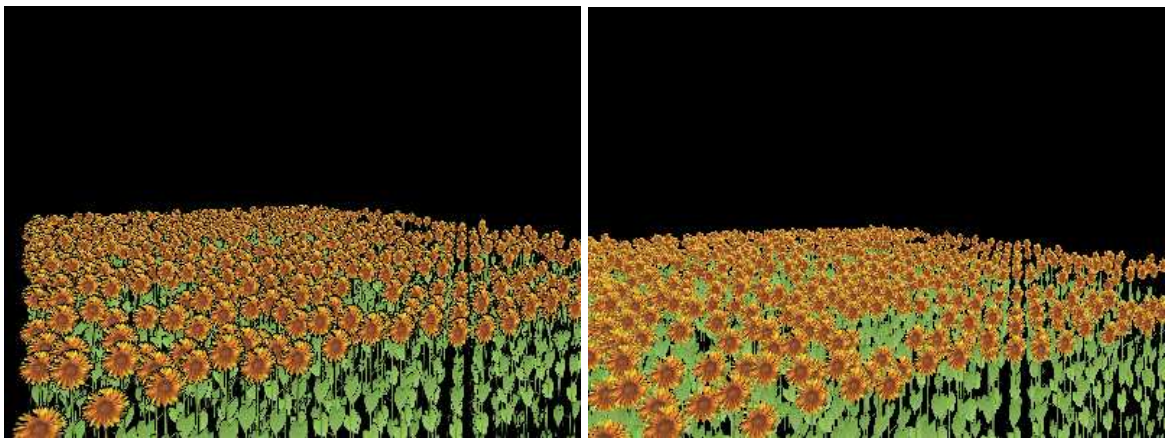
Obr 14:Jednotlivé nastavení prahové hodnoty zleva shora 160, 180, 0 a 255

Na výp. 2 je naznačeno, jak se postupuje při testu na prahovou hodnotu. Toto je součástí vykreslení nosného obdélníku.

```
device.RenderState.AlphaBlendEnable = true;     // Zapnutí průhlednosti
device.RenderState.AlphaTestEnable = true;      // Zapnuti testu pruhlednosti
device.RenderState.AlphaFunction = Compare.Greater;    // Porovnani jednotlivych pixelu
device.RenderState.ReferenceAlpha = 190; // Nastaveni prahove hodnoty
device.RenderState.SourceBlend = Blend.SourceAlpha;
device.RenderState.DestinationBlend = Blend.InvSourceAlpha;
// Vykresleni nosneho obdelniku
device.SetStreamSource(0, verticesF, 0);
device.SetTexture(0, TFront);
device.DrawPrimitives(PrimitiveType.TriangleStrip, 0, 2);
// Konec vykreslovani nosneho obdelniku
device.RenderState.AlphaBlendEnable = false;    // Vypnuti pruhlednosti
```

Výpis 2: Vykreslení nosného obdélníku, včetně nastavení prahové hodnoty

Dále je zde ukázáno, jaký vliv má na vykreslení prahová hodnota. Pokud se nastaví na hodnotu 1.0, chyby vzniklé na okrajích slunečnic jsou velmi výrazné (obr. 15a). Celá scéna je ponořena do temného odstínu. Pokud se však podíváme na tu samou scénu, ale s prahovou hodnotou 200 (obr 15b), slunečnice již mají přirozenou barvu.

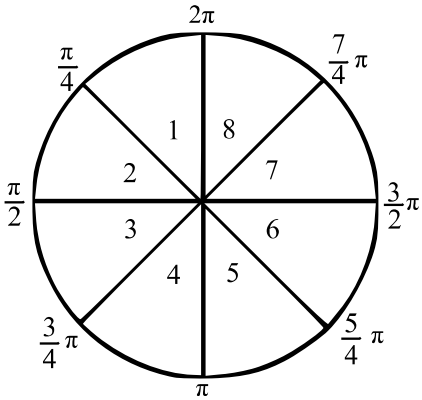


Obr. 15: Pohled na scénu s prahovou hodnotou nastavenou na hodnotu 1.0 (a) a na hodnotou 200(b)

5.6 Další urychlení

Provádění alfatestu je však výpočetně náročné, proto jsem nakonec zvolil jiný způsob. To, že je pole definované jako obdélník, nám poslouží k tomu, že můžeme provádět vykreslování pomocí úhlu pozorovatele, takzvané úhlové vykreslování. Pokud totiž vykreslujeme rostlinky v pořadí od konce směrem k pozorovateli, zaručuje nám to, že se rostlinky nebudou navzájem překrývat. Je to stejný efekt, kterého by bylo dosaženo, pokud by bylo použito řazení objektů.

Pohledový úhel nabývá hodnot 0 až 2π. Proto jsem tento úhel rozdělil do celkem osmi stejných sektorů (viz obr. 16). V každém z nich probíhá jiné pořadí vykreslování. Například pokud se dostane pohledový úhel do sektoru 1, tak vykreslování probíhá od levého zadního rohu pole a pokud se dostane pohledový úhel do sektoru 5, pole se bude vykreslovat od pravého dolního rohu.



Obr. 16: Rozdělení do sektorů podle pohledového úhlu

Narazil jsem však na problém, že když se začali vykreslovat Sprity, jednotlivé části se vzájemně překrývaly. To však není vhodné, proto jsem usoudil, že zkombinuji obě uvedené možnosti a ponechám vykreslování pomocí prostorového úhlu, akorát při vykreslování Spritů navíc použiji i alfatestu.

Dalším urychlením vykreslování je, že se jednoduše nebude vykreslovat nic, co není vidět, takzvané ořezávání. Je jasné, že pokud se bude vykreslovat naprosto vše, program to jedině zpomaluje. Ořezávání probíhá pomocí pohledové pyramidy. To znamená, že pomocí pohledového úhlu a přidané hodnotě na obě strany určím, které rostliny se nebudou vykreslovat. V tomto případě se jedná o ořezávání rostlin, které nejsou v úhlu pohledu $\pi/1.125$, což je 160°. Pole slunečnic se prochází, stejně jako tomu bylo u předchozí metody, pomocí pohledového úhlu. Pokud se narazí na slunečnici, která není vidět, je označena a následně se při vykreslování vynechá.

5.7 Výsledná knihovna

Všechny poznatky zde popsané je výsledně uchováno v knihovně sunflower.dll, která je výstupem této bakalářské práce. Ke komunikaci mezi mou knihovnou a průletovou aplikací slouží několik nejdůležitějších funkcí:

- void setWind(float wPower, float xpower, float zpower) – nastavení hodnot větru, tj. jeho síla, x a z směr šíření
- void setPosition(float xpos, float ypos, float zpos, float yaw) – nastavení pozice pozorovatele, vůči které se počítá vzdálenost pro LOD a natoční Billboardů
- void setSunflower(int possition, string filename) – nahrání Meshe z .x souborů
- bool LoadTextures(string TFrontFile, string TRightFile, string TMeshFile) nahrání textur potřebných k potažení Billboardů
- void Rotate(long ntime, float fps) – hlavní metoda, která způsobuje pohyb rostlin a výpočet LOD.

Pro vkládání jednotlivých modelů je potřeba připomenout, že jsem použil specifickou metodu skládání jednotlivých částí slunečnice. Každý kompletní model slunečnice se skládá ze tří částí a tyto kompletní slunečnice jsem zde použil tři. Nejdůležitější vlastností každé z částí je zřejmě jejich velikost podél osy y. Ta je přesně daná a musí být rovna jedné. V případě, že tomu tak nebude, je možné, že na sebe nebudou jednotlivé části přesně navazovat. Dále musí být tyto části vloženy v přesně daném pořadí. To je popsáno přímo zde,

v knihovně. Toto omezení je zde hlavně z důvodu vyměnitelnosti kterékoliv části modelované rostliny. Veškerá funkčnost této knihovny byla nakonec ověřena v aplikaci k tomuto účelu přímo vytvořené (viz. Aplikace).

6. Měření a výsledky

Před tím, než-li bude uvedena závěrečná sumarizace, představím zde ještě některá měření a pozorování, z nichž poté můžeme odvodit, do jaké míry bylo cílů, mnou vytyčených, dosaženo.

K vytvoření této práce jsem použil hardwarové a softwarové prostředky uvedené v tab. 2.

Počítač	CPU: AMD Sempron 1,8 GHz 3000+ RAM: 1 GB Grafická karta: Radeon X800 XT
Systém	Microsoft Windows XP Professional Verze 2002 Service Pack 2
Vývojové nástroje	Microsoft DirectX SDK (December 2005) Microsoft Visual C# 2005 Profesional Edition .NET framework 2.0

Tab. 2: Hardwarové a softwarové vybavení

Před provedením závěrečného testu jsem se pokusil ještě o dodatečná urychlení zobrazování. Použil jsem k tomu nástrojů na decimaci trojúhelníkových sítí, konkrétně Polygon Cruncheru 7, který spolupracuje s 3DStudiem Max. Snížil jsem počet trojúhelníků až na polovinu a to tak, že jsem manuálně snižoval jejich počet do té míry, kdy ještě nedošlo k poškození kvality modelu a počet trojúhelníků byl co nejmenší. Zrychlení se opravdu dostavilo, konkrétně bylo dosaženo dvojnásobného zrychlení. K tomuto účelu se v počítačové grafice používá proměnná FPS (frames per second), která udává počet snímků za sekundu. Její velikost určuje rychlost zobrazení.

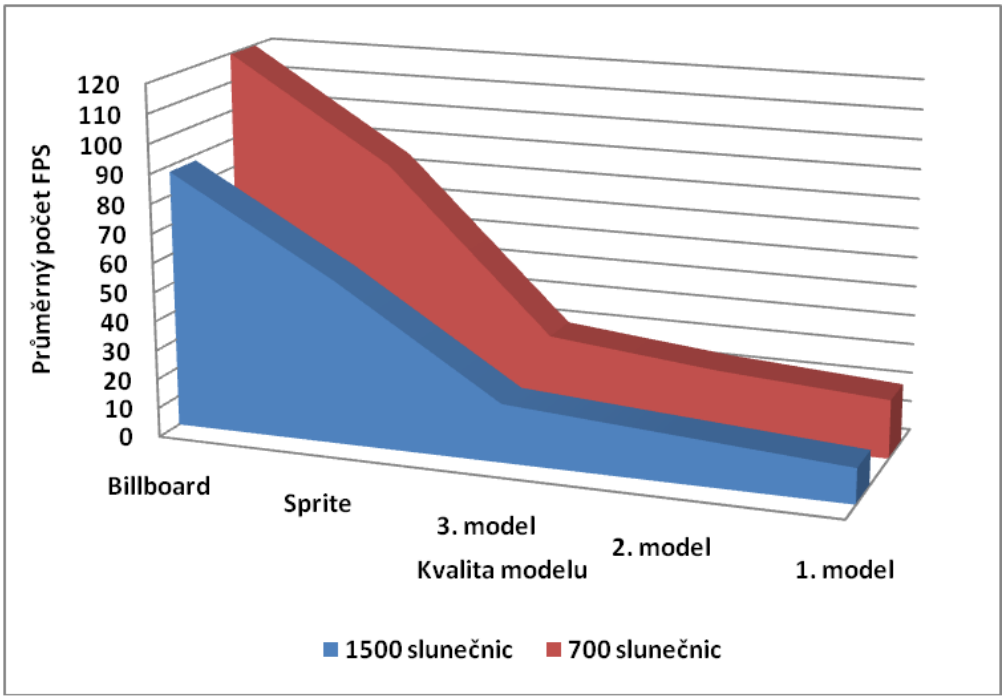
Ke zvětšení FPS by mělo dojít i použitím speciální funkce třídy mesh, funkce optimize. K nárůstu sice došlo, ale pouze na mobilní grafické kartě, která v mnou uvedeném hardwarovém vybavení není.

Jak je ale vidět na níže uvedeném grafu (obr. Graf1) a k němu příslušných tabulkách (tab. 3 a tab. 4), pokus o urychlení pomocí zmenšení počtu trojúhelníků nebyl dostačující. Přesto jsem dosáhl poměrně slušného výsledku.

Kvalita modelu	Poč. troj.	Prům. FPS
1. model	2821000	20
2. model	1458100	24
3. model	801500	30
Sprite	2800	85
Billboard	1400	120

Kvalita modelu	Počet troj.	Prům. FPS
1. model	6045000	12
2. model	3124500	16
3. model	1717500	20
Sprite	6000	56
Billboard	3000	88

Tab. 3 a 4: Hodnoty při vykreslení 700 a 1500 slunečnic



Obr. Graf1: Počet FPS pro jednotlivé druhy representace rostliny.

Protože daný počet FPS stále nebyl ideální, provedl jsem další měření při zapnutí metody LOD, abych mohl odhalit slabá místa a mohl se je pokusit odstranit. Nejprve byl odstraněn pomocí, již zmíněné, metody úhlového vykreslování alfatest, který má vliv pouze na billboardy a sprity. Dostavil se očekávaný nárůst FPS, tudíž jsem zjistil, že jedním z problémů byl opravdu alfatest. Jako další možný problém rychlosti aplikace bylo časté nastavování textur modelům. Když však došlo k jejich vypnutí, nemělo to na výsledný počet FPS, žádný znatelný dopad.

Dalším předpokládaným kamenem úrazu byl počet vykreslovaných trojúhelníků. Proto dalším testem bylo vykreslení pouze jednoho trojúhelníku při použití LOD a při nastavení matice pohybu jako při vykreslování nejkvalitnějšího modelu. Předpokládalo se, že počet FPS by měl být poměrně stálý, ale překvapivě tomu tak nebylo. Při vykreslování s maticí nejkvalitnějšího modelu byl, při pohledu na celé pole, počet FPS o 20 snímků vyšší, než při stejném pohledu s maticí pro LOD.

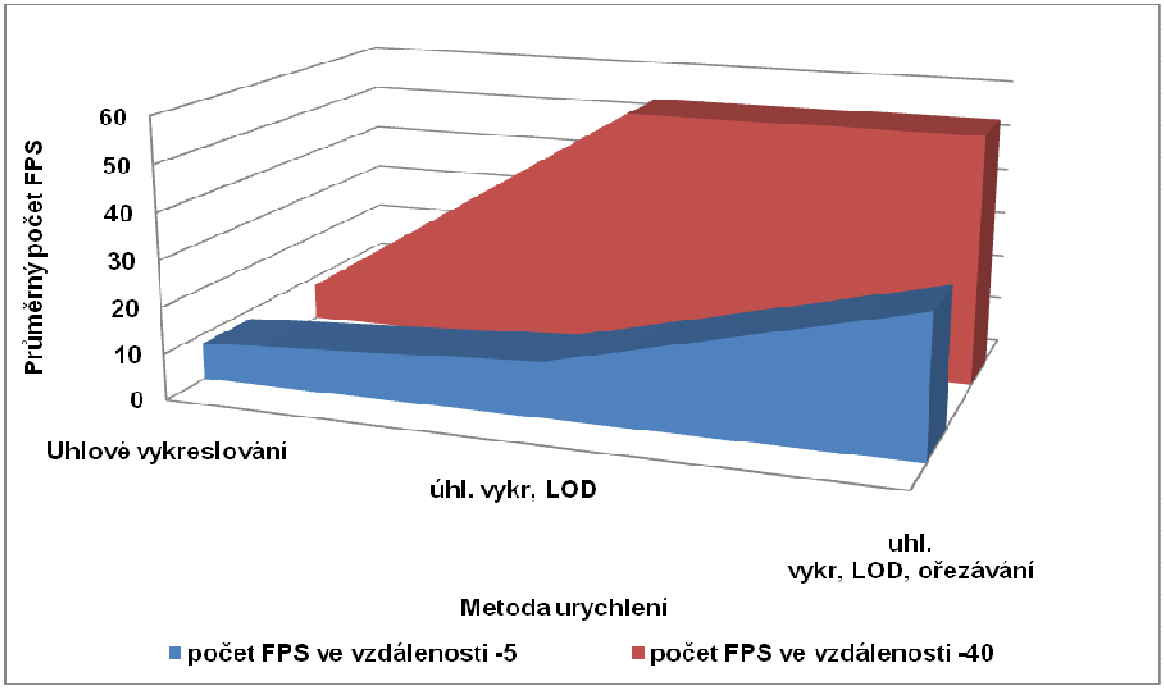
Proto jsem šel ještě dál a pokus jsem zopakoval, ale tentokrát bez vykreslování. Při pohledu na celé pole byl, při nastavení matice nejkvalitnějšího modelu, poloviční počet snímků, než při použití matice LOD. Tudíž jsem usoudil, že jedním z problémů je nastavovaní těchto matic pohybu.

Z tohoto důvodu jsem provedl předposlední test a to, že se bude nastavovat pouze matice translace a ostatní, jako rotace a scaling, teď nepoužiji. Opět jsem zkusil nevykreslovat nic a umístit pozorovatele tak, aby měl výhled na celé pole. Při porovnání FPS pro nastavení kompletní matice pohybu a nastavení pouze matice translace je rozdíl přibližně dvojnásobný. Bohužel v této práci musím nastavovat kompletní matice pohybu, protože by se musely použít postupy pro programování na grafické kartě, což je mimo rozsah této práce.

Závěrečný test se týkal použití všech variant urychlení, které práce poskytuje, tj. vypnutí alfatestu a použití vykreslování pomocí pohledového úhlu, LOD a v neposlední řadě úhlové ořezávání. Výsledky jsou shrnuty v tab. 5 a obr. Graf2.

	Úhlové vykr.	úhlové vykr., LOD	uhlové vykr., LOD, ořezávání
počet FPS ve vzdálenosti -5	8	12	30
počet FPS ve vzdálenosti -40	8	54	54

Tab. 5: Průměrný počet FPS pro 1500 slunečnic při zapnutí jednotlivých optimalizací



Obr. Graf2:Průměrný počet FPS pro 1500 slunečnic při použití jednotlivých metod urychlení.

Z výše uvedených poznatků je vidět, že i přes vysoké nároky na počet trojúhelníků na jednotlivých slunečnicích, dosáhneme použitím metod na urychlení vykreslování dostačujících výsledků.

7. Závěr

V této práci je shrnuto nemalé množství informací týkajících se postupů pro tvorbu rostlin, ať se již jedná o L-Systémy nebo o tvorbu modelů ve 3D studiu Max. Bylo vyzkoušeno několik různých způsobů tvorby modelů, z nichž nejlepší se ukázala být tvorba modelu po jednotlivých částech. Výhodou tohoto postupu je možnost dosahování vysoké variability rostlin, aniž by k tomu bylo zapotřebí velkého množství modelů. Jednotlivé části mohou být například natáčeny, prodlužovány, nebo jednoduše vyměňovány za jiné. Tomu všemu však logicky předcházelo vyřešení problému s propojením jednotlivých částí.

Součástí práce bylo také prostudování materiálů týkajících se metod pro tvorbu úrovní detailů, včetně dalších možností urychlení scény. Zjistil jsem, že při tvorbě úrovně detailů lze použít automatických metod založených na decimaci trojúhelníků, ale přesto jsem zůstal u diskrétních modelů rostlin, jelikož jsou jednodušší na implementaci a zároveň splňují požadovaný vizuální i výkonnostní efekt. Dále jsem se potýkal s problémy poloprůhlednosti u modelů tvořených nosnými obdélníky a také s určením hranice přepnutí mezi jednotlivými stupni modelů.

V neposlední řadě zde byl řešen problém realistického pohybu rostlin. Jako nejvhodnější rostlinu jsem zvolil slunečnici a musel jsem zhodnotit, jak se která část bude pohybovat. Mohl jsem zvolit přístup dokonalého fyzikálního modelu, ten by však ve výsledku nebyl příliš použitelný, jelikož by nemohl být v rozumném čase vypočítán. Z toho důvodu jsem se uchýlil k variantě vizuální. Pozoroval jsem skutečné pole slunečnic a později svůj výtvor s ním porovnával. Dospěl jsem k názoru, že pohyb mnou vytvořených slunečnic se dostatečně podobá pohybu slunečnic reálných.

Veškeré zde uvedené postupy a poznatky byly shrnuty v knihovně, kterou je možno připojit k aplikaci a díky níž se můžeme procházet polem slunečnic. Využitím této knihovny došlo k ověření, že zvolené postupy jsou vyhovující.

Na závěr jsem se však setkal s problémem, který se týká rychlosti vykreslování. Prostřednictvím metod na urychlení vykreslování, které zde byly užity, došlo k dostatečnému urychlení. Pokud bych však chtěl zvýšit výkon aplikace ještě více, musel bych použít jiných prostředků, než jsem nastudoval, například programování pro GPU pomocí shaderů, ale to je již mimo rozsah této práce.

Literatura a užité zdroje

[1] J. Žára, B. Beneš, J. Sochor, P. Felkner: Moderní Počítačová Grafika. Computer Press, 2004.

[2] A. Watt: 3D Computers Graphics. Pearson Education, 2000.

[3] P. Prusinkiewicz, A. Lindenmayer: The Algorithmic Beauty Of Plants. Springer – Verlag New York Inc.1990

[4] L-Systémy: http://www.root.cz/clanky/l-systemy-prirodni-objekty-i-umele-artefakty/

[5] ZPG: http://herakles.zcu.cz/education/zpg/

[6] Ondřej Klučka: Automatické vytváření úrovně detailu, ČVUT Praha FEL, 2007

[7] Tvorba modelů v 3D Studiu Max:
http://www.3dscena.cz/art/3dscena/3dsmax_strom2.html

[8] Jan Kříž: 3ds max – Hotová řešení. Computer Press 2005

[9] Paul Steed: Animace postav v 3ds max a character studiu. Computer Press 2004

[10] Tvorba modelů s kostmi: 3ds max 7, Discreet - Tutorials

[11] H. Hoppe: Progressive meshes. *SIGGRAPH 1996*.
http://research.microsoft.com/~hoppe/pm.pdf

[12] Vladimír Geršl: Modelování a vykreslování trávy pro herní aplikace, ZČU Plzeň, 2007

Příloha

Demonstrační aplikace

K procházení se polem slunečnic jsem vytvořil aplikaci, ke které byla připojena má knihovna. Je zde možnost nahrání vlastních jednotlivých částí rostliny, síly větru a jeho směr. Samozřejmostí je možnost pohybu pomocí myši a klávesnice. Dále je možné měnit směr a sílu větru během procházení.

Ovládání

Ovládání je podobné jako u mnoha akčních her. Používají se následující klávesy:

- A – úkrok na levou stranu
- D – úkrok na pravou stranu
- S – pohyb dozadu
- W – pohyb dopředu
- R – nastavení pozorovatele do pozice, která byla na začátku
- F1 – vykreslení drátěného modelu
- F2 – přičtení inkrementu k síle větru
- F3 – odečtení inkrementu od síly větru
- F4 – změna směru větru na x-ové ose
- F5 – změna směru větru na z-ové ose
- F6 – přepínání velikosti inkrementu mezi 0.1 a 10
- Esc – ukončení aplikace