VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

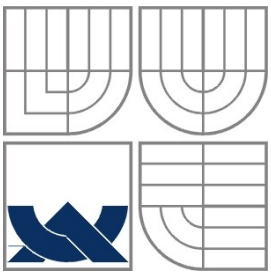
VYUŽITÍ VERTEX A PIXEL SHADERU PRO 3D ZOBRAZENÍ TRAVNATÉHO POROSTU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

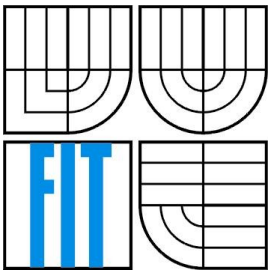
AUTOR PRÁCE
AUTHOR

MARTIN DOKOUPIL

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYUŽITÍ VERTEX A PIXEL SHADERU PRO 3D ZOBRAZENÍ TRAVNATÉHO POROSTU

VERTEX AND PIXEL SHADERS FOR 3D VISUALIZATION OF GRASS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN DOKOUPIL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PŘEMYSL KRŠEK, PH.D.

BRNO 2008

Abstrakt

Tato práce rozebírá problematiku tvorby trojrozměrného travnatého porostu v počítačové grafice. Popisuje vývoj zobrazení porostu v minulosti a také současný přístup k tvorbě. Obsahuje úvod do problematiky shaderů a jejich využití při zobrazení porostu. Okrajově popisuje práci s grafem scény, který byl využit pro implementaci. Detailně popisuje způsob vytvoření tří použitých úrovní detailů, které se využívají, a také způsob vytvoření pohybujícího se porostu. Porovnává rychlost jednotlivých úrovní detailů a zpracovává výsledky zobrazení implementovaného porostu.

Klíčová slova

Porost, tráva, stéblo, shader, vertex, pixel, úroveň detailů, graf scény

Abstract

This thesis describes the way to create three-dimensional visualization of grass in computer graphic. Contains progress of grass visualization from past to present. Contains shader intro and its use for visualization. Text mention scene graph which has been used for implementation. Text describes three levels of detail and implemetation of waving grass. Text compares frame speed for each level of grass and frame speed for complete visualized grass.

Keywords

Grass, blade, shader, vertex, pixel, level of detail, scene graph

Citace

Martin Dokoupil: Využití Vertex a Pixel shaderu pro 3D zobrazení travnatého porostu, bakalářská práce, Brno, FIT VUT v Brně, 2008

Využití Vertex a Pixel shaderu pro 3D zobrazení travnatého porostu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Přemysla Krška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Dokoupil
9.5.2008

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Přemyslu Krškovi, Ph.D. za jeho rady při tvorbě této práce. Zejména bych chtěl poděkovat za jeho motivaci k práci na mé bakalářské práci.

Obsah

1 Úvod

Počítačová grafika patří k rychle se rozvíjejícím odvětvím informačních technologií. Je snahou dosáhnout zobrazení co nejrealističtějších scén. Nejčastěji se počítačová 3D grafika využívá ve filmovém průmyslu a v současné době také u počítačových her. Každé z těchto odvětví využívá jiný přístup. Ve filmovém průmyslu se využívá předrenderované počítačové grafiky, kde je vytvořená scéna spočítána a zobrazena až později. Přitom proces renderování trvá většinou dlouhou dobu. Scéna se zobrazuje až po dokončení výpočtu všech snímků scény. V herním průmyslu se používá zobrazení v reálném čase, kdy se ve stejném okamžiku scéna přepočítává a zobrazuje. Real-time zobrazení je tedy mnohem náročnější na výkon počítače, zároveň není možné vykreslovat tak komplexní scény včetně všech detailů a je třeba dosáhnout kompromisu mezi kvalitou zobrazení a rychlostí. Porost bude zobrazován v reálném čase a proto je nutné brát tuto skutečnost při návrhu v potaz.

Tato práce se zabývá zobrazením travnatého porostu. Implementace je provedena pomocí toolkitu OpenSceneGraph, který je postaven na grafické knihovně OpenGL. Zobrazení pomocí OpenSceneGraph probíhá s využitím grafu scény. Je prováděno v reálném čase a je tedy nutné přistoupit ke kompromisům. Porost musí být možné zobrazovat na současných počítačích a zobrazený porost nesmí být příliš naročný na výpočet a zobrazení scény.

Druhá kapitola obsahuje úvod do problematiky shaderů, popis vertex a pixel shaderů, programovací jazyky shaderů. Dále obsahuje rozbor problematiky zobrazování porostu v minulosti a v současnosti, a využití shaderů pro zobrazení porostu. Také popisuje princip práce s grafem scény.

Ve třetí kapitole nalezneme implementaci porostu: popis jednotlivých úrovní porostu, popis implementace těchto úrovní, a také implementaci pohybu porostu. Popisuje využití shaderů pro zobrazení.

Čtvrtá kapitola zpracovává dosažené výsledky implementace. Porovnává jednotlivé úrovně datailů a jejich rychlost zobrazování. Dále zpracovává výsledky zobrazení kompletního porostu v závislosti na vlastnostech porostu.

Pátá kapitola v závěru zhodnocuje práci a dosažené výsledky. Také popisuje možnosti dalšího pokračování.

2 Rozbor problematiky

Tato kapitola se zaměřuje na problematiku související s tvorbou travnatého porostu. Postupně popisuje základní informace o shaderech, jazyky pro jejich programování a tvorbu programů v jazyce GLSL. Dále popisuje princip tvorby grafu scény s pomocí toolkitu OpenSceneGraph.

2.1 Zobrazení porostu

2.1.1 Historie zobrazování

V počátcích využívání 3D počítačové grafiky nebylo možné zobrazovat reálně vypadající travnatý porost. Neexistovaly grafické akcelerátory a počítání grafiky na procesoru bylo velice náročné. Bylo možné vytvářet pouze jednoduché objekty a nebyly možnosti pro využití textur. Tráva se znázornila pomocí zeleně obarvených polygonů a kladla vysoký důraz na představivost pozorovatele. Po příchodu textur se tyto polygony otexturovali jednoduchým travnatým vzorem. Tento způsob se dlouhou dobu používal u počítačových her a to do doby, než se dostatečně zvýšil výkon grafických karet. Dalším postupem bylo použití svislých polygonů (bilbordů), na kterých byly naneseny stébla pomocí textury s alfa kanálem. Pro zobrazení porostu byly tyto bilbordy umístěny po celé ploše. Tento způsob je v dnešní době ještě využíván u počítačových her.

Obr. 2.1 Hra Operace flashpoint (2001), převzato z [1]

Obr. 2.2 Hra FarCry (2004), převzato z [2]

2.1.2 Zobrazení v současnosti

Moderní hry s důrazem na realistický vzhled ještě používají bilbordy zobrazující porost. Kombinují tento způsob s využitím 3D modelů. Část porostu je vytvořena s využitím bilbordů, a mezi těmito bilbordy jsou vkládány trojrozměrné modely porostu. I přes vysoký výkon dnešních grafických karet nelze vytvořit celou louku vykreslením jednotlivých stébel trávy. Vzhledem k nemožnosti vykreslovat s dostatečně jemným rozlišením a vzhledem k nedokonalosti lidského oka není třeba vzdálená stébla vytvářet. Vzdálená stébla je třeba pouze zobrazit tak, aby se pozorovateli jevila jako stébla skutečná. Je třeba vytvořit několik úrovní detailů, které se budou zobrazovat v jednotlivých vzdálenostech. Základní princip zobrazení pochází z [4]. V nejbližší vzdálenosti je třeba zobrazovat jednotlivá stébla trávy, ve střední vzdálenosti již není třeba zobrazovat jednotlivá stébla. V této vzdálenosti bude dostatečné zobrazit pouze svislý bilbord, na který bude namapována textura porostu. V nejvyšší vzdálenosti není pozorovatel schopen rozlišit jakékoliv detaily. Pozorovatel je schopen rozlišit pouze barvy. Bude tedy stačit pouze zobrazení vodorovného bilboardu, který bude otexturován. Textura bude obsahovat porost při pohledu shora.

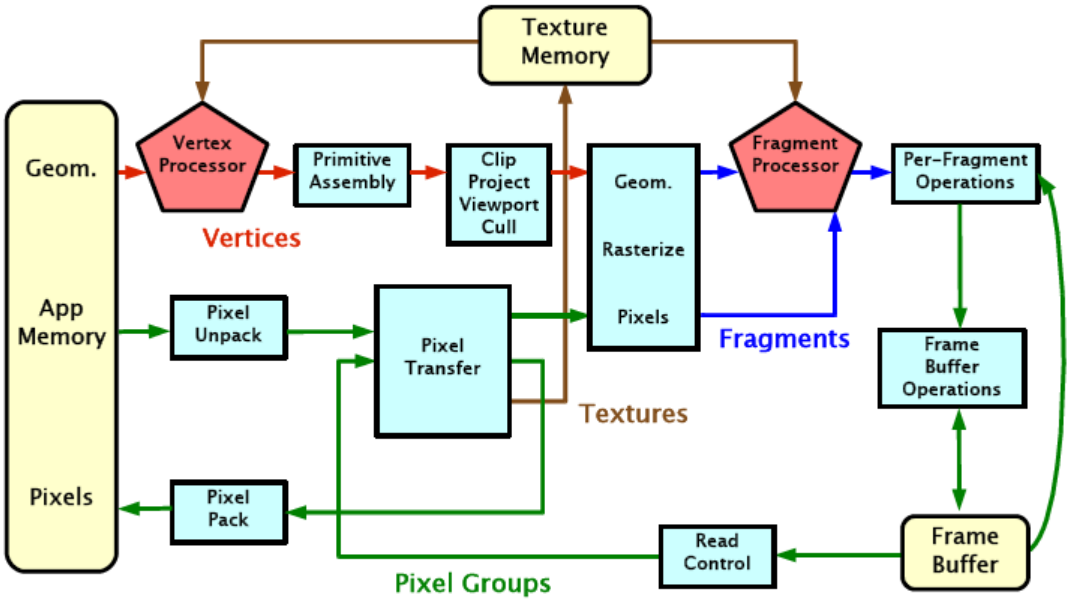
Obr. 2.3 Hra Crysis (2007), převzato z [3]

2.1.3 Animace pohybu porostu

Animace pohybu je ideálním využitím vertex a pixel shaderů. Tvorba pohybu porostu je popsána v [5]. Vertex shadery slouží k animaci pohybu nejbližších stébel trávy. Jednotlivé vektory se vychylují podle programu. Výhodná je animace pohybu pomocí funkce sinus. Pohyb s touto funkcí se blíží skutečnému pohybu stébel ve slabém větru. Vektory ve vyšších částech tvořících stéblo se vychylují s vyšší amplitudou než vektory ve spodní části stébla. Ve střední vzdálenosti vertex shader jednotka vychyluje horní strany bilbordu, který se opět bude pohybovat s funkcí sinus. Využitím pixel shaderu lze změnit barvu od světlejší zelené po tmavší a tím simulovat různé natočení stébel ke světlu. Nejvyšší vzdálenost nevyužívá vertex shader jednotky. Ta nemůže být animována. S využitím pixel shaderu lze měnit barvu stejným způsobem jako u střední vzdálenosti.

2.2 Shadery

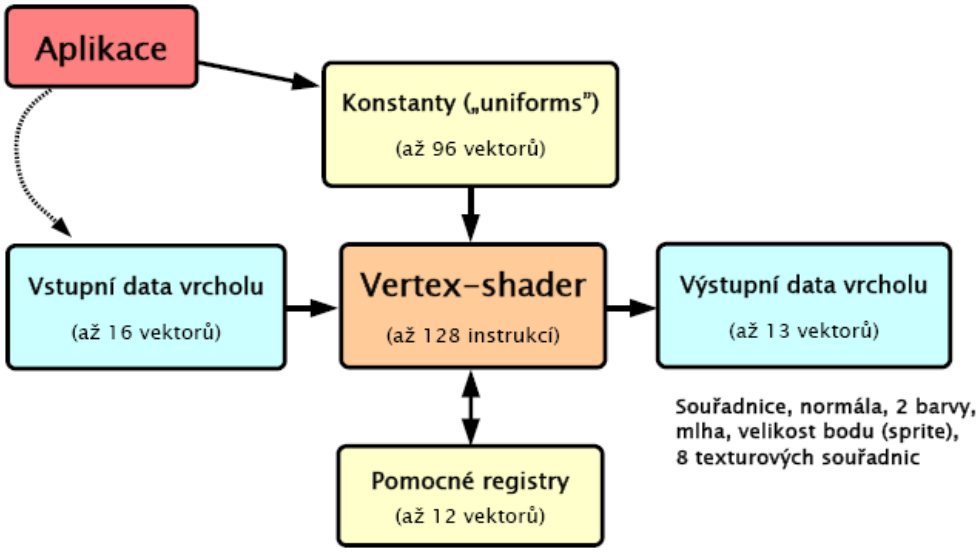
Shadery umožňují vytvořit realističtější a detailnější grafické scény, viz [8]. Ty byly dříve renderovány pomocí rozsáhlých serverových polí. S příchodem novejších, výkonějších grafických karet a OpenGL verze 2 bylo možné pro zobrazování využívat shadery. Shadery jsou programovatelné jednotky grafické karty. To umožňuje vývojářům kontrolovat část operací renderovacího procesu grafické karty. Operace v shader jednotce jsou náročné na výpočet a vyžadují speciální konstrukce, které jsou neefektivní pro použití mikroprocesoru. Umístění shader jednotky do grafického jádra umožňuje specializovat operace takovým způsobem, jakým by to nebylo u mikroprocesoru možné. Použitím shader jednotek lze získat vyšší výkon potřebný pro detailnější scény. V současnosti je možné naprogramovat 2 části: vertex shader a pixel shader.



Obr. 2.4 Pipeline v opengl 2.0, převzato z [6]

2.2.1 Vertex Shadery

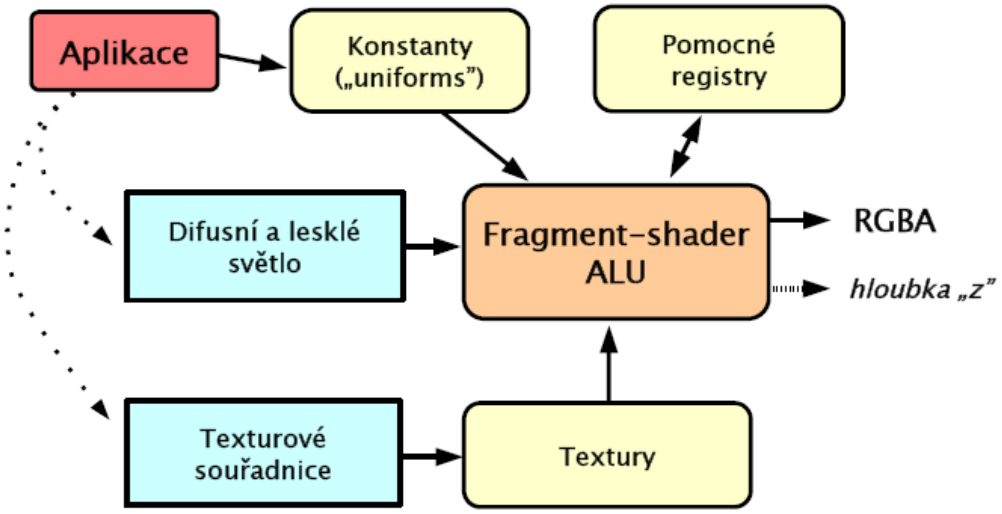
Každý vektor je definován souřadnicemi ve směru os x,y,z. Vektor může být definován také dalšími daty, jako je informace o barvě, alfa kanálu, informace o textuře nebo osvětlení. Vertex jednotka funguje jako černá skříňka. Na vstup do této skříňky vložíme data vektoru. Tam jsou následně zpracovány a na výstupu ze skříňky se objeví vektor s novou pozicí a případná další transformovaná data. Vektor může také projít vertex jednotkou beze změny. Shader jednotka neumí přidávat nebo ubírat vektory, ale pouze modifikuje vstupní vektory. Transformace celých objektů se provede postupným přiváděním jednotlivých vektorů na vstup vertex jednotky a jejich modifikací. Shadery umožňují přeprogramování shader jednotky. Tím lze pro každý objekt scény napsat vlastní program, podle kterého se budou vektory přepočítávat. Dalším využitím vertex shader jednotky může být animace postav, enviromentalní efekty, procedurální deformace, morfing, motion blur – rozmazání, efekt rybího oka a další. Další informace viz [8].



Obr. 2.5 Zjednodušený princip vertex shader jednotky, převzato z [7]

2.2.2 Pixel (fragment) Shadery

Zobrazovaná scéna se na výstupu z grafické karty skládá z pixelů. Každý pixel je nutné vykreslit, obarvit, osvětlit. Scéna se může skládat z několika milionů těchto pixelů. To činí výpočet pixelů náročným. Dříve bylo možné používat pouze textury s nízkým rozlišením, které se mapovaly na poměrně velké polygony. Textury tedy byly rozmazané. S příchodem pixel shaderů lze používat detailnější textury a také nastavovat osvětlení a barvu každého pixelu. Pixel shadery umožňují ukládat více textur jako výšková a normálová mapa, což umožňuje otexturovat model plastickým povrchem. Dalším využitím pixel shaderu je zobrazování 3D scén s komiksovým vzhledem. Více na [9].



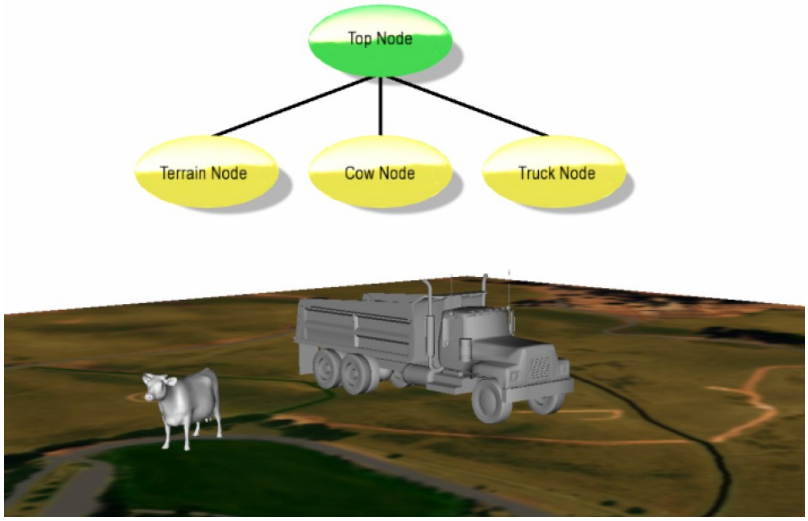
Obr. 2.6 Zjednodušený princip pixel shader jednotky, převzato z [7]

2.2.3 Programovací jazyk shaderů

Pro implementaci byla zvolena knihovna OpenGL. Ta umožňuje použití různých jazyků pro shadery [7]. Jedním jazykem je jazyk Cg, jehož specifikace je od společnosti Nvidia. Dalším jazykem je GLSL (GL shading language), který je přímo součástí knihovny OpenGL. Oba jazyky mají syntaxi podobnou jazyku C a jsou si podobné. Liší se pouze v detailech. GLSL však využívá toho, že je vytvořen jako součást OpenGL a jako rozhraní s touto knihovnou využívá přímo tento jazyk. Program v jazyce Cg je nutné převést do zdrojového kódu symbolických instrukcí a tento výstup využít pro komunikaci. Díky tomu, že není jazyk Cg pevně svázán s knihovnou OpenGL, lze tento jazyk využít i pro jiné knihovny jako třeba DirectX společnosti Microsoft. Možné je také psát program pro shader přímo pomocí symbolických instrukcí. Pro implementaci bakalářské práce byl zvolen jazyk GLSL z důvodu své provázanosti s knihovnou OpenGL.

2.3 Graf scény

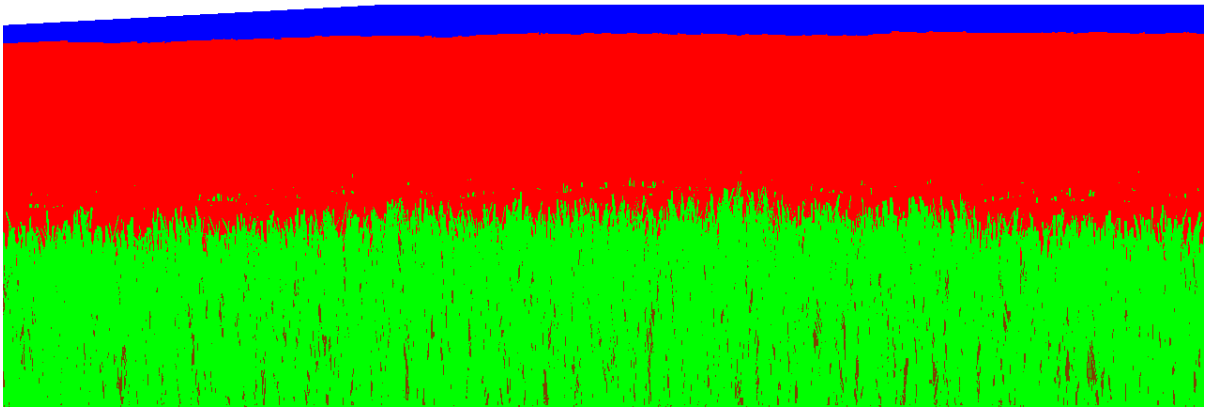
Porost bude implementován pomocí toolkitu OpenSceneGraph. Práce s tímto toolkitem se provádí pomocí grafu scény. Graf scény je hierarchický strom organizující prostorová data pro jejich efektivní vykreslování, viz [10]. Je složen z uzlů a obsahuje jeden kořenový uzel. Každý uzel může obsahovat synovské uzly. Na konci stromu jsou listy, jimiž jsou samotné objekty k vytvoření. Na jeden objekt může ukazovat více uzlů. Uzel stromu může pouze spojovat několik uzlů synovských, ale také může mít další funkci. Mezi tyto funkce patři transformace, která posunuje svoje synovské uzly podle zadání. Dále LOD podle nastavení vykreslí svoje synovské uzly pouze v zadané vzdálenosti od kamery. Uzel může mít také funkci přepínače, který podle svého stavu svoje synovské uzly vykreslí nebo ne.



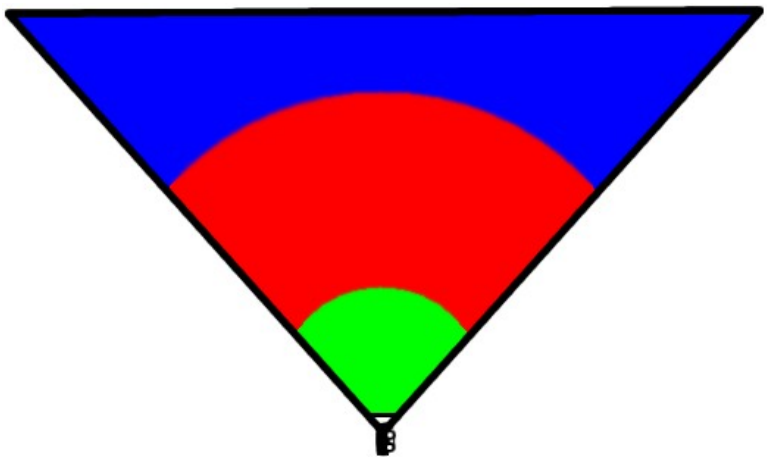
Obr. 2.7 Příklad jednoduchého grafu scény, převzato z [10]

3 Implementace

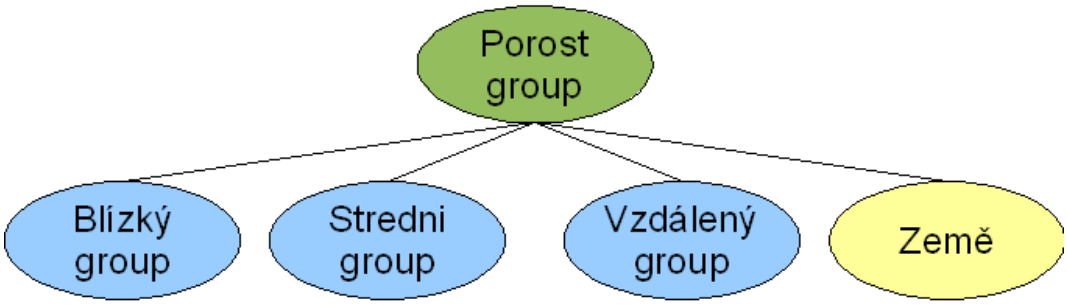
Porost je rozdělen na různé urovně detailů, které se zobrazují v jednotlivých vzdálenostech. Nejbližší vzdálenost je nejdetailnější, nejvzdálenější má naopak nejméně detailů. Porost může mít různou hustotu. Je vytvořen podle obrázku, který určuje, pozici porostu ve scéně a jeho hustotu. Porost je rozdělen na políčka, každé políčko určuje jeden bod obrázku. Barva bodu obrázku udává hustotu porostu na daném políčku. Obrázek je černobílý, černá nulová hodnota (černá barva) znamená bez porostu a nenulová hodnota udává hustotu na políčku. Maximální hodnota (bílá barva) udává maximální hustotu. Maximální hodnota odpovídající bílé barvě je nastavena již při překladu a ostatní hodnoty hustoty se od ní odvíjí. Rozměry obrázku udávají rozměry scény. Podle rozměrů obrázku je pod celým porostem vytvořen polygon země, který je zobrazen stále bez ohledu na vzdálenost od pozorovatele. Tento polygon země má barvu hlíny pod porostem a slouží jako podklad. Jednotlivé úrovně detailů porostu jsou umístěny na tento polygon země.



Obr. 3.1 Zobrazené tři úrovně detailů



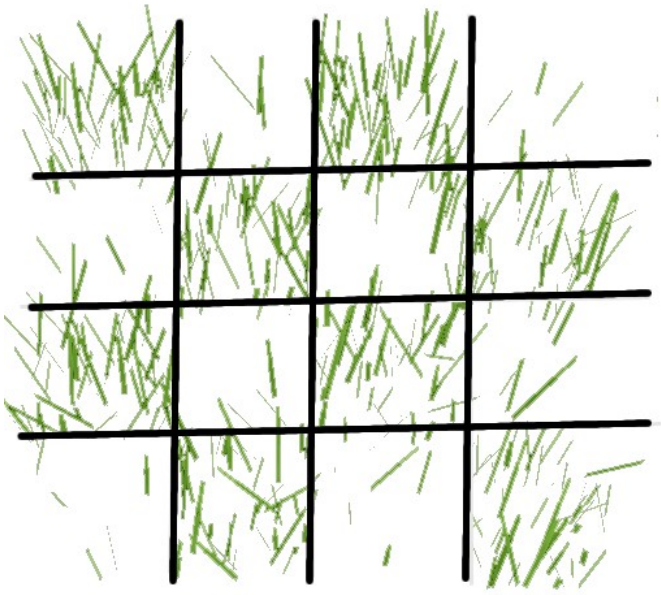
Obr. 3.2 Pohled shora na jednotlivé úrovně detailů



Obr. 3.3 Základní schéma porostu

3.1 Nejbližší vzdálenost

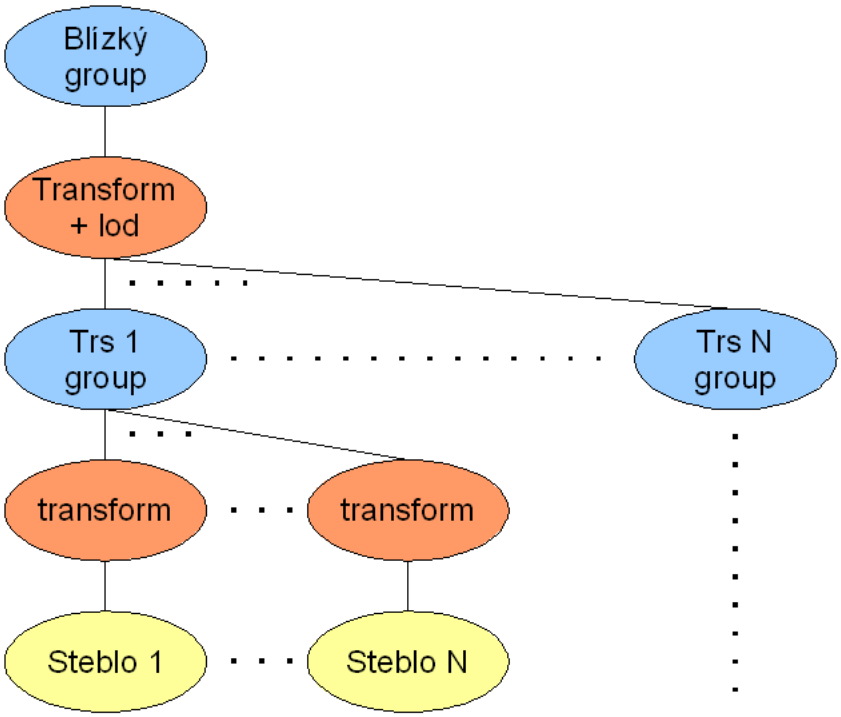
V této vzdálenosti musí být zobrazení co nejdetailnější. Proto jsou zde zobrazovány nejvyšší detaily a tedy jednotlivá stébla. Porost v této vzdálenosti je složen z trsů trávy. Při použití jednoho trsu trávy by pozorovatel objevil, že se neustále opakuje. Je tedy nutné vytvořit větší počet těchto trsů, ale nesmí být jejich počet příliš vysoký. Tím by se výrazně zvýšily nároky na paměť. Tyto trsy lze také pootočit, čímž se dosahne jiného vzhledu dvou stejných sousedních trsů. Protože vytvořené trsy mají čtvercový tvar je možné je pootočit pouze o 0° (původní trs), 90°, 180° a 270°. Při pootočení o jiný úhel by na sebe trsy nenavazovaly a objevovaly se nepřirozená prázdná místa. Použitím čtyř možností otočení získáme čtyřnásobný počet trsů, než jsme původně vytvořili. Při vytvoření porostu se se pro každé políčko porostu náhodně vybere jeden trs z předem vytvořených a náhodně se vybere otočení. Pro vytváření porostu s různou hustotou je vytvořeno více skupin trsů. Podle daného políčka se pak z vybrané skupiny použije jeden z trsů.



Obr. 3.4 Část trávníku, kde se pravidelně střídá nejhustší a nejřidší trs

Protože je zobrazení jednotlivých stébel náročné na výpočet a nebylo by možné zobrazit takové množství polygonů do velké vzdálenosti, je možné vzdálenější stébla vytvořit pouze z jednoho polygonu. Tyto stébla tedy nebudou tak náročná na výpočet a mohou být zobrazeny do vyšší vzdálenosti. Stébla nebudou zaoblená, ale pouze nakloněná. K přepnutí dojde při oddálení, kdy již nebude na první pohled změna viditelná. Méně detailní stébla se vytvoří stejným způsobem jako stébla detailnější.

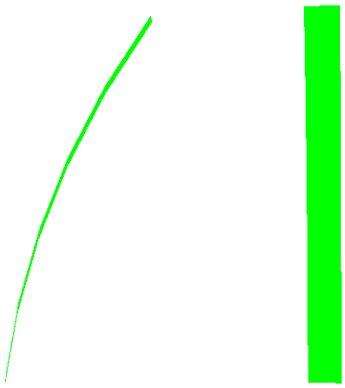
Obr. 3.5 Srovnání vícepolygonových stébel a stébel složených z jednoho polygonu



Obr. 3.6 Schéma nejbližšího porostu

3.1.1 Vytvoření stébla trávy

Travnatý porost se neskládá ze stejných stébel. Je tedy nutné vytvářet stébla s různými rozměry. Stéblo má dva důležité rozměry, kterými jsou výška a šířka stébla. Stébla mají velice malou tloušťku, a proto je možné ji zanedbat. Aby bylo možné stébla ohýbat, je nutné tato stébla vytvořit z několika polygonů. Počet polygonů by neměl být příliš nízký, aby stébla nevypadala hranatě a naopak nesmí být ani příliš vysoký, protože velký počet polygonů zpomaluje výpočet. Je tedy nutné najít nejnižší možný počet polygonů, kdy při ohnutí stéblo vypadá stále dobře. Stéblo trávy má tvar pásku, který se u vrcholu zužuje do špičky. Je vhodné využívat polygony tvaru obdelníku, ze kterých se tento pásek jednoduše složí.Jedna strana tohoto obdelníku je stejná jako šířka stébla a druhá strana polygonu se vypočítá. Vypočítá se jako podíl výšky stébla a počtu polygonů, ze kterých se stéblo skládá. Rozměry všech polygonů jednoho stébla budou stejné.



Obr. 3.7 Pohled na stéblo bez textury složené z pěti polygonů, pohled z boku a zepředu

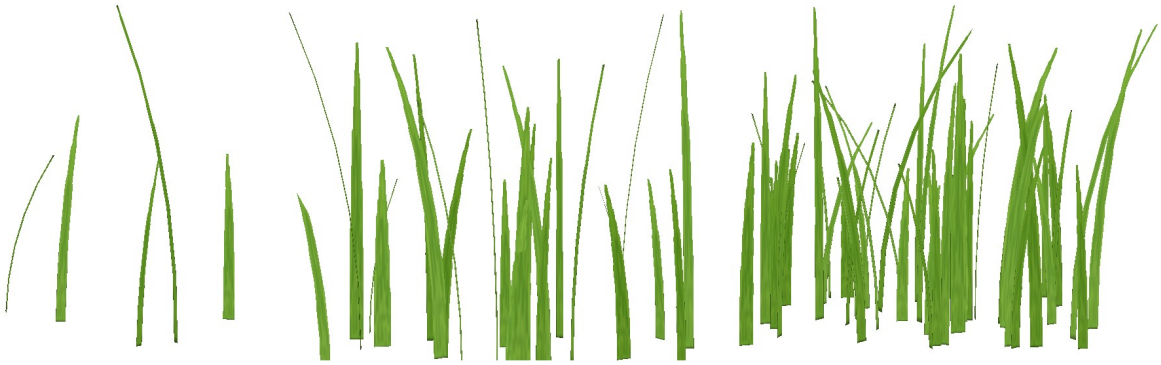
Zůžení stébla do špičky je možné vytvořit pomocí polygonů, ale tento způsob by byl složitější. Vhodnějším způsobem je využití průhledné textury, na které bude nakreslené stéblo a okolí bude průhledné. To docílíme použitím textury s alfakanálem.

Obr. 3.8 Textura stébla, její alfakanál a výsledné stéblo

Tím, že se využije textura, již nebude nutné používat barvu polygonu. Tato barva je složená ze čtyř složek a tyto složky lze využít pro shadery. Je možné shaderům tímto způsobem poslat různé informace. Mezi tyto informace patří frekvence, s jakou se budou stébla pohybovat, a také s jakou počáteční fází se budou pohybovat. Stébla jsou vytvořená stojící vzpřímeně vzhůru. O ohýbání stébel se postará až vertex shader jednotka.

3.1.2 Vytvoření trsu trávy

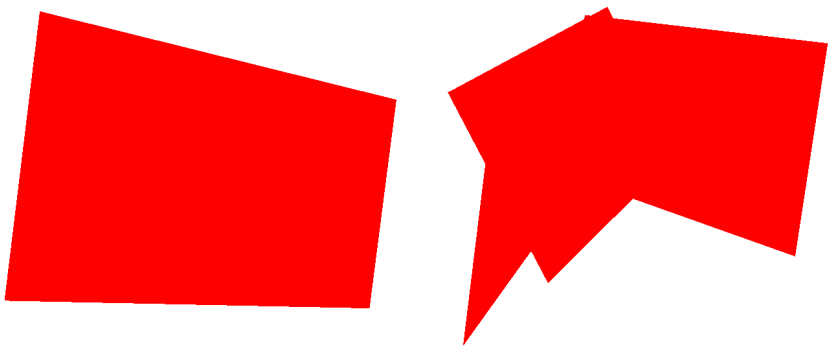
Trs trávy je složený z několika různě vysokých a širokých stébel. Stejným způsobem se vytváří také zde. Trs má tvar obdelníku nebo čtverce a je udán rozměry a hustotou. Trs je vytvořen generováním různých stébel, které jsou náhodně umístěny uvnitř tohoto čtverce. Zároveň je stéblo náhodně pootočeno. Travnatý porost může být různě hustý. Porost může být složený z řídce umístěných stébel, ale také může být složen z těsně umístěných stébel. Čím těsněji jsou stébla umístěna, tím hustější je porost. Vygenerováním vyššího počtu stébel lze dosáhnout vyšší hustoty. Hustota tohoto trsu udává kolik stébel se na tomto poli nalézá. Vyšší hustota znamená hustější porost.



Obr. 3.9 Trsy trávy s různou hustotou

3.2 Střední vzdálenost

Ve střední vzdálenosti je nutné snížit počet vykreslovaných polygonů, proto je třeba zobrazit několik stébel na jednom bilbordu. Jediný bilbord by byl dobře viditelný pouze z jedné strany. Vhodné je použití více než jednoho bilbordu. Naopak počet musí být menší, protože při vyšším počtu bilbordů by tento způsob přišel o výhodu jednoduchosti pro výpočet. Bilbordy jsou rozmísťovány po políčkách podobně jako trsy v nejbližší vzdálenosti. Aby nebyla vidět pravidelná síť bilbordů, je možné je náhodně umísťovat uvnitř políček. Ke sledování bilbordů z různých úhlů je třeba každý pootočit. Toto pootočení se provede náhodně. Tím budou bilbordy viditelné ze všech stran. Jednotlivá stébla porostu se ohýbají, u bilbordu je jedinou možností jeho naklonění. O nahnutí a pohyb bilbordu se postará vertex shader jednotka.

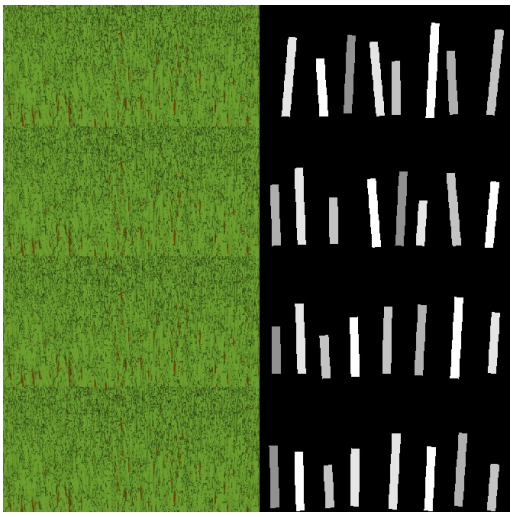


Obr. 3.10 Jeden bilbord a více různě natočených bilbordů

Tento bilbord využije pro zobrazení porostu průhlednou texturu tak, aby byla viditelná stébla. Textura musí vypadat podobně jako porost v bližší vzdálenosti pro méně viditelný přechod mezi vzdálenostmi. Kdyby byl přechod mezi nejbližší a touto střední vzdáleností skokový, pozorovatel by tuto změnu na první pohled viděl. Je tedy možné tento přechod provést postupně, kdy se nejdříve zobrazí jeden bilbord a se zvyšující vzdáleností se přidávají další bilbordy a jednotlivých stébel bude naopak ubývat.

Obr. 3.11 Jeden bilbord a více různě natočených bilbordů po otexturování

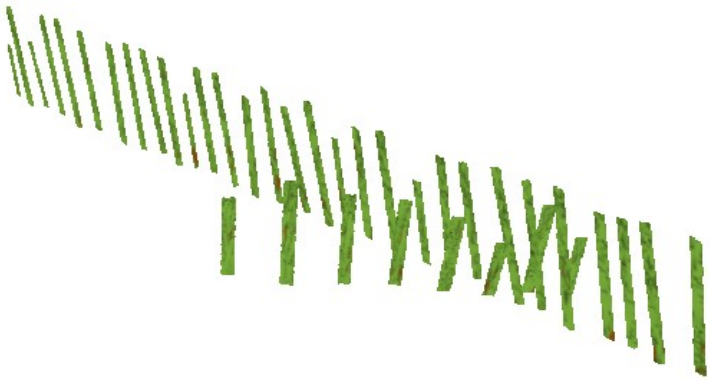
U jednotlivých stébel je vytvoření porostu s různou hustotou jednoduché. U střední vzdálenosti je vytvoření porostu s různou hustotou složitější. Lze to provést zvýšením nebo snížením počtu bilbordů. Toto řešení může výrazně zvýšit počet vykreslovaných polygonů a tím ztratit výhodu použití těchto bilbordů. Lepším řešením je využití alfakanálu. Textura s alfakanálem je změněna tak, že nevyužívá pouze hodnoty průhledná a neprůhledná, ale jednotlivá stébla mají různou hodnotu alfakanálu. Stébla se potom zobrazují s pomocí prahování. Pokud je hodnota alfakanálu vyšší než práh, stéblo se zobrazí, pokud je nižší nezobrazí se. Různá hustota se tedy bude provádět změnou tohoto prahu. Čím vyšší hustotu bude porost mít, tím nižší bude hodnota prahu. Aby nevypadal celý porost u bilbordů stejně, je vytvořeno více textur. Při každém vytvoření bilbordu se pak náhodně vybere jedna z textur. Všechny textury je možné vložit do jednoho obrázku, a tím se přesune do grafické karty textura pouze jednou. Bilbord potom vybírá texturu podle souřadnic, které získá při vytvoření.



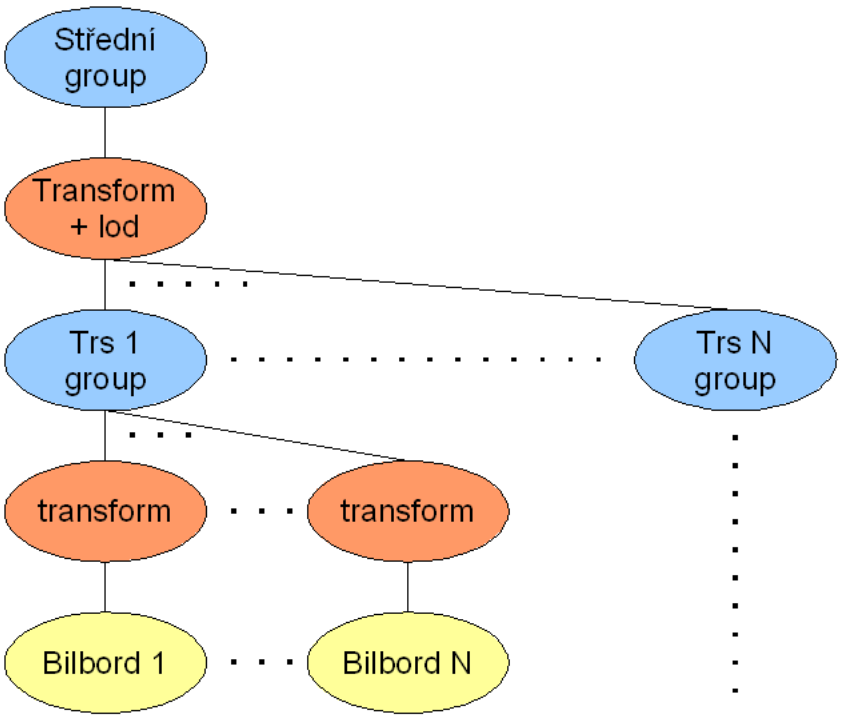
Obr. 3.12 Textura pro bilbordy s alfakanálem

Pokud by byl počet stébel na textuře příliš vysoký, mohla by stébla vedle sebe splynout do jednoho a ve výsledku by se porost podobal obarvenému obdelníku, což by nevypadalo přirozeně. Pokud je hustota porostu nad 50% maximální hustoty, je vytvořeno více bilbordů než u nižších hustot. Prahová hodnota alfakanálu se poté posune, aby výsledný porost odpovídal požadované hustotě.

Aby bylo možné bilbordy zobrazovat do ještě vyšší vzdálenosti, je možné využít další úrovně. Zde budou opět bilbordy, ale budou delší. Původně tyto bilboardy zabíraly jedno políčko. Tato políčka lze spojit a bilboardy budou umísťovány v této skupině. Šířka bilboardu se změní z šířky políčka na šířku skupiny. Tím lze ušetřit počet zobrazených polygonů. Je nutné najít vhodnou velikost, aby se využití této úrovně vyplatilo. Čím větší bude počet políček, přes které je umístěn bilboard, tím menší bude jejich celkový počet a vyšší úspora při vykreslování. Pokud by se použily bilboardy přes velký počet políček, pozorovatel by poznal změnu ve stylu umisťování objektů a porost nevypadal pravidelně. Je tedy nutné najít vhodnou velikost těchto vzdálenějších bilbordů. Způsob vytvoření je stejný jako u bilbordů menších.



Obr. 3.13 Porovnání původního a širšího bilbordu

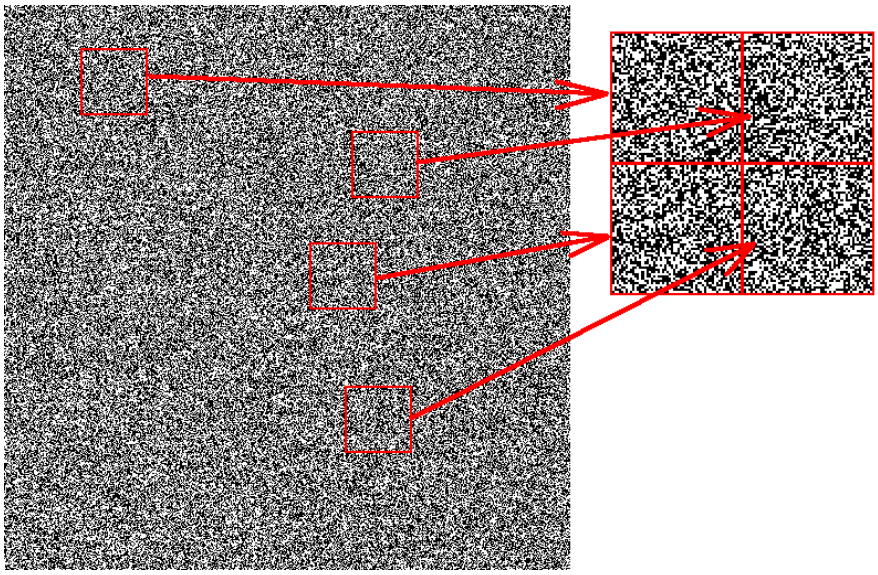


Obr. 3.14 Schéma střední vzdálenosti

3.3 Nejvyšší vzdálenost

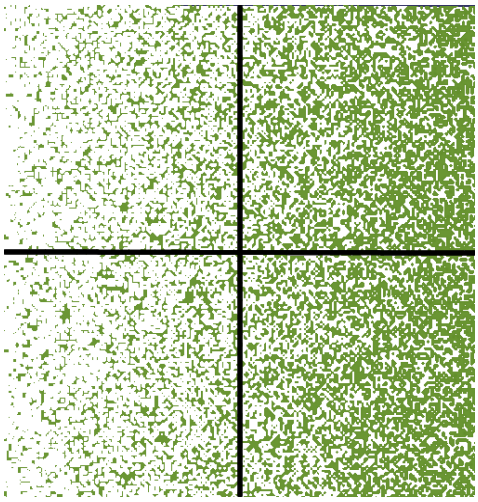
V nejvyšší vzdálenosti je dostačující zobrazení pouze textury porostu. Tato textura je umístěna na zemi horizontalně s povrchem. U této textury lze také využít alfa kanál. Textura musí mít takový vzhled, aby navazovala na blíže zobrazený porost. Zároveň musí políčka tohoto vzdáleného porostu navazovat vzájemně na sebe.

Porost s různou hustotou se provede podobně jako u střední vzdálenosti. To znamená pomocí různých hodnot alfakanálu textury a následným prahováním této hodnoty. Vzdálený porost se může na první pohled jevit jako šum. Toho lze využít vygenerováním zašuměné textury. Alfa kanál je vytvořen pomocí šumu. Tento je následně prahován a podle výsledku je bod průhledný nebo se zobrazí barva porostu. Šum lze vytvořit vygenerováním nebo vytvořením obrázku. Generování šumu je náročné na výpočet, a proto je vytvořen obrázek. S použitím dostatečně velkého obrázku lze celý vzdálený porost vytvořit nepravidelně, a to náhodným vybráním částí obrázku. Téměř vždy bude vycházet jiná část obrázku v závislosti na generování souřadnic.



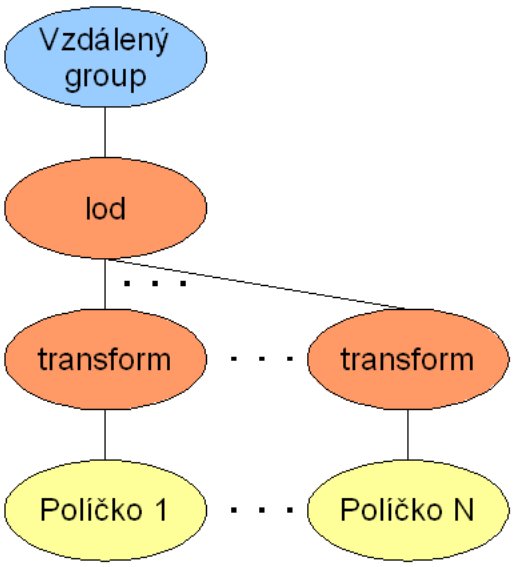
Obr. 3.15 Výběr textury šumu pro vzdálená políčka

Šum na sebe v každém případě navazuje a nejsou tedy vidět hranice polygonů. Tato návaznost není vidět pouze pokud mají sousední políčka stejnou hustotu. Při různých hustotách je hranice viditelná. Proto je třeba vypočítat pro každý vrchol polygonu hustotu v tomto místě a uložit ji. Sousední vrcholy budou mít stejný výsledek výpočtu hustoty daného místa. Hustota se tedy uvnitř polygonu mění v závislosti na pozici. Tím nejsou jasně viditelné hrany mezi sousedními polygony.



Obr. 3.16 Čtyři pole vzdáleného porostu s rostoucí hustotou

Aby se snížil počet polygonů nutných k vykreslení je možné spojit několik políček do jednoho. Počet těchto spojených políček nesmí být příliš velký. Na krajích polygonu může být řídký porost a uvnitř se objeví část hustého, při použití velkých ploch by se porost neobjevil. Při přiblížení k tomuto místu by došlo k přepnutí úrovně detailů a zobrazený řídký porost by se najednou změnil na hustý. S tím by pozorovatel nebyl spokojen. Velikost pole po spojení musí být takové, že na dané ploše se nesmí hustota příliš měnit.



Obr. 3.17 Schéma vzdáleného porostu

3.4 Pohyb porostu

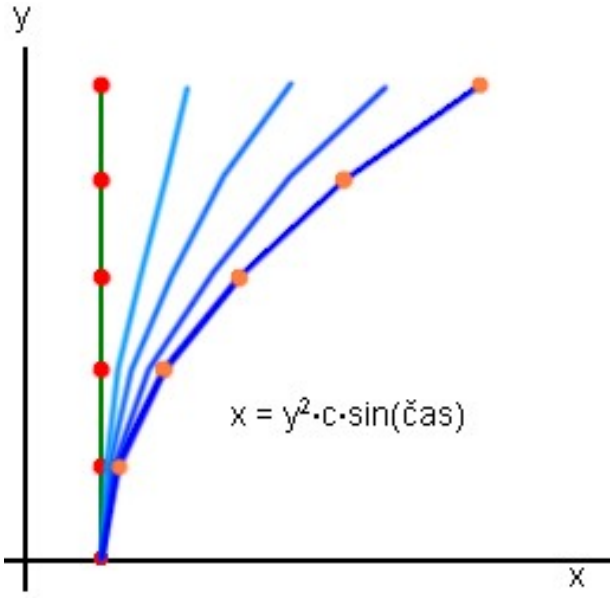
Pohyb porostu je třeba popsat pomocí shader jednotek. Musí být napsáno více programů pro různé úrovně detailů. Tyto programy budou 3, stejně jako počet úrovní detailů. V nejbližší vzdálenosti je to pohyb jednotlivých stébel, ve střední vzdálenosti pohyb bilbordů a pohyb nejvzdálenějšího porostu.

3.4.1 Pohyb stébel

Shaderům je třeba předat nějakou hodnotu, podle které by poznala, kde se mají jednotlivá stébla scény v čase nacházet. To je možné pomocí hodnoty simulačního času. Tento čas je předán shaderům a podle jeho hodnoty je provedena modifikace. Vhodnou modifikací je funkce sinus, které je předán jako parametr hodnota simulačního času. Funkce sinus je výhodná díky svému oboru hodnot, protože tyto hodnoty jsou od -1 po 1. Stéblo je ve své základní poloze při hodnotě 0 a při ostatních hodnotách dochází k vychylování. Znaménko určí, na kterou stranu se bude stéblo vychylovat. Program bude napsaný jeden pro všechna stébla. Lze využít toho, že všechna stébla mají stejnou texturu a tuto texturu poslat do grafické karty pouze jednou. Tím se výrazně sníží množství dat nutných k přesunu do grafické karty a také zrychlí vykreslování. Spočítání funkce sinus by bylo možné provést ještě před vstupem do shaderu, potom by ale dával stejné výsledky pro všechny stébla, a ty by se pohybovaly stejně. To by nepůsobylo dobrým dojmem. Je tedy nutné počítat vychýlení pro každé stéblo. Pokud je předán funkci sinus pouze simulační čas, dává pořád stejné výsledky pro všechny stébla a počítání pro každé stéblo při stejném vychýlení ztrácí význam. Je tedy nutné pro každé stéblo uložit jeho počáteční posunutí. Funkci se předá simulační čas a počáteční posunutí a výsledek bude

jiný pro každé stéblo. Aby se stébla pohybovala s různou frekvencí je možné pro každé uložit ještě frekvenci. Výsledek pro každé stéblo bude dán simulačním časem, frekvencí pohybu a počátečním posunutím. Tuto frekvenci a posunutí je nutné shaderům předat. Barva stébla je určena texturou a vektor barvy není využit. To umožňuje hodnoty předat pomocí jednotlivých barevných složek.

Naklonění stébla se provádí s využitím vertex shader jednotky. Vstupní vektor je posunut tak, aby bylo stéblo zahnuté. To je možné provést změnou x-ové souřadníce. Pro zahnutí stébla je nutné zvyšovat x-ovou souřadnici rychleji než roste výška stébla. Program je možné napsat různými způsoby. Muže být napsán tak, že je výpočetně náročnější, ale dává lepší výsledky, nebo méně náročný, rychlejší, ale s horšímy výsledky. Jak je vidět na obrázku 3.18 se při posunování x-ové souřadnice stéblo protahuje. Aby bylo stéblo stále stejně dlouhé, je nutné se zvyšováním x-ové souřadnice snižovat souřadnici y-ovou.



Obr. 3.18 Ohýbání stébla

Pixel shader jednotka pouze nastaví barvu pixelu podle textury. Také je možné barvu pozměnit, aby se měnila s pohybem stébla, kde podle výsledku funkce sinus z vertex shaderu je barva pixelu modifikována.

3.4.2 Pohyb bilboardu

Pro pohyb bilbordu pomocí shaderů bude opět třeba využít simulačního času. Velikost výchylky se získá funkcí sinus vypočítanou pro každý bilbord. Funkci sinus je také předána frekvence pohybu a počáteční posunutí. Ty jsou do grafické karty přesunuty pomocí prvních dvou složek barevného vektoru, který není při zobrazování využit. Z hodnot frekvence a posunutí je vypočtena výchylka v intervalu -1 a 1. Textura bilbordu je také pro všechny bilboardy stejná, proto je do grafické karty přesunována pouze jednou.

Vertex shader jednotka slouží k naklánění bilbordu. Spodní dvojice vektorů není vůbec při pohybu modifikována. Horní dvojice vektorů je posouvána, a tím dochází k naklánění bilbordu. Aby se všechny nepohybovaly ve stejném intervalu je využita třetí složka barevného vektoru pro přenos konstanty maximální výchylky. Při výpočtu je tato možnost zohledněna a promítnuta do výsledného posunutí. Funkce pro posun vektorů a tedy naklánění je odvozena z programu pro naklánění jednotlivých stebel, aby se pohyb bilbordu podobal pohybu stébel. Program muže být zjednodušen, protože modifikuje mnohem jednodužší objekt. Poslední složka barevného vektoru slouží pro přenos hustoty porostu. S touto hodnotou hustoty vertex shader jednotka nepracuje, ta je přesunuta do pixel shader jednotky.

Jak již bylo zmíněno dříve je pixel shader jednotka využita pro vytvoření bilbordu s různou hustotou. Podle hodnoty hustoty získané z vertex shader jednotky je prahován alfa kanál textury.Podle výsledku je tento bod textury zobrazen, nebo v druhém případě je průhledný. Barva bodu je také modifikována v závislosti na naklonění. Naklonění je pixel shader jednotce předáno z vertex shader jednotky a naklonění definuje výsledek vypočítané hodnoty funkce sinus.

Obr. 3.19 Nakláněný bilbord (bez textury porostu)

3.4.3 Vzdálený pohyb

U nejvzdálenějšího porostu se již nevyužívá pohybu modifikací ve vertex shader jednotce, protože se jedná pouze o texturu ležící na povrchu země. Pohyb lze částečně nahradit modifikací jednotlivých bodů textury v pixel shader jednotce.

Pixel shader jednotka se používá stejně jako u bilbordu k zobrazení porostu s různou hustotou. Hustota porostu je do shader jednotky přesunuta s využitím barevného vektoru. Alfakanál textury je prahován pomocí hodnoty hustoty. Ve vertex shader jednotce je spočítána hodnota funkce sinus ze simulačního času. U nejvzdálenějšího porostu není třeba brát v úvahu frekvenci a počet posunutí, protože se bude modifikovat každý pixel ne celý polygon. Pokud by měly sousední políčka různé výsledky funkce, byl by přechod mezi těmito políčky viditelný. Barva pixelu je modifikována podle výsledku funkce sinus.

4 Výsledky

4.1 Výsledky nejbližšího porostu

U nejbližšího porostu je vhodné porovnání stébel jednoduchých a detailnějších. Tabulká udává počet zobrazených snímků za sekundu (Frames Per Second) a počet polygonů nutných k vykreslení pro oba druhy stébel. Výsledky v tabulce jsou získány s nejhustším porostem. Hodnoty jsou pouze přibližné. Dohlednost je v tomto případě neomezená, zobrazují se tedy všechny stébla celé mapy. Porovnání vzhledu těchto stébel je na obrázku 3.5 v kapitole 3.1.

	Detailní stébla		Jednoduchá stébla	
Rozměr mapy	FPS	Polygonů	FPS	Polygonů
1x1	1050	250	1080	50
2x2	390	1 000	420	200
4x4	110	4 000	112	800
8x8	29	16 000	29,03	3 200
16x16	7,15	64 000	7,12	12 800
32x32	1,75	256 000	1,77	51 200
64x64	0.42	1 024 000	0.43	204 800

Tab. 4.1 Srovnání detailních a jednoduchých stébel

Z tabulky je tedy vidět, že mezi detailními a jednoduchými stébly není takový výkonostní rozdíl, jaký by se dal očekávat. Počet polygonů nutných k vykreslení je pětinásobný, zvýšení se pohybuje v jednotkách procent. Graf scény je v obou případech stejný, liší se pouze výsledným objektem. Zobrazení objektu tedy není tak náročné v porovnání se zpracováním a transformací objektů grafu. Řešení viditelnosti a překrývání stébel je tak náročné, že vyšší počet polygonů již rychlost příliš neovlivní. Dále mohlo dojít k mírnému ovlivnění, že ostatní objekty scény byly vytvořeny, došlo pouze k jejich odpojení z grafu scény. Díky získaným výsledkům je tedy možné v rámci optimalizace část s jednoduchými stébly vynechat a využít pouze stébla detailní.

4.2 Výsledky středně vzdáleného porostu

Stejně jako u nejbližší vzdálenosti je i v té střední vhodné porovnat dva naimplementované způsoby vytváření bilbordů. Tyto způsoby se liší pouze v šířce použitého bilbordu. Větší bilbordy jsou umístěny přes několik políček. Více o implementaci lze nalézt v kapitole 3.2. Stejně jako v

předchozím případě je vytvářen porost s nejvyšší možnou hustotou. Získané hodnoty jsou pouze přibližné.

Rozměr mapy	Menší bilbordy		Větší bilbordy	
	FPS	Polygonů	FPS	Polygonů
1x1	865	3	607	20
2x2	570	12	648	20
4x4	344	48	486	20
8x8	240	192	393	80
16x16	119	768	265	320
32x32	29,5	3 072	93	980
64x64	7,35	12 288	26,7	3 380

Tab. 4.2 Srovnání bilbordů

Oproti výsledku v nejbližší vzdálenosti je zde výsledek uspokojivější. U větších rozměrů mapy dochází s použitím větších bilbordů k úspoře. Použití větších bilbordů umožňuje 4-krát větší mapy se srovnatelnými výsledky. Větší bilbordy není vhodné využívat na malých plochách ,jak je vidět v tabulce u menších rozměrů, kde je mapa menší než je šířka většího bilbordu. Menší bilbordy je nutné ponechat pro navázání mezi trsy nejbližší vzdálenosti a většími bilbordy. Plocha využití větších bilbordů tedy musí být dostatečně velká, aby docházelo k úspoře. Protože dává využití dvou úrovní bilbordů dobré výsledky, je vhodné tento způsob ponechat a pouze optimalizovat počet bilbordů na jednom políčku.

4.3 Výsledky vzdáleného porostu

U vzdáleného porostu je pouze jedna úroveň. Pro celkový přehled v jednotlivých vzdálenostech je vhodné výsledky v této části doplnit. Více o implementaci lze nalezt v katitole 3.3. Výsledky jsou pro porost s nejvyšší hustotou. V této vzdálenosti na hustotě nezáleží a pro všechny hustoty dává stejné výsledky. Výsledky jsou pouze přibližné a výrazným způsobem závisí na aktuální pozici kamery.

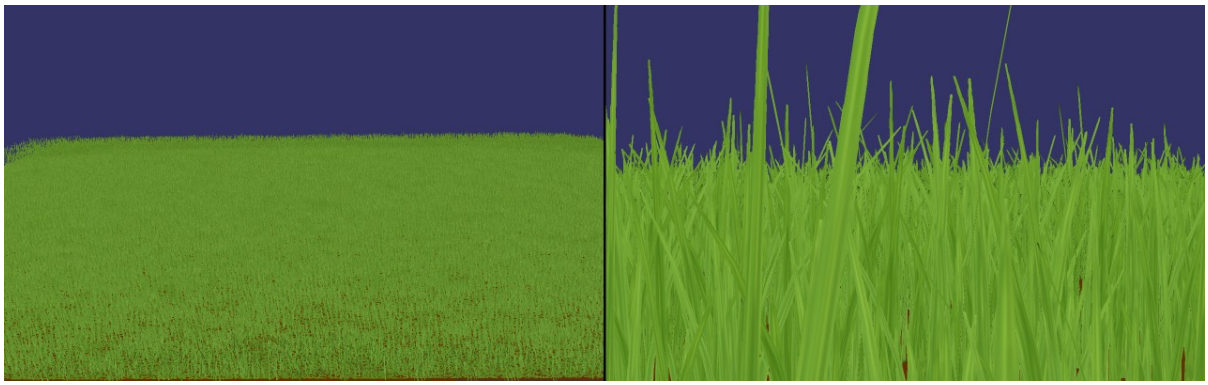
Rozměr mapy	FPS	Polygonů
1x1	-	0
2x2	-	0
4x4	1 200	1
8x8	1 057	4
16x16	992	9
32x32	992	36
64x64	470	169

Tab. 4.3 Výsledky vzdáleného porostu

Oproti předchozím úrovním je patrné zvýšení výkonu. V předchozí střední úrovni detailů byl výkon největší mapy v řádu desítek snímků za sekundu. U nejvyšší uřovně je již v řádu stovek snímků za sekundu. U větších map již narůstá počet polygonů nutných k vykreslení. Pokud by byl porost vytvářen na mnohem větších mapách, bylo by vhodné použít další podúroveň, kde by byly vytvářeny mnohem větší polygony. Tím by se počet polygonů výrazně snížil.

4.4 Výsledky kompletního porostu

Výsledky je možné zpracovat ze dvou hledisek. První je rychlost vykreslování v závislosti na velikosti mapy a druhé rychlost vykreslování při různých hustotách porostu. Dále jsou porovnány výsledky rychlosti zobrazení při statickém porostu a pohybujícím se porostu.



Obr. 4.1 Vzdálené a blízké umístění kamery

4.4.1 Velikost mapy

Velikost mapy v tomto případě není třeba testovat od nejmenší velikosti. Menší mapy měly význam pro porovnání výkonu jednotlivých urovní. U menších map by se nedostalo na zobrazení vzdálenější urovně, a tím by výsledky byly zkreslené. Měření probíhá na porostu s nejvyšší hustotou. Výsledky jsou pouze přibližné a zavisí na aktuální pozici kamery ve scéně. Protože silně závisí na přiblížení k porostu, je počet snímků změřen dvakrát. Vzdálenější scéna, kde je kamera umístěna v takové vzdálenosti, ve které dochází k přepnutí mezi blízkou a střední úrovní. U blízké scény je kamera umístěna přímo u jednotlivých stébel. Oba pohledy kamery jsou na obrázku 4.2. Dále je doplněna doba nutná k vytvoření a zobrazení porostu od spuštění programu.

Rozměr mapy	FPS Vzdálená	FPS Blízká	Doba spuštění
32x32	15,2	5,49	2,2s
48x48	12,07	4,85	2,8s
64x64	8,65	4,54	4,2s
96x96	6,12	3,28	6,1s
128x128	4,84	2,82	9,3s
192x192	3,55	2,24	23,5s
256x256	2,64	1,82	1,5min

Tab. 4.4 Výsledky zobrazení porostu v závislosti na velikosti mapy

Rychlost zobrazení kompletního porostu se pohybuje na úrovni středních detailů. Počet snímků za sekundu se snižuje úměrně ke zvětšení velikosti mapy. Toto zpomalení je dáno zvětšováním grafu scény. Doba nutná ke spuštění se zvyšuje s velikostí. U mapy s rozměry 256x256 doba spuštění výrazně narostla. Při optimalizaci by bylo třeba snížit tento čas nutný ke spuštění.

4.4.2 Hustota porostu

Hustota je udána indexem hustoty. Index hustoty je hodnota od 1 do 10, menší čísla znamenají nižší hustotu, vyšší čísla vyšší hustotu. Tabulka udává počet snímků za sekundu při dané hustotě. Pro testování rychlosti zobrazování porostu o různých hustotách je třeba zvolit jeden rozměr mapy. Měření proběhlo pomocí mapy s rozměrem 64x64. Měření je stejně jako v předchozím případě provedeno kamerou ve dvou pozicích. Pozice kamery jsou na obrázku 4.1

Hustota	FPS Vzdálená	FPS Blízká
1	13	15,65
2	12,8	12,12
3	12,63	10,53
4	12,6	8,65
5	12,35	7,83
6	9	6,16
7	9,35	5,72
8	9	5,01
9	9	4,72
10	9,2	4,24

Tab. 4.5 Výsledky zobrazení porostu v závislosti na hustotě

U vzdáleného pohledu kamery je vidět, že hustota porostu nemá vliv na výkon. Rozdíl je pouze u první a druhé poloviny tabulky. Tento rozdíl je způsoben vytvářením vyššího počtu bilbordů u druhé poloviny tabulky. S kamerou umístěnou blízko stébel je u méně hustého porostu rychlost vyšší.

U blízkého pohledu s malou hustotou se nemusí zobrazovat takové množství stébel a lze tedy dosáhnout vyšší rychlosti.

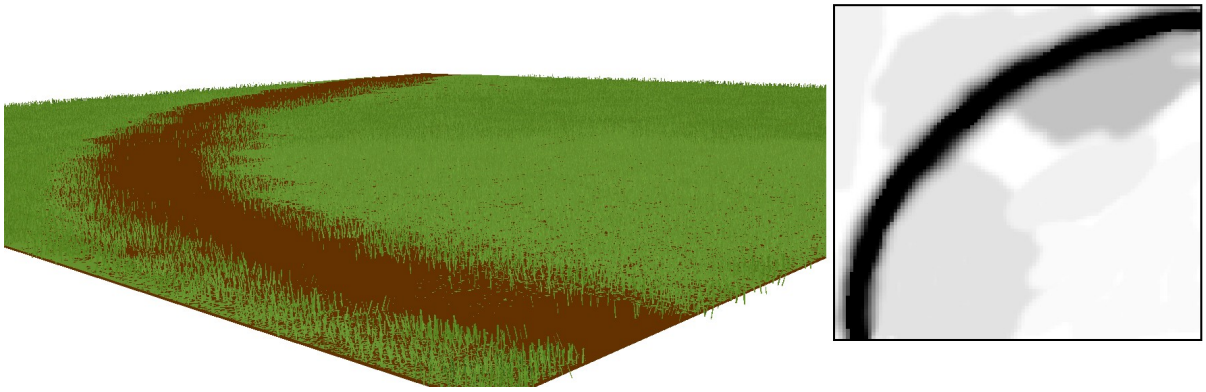
4.4.3 Pohyb porostu

Rychlost v závislosti na pohybu je měřena na mapě z obrázku 4.2. Rychlost je stejně jako v předchozích případech měřena dvěma kamerami s různým umístěním viz obrázek 4.1.

	FPS Vzdálená	FPS Blízká
Statický porost	7,77	3,78
Porost v pohybu	6,54	3,59

Tab. 4.6 Výsledky zobrazení porostu v závislosti na pohybu

Pohyb porostu nemá příliš vliv na rychlost zobrazení. Rychlost pohybujícího se porostu je jen mírně nižší než rychlost statického porostu.



Obr. 4.2 Výsledný porost a jeho mapa

5 Závěr

Cílem práce bylo zobrazení porostu s využitím shaderů. Zobrazení plně trojrozměrného porostu je stále velice náročné. To je zřejmé z dosažených výsledků. Výsledky byly měřeny na dnes již starším hardwaru. Při použití moderního hardwaru by bylo možné dosáhnout lepších výsledků, ale stále by tyto výsledky nebyly použitelné pro reálné nasazení v počítačových hrách.

Přínosem pro mne bylo seznámení s toolkitem OpenSceneGraph a s principem grafu scény, který toolkit využívá. Dále jsem se seznámil s principem shaderů a jejich programováním v jazyce GLSL. Zjistil jsem, jaké problémy nastávají při zobrazení porostu a také proč zatím nedochází k masivnímu rozšíření plně trojrozměrného porostu.

Pohyb porostu v mírném větru je vytvořen náhodným nakláněním jednotlivých stébel. Jako další rozšíření by bylo možné brát v úvahu další skutečnost a to sílu větru. Shaderům by se předal spolu s objektem i směr a síla větru. Shader nakloní stéblo podle informací o větru. Dalším možným rozšířením je umístění různých rostlin jako je plevel nebo květiny do porostu. Tyto další rostliny by se vytvořily podobným způsobem jako je zobrazen porost. Protože by těchto rostlin nebyl takový počet jako je stébel, mohly by se modely zobrazovat do mnohem větší vzdálenosti.

V rámci optimalizace je třeba snížit rychlost spuštění scény, která pro velkou mapu dosáhla velmi vysokého času. Také by bylo třeba optimalizovat strukturu grafu scény pro co nejvyšší výkon. Dále je podle výsledků možné zrušit méně detailní stébla v nejbližší úrovní, jelikož jejich použitím nedosáhneme požadovaného zrychlení.

Literatura

[1] Operace Flashpoint – screenshot. Codemasters, Bohemia Interactive. 2001. Dostupné na http://www.gamespot.com/pc/action/operationflashpointcwc/images.html (duben 2008)

[2] FarCry – screenshot. Ubisoft, Crytec. 2004. Dostupné na http://www.gamespot.com/pc/action/farcry/images.html (duben 2008)

[3] Crysis – screenshot. EA Games, Crytec. 2007. Dostupné na http://www.gamespot.com/pc/action/crysis/images.html (duben 2008)

[4] Boulanger, K., Pattanaik, S., Bouatouch, K.: Rendering Grass Terrains in Real-Time with Dynamic Lightning. University of Central Florida USA, IRISA Rennes France. 2006. Dostupné na http://www.irisa.fr/bunraku/GENS/kboulang/publications/grassSiggraph2006.pdf (duben 2008)

[5] Isidoro, J., Card, D., aj.: Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks. Wordware Publishing, Inc. Plano Texas USA, 2002. strana 334. Dostupné na http://ati.amd.com/developer/shaderx/ShaderX_AnimatedGrass.pdf (duben 2008)

[6] Pelikán, J.: 3D akcelerátory – historie a architektura. MFF UK. Praha. 2008. Dostupné na http://cgg.ms.mff.cuni.cz/~pepca/lectures/pdf/hwintro.pdf (duben 2008)

[7] Pelikán, J.: Programování GPU. MFF UK. Praha. 2005. Dostupné na http://cgg.ms.mff.cuni.cz/~pepca/lectures/pdf/hwshaders.pdf (duben 2008)

[8] Programmable Vertex Shaders: Technical Brief. Nvidia. 2001. Dostupné na http://www.nvidia.com/attach/4049 (duben 2008)

[9] Programmable Pixel Shaders: Technical Brief. Nvidia. 2001. Dostupné na http://www.nvidia.com/attach/162 (duben 2008)

[10] Martz, P.: OpenSceneGraph Quick Start Guide. Skew Matrix Software. California USA. 2007. Dostupné na http://www.lulu.com/content/767629 (duben 2008)