VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RENDEROVÁNÍ TRÁVY

GRASS RENDERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **TOMÁŠ BERDIS**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ STARKA**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Berdis Tomáš**
Program: Informační technologie
Název: **Renderování trávy**
Grass Rendering
Kategorie: Počítačová grafika

Zadání:

1. Nastudujte techniky renderování trávy (osvětlování a jednoduché animace).
2. Navrhněte aplikaci renderující travnaté plochy.
3. Vytvořte navrženou aplikaci.
4. Vytvořte krátké video prezentující práci.

Literatura:

- Po dohodě s vedoucím.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 a část bodu 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Starka Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Tráva je nedielnou súčasťou väčšiny vonkajších scén. V skutočnom svete sa vyskytuje vo veľkom počte, čo prináša mnohé výzvy pri jej renderovaní v počítačovej grafike. Táto práca sa zaoberá technikami renderovania trávy v reálnom čase. Výsledkom práce je aplikácia renderujúca trávové povrchy využitím hardvérovej teselácie.

Abstract

Grass is an integral part of most of the outdoor scenes. In the real world, grass occurs in big numbers, which brings a lot of challenges during rendering in computer graphics. This thesis discusses various techniques for real-time rendering. Result of this thesis is an application capable of rendering grass fields using hardware tessellation.

Kľúčové slová

renderovanie trávy, C++, OpenGL, teselácia, simulácia vetra, výšková mapa, LoD

Keywords

grass rendering, C++, OpenGL, tessellation, wind simulation, height map, LoD

Citácia

BERDIS, Tomáš. *Renderování trávy*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Starka

Renderování trávy

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Tomáša Starky. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Tomáš Berdis
10. mája 2021

Poďakovanie

Ďakujem svojmu vedúcemu Ing. Tomášovi Starkovi za trpezlivosť a odbornú pomoc pri riešení tejto práce. Ďalej chcem poďakovať Frankovi Herbertovi, Maynardovi Jamesovi Keenanovi a Trentovi Reznorovi za ich umeleckú tvorbu, ktorá ma pri písaní tejto práce inšpirovala.

Obsah

Kapitola 1

Úvod

Rozsiahle prírodné scény sú dnes často potrebné, pre uskutočnenie kreatívnej vízie autora a poskytnutie realistického dojmu pre hráča, či diváka. Dôležitou súčasťou väčšiny takýchto scén je prirodzene tráva. Steblá trávy sa v reálnom svete vyskytujú v síce pomerne jednoduchých tvaroch, ale zato vo veľkých množstvách, čo značne komplikuje ich zobrazovanie v počítačovej grafike.

Filmový priemysel má oproti tomu hernému značnú výhodu v tom, že nemusí svoje scény vykresľovať v reálnom čase. Vykresľovanie tak môže trvať reálne aj niekoľko hodín, či dní. Ale keďže hráč v počítačovej hre môže so scénou interagovať a pohybovať sa v nej, je potrebné scénu vykresľovať v reálnom čase. To prináša mnohé výzvy z výkonového, ako aj dizajnového hľadiska.

Na počiatkoch herného priemyslu sa pre reprezentáciu trávy používali jednoduché trávové textúry, kvôli zníženiu nárokov na výkon. Táto technika sa používa aj dnes, ale skôr v kombinácií z ďalšími technikami.

S narastajúcim grafickým výkonom a z neho vyplývajúcimi možnosťami vykresliť na scénu viac objektov, sa začali využívať novšie techniky. Tráva sa začala vykresľovať ako textúrovaný quad, čo dodalo scénam viac hĺbky. Tieto quady sú však dvojrozmerné, z čoho vyplývajú ich obmedzenia v 3D priestore. Jedným z hlavných problémov sú pozorovacie uhly. Ak sa na takúto trávu pozrieme z diaľky, alebo z menšieho uhla, ilúzia trávy ostáva zachovaná. So vzrastajúcim uhlom sa však táto ilúzia vytráca a objavujú sa nedostatky. Možnosti ako tieto nedostatky minimalizovať, alebo sa im vyhnúť úplne riešia mnohé metódy predstavené v tejto práci.

Doposiaľ najlepším, no zároveň aj najdrahším spôsobom ako reprezentovať trávu v počítačovej grafike je modelovanie jednotlivých stebiel pomocou geometrie. Tento prístup umožňuje komplexnejšiu simuláciu pohybu stebiel, keďže vieme steblá ovplyvniť individuálne. To je možné využiť napríklad pri simulácií vetra a kolízií trávy s inými objektmi.

Súčasne vyvinuté techniky pre vykresľovanie trávy poskytujú mnohé možnosti pre dostatočne realistické zobrazenie trávy. Faktom ale ostáva to, že stále nedisponujeme výkonom dostatočným na to, aby sme mohli realisticky simulovať všetky aspekty prírodných scén. V reálnych aplikáciách je preto nutné využiť kombinácie viacerých techník a znižovať úroveň detailu tam, kde to vadí najmenej pre zachovanie uveriteľnosti výslednej scény.

Kapitola 2

Renderovanie trávy v počítačovej grafike

Tráva je dôležitou súčasťou väčšiny prírodných scén. Pri digitálnej simulácií takýchto scén je cieľom zachovať čo najväčšiu uveriteľnosť a detail. Typickým problémom je ale hardvérové obmedzenie, ktoré neumožňuje vyrenderovať toľké množstvo trávy aké by existovalo v ekvivalentnej reálnej scéne. Ešte k tomu aj v reálnom čase. Existuje množstvo techník, ktoré tento problém riešia. Vybrané techniky sú popísané v nasledujúcich sekciách.

2.1 Prehľad existujúcich techník

Jednou z najbežnejších techník používaných hlavne v starších aplikáciách je mapovanie 2D textúry na terén. Problémom tejto techniky je najmä absencia objemu trávy a nemožnosť jej animácie. Aj keď je mapovanie textúry implementačne a výkonovo menej náročné, takáto tráva nevyzerá dobre pod malými pozorovacími uhlami.

Ďalšou možnosťou je využitie tzv. obrazovo založených techník[1][9, 4, 8]. Tu sa textúra mapuje na vertikálny objekt namiesto zeme. To síce zlepšuje viditeľnosť trávy z menších uhlov, ale naopak zhoršuje dojem pri pohľade z výšky. Takto vytvorené trávnaté objekty je možné animovať, no len ako celok. Najlepšie vyzerajúce, no zároveň aj najdrahšie z hľadiska výkonu je renderovanie jednotlivých stebiel trávy ako geometrie[1, 6, 2, 7, 14]. Techniky využívajúce geometriu umožňujú animáciu individuálnych stebiel trávy. Vo výsledku je možné realistickejšie simulovanie kolízii s objektami, ako aj plynulejšie ohýbanie stebiel trávy podľa smeru vetra.

V praxi sa využíva kombinácia vyššie zmienených prístupov. Jednotlivé techniky sa menia často podľa vzdialenosti od kamery, teda LOD[2], ktorý chceme dosiahnuť.

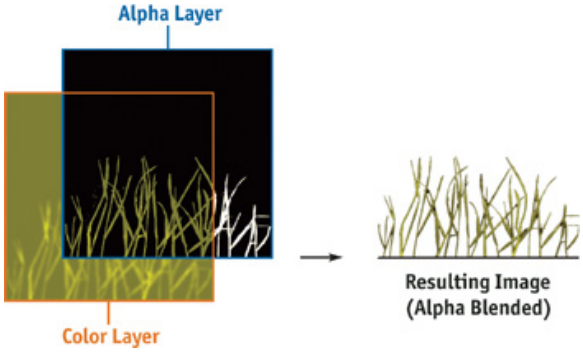
2.2 Obrazovo založené techniky

V tejto sekcii sú popísané techniky, ktoré pre vykreslenie trávy využívajú 2D objekty, na ktorých je namapovaná polo-priehľadná textúra. Takéto objekty môžu byť reprezentované statickým *quadom*[3], alebo tzv. *billboardom*. Billboardy sú podobné quadom, s tým rozdielom, že billboardy sú vždy orientované ku kamere.

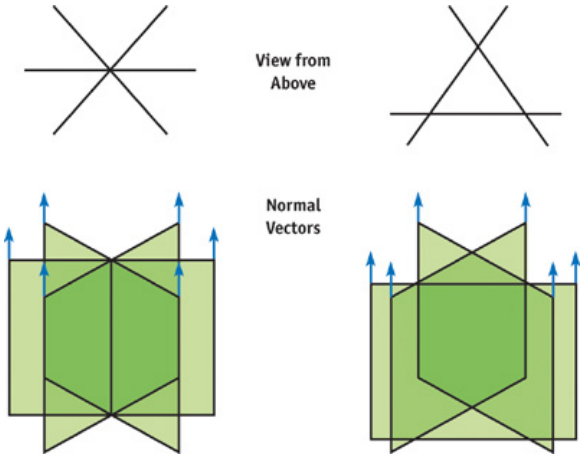
[1] Angl. image-based techniques.
[2] Angl. level of detail - úroveň detailu.
[3] Quad - 2D objekt obdĺžnikového tvaru v 3D scéne.

2.2.1 Vzájomne prekrížené quady

Kurt Pelzer vo svojom príspevku v knihe GPU Gems[9] popisuje možnosť zaplnenia scény trávou pomocou trávových objektov. Každý takýto objekt je tvorený troma quadmi, na ktorých je nanesená polo-priehľadná textúra trávy (viz. obr. 2.1). Jednotlivé quady sú vzájomne prekrížené, kvôli viditeľnosti z viacerých strán (viz. obr. 2.2). Takto vytvorené objekty sú následne rozmiestnené na veľkú plochu tesne vedľa seba. Pri vykresľovaní je potrebné objekty zoradiť od konca a použiť *alpha blending*[4] pre umožnenie viditeľnosti viacerých polo-priehľadných textúr za sebou.



Obr. 2.1: Polo-priehľadná textúra trávy aplikovaná na quad. Z nízkych pozorovacích uhlov imituje trávové steblá. Na ľavej strane je možné vidieť oddelenú farebnú vrstvu od alfa vrstvy. Na strane pravej výsledný obrázok s použitím alpha blendingu.[9]



Obr. 2.2: Možnosti umiestnenia quadov s textúrou trávy, pre vytvorenie trávového objektu, ktorý je možné pozorovať zo všetkých bočných strán. Šípky nahor predstavujú normály.[9]

Pre dosiahnutie vierohodnosti scény s takto vytvorenou trávnatou lúkou je potrebné trávu rozhýbať vo vetre. Pelzer tu popisuje tri spôsoby animácie, teda simulácie vetra. Pozri 2.3. Každý so sebou samozrejme nesie výhody aj nevýhody. Kalkulácia využíva trigonometrické funkcie, konkrétne sínus a kosínus pre posunutie horných vertexov každého objektu v smere vetra. Polohu vertexov je jednoduché určiť vo vertex shaderi, kde je ju

[4] Alpha blending - proces kombinovania obrázku s pozadím, pre dosiahnutie čiastočnej, alebo úplnej priehľadnosti.

možné odvodiť z textúrových súradníc (napr. v štandarde OpenGL to budú hodnoty t/v blízke jednotke).

Animácia klastrov

V tomto prípade je posun horných vertexov aplikovaný na blízku skupinu trávových objektov, tzv. klaster. Pre vytvorenie prirodzene pôsobiacej animácie je potrebné zvoliť nie príliš veľkú veľkosť klastra. Pozri 2.3a.

Vektor posunu je vypočítaný na CPU a ďalej poslaný do vertex shadera ako konštanta. To umožňuje využitie výpočtovo náročnejších algoritmov pre komplexnejšiu animáciu. Zároveň je ale nutné túto konštantu meniť pre každý klaster, čo má za následok väčší počet volaní kreslenia (angl. *draw call*) grafickej karty a teda horší výkon. Nevýhodou môže aj byť viditeľnosť klastrov na lúke, keďže sa všetky vertexy v rámci klastra pohybujú synchrónne.

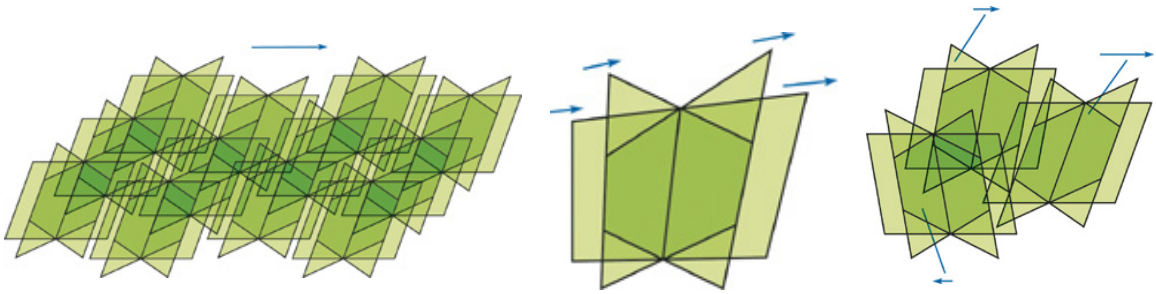
Animácia vertexov

Výkonnostný problém pri animovaní celých klastrov rieši presunutie výpočtov do vertex shadera. Tu sa nová pozícia vypočíta pre každý horný vertex samostatne, bez potreby zásahu zo strany CPU, takže na vykreslenie celej lúky stačí jedno volanie kreslenia. Nová pozícia sa počíta na základe pôvodnej pozície vertexu vo svete.

Pretože je posun vertexov v blízkej oblasti veľmi podobný, chýba trávniku lokálny chaos a vo výsledku pôsobí veľmi homogénne. Tento jav je možné potlačiť použitím pseudonáhodnej funkcie vo vertex shaderi.

Animácia trávových objektov

Poslednou spomenutou možnosťou je kombinácia vyššie spomínaných techník. Výsledná pozícia horného vertexu sa tentokrát počíta na základe pozície stredu trávového objektu. Pretože susedné trávové objekty majú teraz rozdielne animácie, dochádza k lokálnemu chaosu. Pozri 2.3c. Keďže sa pri výpočte využíva pozícia stredu trávového objektu, je potrebné túto pozíciu vertex shaderu poskytnúť, čo o niečo zvyšuje pamäťové nároky.



(a) Animácia klastrov.

(b) Animácia vertexov.

(c) Animácia trávových objektov.

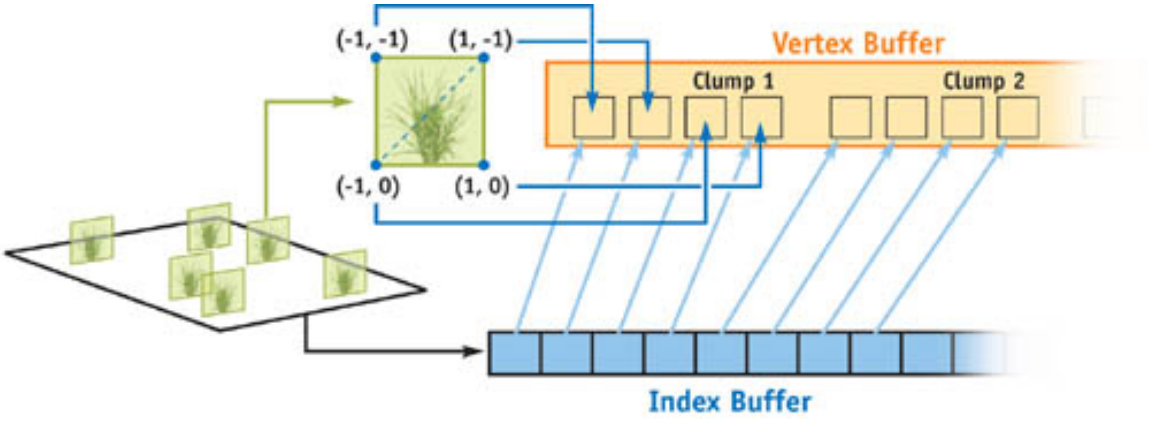
Obr. 2.3: Spôsoby animácie trávových objektov.[9]

2.2.2 Billboardy orientované ku kamere

Technika navrhnutá Whatleym[15] využíva namiesto viacerých statických quadov použitých u Pelzera[9], jeden quad (billboard) orientovaný vždy smerom ku kamere. Využitie billboardov zachováva konštantnú vizuálnu hĺbku pri pohľade zo všetkých strán (aj pri pohľade zhora). Terén je v tomto prípade rozdelený podľa rovnomernej mriežky na jednotlivé bunky. Vietor je simulovaný podobne ako u Pelzera.

Spôsob vykreslenia

Každá vrstva trávy (t.j. všetka tráva, ktorá používa rovnakú textúru a parametre) je reprezentovaná párom vertex a index bufferu (pozri obr. 2.4). Tieto buffer-e sú obsiahnuté v rámci jednej bunky mriežky. Cieľom je zaplniť buffer-e veľkým počtom quadov a všetky vykresliť naraz v rámci jedného draw callu. Všetky vertexy zo začiatku obsahujú len pozíciu ich quadu vo svete. Neskôr sú v rámci vertex shadera dopočítané ich pozície tak, aby celý quad smeroval ku kamere.



Obr. 2.4: Zaplnenie buffer-a jedným typom trávového quadu.[15]

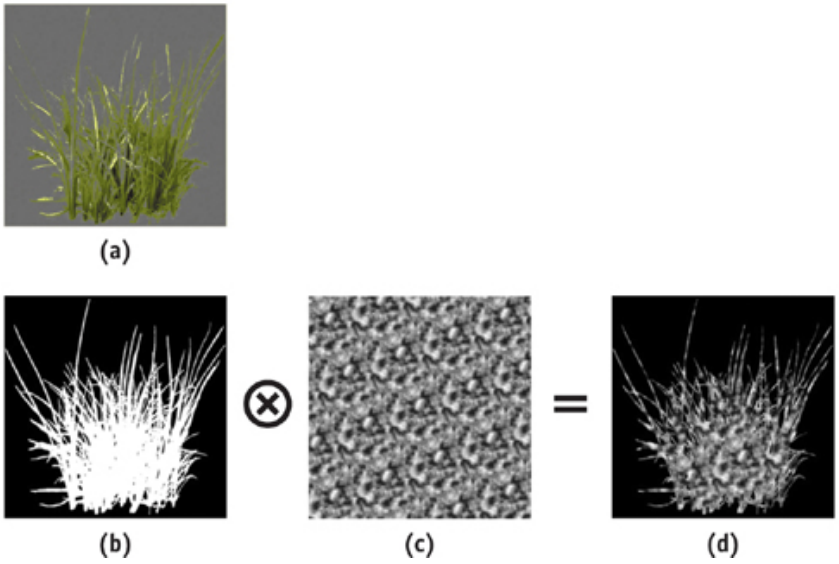
Simulácia priehľadnosti pomocou efektu rozpustenia

Pri renderovaní trávy je vhodné použiť priehľadnosť pre zlepšenie prechodov medzi časťami trávnika a dojmu strácajúcej sa trávy v diaľke. Priehľadnosť založená na alfe, je ale výkonovo náročná, keďže vyžaduje zoradenie objektov pred blendingom. Namiesto toho je možné priehľadnosť simulovať pomocou *efektu rozpustenia* (angl. *dissolve effect*).

Tento efekt je dosiahnutý moduláciou alfa kanálu textúry trávy šumovou textúrou (vhodný je napr. Perlinov šum[10]). Následne je použitý alfa test pre elimináciu nepotrebných pixelov. Tento proces je zobrazený na obr. 2.5.

Osvetlenie

Osvetlenie billboardu závisí v tomto prípade na osvetlení terénu pod ním, teda na jeho normále. Normála polygónu pod billboardom je určená počas umiestňovania billboardov na trávnik a posunutá do časti generovania billboardu. S týmto prístupom je možné vykonať rovnaké výpočty osvetlenia billboardu, ako pri osvetľovaní terénu pod ním.



Obr. 2.5: Aplikovanie efektu rozpustenia na textúru trávy. a) pôvodná textúra, b) alfa kanál pôvodnej textúry, c) textúra vygenerovaná použitím Perlinovho šumu[10], d) nový alfa kanál vytvorený moduláciou pôvodného alfa kanálu a šumovej textúry.[15]

2.2.3 Quady v mriežke s využitím ray tracingu

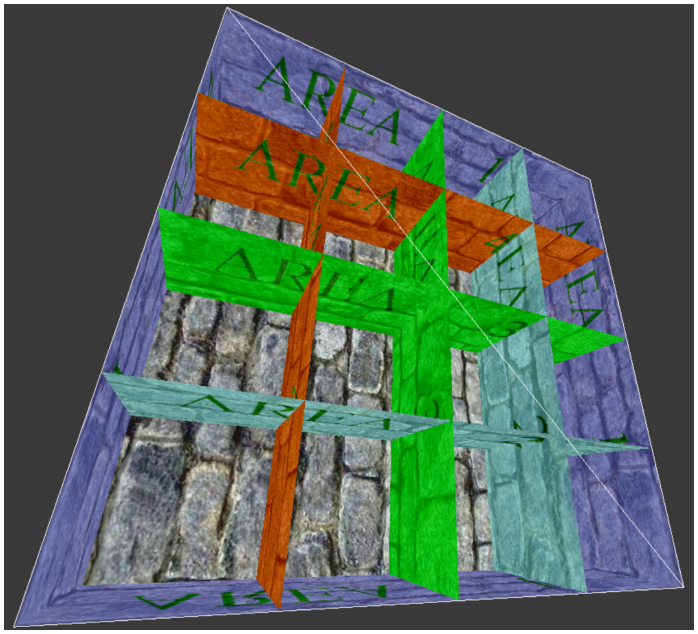
Habel et al.[4] modeluje trávu ako kolekciu textúrovaných quadov usporiadaných v pravidelnej mriežke. Pozri obr. 2.6. Namiesto renderovania quadov použitím polygónov, sú virtuálne quady definované na tzv. *nosnom polygóne* a výsledný obraz vzniká *ray tracovaním*[5] lúča prechádzajúceho týmito quadmi, kým sa nedosiahne zadaná úroveň nepriehľadnosti. Quady sú usporiadané spredu do zadu. Všetky výpočty prebiehajú vo fragment shaderi, takže je implementovanie tejto techniky vhodné pre už existujúce renderovacie systémy. Takto vytvorená tráva môže byť definovaná ako materiál a nevyžaduje dodatočné zmeny v scéne.

Ray tracing trávy

Mesh, na ktorý bude tráva vykreslená je rozdelený na viacero kusov trávnika. Základný kus trávnika pozostáva z textúry terénu a textúry obsahujúcej jednu pod-textúru pre každý jeho virtuálny quad. Pre obe osi mriežky sú využité rovnaké textúry. Ukážka textúr je na obr. 2.8.

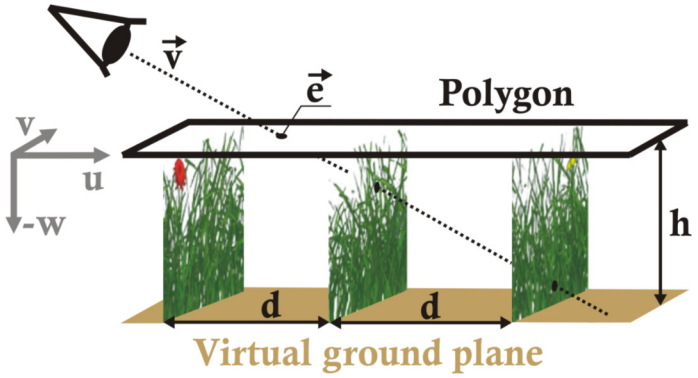
Fragment shader vrhá lúče na schránku definovanú nosným polygónom na vrchu a virtuálnym spodným polygónom, ktorý je posunutý pozdĺž negatívnej osi w dotykového priestoru (angl. *tangent space*). Tráva môže byť aplikovaná na akýkoľvek mesh, ktorý má definovaný dotykový priestor (u, v, w). Vstupom do fragment shadera sú vektory $\vec{v}$ (pohľadový vektor smerujúci od kamery) a $\vec{e}$ (vstupný bod lúča). Vzdialenosť medzi quadmi d a výška schránky h sú určené vopred. Pozri obr. 2.7. Na základe týchto parametrov je vypočítaná pozícia prvého quadu, ktorý tento lúč pretne. Pre každý ďalší quad, ktorý lúč pretne sú vypočítané korešpondujúce súradnice, až kým sa nedosiahne spodného polygónu. Každý pretnutý bod

[5]Ray tracing - renderovacia technika, ktorá využíva sledovanie lúča smerujúceho z kamery do scény a jeho odrazov, pre vypočítanie výsledného obrazu.

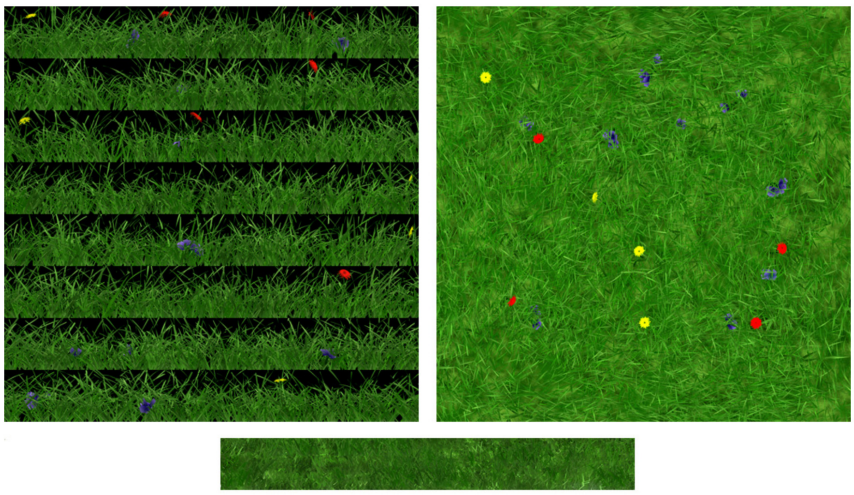


Obr. 2.6: Obrázok ilustruje usporiadanie quadov do rovnomernej mriežky.[4]

(jeho korešpondujúci pixel na textúre) prispieva k výslednej farbe pixelu určitou úrovňou alfy. Po dokončení ray tracingu sa zvyšná alfa doplní z plne nepriehľadnej časti textúry.



Obr. 2.7: Renderovanie trávových quadov použitím ray-tracingu. Výsledná tráva je vykreslená na polygón zobrazený na obrázku. Vektor $\vec{v}$ predstavuje pohľadový vektor smerujúci od kamery, vektor $\vec{e}$ reprezentuje lúč vstupujúci do schránky, d je vzdialenosť medzi jednotlivými quadmi a h určuje vzdialenosť polygónu od virtuálnej zeme.[4]



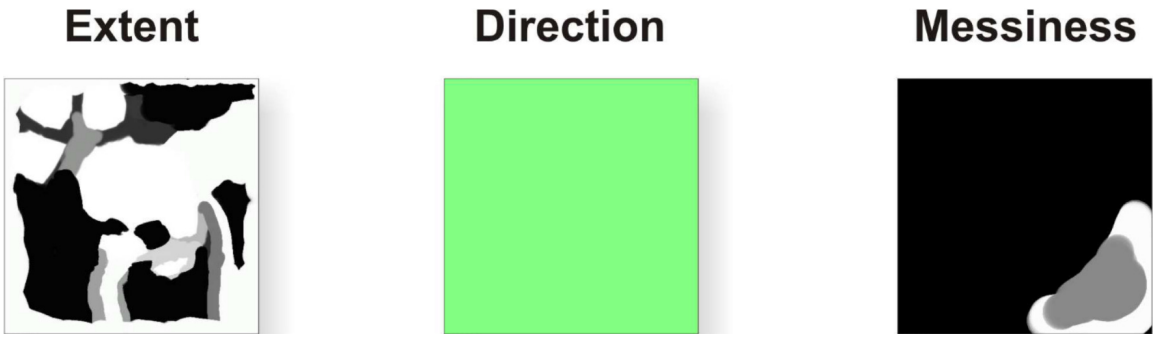
Obr. 2.8: Obrázok naľavo ukazuje textúry pre jednotlivé quady mriežky. Napravo je zobrazená textúra terénu. V spodnej časti je plne nepriehľadná textúra použitá na doplnenie chýbajúcej alfy pri ray-tracingu.[4]

2.2.4 Metóda Orthmann(2009)

Orthmann et al.[8] generuje trávové quady procedurálne na grafickej karte, pomocou *geometry shadera*. Navrhnutý kolízny systém sa stará o detekciu kolízií quadov s inými objektmi a vetrom, odozvu na tieto kolízie a návrat do pôvodného stavu. Vykreslenie deformovaných, alebo nederfomovaných quadov záleží na výstupe z kolízneho systému.

Procedurálne generovanie quadov

Zhluk trávy je ako aj v predchádzajúcich metódach reprezentovaný quadom s polo-priehľadnou textúrou. Jednotlivé quady sú vytvorené vo fáze pred-spracovania na GPU. Informácie o rozložení trávnika udáva séria textúr popísaná na obr. 2.9. *Geometry shader* vytvorí ná-



Obr. 2.9: Textúry použité na generovanie trávových quadov. Textúra vľavo udáva oblasti, kde sa tráva vyskytuje. Stredová textúra udáva smer v ktorom tráva rastie. Textúra vpravo definuje mieru náhodnosti stebiel trávy.[8]

hodnú množinu quadov. Tieto quady si ukladajú informácie o pozícií, orientácií, stave kolízie a index textúry, odkazujúci na pole textúr, ktoré obsahuje textúry trávy. Quady sú po vytvorení umiestnené do jedného veľkého buffer-u (rovnako ako pri metóde Whatley[15]), pre obmedzenie počtu volaní kreslenia (angl. render call). Pre hrubú detekciu kolízií na CPU

sú tieto quady rozdelené do hierarchickej štruktúry *octree*[6], podľa pozície na meshi terénu. Každý z uzlov si ukladá indexy do vertex buffer-u, kde sú quady umiestnené a ďalšie stavové informácie.

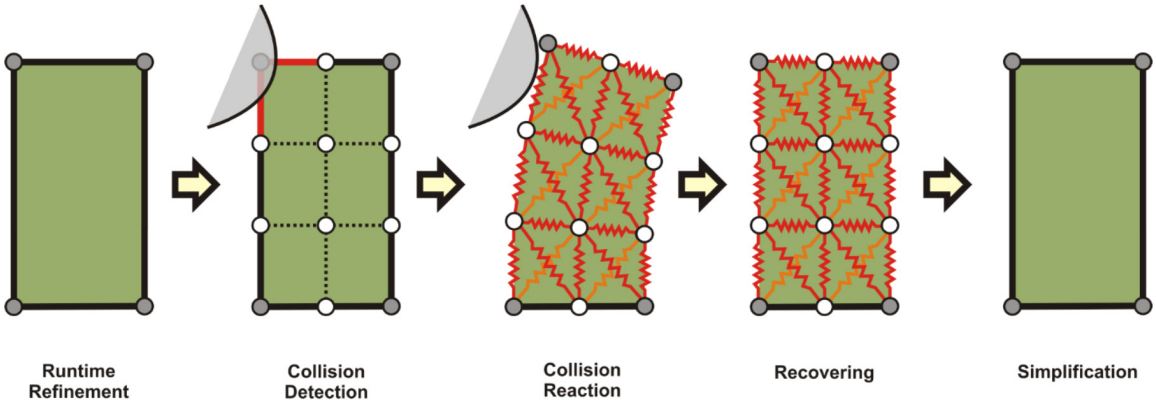
Kolízny systém

Horné dva vertexy quadu sú transformované veternou funkciou pozostávajúcej z viacerých trigonometrických funkcií [9, 15]. Kolízny systém je rozdelený na CPU a GPU časti. CPU časť vykonáva hrubé spracovanie. GPU časť pozostáva z dvoch procedúr, jedna vykonáva kolízny test a odozvu, zatiaľ čo druhá obnovuje steblá do pôvodného stavu.

Hrubé spracovanie na CPU je vykonávané pre každý snímok. Hraničné objemy (angl. *bounding volumes*) objektov, ktoré môžu kolidovať, sú otestované na kolíziu s hraničnými schránkami (angl. *bounding boxes*) octree štruktúry. Podľa stavu každého uzla a výsledku testu je uzol označený buď ako *pravdepodobne kolidujúci*, *nekolidujúci* alebo *obnovujúci sa*.

Geometria priradená uzlu, ktorý je označený ako *pravdepodobne kolidujúci* je spracovaná prvým geometry shaderom. Ten pre zadaný quad vypočíta hraničnú sféru (angl. *bounding sphere*), na ktorej prevedie kolízny test s hraničnými sférami možných kolidujúcich objektov. Ak je tento test úspešný, prevedie sa druhý, presnejší test. Quad sa rozdelí na menšie časti a pre všetky jeho vertexy je určené, či ležia vnútri, alebo vonku kolidujúceho objektu. Vertexy ležiace v objekte sú následne posunuté von z objektu.

Každá kolízia uzlu resetuje *počítadlo obnovenia*. Počas doby obnovovania je uzol označený ako *obnovujúci sa* a vertexy jeho quadu sa postupne vracajú na pôvodnú pozíciu. Po návrate do pôvodného stavu je quad zjednodušený odstránením prebytočných vertexov. Celý kolízny proces je zobrazený na obr. 2.10.



Obr. 2.10: Kolízny systém použitý v metóde Orthmann (2009). Po detekovaní kolízie s objektom je quad rozdelení na menšie časti. Novo-vzniknuté vertexy sú posunuté mimo objekt, ak sa v ňom predtým nachádzali. Po kolízií sa vertexy postupne vracajú do pôvodného stavu. V poslednom kroku je quad zjednodušený na pôvodnú podobu. [8]

[6]Octree - hierarchická dátová štruktúra, v ktorej má vnútorný uzol vždy presne osem potomkov.

2.3 Techniky využívajúce geometriu

V nasledujúcej sekcií sú popísané vybrané techniky, ktoré modelujú steblá trávy ako geometriu. Jedná sa o výpočtovo a implementačne náročnejšie techniky, ktoré ale poskytujú väčšie možnosti pri simulácií pohybu stebiel a ich osvetlení.

2.3.1 Predgenerované steblá trávy

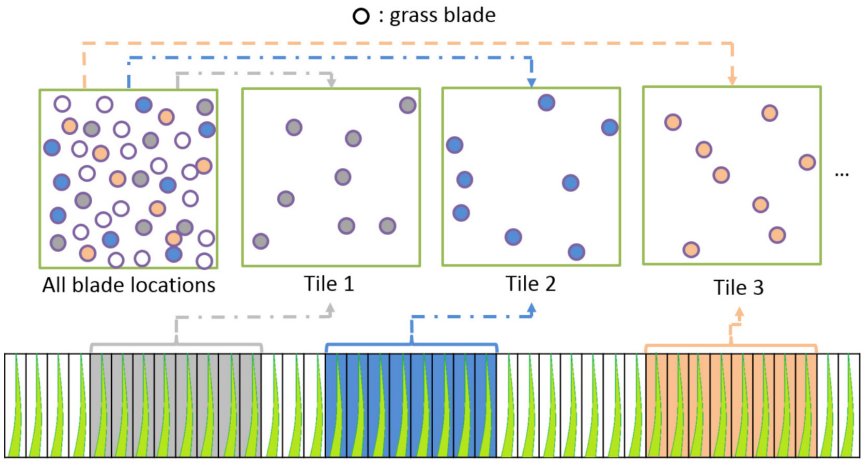
Fan et al.[2] využíva predgenerovaný štvorec trávnika, ktorý obsahuje zoznam všetkých stebiel, ich základnú geometriu a pozíciu v tomto štvorci trávnika. Celý trávnik je rozdelený na tzv. *dlaždice* (angl. *tiles*), ktorých dáta stebiel sú inštancované z predgenerovaného štvorca.

Umiestnenie stebiel

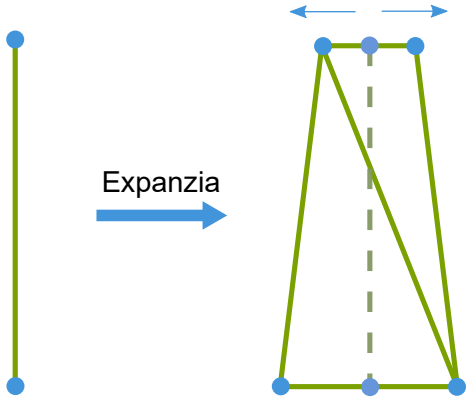
Každá dlaždica má rovnaký tvar ako predgenerovaný štvorec a referuje na podmnožinu stebiel z tohto štvorca. Začiatočná pozícia v zozname stebiel pre danú dlaždicu je daná pseudo-náhodnou funkciou, ktorá má na vstupe index dlaždice. Pri renderovaní dlaždice je každé steblo získané z predgenerovaného štvorca na základe začiatočnej pozície dlaždice a indexu stebla tejto dlaždice. Pozri obr. 2.11.

Expanzia geometrie

Steblá sú v predgenerovanom zozname reprezentované krivkou, ktorú je pri renderovaní potrebné expandovať na trojuholníky. Každý čiarový segment tejto krivky je expandovaný na jeden degenerovaný quad. Šírka expanzie je postupne znižovaná smerom k vrcholu stebla. Tento proces je zobrazený na obrázku 2.12.



Obr. 2.11: Ukážka predgenerovaného štvorca trávnika (vľavo hore), ktorý obsahuje zoznam vopred vymodelovaných stebiel (dole) reprezentovaných krivkami. Každá dlaždica trávnika je určitou podmnožinou predgenerovaného štvorca, z ktorého si vyberá bloky stebiel.[2]



Obr. 2.12: Ukážka expanzie segmentu krivky na dva trojuholníky. S rastúcou výškou stebla sa šírka medzi vertexami znižuje.[2]

2.3.2 Modelovanie stebiel využitím teselácie

Technika navrhnutá Jahrmann et al.[6] renderuje všetky steblá trávy ako geometriu využitím hardvérovej teselácie. Jahrmann et al. popisujú vo svojom článku dva odlišné postupy na základe veľkosti trávnatej plochy.

Prvý z nich je vhodný pre menšie trávnaté plochy. V tomto prípade sa ukladajú geometrické dáta každého stebla individuálne. To vedie k dostačujúcej náhodnosti rozmiestnenia stebiel, takže trávnik pôsobí prirodzene. Keďže je ale každé steblo uložené v pamäti, je tento postup ťažko škálovateľný pre väčšie plochy.

Druhý postup využíva inštancovanie. V pamäti je uložená len jediná inštancia časti trávnika, ktorá je následne vykreslená niekoľko krát. To umožňuje efektívne renderovať veľké trávnaté plochy v reálnom čase. Bežnou vadou inštancovania sú opakujúce sa vzory, ale keďže je každé steblo generované v shaderoch, dá sa táto vada ľahko maskovať. Tvar stebla je možné ovplyvniť pozíciou stebla v scéne a zakryť tým prechody medzi kusmi trávnika.

Inicializácia

Celý trávnik je reprezentovaný ako mriežka fixnej veľkosti. Každá bunka tejto mriežky predstavuje jeden kus trávnika. Veľkosť buniek závisí od konkrétnej scény, určuje sa teda podľa požadovanej úrovne detailu. Nesmie byť ale príliš veľká, pretože výrazne ovplyvňuje výkon aplikácie. Každý kus trávnika má v sebe uložené všetky potrebné dáta pre vykreslenie stebiel trávy, ktoré na ňom rastú.

Výsledný vzhľad trávnika je určený viacerými textúrami a parametrami, ktoré sú špecifikované počas inicializácie a slúžia ako vstup pre vykresľovací proces. Jednotlivé parametre sú popísané nižšie.

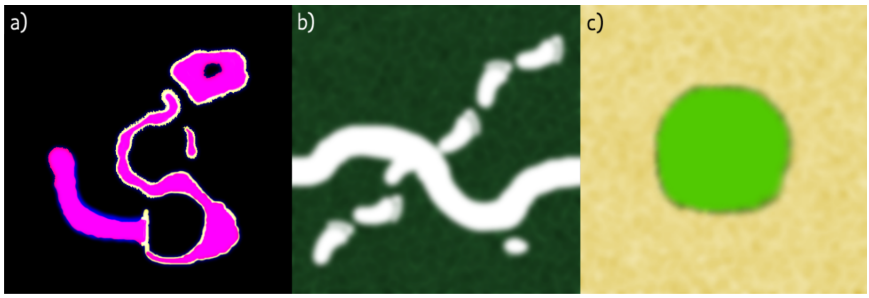
- Hustota: určuje počet stebiel trávy inicializovaných v bunke 1x1
- Šírka: minimálna a maximálna šírka stebla
- Výška: minimálna a maximálna výška stebla
- Faktor ohybu: určuje maximálnu úroveň ohnutia stebla
- Faktor hladkosti: špecifikuje maximálnu úroveň teselácie stebla

Zatiaľ čo celkový vzhľad určujú vyššie popísané parametre, vzhľad trávy v rozličných častiach scény je ovplyvnený troma textúrami. Pozri 2.13.

Mapa hustoty stanovuje hustotu a výšku trávy. Červený kanál udáva hustotu stebiel. Zelený kanál určuje miesta, kde sa bude generovať aj tráva s nižšími detailmi, ktorá je využitá napríklad pri vykresľovaní odrazov vodnej plochy. Modrý kanál obsahuje výšku stebla v danom bode.

Mapa terénu predstavuje textúru pod trávou. Keďže nie je tráva na všetkých miestach dostatočne hustá, je potrebné plochu pod trávou pokryť textúrou.

Mapa vegetácie obsahuje informáciu o zafarbení trávy v jednotlivých častiach trávnika. Využíva sa napríklad, keď chceme v určitých miestach rozlíšiť trávu, ktorá je suchá, alebo inak zafarbená.



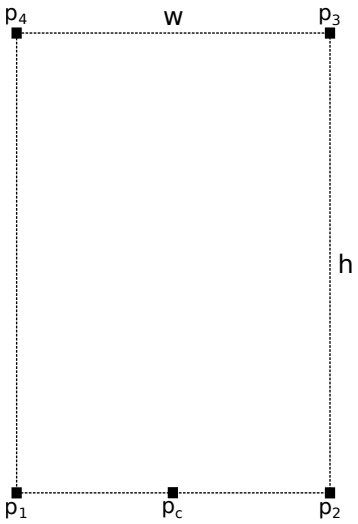
Obr. 2.13: Textúry použité pri generovaní trávy. a) mapa hustoty, b) mapa terénu, c) mapa vegetácie. Prevzaté z [6].

Vytváranie geometrie

Po konci inicializácie sa začne generovať základná geometria pre nízko-detailnú a vysoko-detailnú verziu trávnika. Navzorkuje sa mapa hustoty a pre každý jej pixel sa vyráta výsledná pozícia vo svete. Táto oblasť sa vynásobí so vzorkou červeného kanála a parametrom hustoty, pre určenie počtu stebiel, ktoré sa majú v tejto oblasti vygenerovať. Pre každé steblo sa vyráta náhodná pozícia a pridá sa do zoznamu stebiel s vysokým detailom. Ak je v tejto oblasti zelený kanál nenulový, vygeneruje sa geometria aj do zoznamu stebiel s nízkym detailom. Počiatočnú geometriu pre každé steblo predstavuje jeden obrysový obdĺžnik s nasledujúcimi vertexami:

$$
\begin{aligned}
w &= w_{min} + R \cdot (w_{max} - w_{min}) \\
h &= (h_{min} + R \cdot (h_{max} - h_{min})) \cdot hustota_b \\
\mathbf{p}_1 &= \mathbf{p}_c + [\text{-}0.5 \cdot w, 0.0, 0.0] \\
\mathbf{p}_2 &= \mathbf{p}_c + [\ 0.5 \cdot w, 0.0, 0.0] \\
\mathbf{p}_3 &= \mathbf{p}_c + [\ 0.5 \cdot w, h, 0.0] \\
\mathbf{p}_4 &= \mathbf{p}_c + [\text{-}0.5 \cdot w, h, 0.0]
\end{aligned}
\tag{2.1}
$$

Kde w je šírka a h je výška stebla s maximálnymi (w_{max}, h_{max}) a minimálnymi (w_{min}, h_{min}) hodnotami. R je náhodne vygenerované číslo od nula do jedna, $hustota_b$ je hodnota modrej vzorky získaná z mapy hustoty. $\mathbf{p}_c$ je stredový bod stebla (súradnice [0.0, 0.0, 0.0]) a $\mathbf{p}_i$ je i-tý bod obdĺžnika. Výsledné body je možné vidieť na obr. 2.14.



Obr. 2.14: Obrysový obdĺžnik použitý pre generovanie tvaru stebla trávy. Body p_1 až p_4 predstavujú okrajové body obdĺžnika a p_c je stredový bod stebla. w je šírka a h výška konkrétneho stebla.

Každý zo štyroch bodov $\mathbf{p}_i$ predstavuje vertex, ktorý je poslaný na grafickú kartu spolu s ďalšími hodnotami. Okrem pozície vertexu (x_p, y_p, z_p) sa pripája pozícia stredového bodu stebla (x_c, y_c, z_c) s hodnotou y_c rovnej nule, ak sa jedná o dolný vertex a hodnotou jedna, ak sa jedná o horný vertex. Ďalej sa posielajú textúrové súradnice (s, t) a sada náhodne vygenerovaných hodnôt (r_0 - r_7) pre dosiahnutie unikátnosti každého stebla. Vo výsledku tak každý vertex pozostáva zo štyroch vektorov o veľkosti štyroch floatov, nasledovne:

$$
\begin{aligned}
vec_1 &= \{x_p, y_p, z_p, r_0\} \\
vec_2 &= \{x_c, y_c, z_c, r_1\} \\
vec_3 &= \{s, t, r_2, r_3\} \\
vec_4 &= \{r_4, r_5, r_6, r_7\}
\end{aligned}
$$

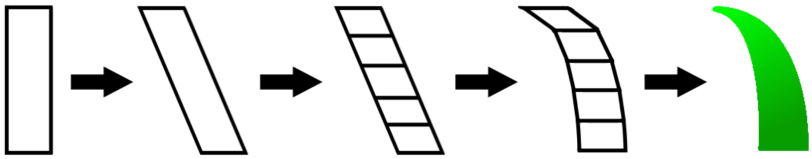
Generovanie stebla

Obdĺžnik z predchádzajúceho kroku je ďalej poslaný do teselačného shadera, kde je rozdelený na viacero menších obdĺžnikov. Novovzniknuté vertexy sú následne zarovnané podľa dvoch paralelných splinov[7]. Spliny sú vytvorené pomocou vrchných a spodných vertexov vstupného obdĺžnika a dvoch dodatočných kontrolných bodov, určujúcich zahnutie stebla. Finálny tvar stebla určuje alfa textúra. Tento proces je popísaný nižšie a zobrazený na obr. 2.15.

Počas vertex shader fázy sa na všetky vertexy aplikuje rotácia okolo centra, pre dosiahnutie náhodnej orientácie stebla. Ako uhol sa použije jedna z náhodných hodnôt. Vrchné vertexy (identifikované pomocou hodnoty y stredového bodu) sa posunú náhodne v osiach x a z. Vektor posunu je daný dvoma ďalšími náhodnými hodnotami R_i vynásobenými faktorom ohybu b nasledovne:

$$
posun = \begin{bmatrix} b \cdot (2R_1 - 1) \\ 0 \\ b \cdot (2R_2 - 1) \end{bmatrix} \tag{2.2}
$$

[7]Spline - polynomická funkcia definovaná po intervaloch, aproximujúca krivku.



Obr. 2.15: Obrázok ukazuje postup generácie stebla. V prvom kroku je zobrazený obrysový quad, ktorý určuje hranice stebla. V ďalšom kroku sú posunuté horné vertexy v osiach x a z. Posunutý quad je ďalej rozdelený na viacero menších quadov využitím teselácie. Novovzniknuté vertexy sú následne zarovnané pozdĺž dvoch paralelných splinov. Posledný krok zobrazuje aplikáciu alfa textúry a ofarbenie vo fragment shaderi. [6].

Tessellation control shader slúži na výpočet úrovne teselácie. Najskôr sa vypočíta vzdialenosť stebla od kamery a určí sa maximálna vzdialenosť vykresľovania. Výsledná úroveň $\mathbf{n}$ vznikne lineárnym interpolovaním hodnôt od nuly až po faktor hladkosti s. Tento výpočet je popísaný rovnicou 2.3, kde d je vzdialenosť stebla od kamery a d_{max} je maximálna vzdialenosť vykresľovania.

$$\mathbf{n} = \lceil s \cdot (1 - \frac{d}{d_{max}}) \rceil \tag{2.3}$$

Hodnota $\mathbf{n}$ je použitá ako *vnútorná úroveň teselácie* a *vonkajšia úroveň teselácie* pre pravú a ľavú stranu obdĺžnika. Ak je vzdialenosť od kamery väčšia ako maximálna vzdialenosť je táto hodnota nulová. Nulová hodnota znamená kompletné zahodenie stebla. Nenulová hodnota $\mathbf{n}$ znamená rozdelenie obdĺžnika na $\mathbf{n}$ pod-obdĺžnikov pozdĺž osi y.

V tom istom shaderi sa ďalej vypočítajú dva dodatočné kontrolné body, ktoré pomôžu určiť tvar stebla. Kontrolný bod $\mathbf{h}$ pre ľavú a pravú stranu obdĺžnika sa vypočíta spôsobom popísaným v rovnici 2.4, kde $\mathbf{l} = (\mathbf{l}_x,\mathbf{l}_y,\mathbf{l}_z)$ a $\mathbf{u} = (\mathbf{u}_x,\mathbf{u}_y,\mathbf{u}_z)$ predstavujú dolný a horný vertex a R_i sú náhodné hodnoty.

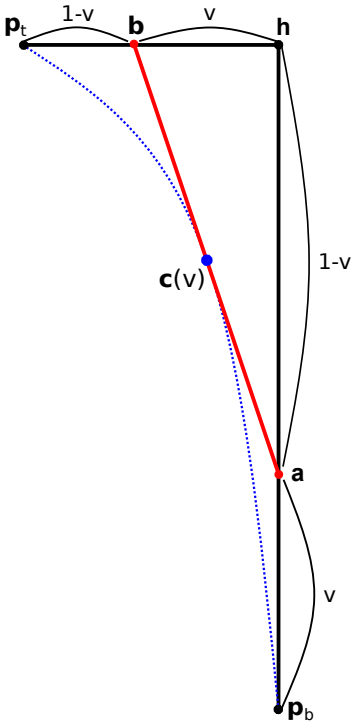
$$\mathbf{h} = \begin{bmatrix} \mathbf{l}_x \cdot R_1 + \mathbf{u}_x \cdot (1 - R_1) \\ \mathbf{l}_y \cdot R_2 + \mathbf{u}_y \cdot (1 - R_2) \\ \mathbf{l}_z \cdot R_1 + \mathbf{u}_z \cdot (1 - R_1) \end{bmatrix}, R_1 \in \left\langle -\frac{1}{4}, \frac{1}{4} \right\rangle, R_2 \in \left\langle \frac{3}{4}, \frac{5}{4} \right\rangle \tag{2.4}$$

V tessellation control shaderi sú vertexy teselovaného obdĺžnika zarovnané pozdĺž dvoch paralelných kvadratických splinov. Každý zo splinov je definovaný troma kontrolnými bodmi. Pre vypočítanie bodu na krivke $c(v)$ s parametrom v je použitý algoritmus De Casteljau[3]. Parameter v predstavuje pomer i/n pre i-tý pod-obdĺžnik, kde n je úroveň teselácie (počet pod-obdĺžnikov). Vyhodnotenie vertexu na jednom spline je možné vidieť v rovnici 2.5 a na obrázku 2.16a, kde $\mathbf{p}_b$ a $\mathbf{p}_t$ predstavujú dolný a horný vertex a $\mathbf{h}$ je dodatočný kontrolný bod pre daný spline. Z týchto parametrov je taktiež možné vypočítať dotyčnicu $\vec{t}$ splinu.

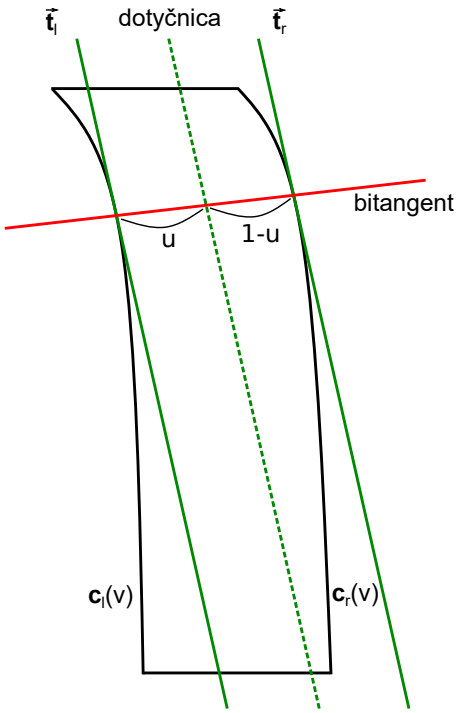
$$\begin{aligned} \mathbf{a} &= \mathbf{p}_b + v \cdot (\mathbf{h} - \mathbf{p}_b) \\ \mathbf{b} &= \mathbf{h} + v \cdot (\mathbf{p}_t - \mathbf{h}) \\ \mathbf{c}(v) &= \mathbf{a} + v \cdot (\mathbf{b} - \mathbf{a}) \\ \vec{t} &= \frac{\mathbf{b} - \mathbf{a}}{\|\mathbf{b} - \mathbf{a}\|} \end{aligned} \tag{2.5}$$

Po vypočítaní bodov na oboch splinoch ($\mathbf{c}_l$(v) a $\mathbf{c}_r$(v) pre ľavý a pravý spline), finálna pozícia vertexu a dotyčnica môžu byť interpolované použitím parametra v. Bitangent[8] vyplýva z vypočítaných bodov na splinoch. Normálu je možné vypočítať vektorovým súčinom dotyčnice a bitangentu. Tento postup je popísaný v rovnici 2.6 a znázornený na obrázku 2.16b.

$$\begin{aligned} pozícia &= \mathbf{c}_l(v) \cdot (1-u) + \mathbf{c}_r(v) \cdot u \\ bitangent &= \frac{\mathbf{c}_r(v) - \mathbf{c}_l(v)}{\|\mathbf{c}_r(v) - \mathbf{c}_l(v)\mathbf{a}\|} \\ dotyčnica &= \frac{\vec{t_l} \cdot (1-u) + \vec{t_r} \cdot u}{\|\vec{t_l} \cdot (1-u) + \vec{t_r} \cdot u\|} \\ normála &= \frac{dotyčnica \times bitangent}{\|dotyčnica \times bitangent\|} \end{aligned} \tag{2.6}$$



(a) Využitie algoritmu De Casteljau[3] pre vypočítanie bodu na krivke $\mathbf{c}$(v) parametrom v. Body $\mathbf{p}_b$, $\mathbf{p}_t$ a $\mathbf{h}$ predstavujú kontrolné body kubického splinu.



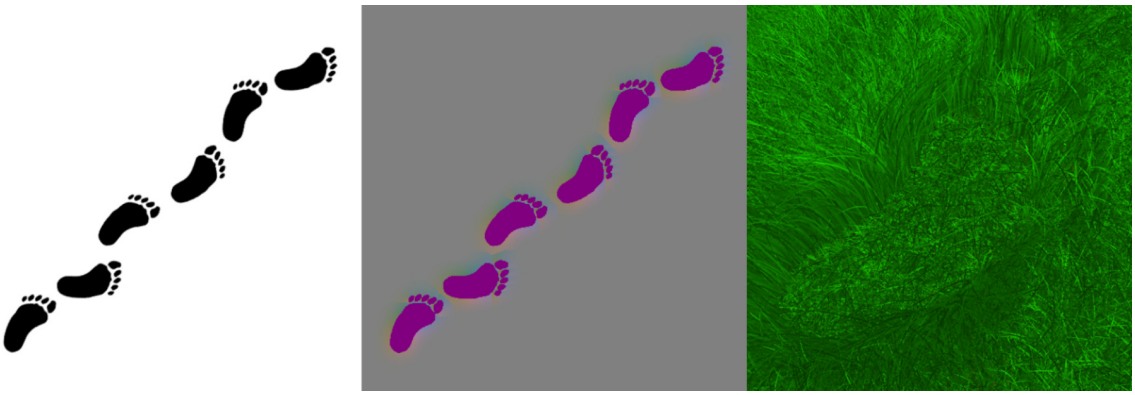
(b) Ukážka výpočtu dotyčnice a bitangentu stebla trávy.

Obr. 2.16: Generovanie výslednej geometrie stebla trávy. Prevzaté a upravené z [6].

Finálny tvar stebla je formovaný maskovaním alfa textúry. Difúzna farebná zložka je vzorkovaná buď z difúznej textúry, alebo vegetačnej mapy, popísanej v časti *Inicializácia*. Pre dosiahnutie lepšej farebnej variácie jednotlivých stebiel sú jednotlivé farebné komponenty modifikované troma náhodnými hodnotami. Pre simulovanie jednoduchého tieňa je

[8]Bitangent - čiara, ktorá je dotyčnicou krivky v dvoch rozdielnych bodoch. *https://mathworld.wolfram.com/Bitangent.html*

Obr. 2.17: Textúry pre finálny vzhľad stebla. Alfa textúra (hore) pre určenie tvaru a difúzna textúra (dole) pre určenie farby.[6]



Obr. 2.18: Obrázok naľavo ukazuje tvar stôp, v strede je zobrazená korešpondujúca force mapa a napravo je výsledná scéna s aplikovanou force mapou.[6]

možné steblo v spodnej časti stmaviť a v hornej naopak zosvetliť. To sa dá dosiahnuť násobením farby vertikálnou pozíciou textúry. Ukážku alfa textúry a difúznej textúry je možné vidieť na obrázku 2.17.

Vietor a kolízie

Pre simulovanie pohybu trávy vo vetre je využitá funkcia vetra $w(\mathbf{p})$. Parametrami funkcie sú pozícia v scéne $\mathbf{p} = (\mathbf{p}_x,\mathbf{p}_y,\mathbf{p}_z)$ a aktuálny čas t. Táto funkcia vypočítava vietor ako dve prekrývajúce sa vlny (sínus a kosínus) pozdĺž osi x a jednu kosínovú vlnu pozdĺž osi z. Funkcia je popísaná rovnicou 2.7, kde konštanty c_i upravujú tvar vetra a ε predstavuje malé číslo, pre prípad delenia nulou. Výsledok je následne aplikovaný ako posun horných vertexov stebla.

$$\begin{aligned} w(\mathbf{p}) &= \sin(c_1 \cdot a(\mathbf{p})) \cdot \cos(c_3 \cdot a(\mathbf{p})) \\ a(\mathbf{p}) &= \pi \cdot \mathbf{p}_x + t + \frac{\frac{\pi}{4}}{|\cos(c_2 \cdot \pi \cdot \mathbf{p}_z)| + \varepsilon} \end{aligned} \tag{2.7}$$

Okrem pohybu vo vetre môže byť každé steblo ovplyvnené aj vonkajšími vplyvmi ako je napríklad pohyb objektov po trávniku. Pre tento prípad je využitá force (silová) mapa, ktorá indikuje smer tlaku na steblo na danej pozícií. V mieste úplného stlačenia smerujú vektory mapy k negatívnemu y (smer dole) zatiaľ čo na okrajoch smerujú vektory von z centra objektu. Vektory v oblasti okolo objektu smerujú takisto z centra objektu, ale už so znižujúcou sa dĺžkou. Veľkosť takejto oblasti záleží na sile dopadu daného objektu. Príklad takejto force mapy je zobrazený na obrázku 2.7. Vzorkovaný vektor môže byť potom pridaný do posunu horných vertexov stebla.

Rozšírenie metódou Jahrmann-Wimmer (2017)

Jahrmann et al. predstavujú v novom článku[7] rozšírenie pôvodnej metódy. Toto rozšírenie dovoľuje vykresliť trávu na povrch ľubovoľného tvaru a orientácie, teda akýkoľvek 3D model. Metóda využíva fyzikálny model počítaný pre každé steblo, ktorý berie do úvahy pôsobenie vetra, kolízií a gravitácie. Obrázok 2.19 ukazuje trávu vykreslenú touto metódou na komplexných 3D objektoch.

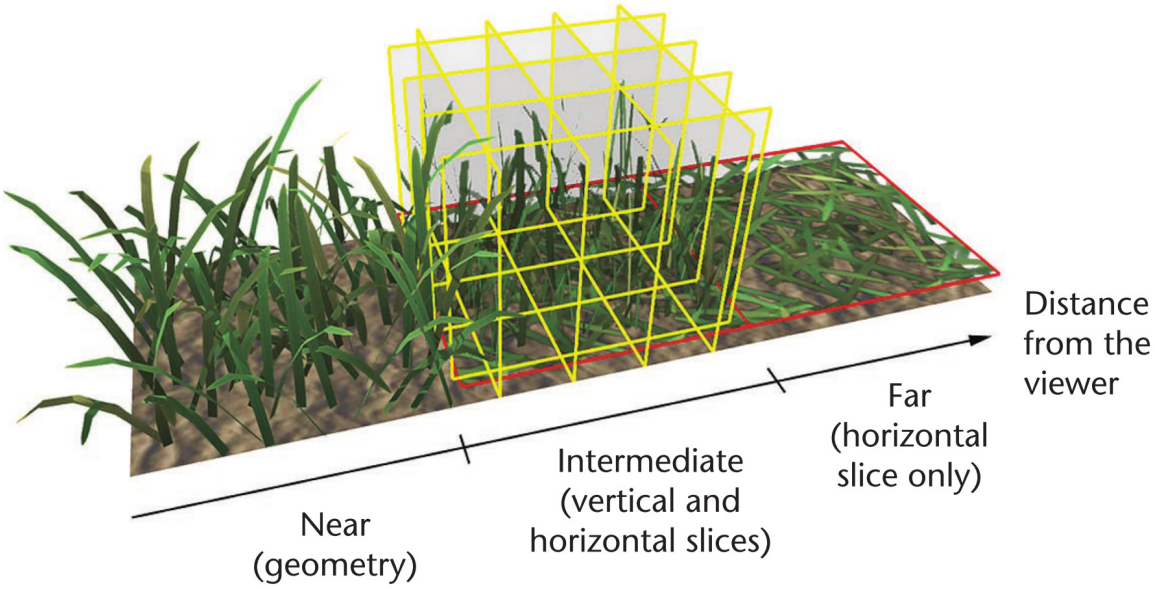
Obr. 2.19: Tráva vykreslená na komplexných 3D objektoch využitím metódy Jahrmann-Wimmer (2017). Objekty používajú pre vzhľad trávy rôzne farebné textúry.[7]

2.4 Hybridné techniky

Táto sekcia obsahuje techniky, ktoré využívajú kombináciu vyššie zmieňovaných techník, alebo zasahujú do iných oblastí počítačovej grafiky.

2.4.1 Metóda Boulanger(2009)

Boulanger et al.[1] využíva pre renderovanie kombináciu troch techník, meniacich sa podľa LOD. Tráva najbližšie ku kamere je modelovaná ako geometria, v strednej vzdialenosti sú využité tzv. *zväzkové diely* (angl. *volume slices*) a najvzdialenejšiu trávu reprezentujú horizontálne 2D diely pre steblá menšie ako jeden pixel. Pozri obr. 2.20. Aj keď je využitá LOD schéma, špecifikovať parametre každého stebla by zaberalo veľa pamäte. Preto je využité inštancovanie. Terén je ako aj v predchádzajúcich metódach rozdelený podľa rovnomernej mriežky. Cez každú bunku tejto mriežky je preložená základná *dlaždica* (angl. *grass patch*), ktorá obsahuje niekoľko tisíc stebiel.



Obr. 2.20: Zobrazené sú tri úrovne detailu. Každá z nich využíva jednu techniku renderovania trávy. Na ľavo je tráva, ktorá je najbližšie ku kamere modelovaná ako geometria. V strede sú využité tzv. *zväzkové diely* (vertikálne a horizontálne). Na pravo, kde je tráva od kamery najďalej, sú využité len horizontálne diely.[1]

Geometria

Základná dlaždica v tomto prípade obsahuje steblá reprezentované geometriou. Steblo trávy je aproximované dvoj-strannými *obdĺžnikovými pásmy* (angl. *quad strips*) o nulovej hrúbke. Výsledný tvar dodáva alfa kanál textúry, ktorá tento pás pokrýva. Tento kanál obsahuje antializovanú šedotónovú verziu stebla. Pre určenie tvaru stebla sú využité trajektórie generované particle systémom[13] (*systémom častíc*). Častica je pod vplyvom gravitácie vypustená skoro vertikálne od koreňa stebla. Pozícia častice je vyhodnotená viac krát za sebou, čo poskytuje referenčné body, z ktorých sa určia pozície vertexov.

Zväzkové diely

V tomto prípade je základnou dlaždicou zväzok, ktorý obsahuje 2D diely. Šírka a hĺbka zväzku korešpondujú s rozmermi bunky, zatiaľ čo najvyššie steblo trávy určuje jeho výšku. Tento zväzok reprezentujú polo-priehľadné 2D diely, ktoré sú zarovnané podľa svetových osí. Každý diel predstavuje quad, na ktorom je namapovaná 2D textúra. Vzdialenosť medzi dielmi je fixná pre zjednodušenie inštancovania. Pretože je tráva v reálnom svete v podstate vertikálna, používa sa len jeden horizontálny diel, pre pohľad z výšky. Zvyšné diely sú vertikálne.

Horizontálne diely

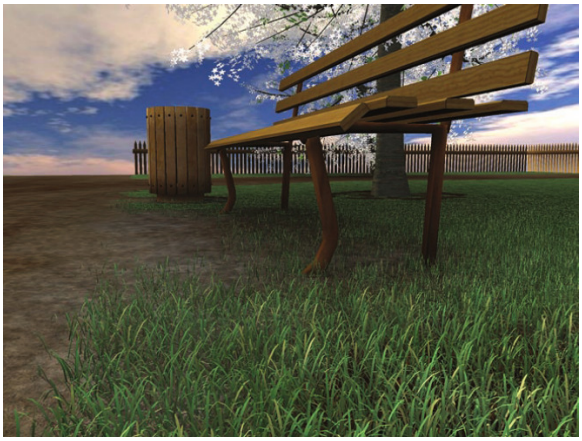
Tráva v pozadí sa javí byť veľmi malá, preto sa na jej reprezentáciu použije len jeden horizontálny diel pre každú viditeľnú bunku trávnika.

Manažment hustoty

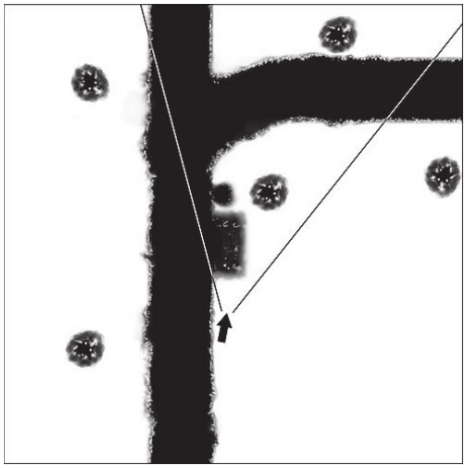
Hustota trávy v jednotlivých bodoch terénu je definovaná *mapou hustoty*. Každé steblo základnej dlaždice si ukladá hodnotu v rozmedzí od nula do jedna. Táto hodnota sa nazýva *prah hustoty*. Počas renderovania je pre každú dlaždicu získaná vzorka z mapy hustoty. Ak je hodnota tejto vzorky menšia ako *prah hustoty* daného stebla, steblo sa zahodí.

Pri geometrii je *prah hustoty* nastavený ako konštantná hodnota pre každý vertex stebla. Porovnanie vzorky získanej z *mapy hustoty* a *prahu hustoty* je vykonávané pre každý fragment. Spracovávanie pre každý vertex by značne obmedzilo výkon. Ukážku aplikovania hustoty na geometriu je možné vidieť na obr. 2.21.

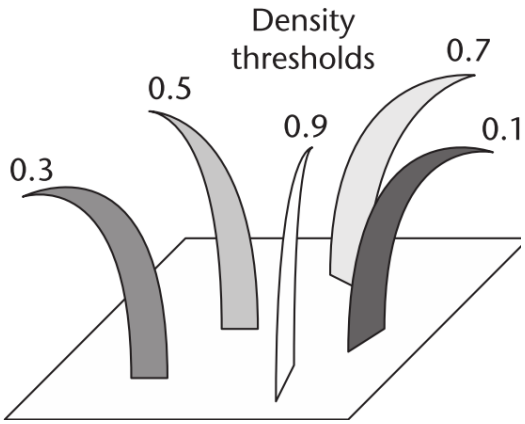
Pri určovaní hustoty stebiel 2D dielov je použitá samostatná šedotónová textúra (pozri obr. 2.22). Hodnoty šedi udávajú hodnotu *prahu hustoty* pre každé steblo. Pre vygenerovanie šedotónovej textúry je vykreslená dlaždica s geometriou medzi dve *vystrihovacie roviny* (angl *clipping planes*). Farba stebiel tejto dlaždice je rovná hodnote šedi, proporciálnej k ich hodnote *prahu hustoty*.



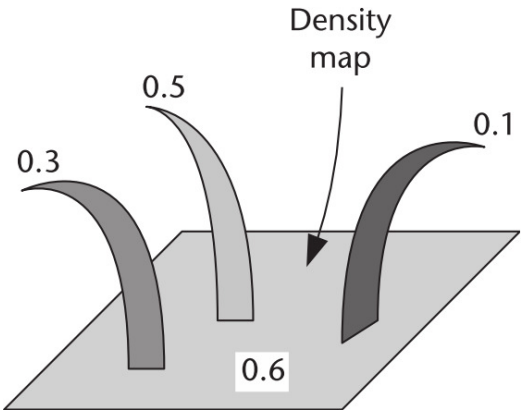
(a) Výsledná hustota.



(b) Mapa hustoty.

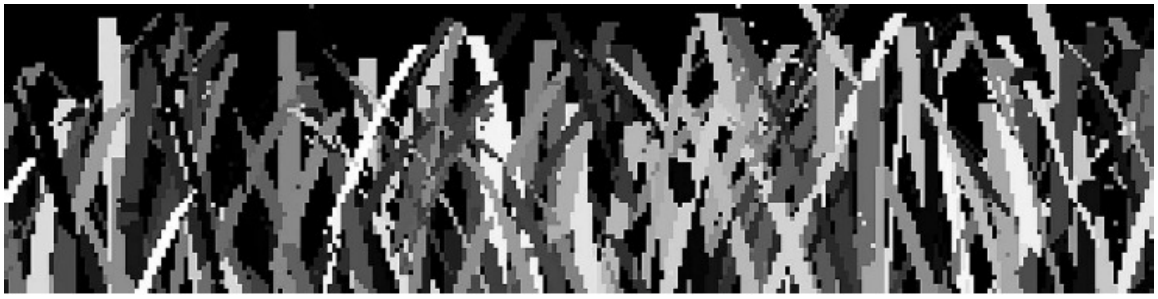


(c) Hodnoty prahov hustoty stebiel.



(d) Zvyšné steblá po aplikovaní hustoty.

Obr. 2.21: Určovanie hustoty trávy pri geometrii. (a) Ukážka scény s výslednou trávou. (b) Mapa hustoty definuje hustotu stebiel v každom bode terénu. Biele miesta predstavujú maximálne hustú trávu, zatiaľ čo čierne miesta neobsahujú žiadne steblá. Šípka označuje smer kamery. Každé steblo v základnej dlaždici má určený prah hustoty (c). Ak je hodnota vzorky menšia ako prah hustoty daného stebla, steblo sa zahodí (d).[1]



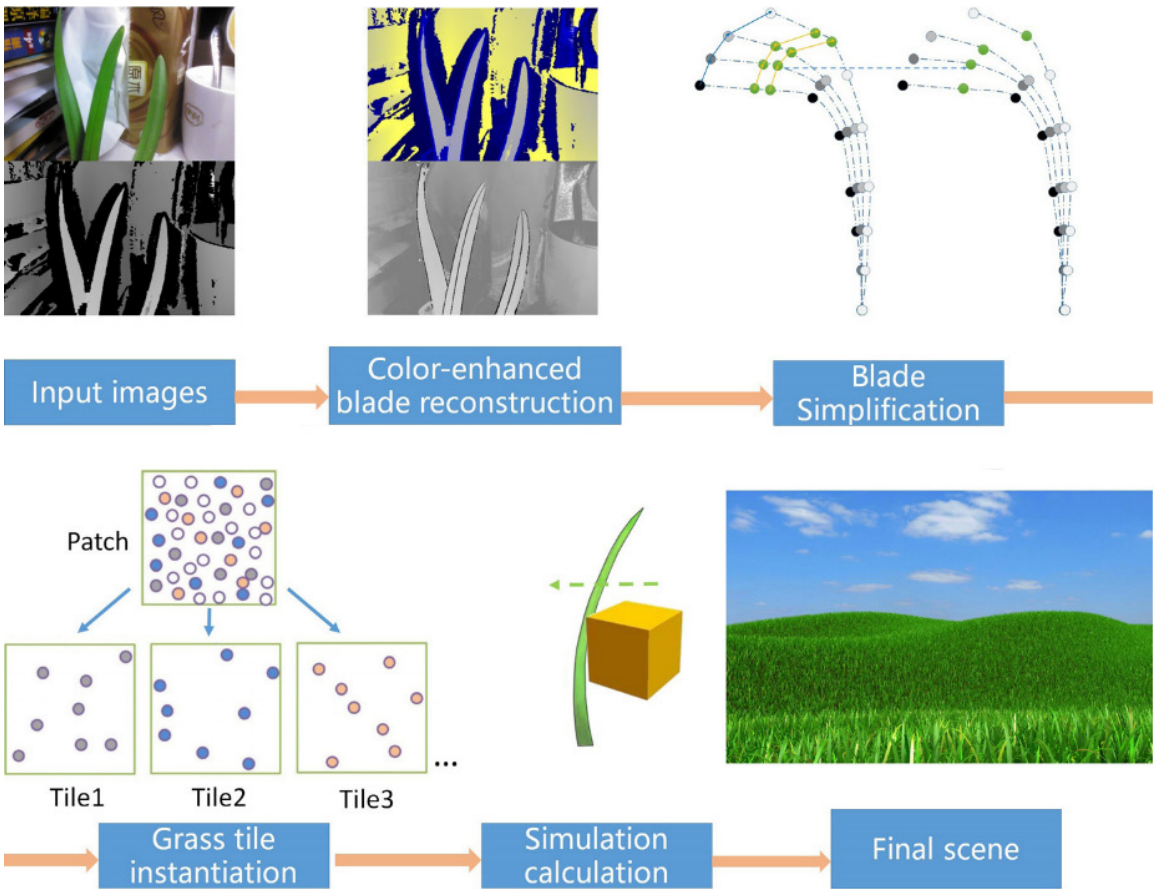
Obr. 2.22: Šedotónová textúra použitá na určenie prahu hustoty jednotlivých stebiel. Prah hustoty určuje zahodenie stebla, ak je vzorka mapy hustoty menšia ako daný prah.[1]

2.4.2 Rekonštrukcia stebiel trávy z obrázkov

Wang et al.[14] predstavuje techniku rekonštrukcie modelu stebla z farebnej hĺbkovej mapy. Takto získaný model je následne zjednodušený kvôli zníženiu počtu vertexov a vykreslený. Obrázky sú získané z kamery založenej na snímaní *štruktúrovaného svetla* [9].

Model stebla získaného z hĺbkovej mapy je zjednodušený na čiarové segmenty a ďalej expandovaný, podobne ako pri metóde Fan et al.[2]. Pre simuláciu stebiel trávy táto technika rozširuje inú metódu, navrhnutú na simuláciu vlasov (Han et al.[5]). Manažment dlaždíc trávnika používa princípy metódy Fan et al., ktorú implementuje celú na GPU, čo zmierňuje nápor na CPU. Celý proces je zobrazený na obrázku 2.23.

[9] Štruktúrované svetlo - proces projekcie známeho vzoru (napr. mriežky) do scény. Na základe deformácie tohto vzoru na objekte, je možné vypočítať informácie o hĺbke a povrchu daného objektu.



Obr. 2.23: Rekonštrukcia stebiel trávy z obrázkov. Model stebla, pozostávajúci z čiarových segmentov je získaný z farebnej hĺbkovej mapy. Tento model je následne zjednodušený a umiestnený do zoznamu stebiel. Steblá sú rozdelené do skupín podľa dlaždíc trávnika, na ktorých sa nachádzajú. Následne prebehne simulácia ich pohybu použitím techniky pre simuláciu vlasov a nakoniec sú vykreslené.[14]

2.5 Súčasne používané techniky

Táto sekcia obsahuje vybrané ukážky hlavných typov techník renderovania trávy, popísaných v kapitole 2. Ide o praktické ukážky 3D hier, vydaných v rokoch 2019 - 2020 na platforme *Playstation 4*.

Textúrované quady

V súčasnosti najprevalentnejšou technikou na vykreslenie trávových plôch je využitie quadov s polo-priehľadnou textúrou. Táto technika je jednoducho implementovateľná a nevyžaduje toľko prostriedkov ako použitie geometrie. Jednou z najnovších hier, ktorá túto techniku využíva pre renderovanie všetkej trávy je napríklad *Death Stranding* (pozri obr. 2.24).

Obr. 2.24: Snímok obrazovky zachytený z hry Death Stranding na konzole PS4 Pro. Hra využíva textúrované quady pre zhluky trávy (možné vidieť v spodnej časti). Terén pokrýva výrazná trávová textúra.

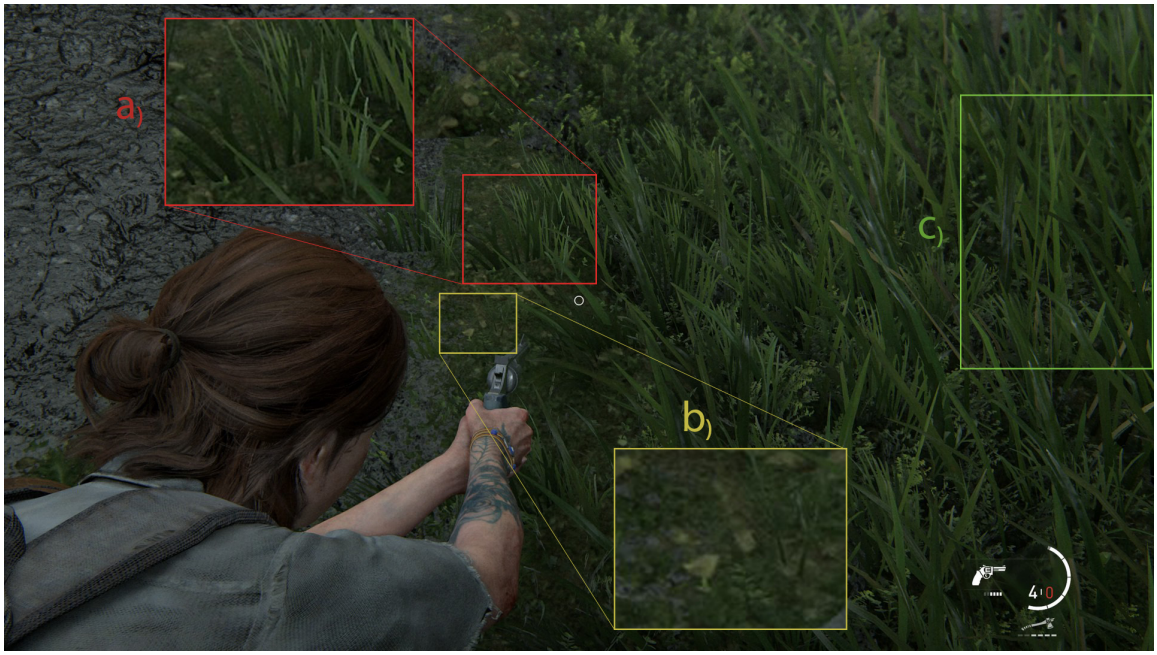
Geometria

Pre dosiahnutie čo najväčšieho detailu pri vykresľovaní trávy je najvhodnejšie modelovať steblá trávy pomocou geometrie. Hra *Ghost of Tsushima* využíva túto techniku pre drvivú väčšinu trávy s scéne. Steblá trávy na seba vzájomne vrhajú tiene a pohybujú sa pod náporom vetra samostatne. Aj napriek hardvérovým limitáciám konzoly Playstation 4 je v hre tráva vykreslená v dostatočnej hustote, aby pôsobila autenticky. Ukážku je možné vidieť na obr. 2.25.

Obr. 2.25: Snímok obrazovky zachytený z hry Ghost of Tsushima na konzole PS4 Pro. Táto hra modeluje väčšinu trávy ako geometriu. Na snímke je možné vidieť jednotlivé steblá deformované pod váhou koňa.

Kombinácia techník

Hra *The Last of Us Part II* využíva kombináciu hneď niekoľkých techník pre dosiahnutie autenticky pôsobiacej vegetácie, ako je možné vidieť na obrázku 2.26. Pre hustejšiu časť trávnika, kde má hráč možnosť schovať sa je použitá geometria, keďže umožňuje ohýbanie jednotlivých stebiel vplyvom pohybu hráča. Prechody medzi hustou trávou a zemou, alebo miesta s menej hustou trávou sú pokryté dvojrozmernými quadmi. Zatiaľ čo plocha pod týmito objektami je pokrytá textúrou trávy.



Obr. 2.26: Snímok obrazovky zachytený z hry The Last of Us Part II na konzole PS4 Pro. V tomto prípade je využitých niekoľko techník pre vykreslenie trávnika. a) 2D billboardy sú využité pri prechode medzi hustou časťou trávnika a terénom bez trávy. b) Terén pod trávou pokrýva trávová textúra. c) 3D geometria je využitá v časti, kde je tráva hustejšia a hráč s ňou môže interagovať.

Kapitola 3

Návrh aplikácie pre renderovanie trávy

Cieľom tejto práce je vytvorenie demonštračnej aplikácie renderujúcej trávnaté plochy. Aplikácia by mala implementovať jednoduché animácie a osvetlenie trávy. V tejto kapitole je popísaná zvolená metóda pre implementáciu renderovania trávy a ďalšie aspekty návrhu takejto aplikácie.

3.1 Použitá metóda

Pre generovanie stebiel trávy bude využitá metóda Jahrmann-Wimmer (2013)[6], ktorá je popísaná v sekcií 2.3.2. Táto metóda umožňuje vykresliť jednotlivé steblá trávy ako geometriu, čo dovoľuje sledovať trávu z viacerých uhlov a umožňuje simuláciu pohybu vo vetre pre každé steblo. Zvolená metóda taktiež predstavuje zo všetkých popísaných možností renderovania geometrie najmenšiu záťaž na CPU. Záťaž grafickej karty je možné takisto redukovať použitím inštancovania, popísaného v článku. Rozšírenie metódou Jahrmann-Wimmer (2017) použité nebude, pretože rozsah jeho implementácia značne prekračuje rozsah tejto práce[1].

3.2 Reprezentácia terénu

Vygenerované steblá trávy je potrebné umiestniť na terén. Najjednoduchším spôsobom je vytvoriť obdĺžnikový polygón, ktorý by reprezentoval plochú zem. Výsledná scéna by ale pôsobila neprirodzene a nebolo by možné pozorovať správanie trávy pri rozdielnych výškach. Ďalším spôsobom je načítavanie terénu ako vopred vytvoreného mesha. Táto možnosť rieši predchádzajúce problémy, ale neposkytuje možnosť užívateľovi jednoducho generovať nový terén. Riešením zvoleným v tejto aplikácií je generovanie terénu z textúry, ktorá je načítaná zo súborového systému. Užívateľ si bude schopný vybrať z viacerých predvytvorených textúr, alebo si načítať vlastnú.

Terén bude reprezentovaný mriežkou, ktorej rozmery bude možné špecifikovať pri generovaní nového terénu. Ďalej bude možné zvoliť rozlíšenie (počet riadkov a stĺpcov mriežky), teda presnosť vzorkovania z textúry terénu.

[1] Implementáciu tohto rozšírenia je možné vidieť v diplomovej práci pána Procházky[12].

3.3 Vyplnenie scény trávou

Rovnako ako terén, bude aj trávnik reprezentovaný mriežkou. Jednotlivé bunky mriežky budú predstavovať dlaždice s pevne daným počtom stebiel. Geometria stebiel bude vygenerovaná len raz, pre jednu dlaždicu. Tá sa vykreslí niekoľko krát, pre každú bunku mriežky využitím inštancovania. Tento prístup výrazne obmedzí prenosy dát medzi CPU a GPU, a prispeje k lepšiemu výkonu aplikácie.

Keďže musí mať každá dlaždica inú pozíciu v scéne, je potrebné vypočítať transformačné matice pre každú inštanciu a tieto dáta poslať na GPU spolu predgenerovanou geometriou.

Modelovanie stebiel

Geometria stebla bude vytvorená teseláciou jednoduchého quadu. Quady budú vygenerované s náhodnými rozmermi v rozsahoch špecifikovaných užívateľom. Zahnutie stebla bude definované dvoma paralelnými splinami (pozri obr. 2.16a) a jeho finálny tvar určí alfa textúra.

Úroveň detailu

Tráva v diaľke bude vykresľovaná s nižším detailom ako tráva v blízkosti kamery, kvôli obmedzeniu záťaže na GPU. Úroveň detailu je možné ovplyvniť postupným znižovaním úrovne teselácie. Pre ďalšie zlepšenie výkonu a redukciu aliasovania vzdialených častí trávnika je vhodné v diaľke vykresľovať menší počet stebiel. Podobne ako pri znižovaní úrovne teselácie sa bude so vzrastajúcou vzdialenosťou znižovať aj počet vykreslených stebiel. To je možné dosiahnuť ich postupným zahadzovaním, teda nastavením úrovne teselácie na nulu.

Animácie a Osvetlenie

Aby tráva v scéne nepôsobila staticky, je potrebné v scéne simulovať vietor. Pre simuláciu pohybu trávy vo vetre bude využitá funkcia vetra popísaná v rovnici 2.7.

Osvetlenie stebiel je možné dosiahnuť postupným prechodom farieb od vrcholu k spodnej časti stebla. Farba v hornej časti, kde je steblo vystavené najväčšiemu množstvu slnečného svetla bude svetlejšia. Naopak, farba v spodnej časti bude o niečo tmavšia, čo vytvorí jednoduchý efekt tieňa v spodnej časti trávy.

Kapitola 4

Implementácia

V tejto kapitole sú popísané implementačné detaily demonštračnej aplikácie renderujúcej trávnaté povrchy.

4.1 Použité technológie

Aplikácia je vytvorená v jazyku *C++* s využitím *OpenGL* ako grafického API. Pre prácu s ním je využitý framework *GPUEngine*[1], ktorý poskytuje rozhranie pre prácu s týmto API. *GPUEngine* je využitý taktiež pre načítavanie shaderov zo súborového systému.

Grafický framework *Qt*[2] zabezpečuje vytvorenie okna aplikácie v operačnom systéme, sprístupnenie *OpenGL API*, odchytávanie užívateľských vstupov a načítavanie textúr zo súborového systému.

Aplikácia poskytuje užívateľovi možnosť upravovať parametre generovania terénu a trávy počas behu aplikácie, prostredníctvom grafického rozhrania. Vykreslenie prvkov užívateľského rozhrania zabezpečuje knižnica *Dear ImGui*[3]. Pre zabezpečenie spolupráce tejto knižnice s frameworkom *Qt* je použitá knižnica *qtimgui*[4].

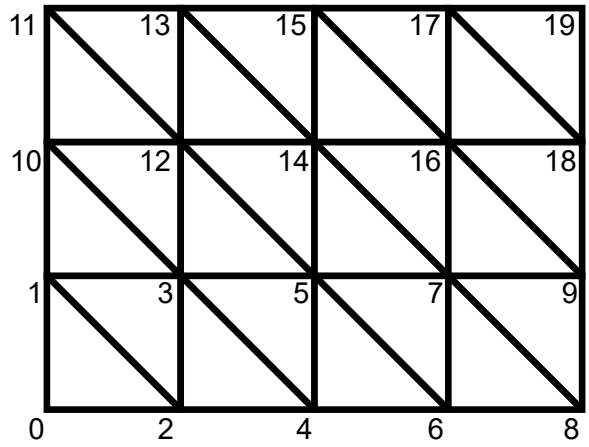
Pre automatizáciu prekladu je využitý nástroj *CMake*[5].

4.2 Generovanie terénu

Terén je generovaný na CPU ako mriežka vopred určených rozmerov. Vertexy mriežky sú posunuté pozdĺž osi y vzorkovaním výškovej mapy vo vertex shaderi. Textúrové súradnice sú vypočítané taktiež vo vertex shaderi na základe pozície vertexov v scéne a rozmerov terénu. Pre reprezentáciu terénu je použitá primitíva *trojuholníkových pásov* (angl. *triangle strips*), ktoré sú ukončované na konci riadku tzv. *restart indexom*. Každý riadok reprezentuje jeden trojuholníkový pás (viz. obr. 4.1). Počet riadkov a stĺpcov, ako aj rozmery mriežky sú špecifikované užívateľom. Keďže majú jednotlivé riadky spoločné vertexy je optimálne použiť indexové renderovanie. Výpočet vertexov a indexov mriežky terénu je popísaný algoritmom 1.

Generovanie vertexov začína v ľavom dolnom rohu mriežky. Stred terénu je v bode [0,0,0]. Funkcia *mix()* knižnice *GLM* určí pozíciu vertexu na ose x v rozmedzí šírky terénu

[1] Dostupné z: https://github.com/Rendering-FIT/GPUEngine
[2] Dostupné z: https://www.qt.io/
[3] Dostupné z: https://github.com/ocornut/imgui
[4] Dostupné z: https://github.com/seanchas116/qtimgui
[5] Dostupné z: https://cmake.org/



Obr. 4.1: Séria trojuholníkových pásov reprezentujúcich mriežku terénu. Čísla označujú indexy vertexov.

interpolovaním normalizovanej pozície stĺpca. Pri určovaní pozície na ose z sa znamienka vymenia, kvôli orientácií osi z v štandarde *OpenGL*. Funkcia *vec2(x,y)* knižnice *GLM* vytvorí zo zadaných hodnôt vektor. Pri generovaní indexov pri prechode riadkom sa do úvahy berie aj riadok nad ním, preto sa posledným riadkom neprechádza.

Takto vytvorená plochá mriežka je ďalej tvarovaná vo vertex shaderi. Tu sa na vertexy aplikuje vertikálny posun pozdĺž osi y. Nové pozície sú vypočítané vzorkovaním modrého kanálu výškovej mapy, ktorá je spoločná aj pre generovanie trávy.

4.3 Vykreslenie trávnika

Pri vykresľovaní trávy sa trávnik rozdelí na štvorcové dlaždice rovnomernej mriežky, ktorej veľkosť je určená užívateľom. Geometria stebiel sa generuje len pre jednu dlaždicu. Táto dlaždica sa následne vykresľuje viac-krát, aby sa obmedzili pamäťové nároky a počet volaní kreslenia (angl. *draw call*) grafickej karty. Počet stebiel (určený užívateľom) generovaných v tejto dlaždici je takto rovnaký pre všetky ostatné dlaždice. Jediným spôsobom ako je možné upraviť počet stebiel danej dlaždice je steblá pri vykresľovaní zahadzovať.

4.3.1 Generovanie quadov

Pred samotnou generáciou geometrie stebla je potrebné najskôr určiť množinu náhodných hodnôt pre dané steblo, ktoré unikátne ovplyvnia jeho vzhľad. Tieto hodnoty sú v nasledu-

Algoritmus 1 Generovanie mriežky terénu

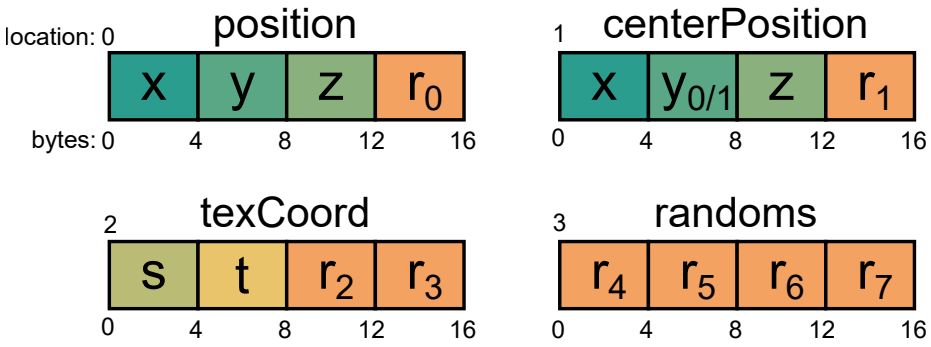
```
1: /* Generovanie vertexov (x,z)*/
2: for row = 0, ..., rows − 1 do
3:     for col = 0, ..., cols − 1 do
4:         normalizedLength = row/(rows − 1);
5:         normalizedWidth = col/(cols − 1);
6:         xOffset = mix(−terrainWidth/2, terrainWidth/2, normalizedWidth);
7:         zOffset = mix(terrainLength/2, −terrainLength/2, normalizedLength);
8:         terrainVertices.push(vec2(xOffset, zOffset));
9:     end for
10: end for
11: /* Generovanie indexov */
12: restartIndex = terrainVertices.size();
13: for row = 0, ..., rows − 2 do
14:     for col = 0, ..., cols − 1 do
15:         for k = 0, 1 do
16:             currentRow = row + k;
17:             index = currentRow ∗ cols + col;
18:             terrainIndices.push(index);
19:         end for
20:     end for
21:     terrainIndices.push(restartIndex);
22: end for
```

júcich rozsahoch:

$r_0 \in <0.00, 360.00>$	Uhol rotácie stebla
$r_1 \in <-1.00, 1.00>$	Posunutie horných vertexov na osi x
$r_2 \in <-1.00, 1.00>$	Posunutie horných vertexov na osi z
$r_3 \in <-0.25, 0.25>$	Variácia kontrolného bodu (De Casteljau) na osi x a z
$r_4 \in <0.75, 1.25>$	Variácia kontrolného bodu (De Casteljau) na osi y
$r_5 \in <0.00, 0.05>$	Farebná variácia červeného kanálu
$r_6 \in <0.00, 0.05>$	Farebná variácia zeleného kanálu
$r_7 \in <0.00, 0.05>$	Farebná variácia modrého kanálu

Prvým krokom pri vytváraní stebla je vygenerovanie obrysového obdĺžnika, ktorý reprezentuje jeho počiatočné dimenzie. Generovanie štyroch rohových vertexov tohto obdĺžnika je popísané v rovnici 2.1 metódy Jahrmann-Wimmer (2013). Vo výslednom výpočte však nie je použitá hodnota $hustota_b$, pretože tá sa vzorkuje neskôr vo vertex shaderi. Steblo je následne náhodne posunuté v rámci dlaždice a pre každý vertex sú pridané informácie o jeho stredovom bode. Súradnici y stredového bodu je priradená hodnota nula, alebo jedna, podľa toho či sa jedná o spodný, alebo horný vertex. Ďalej sú vertexom priradené textúrové súradnice a náhodné hodnoty. Tieto náhodné hodnoty sú pre každý vertex daného obdĺžnika rovnaké. Atribúty výsledného vertexu sú popísané na obrázku 4.2.

Pre poslanie dát na grafickú kartu je využitá primitíva GL_PATCHES, ktorá obsahuje štyri vertexy obrysového obdĺžnika. Pretože je pre každú dlaždicu trávnika použitá rovnaká geometria, je využité inštancované renderovanie. To v praxi znamená, že je geometria



Obr. 4.2: Atribúty vertexu obrysového obdĺžnika. Atribút *position* reprezentuje pozíciu (x,y,z) vertexu v 3D priestore. Atribút *centerPosition* obsahuje pozíciu (x,z) stredového bodu stebla. Hodnota y je rovná nule, alebo jednotke, podľa toho, či sa jedná o dolný, alebo horný vertex. *texCoord* obsahuje textúrové súradnice (s,t) vertexu. Zvyšné hodnoty atribútov a atribút *randoms* zapĺňajú náhodné hodnoty (r_0 - r_7).

stebiel dlaždice skopírovaná do grafickej pamäte len raz a je využité len jeden render call. Dlaždica sa ale vykreslí niekoľko-krát (podľa počtu inštancií) vždy s inou transformačnou maticou. Po výpočte stredových pozícií dlaždíc na CPU sa vypočítajú korešpondujúce transformačné matice, ktoré sú uložené do SSBO (Shader Storage Buffer Object). Spomínaný SSBO je následne naviazaný na shader program pre vykreslenie trávy a vykoná sa render call použitím OpenGL funkcie *glDrawArraysInstanced* s počtom inštancií rovnému počtu dlaždíc trávnika.

4.3.2 Tvarovanie stebla

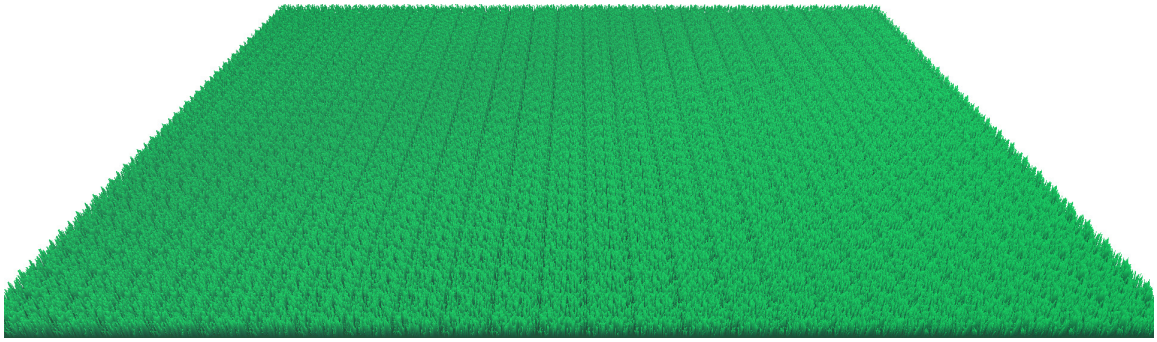
Táto podsekcia popisuje spracovanie vstupného obdĺžnika na GPU, jeho tvarovanie a následné vykreslenie na obrazovku. Pre tento proces sú využité štyri druhy shaderov z OpenGL pipeline, a to *vertex*, *tessellation control*, *tessellation evaluation* a *fragment* shader. Tento proces popisuje obrázok 2.15.

Vertex shader

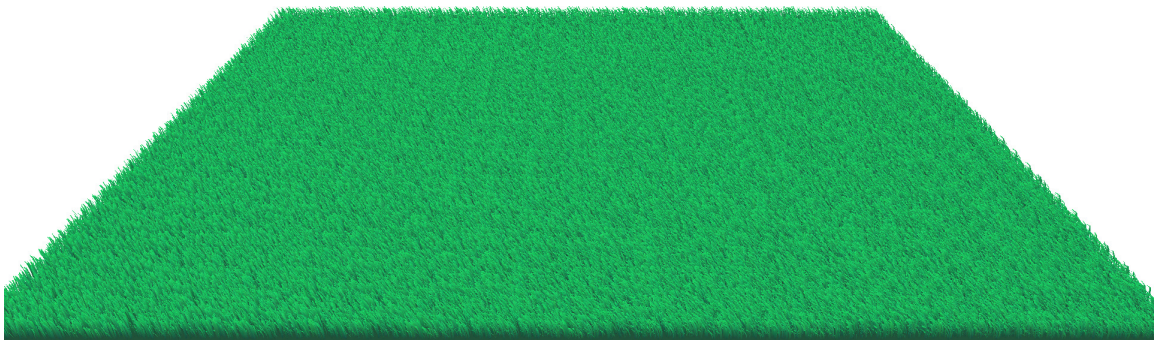
Na začiatku vykresľovacieho procesu má každá dlaždica stred v bode [0,0,0]. V prvom kroku vertex shadera je každá dlaždica otočená o násobok 90-tich stupňov, podľa náhodnej hodnoty vygenerovanej na CPU pre každú dlaždicu. Rotácia [6] dlaždíc je použitá kvôli zamedzeniu viditeľnosti mriežky trávnika. Jedným zo spôsobov riešenia tohto problému je popísaný v článku [6], ktorý navrhuje ovplyvnenie náhodných premenných r_0 - r_7 podľa pozície v scéne. Táto technika sa ale pri implementácií ukázala byť nepostačujúca, preto bola zvolená technika otáčania dlaždíc použitá v článku [1]. Na obrázku 4.3 je možné vidieť mriežku trávnika. Na obrázku 4.4 je použitá rotácia dlaždíc, ktorá mriežku skryje.

Pre vzorkovanie z výškovej mapy je najprv potrebné určiť stredovú pozíciu stebla na trávniku. V prvom rade je z transformačnej matice dlaždice určená globálna pozícia vertexu

[6] Funkcia *rotate* pre rotáciu bodu okolo stredového bodu použitá v tomto vertex shaderi je prevzatá z príspevku užívateľa *iscariot* na fóre StackOverflow dostupného z nasledujúcej adresy: https://stackoverflow.com/questions/61998702/opengl-es-2-0-rotate-point-around-pivot-point-2d-vertex-shader. Navštívené 02.05.2021.



Obr. 4.3: Trávnik bez rotácie dlaždíc. Mriežka je viditeľná z kolmých uhlov.



Obr. 4.4: Dlaždice sú otočené o náhodný násobok 90-tich stupňov. Mriežka v tomto prípade viditeľná nie je.

v scéne (neskôr použitá pri kalkulácií vetra). Ďalej je z tejto pozície vypočítaná relatívna pozícia na trávniku, ktorá slúži ako textúrová súradnica.

Keďže je tráva v odlišných častiach trávnika rôzne hustá, je potrebné v týchto miestach vykresliť iný počet stebiel. Počet stebiel je ale pre každú dlaždicu pevne daný, preto je potrebné prebytočné steblá zahadzovať. Jedným zo spôsobov zahodenia celého stebla je nastavenie úrovne vonkajšej teselácie v *tessellation control* shadery na nulu. Týmto sa zahodí celá primitíva *GL_PATCHES* a steblo sa ďalej nevykresľuje. Pravdepodobnosť zahodenia stebla je daná hodnotou červeného kanálu výškovej mapy (pozri obr. 4.5a). Zahodenie je vypočítané rovnicou 4.1, kde r_1 je náhodná hodnota v rozsahu [-1.0,1.0], *cervena_vzorka* je hodnota vzorky červeného kanálu výškovej mapy. Ak je hodnota zahodenia väčšia ako jedna, je steblo označené na zahodenie. Toto riešenie efektívne zahadzuje steblá trávy podľa určenej pravdepodobnosti.

$$zahodenie = |r_1| + (1 - cervena\_vzorka) \tag{4.1}$$

Vertexy stebla sa ďalej otočia okolo jeho stredu o uhol rovný hodnote r_0. Posunú sa pozdĺž osi y podľa výšky terénu v danej oblasti. Hodnota posunu závisí od modrého kanálu výškovej mapy (pozri obr. 4.5c) a hodnoty maximálnej výšky terénu, špecifikovanej užívateľom (rovnako ako pri tvarovaní mriežky terénu).

Zelený kanál výškovej mapy udáva výšku stebiel v danej oblasti (pozri obr. 4.5b). Pri zmene výšky stebla je taktiež potrebné zmeniť jeho šírku, aby boli zachované proporcie. Toto škálovanie je popísané algoritmom 2, kde hodnoty pos_{xyz} označujú pozície rohových vertexov a hodnoty $center_{xyz}$ označujú pozíciu stredového bodu. *heightSample.g* reprezentuje vzorku zeleného kanálu výškovej mapy a *terrainHeight* výšku terénu. Steblá menšie ako

heightTreshold sú označené na zahodenie. Obrázok 4.6 zobrazuje výsledný efekt aplikovania všetkých kanálov výškovej mapy.

Algoritmus 2 Škálovanie stebla

```
1: if heightSample.g > heightTreshold then
2:     posX = posX + (centerX - posX) * (1 - heightSample.g);
3:     posZ = posZ + (centerZ - posZ) * (1 - heightSample.g);
4:     if centerY > 0.99f then                    ▷ Ak sa jedná o horný vertex
5:         posY = posY - ((1 - heightSample.g) * (posY - terrainHeight));
6:     end if
7: else
8:     vDiscardBlade = 1;                         ▷ Označ steblo na zahodenie
9: end if
```

Poslednou úpravou je posunutie horných vertexov pre určenie ohybu stebla a simuláciu vetra. Vektor posunu horných vertexov v rovnici 2.2, ako aj posun podľa vetra je naviac vynásobený zelenou vzorkou výškovej mapy, aby bol posun prispôsobený výške stebla. Pre simuláciu lokálneho chaosu spôsobeného vetrom je použitá funkcia vetra metódy Jahrmann-Wimmer (2013) popísaná v rovnicou 2.7. Perióda tejto funkcie sa mení v závislosti na čase, pre dosiahnutie realistickejších zmien. Pre simulovanie dlhších veterných vĺn je použitá funkcia inšpirovaná implementáciou v hre Horizon Zero Dawn[7], používajúca dve sínusové vlny. Kombináciou týchto funkcií je dosiahnutá realistickejšia simulácia vetra. Výsledná výpočet je zobrazený v rovnici 4.2, kde k_1 - k_3 určujú parametre funkcie, modifikovateľné užívateľom. Premenné c_{xyz} reprezentujú pozíciu stredového bodu stebla. Parameter t udáva čas simulácie a hodnota ε predstavuje malé číslo, pre prípad delenia nulou. Hodnoty x a z predstavujú pozíciu vertexu.

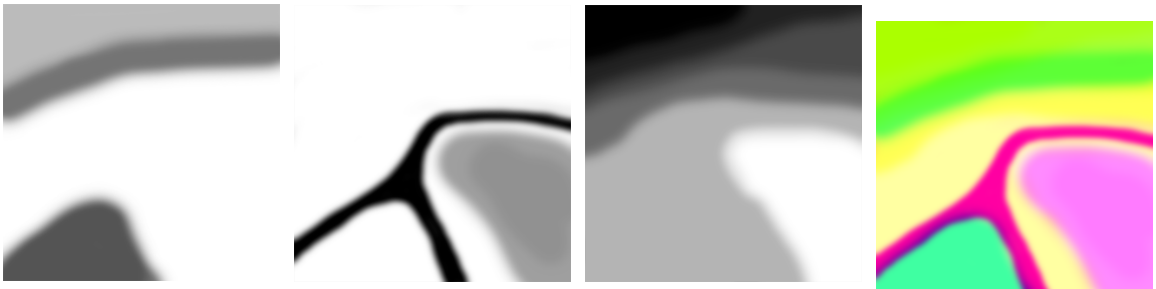
Pred výstupom z vertex shadera sú pozície vertexov a stredového bodu presunuté na pozíciu svojej dlaždice.

$$
\begin{aligned}
a &= \pi \cdot \mathbf{c}_x + t + \frac{\frac{\pi}{4}}{|\cos(k_2 \cdot \pi \cdot \mathbf{c}_z)| + \varepsilon} \\
kratky\_posun &= \sin(k_1 \cdot a) \cdot \cos(k_3 \cdot a) \\
dlhy\_posun\_x &= 2 \cdot \sin(1 \cdot (\mathbf{c}_x + \mathbf{c}_y + \mathbf{c}_z + t)) + 1 \\
dlhy\_posun\_z &= 1 \cdot \sin(2 \cdot (\mathbf{c}_x + \mathbf{c}_y + \mathbf{c}_z + t)) + 0.5 \\
\mathbf{x} &= \mathbf{x} + zelena\_vzorka \cdot kratky\_posun + zelena\_vzorka \cdot dlhy\_posun\_x \\
\mathbf{z} &= \mathbf{z} + zelena\_vzorka \cdot kratky\_posun + zelena\_vzorka \cdot dlhy\_posun\_z
\end{aligned}
\tag{4.2}
$$

Tessellation control shader

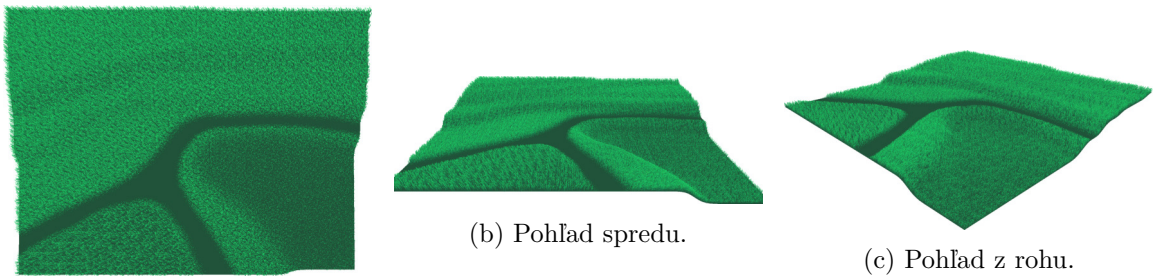
Pre určenie úrovne teselácie (LOD) je najskôr potrebné vypočítať vzdialenosť stebla od kamery. Pozícia kamery je dostupná prostredníctvom uniform premennej shader programu. Výpočet úrovne teselácie je popísaný rovnicou 2.3.

[7] Dostupné z: https://gdcvault.com/play/1025530/Between-Tech-and-Art-The (strana 25)



(a) Červený kanál (hustota trávy). (b) Zelený kanál (výška stebiel). (c) Modrý kanál (výška terénu). (d) Plná textúra.

Obr. 4.5: Ukážka výškovej mapy.



(a) Pohľad z vrchu. (b) Pohľad spredu. (c) Pohľad z rohu.

Obr. 4.6: Trávnik po aplikovaní všetkých kanálov výškovej mapy.

Kvôli zníženiu nárokov na výkon aplikácie a obmedzeniu aliasovania je vhodné znížiť množstvo vykresľovaných stebiel so vzrastajúcou vzdialenosťou. To je možné dosiahnuť postupným zvyšovaním pravdepodobnosti zahodenia, podobne ako pri obmedzovaní hustoty stebiel rovnicou 4.3. Ak je hodnota zahodenia väčšia ako jedna, je premenná *tessellationLevel* nastavená na nulu. Táto hodnota je nastavená na nulu aj v prípade, že bol vo vertex shaderi nastavený príznak *vDiscardBlade* na jednotku.

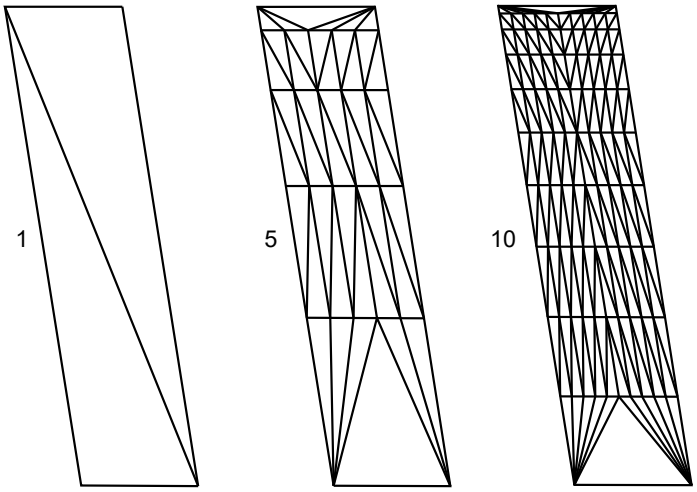
$$zahodenie = |r_1| + (vzdialenost\_od\_kamery / maximalna\_vzdialenost\_vykreslovania) \tag{4.3}$$

Proces teselácie rozdelí ľavú a pravú stranu quadu na n rovnakých častí pridaním nových vertexov. Vnútro quadu sa taktiež rozdelí na n riadkov a n stĺpcov (okrem prvého a posledného riadku). Premenné *gl_TessLevelOuter[0]*, *gl_TessLevelOuter[2]*, *gl_TessLevelInner[0]* a *gl_TessLevelInner[1]* sú nastavené na hodnotu *tessellationLevel*. Zvyšné premenné majú hodnotu jedna. Nakoniec sú vypočítané dodatočné kontrolné body pre algoritmus De Casteljau použitý v ďalšom shaderi.

Ďalšou fázou teselácie je *tessellation primitive generator*, ktorý programovateľný nie je. Tento generátor vezme vertexy z vertex shadera, interpretuje ich ako abstraktný quad, ktorý rozdelí podľa hodnôt špecifikovaných v *tessellation control shaderi*. Výsledok teselácie je možné vidieť na obr. 4.7.

Tessellation evaluation shader

Výstupom *tessellation primitive generátora* je abstraktný quad s novým počtom vertexov. Vertexy tohto quadu však obsahujú len pozíciu relatívnu k tomuto quadu, preto je potrebné



Obr. 4.7: Teselácia quadu vystupujúceho z vertex shadera. Čísla udávajú úroveň teselácie n. Ľavá a pravá vonkajšia úroveň a obe vnútorné úrovne teselácie sú nastavené na hodnotu n. Spodná a horná vonkajšia úroveň sú nastavené na hodnotu jedna. Body stretu úsečiek reprezentujú novo-vygenerované vertexy.

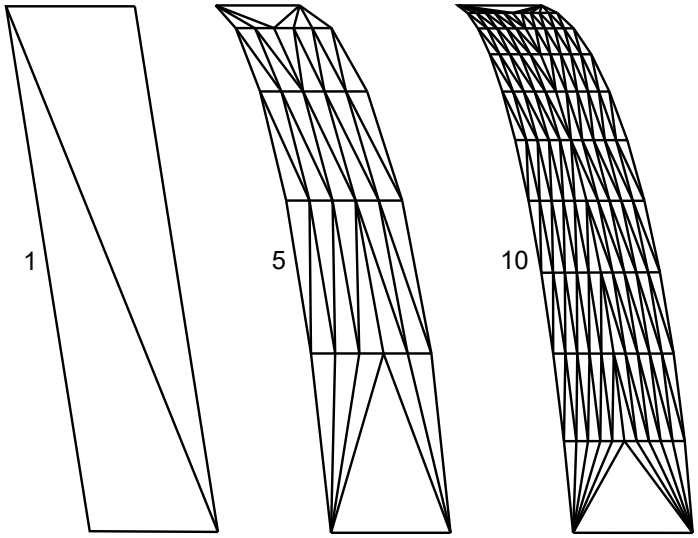
dopočítať ich pozíciu v scéne, ako aj textúrové súradnice. Pozíciu na ose x reprezentuje premenná u a na ose y premenná v. Hodnoty týchto premenných prakticky reprezentujú textúrové súradnice, preto na ich určenie nie sú potrebné ďalšie kalkulácie.

Prvým krokom je vypočítanie bodov na oboch splinoch, ktoré určujú ohnutie stebla. Kontrolné body týchto splinov pozostávajú zo spodného a horného vertexu pôvodného obdĺžnika, ako aj dodatočného kontrolného bodu vypočítaného v *tessellation control shaderi*. Výpočet bodu na spline algoritmom De Casteljau je popísaný rovnicou 2.5, kde parameter v reprezentuje vyššie spomínanú hodnota v. Po vypočítaní bodov na oboch splinoch, je výsledný pozícia vertexu určená interpolovaním medzi týmito dvoma bodmi hodnotou u. Finálna pozícia vertexov je znázornená na obr. 4.8.

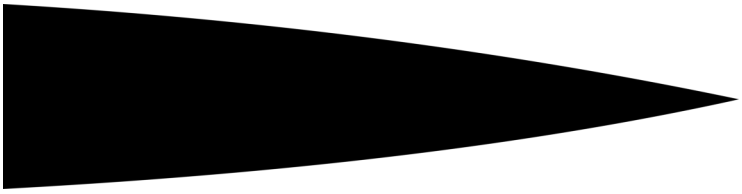
V poslednom kroku je pozícia vertexov vynásobená projekčnou a pohľadovou maticou, pre určenie výslednej pozície na obrazovke.

Fragment shader

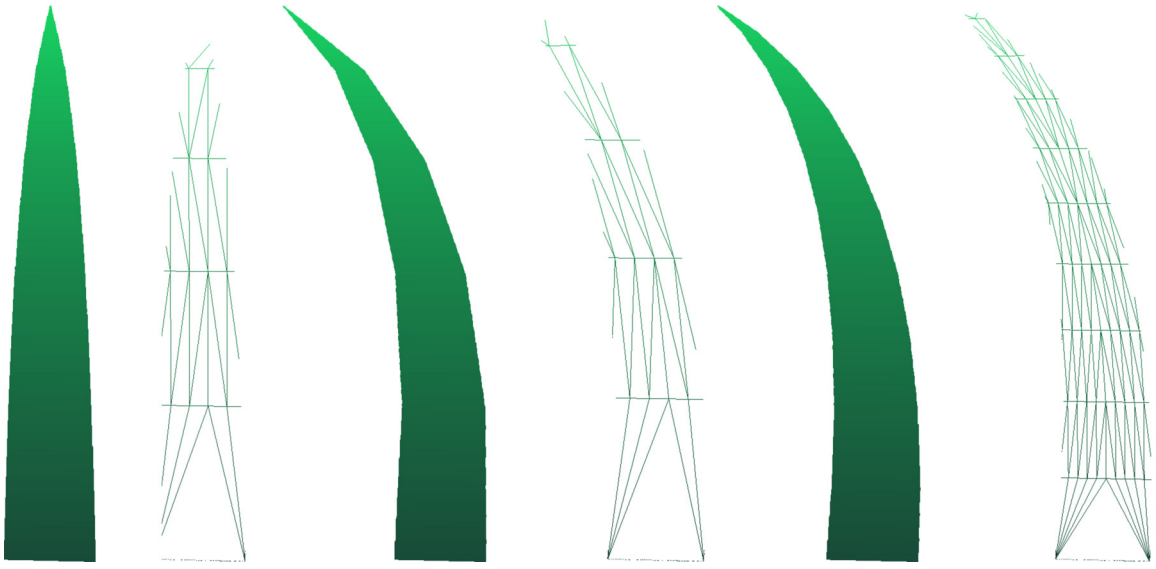
V poslednej fáze je na výsledné steblo použitá alfa textúra 4.9, využitím alfa testovania. Týmto procesom sa zahodia prebytočné fragmenty a steblo dostane finálny tvar. Farba jednotlivých fragmentov je určená interpolovaním hodnôt medzi dvoma farbami, podľa vertikálnej textúrovej súradnice. Horné fragmenty majú farbu svetlejšiu, zatiaľ čo spodné farbu tmavšiu. Týmto prechodom je vo výsledku dosiahnuté jednoduché osvetlenie, ktoré simuluje tieň v spodnej časti stebla. Výsledok fragment shadera je možné vidieť na obr. 4.10.



Obr. 4.8: Steblo tvarované v tessellation evaluation shaderi. Vertexy sú zarovnané pozdĺž dvoch paralelných splinov. Čísla udávajú úroveň teselácie.



Obr. 4.9: Ukážka alfa textúry využitej na formovanie výsledného tvaru stebla, použitím alfa testovania.



Obr. 4.10: Ukážka stebiel trávy po aplikovaní alfa textúry. Každý pár predstavuje vykreslené steblo trávy v plnej farbe (vľavo) a jeho kostru (vpravo). Ľavý pár zobrazuje steblo bez použitia ohybu s úrovňou teselácie rovnej 5. Pár v strede zobrazuje steblo s použitím ohybu a úrovňou teselácie rovnou 5. Napravo je to isté steblo s úrovňou teselácie rovnej 10. Snímok je zachytený z výslednej aplikácie.

Kapitola 5

Záver

Táto práca sa zaoberá problematikou vykresľovania trávy v počítačovej grafike. V teoretickej časti sú predstavené rôzne prístupy k tejto problematike. Rozobrané sú konkrétne metódy a ich využitie v reálnych aplikáciách.

Výsledkom práce je demonštračná aplikácia, ktorá podľa vstupných parametrov a textúr dokáže vygenerovať terén, na ktorý následne vykreslí trávu. Parametre trávy a terénu je možné meniť v reálnom čase. Tráva využíva jednoduché osvetlenie a jej pohyb vo vetre je simulovaný pre každé steblo individuálne. Repozitár obsahujúci zdrojové kódy aplikácie je dostupný na portáli *GitHub*[1].

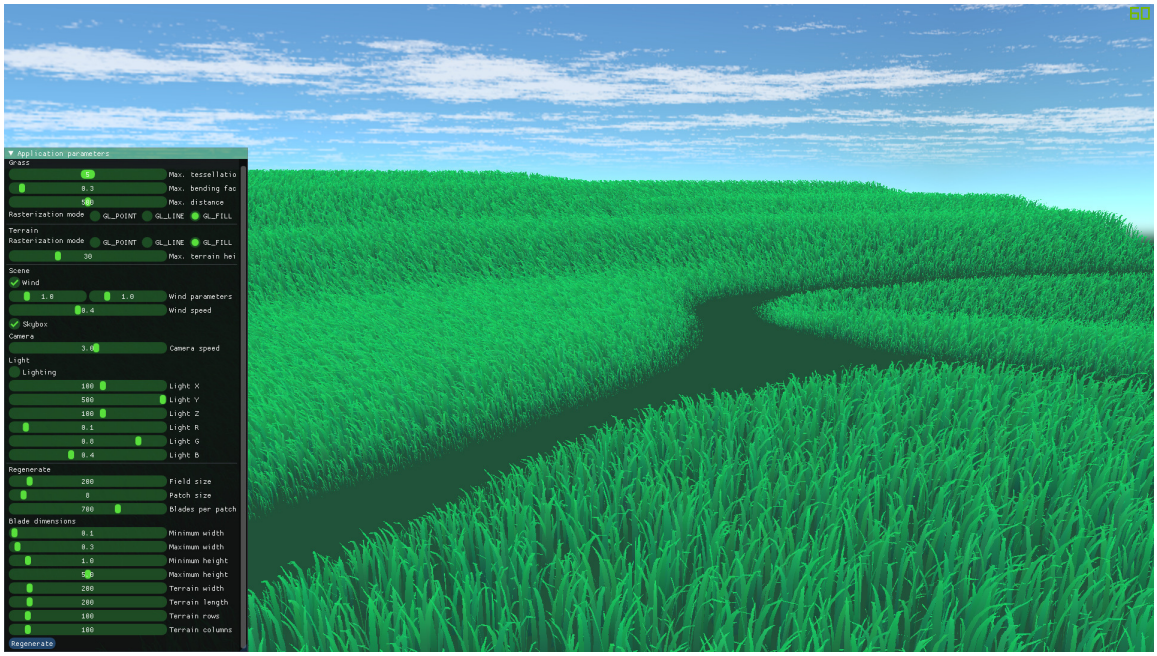
Vybraná technika modeluje trávu ako geometriu, čo poskytuje mnoho možností ako zlepšiť výsledný dojem zo scény. Implementované riešenie neberie do úvahy pozíciu slnka (alebo iných zdrojov svetla) v scéne. To môže byť doplnené dopočítaním normál stebiel (popísaným v metóde *Jahrmann-Wimmer (2013)*[6]) a ich následným použitím pri implementácií *Phongovho osvetľovacieho modelu*[11].

Simulácia vetra využíva goniometrické funkcie pre určenie polohy stebla v danom čase. Tento postup môže pri pozornom sledovaní z dostatočnej vzdialenosti pôsobiť umelo. To je spôsobené periódami použitých funkcií (zvyčajne sínus, alebo kosínus), ktoré sa prejavujú ako pravidelné vlny. Riešením tohto problému môže byť simulovanie tzv. *veternej mapy* (angl. *wind map*), ktorej vektory sú použité pre posun jednotlivých stebiel.

Ďalším možným rozšírením je metóda *Jahrmann-Wimmer (2017)*[7], ktorá predstavuje vylepšený fyzikálny model stebiel. Tento model dovoľuje vykresliť trávu nie len na typický terén, ale aj na ľubovoľný 3D model. Taktiež sú v tejto metóde predstavené pokročilé možnosti orezávania stebiel, ktoré nie sú od kamery viditeľné, čo značne zníži nároky na výkon. Keďže implementácia v tejto práci nerieši kolízie stebiel s inými objektmi, je možné využiť aj kolízny systém tejto metódy.

Snímok 5.1 zobrazuje vzhľad výslednej aplikácie implementovanej v tejto práci.

[1] Dostupné z: https://github.com/TomasBerdis/GrassRenderer.



Obr. 5.1: Snímok zachytený z výslednej aplikácie.

Literatúra

[1] Boulanger, K., Pattanaik, S. N. a Bouatouch, K. Rendering Grass in Real Time with Dynamic Lighting. *IEEE Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society Press. 2009, zv. 29, č. 1, s. 32–41. DOI: 10.1109/MCG.2009.14. ISSN 0272-1716.

[2] Fan, Z., Li, H., Hillesland, K. a Sheng, B. Simulation and Rendering for Millions of Grass Blades. In: *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*. New York, NY, USA: Association for Computing Machinery, 2015, s. 55–60. ISBN 9781450333924. Dostupné z: https://doi.org/10.1145/2699276.2699283.

[3] Farin, G. a Hansford, D. *The Essentials of CAGD*. Október 2000. ISBN 9780429062414.

[4] Habel, R., Wimmer, M. a Jeschke, S. Instant Animated Grass. *Journal of WSCG*. január 2007, zv. 15, 1-3, s. 123–128. ISSN 1213-6972. ISBN 978-80-86943-00-8. Dostupné z: https://www.cg.tuwien.ac.at/research/publications/2007/Habel_2007_IAG/.

[5] Han, D. a Harada, T. Real-time hair simulation with efficient hair style preservation. *VRIPHYS 2012 - 9th Workshop on Virtual Reality Interactions and Physical Simulations*. Január 2012, s. 45–51. DOI: 10.2312/PE/vriphys/vriphys12/045-051.

[6] Jahrmann, K. a Wimmer, M. Interactive Grass Rendering Using Real-Time Tessellation. In: Oliveira, M. a Skala, V., ed. *WSCG 2013 Full Paper Proceedings*. Jún 2013, s. 114–122. ISBN 978-80-86943-74-9. Dostupné z: https://www.cg.tuwien.ac.at/research/publications/2013/JAHRMANN-2013-IGR/.

[7] Jahrmann, K. a Wimmer, M. Responsive Real-Time Grass Rendering for General 3D Scenes. In: *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, Február 2017, s. 6:1–6:10. ISBN 978-1-4503-4886-7. Dostupné z: https://www.cg.tuwien.ac.at/research/publications/2017/JAHRMANN-2017-RRTG/.

[8] Orthmann, J., Rezk Salama, C. a Kolb, A. GPU-based Responsive Grass. In:. Január 2009, sv. 17, 1-3, s. 65–72. ISSN 1213–6972. Dostupné z: http://hdl.handle.net/11025/1290.

[9] Pelzer, K. GPU Gems: Programming Techniques Tips & Tricks for Real Time Graphics. In: Fernando, R., ed. Addison-Wesley Professional, 2004, kap. 7

Rendering Countless Blades of Waving Grass, s. 107–121. ISBN 978-0-321-22832-1. Dostupné z: https://dl.acm.org/doi/book/10.5555/983868.

[10] Perlin, K. An Image Synthesizer. In: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1985, s. 287–296. SIGGRAPH '85. DOI: 10.1145/325334.325247. ISBN 0897911660. Dostupné z: https://doi.org/10.1145/325334.325247.

[11] Phong, B. T. Illumination for Computer Generated Pictures. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. jún 1975, zv. 18, č. 6, s. 311–317. DOI: 10.1145/360825.360839. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/360825.360839.

[12] Procházka, M. *Vykreslování trávy v reálném čase [online]*. 2020 [cit. 2020-11-14]. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedúci práce Chmelík, J. Dostupné z: https://is.muni.cz/th/x4oux/.

[13] Reeves, W. T. a Blau, R. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. In:. New York, NY, USA: Association for Computing Machinery, 1985, s. 313–322. SIGGRAPH '85. DOI: 10.1145/325334.325250. ISBN 0897911660. Dostupné z: https://doi.org/10.1145/325334.325250.

[14] Wang, S., Ali, S. G., Lu, P., Li, Z., Yang, P. et al. GPU-based Grass Simulation with Accurate Blade Reconstruction. In: Magnenat Thalmann, N., Stephanidis, C., Wu, E., Thalmann, D., Sheng, B. et al., ed. *Advances in Computer Graphics*. Cham: Springer International Publishing, 2020, s. 288–300. ISBN 978-3-030-61864-3.

[15] Whatley, D. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. In: Pharr, M. a Fernando, R., ed. Addison-Wesley Professional, 2005, kap. 1 Toward Photorealism in Virtual Botany, s. 7–25. ISBN 978-0-321-54541-1. Dostupné z: https://dl.acm.org/doi/book/10.5555/1406887.