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

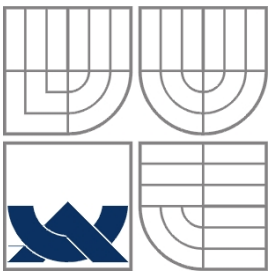
VIZUALIZÁCIA A GENEROVANIE REALISTICKÝCH MODELOV STROMOV

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

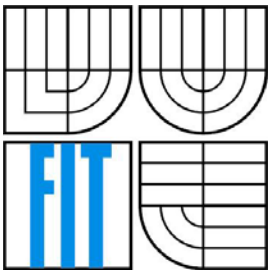
AUTOR PRÁCE
AUTHOR

TOMÁŠ PAFČO

BRNO 2007



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

VIZUALIZÁCIA A GENEROVANIE REALISTICKÝCH MODELOV STROMOV

VIZUALIZATION AND GENERATION OF REALISTIC TREE MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — TOMÁŠ PAFČO
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN PEČIVA
SUPERVISOR

BRNO 2007

Abstrakt

Táto bakalárska práca sa týka generovania realistických modelov stromov a ich následnej vizualizácie v reálnom čase. Navrhnutá metóda generovania ponúka široké spektrum nastaviteľných parametrov, vďaka čomu umožňuje vytvárať konkrétne druhy listnatých aj ihličnatých stromov s pomerne vysokou mierou vierohodnosti. Zároveň umožňuje do určitej miery nastaviť úroveň detailov a zložitosť geometrie stromu, čím je možné ovplyvniť výpočtovú náročnosť vykresľovania a počet snímkov za sekundu pri zobrazovaní modelu. Pri vizualizácii stromov bola využitá voľne dostupná sada knižníc *Open Scene Graph*. Zároveň bola v jazyku C++ vytvorená demonštračná aplikácia, ktorá umožňuje detailné prezeranie niekoľkých druhov stromov. Aplikácia je spustiteľná v prostredí Microsoft Windows XP.

Kľúčové slová

stromy, 3D, vizualizácia v reálnom čase, procedurálne generovanie, OpenSceneGraph

Abstract

This thesis deals with generation of realistic tree models and theirs real-time visualization. Proposed generating method offers versatile spectrum of variable parameters, so it is able to create particular species of broad-leaved or coniferous trees with relative high amount of authenticity. It is also possible to manage level of details and complexity of tree geometry, which influence the rendering time and amount of frames per second. An OpenSceneGraph, the open-source toolkit, was used to visualization. A demonstrational application was created in C++ to visualize several types of trees on WinXP platform.

Keywords

trees, 3D, real-time visualization, procedural generation, OpenSceneGraph

Citácia

Tomáš Pafčo: Vizualizácia a generovanie realistických modelov stromov, bakalárska práca, Brno, FIT VUT v Brne, 2007

Vizualizácia a generovanie realistických modelov stromov

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jána Pečivu.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tomáš Pafčo
1. 5. 2007

Poďakovanie

Chcel by som sa poďakovať Ing. Jánovi Pečivovi za poskytnutie odbornej pomoci, cenných rád a pripomienok, ktoré mi pomáhali pri tvorbe tejto práce.
Zároveň by som chcel vysloviť poďakovanie Pavlovi Matějkovi a Tomášovi Zuntovi za vecnú kritiku dosiahnutých výsledkov, hlavne po vizuálnej stránke, a Mariánovi Krivdovi za množstvo cenných programátorských rád.

Obsah

1 Úvod

Jednou z významných výziev v oblasti trojrozmernej počítačovej grafiky je generovanie a vizualizácia realistických prírodných scenérii, predovšetkým stromov a lesných scén. V súčasnosti sa totiž väčšina takých interaktívnych aplikácií, ako sú napríklad letecké simulátory, virtuálna realita alebo počítačové hry, odohráva v prírodných scénach. A sú to práve stromy a vegetácia, ktoré týmto scénam pridávajú na realistickosti a vierohodnosti.

Modely stromov môžu byť vytvorené dvoma spôsobmi, a to ručne alebo pomocou vhodného algoritmu. Vyžitie algoritmu (tento spôsob býva označovaný aj ako tzv. procedurálne generovanie) však prináša viacero výhod. Napríklad, procedurálne generovanie stromu trvá neporovnateľne kratšie, ako jeho ručné modelovanie. Zároveň umožňuje vytvárať veľké množstvo rôznych stromov určitého typu pomocou zmeny príslušnej náhodnej premennej v generátore. Generované stromy taktiež umožňujú jednoduchú parametrizáciu, čo znamená, že napríklad vek stromu, ročné obdobe, typ stromu, dĺžka vetiev a obrovské množstvo ďalších parametrov je možné využiť ako vstup generujúceho algoritmu a tak vytvárať unikátne stromy, presne spĺňajúce požiadavky cieľovej aplikácie alebo virtuálneho prostredia.

Táto práca pojednáva o súčasných postupoch algoritmického generovania realistických stromov a možnostiach ich vizualizácie v reálnom čase, pokiaľ možno v čo najvyššej vizuálnej kvalite. V práci je zároveň prezentovaná nová metóda generovania stromov, ktorá bola navrhnutá tak, aby bola na jednej strane jednoduchá, ale aby umožňovala vytvárať realistické stromy rôzneho druhu porovnateľné so skutočnými stromami. Ďalšou požiadavkou bola možnosť kontroly nad počtom jednotlivých biologických častí stromu, počtom polygónov a celkovou úrovňou detailov.

Z tematického hľadiska je práca členená nasledujúcim spôsobom: V prvej kapitole sú predstavené v súčasnosti najpoužívanejšie metódy generovania vegetácie, predovšetkým však stromov. Ide hlavne o parametrizované generovanie a L-systémy. Ďalšia kapitola pojednáva o postupoch využívaných pri ich vizualizácii v reálnom čase. Najbežnejšie postupy sú: kontrola úrovne detailov (LOD – „level of details“), billboardy a impostory. Tretia kapitola prezentuje novú metódu generovania stromov a demonštračnú aplikáciu, ktorá túto metódu využíva.

2 Metódy generovania stromov

Na to aby scéna pôsobila realisticky je potrebné, aby boli stromy trojrozmerné. Našťastie, vo väčšine prípadov stromy tvoria len akúsi kulisu, alebo pozadie, preto sú len málokedy vyššie ako 20 až 30% obrazovky. Vďaka tomu jemné detaily, ako zakrivenie listov alebo žilnatá štruktúra niesú dôležité. Naproti tomu, štruktúra vetiev musí byť v tomto rozlíšení veľmi presná, pretože listy neskrývajú vetvy kompletne ani u husto-listnatých stromov.

Generátor musí umožňovať vytvárať širokú paletu bežných typov stromov a taktiež musí byť schopný zaobchádzať s určitou mierou náhodnosti tak, aby bolo možné vytvárať veľké množstvo odchýliek v štruktúre stromov vychádzajúcich z rovnakého návrhového vzoru. Používanie generátora musí byť ľahko pochopiteľné aj pre bežného užívateľa, ktorý ma len všeobecnú znalosť základnej geometrie. To vylučuje využitie akýchkoľvek parametrov vyžadujúcich pochopenie náročnejších matematických princípov, ako sú napríklad diferenciálne rovnice. Generátor musí byť taktiež stabilný a ľahko použiteľný. Užívateľom zadané rovnice by mohli ľahko spôsobiť nepredvídateľné chovanie, preto tie aspekty modelu generovania, ktoré by mohli byť náročné na kontrolu, bývajú spravidla užívateľovi neprístupné a nahradené intuitívnymi parametrami. Naproti tomu, model generovania by mal čo najmenej obmedzovať užívateľa pri návrhu stromu.

V nasledujúcej časti bude priblížených niekoľko v súčasnosti najpoužívanejších metód generovania rastlín, pričom dôraz bude kladený predovšetkým na stromy.

2.1 Parametrizované generovanie

Jedným zo spôsobov ako zjednodušiť proces vytvárania rastliny je reprezentovať konkrétny druh ako kolekciu parametrov ktorých hodnotu upravuje užívateľ. Tento prístup vyžívajú Joseph Pern a Jason Weber vo svojej práci [1], zameranej na vytváranie veľmi realistických stromov. Sústredili sa na určitú množinu stromov a určili okolo 30 parametrov ktorými je možné popísať rôzne druhy stromov. Systém vetiev je rozdelený na niekoľko rekurzívnych úrovní s tým, že kmeň reprezentuje úroveň 0, primárne vetvy vyrastajúce z kmeňa majú úroveň 1, sekundárne vetvy vyrastajúce z primárnych majú úroveň 2, atď. Vetvy môžu mať synovské vetvy po celej svojej dĺžke, ale taktiež sa môžu rozbiehať na niekoľko paralelných vetví. Tieto paralelné vetvy zostávajú na rovnakej rekurzívnej úrovni ako pôvodná vetva. Každá úroveň vetiev má svoju vlastnú množinu parametrov, ako je napríklad uhol medzi rodičovskou a synovskou vetvou, dĺžka vetiev, zakrivenie a pod. Okrem toho existujú ešte globálne parametre ktoré určujú napríklad tvar stromu alebo jeho výšku. Tvar stromu je definovaný pomocou tzv. obálky, čo je vlastne paremtrizovaný priestor, pričom dĺžky všetkých vetiev sú prispôsobené tak aby sa celý strom nachádzal vo vnútri tohto priestoru.

Parametre sú na dostatočne vysokej úrovni abstrakcie, čo umožňuje ich nastavenie aj na základe pozorovania reálneho stromu a prípadne manuálneho zmerania veľkostí uhlov a relatívnych dĺžok.

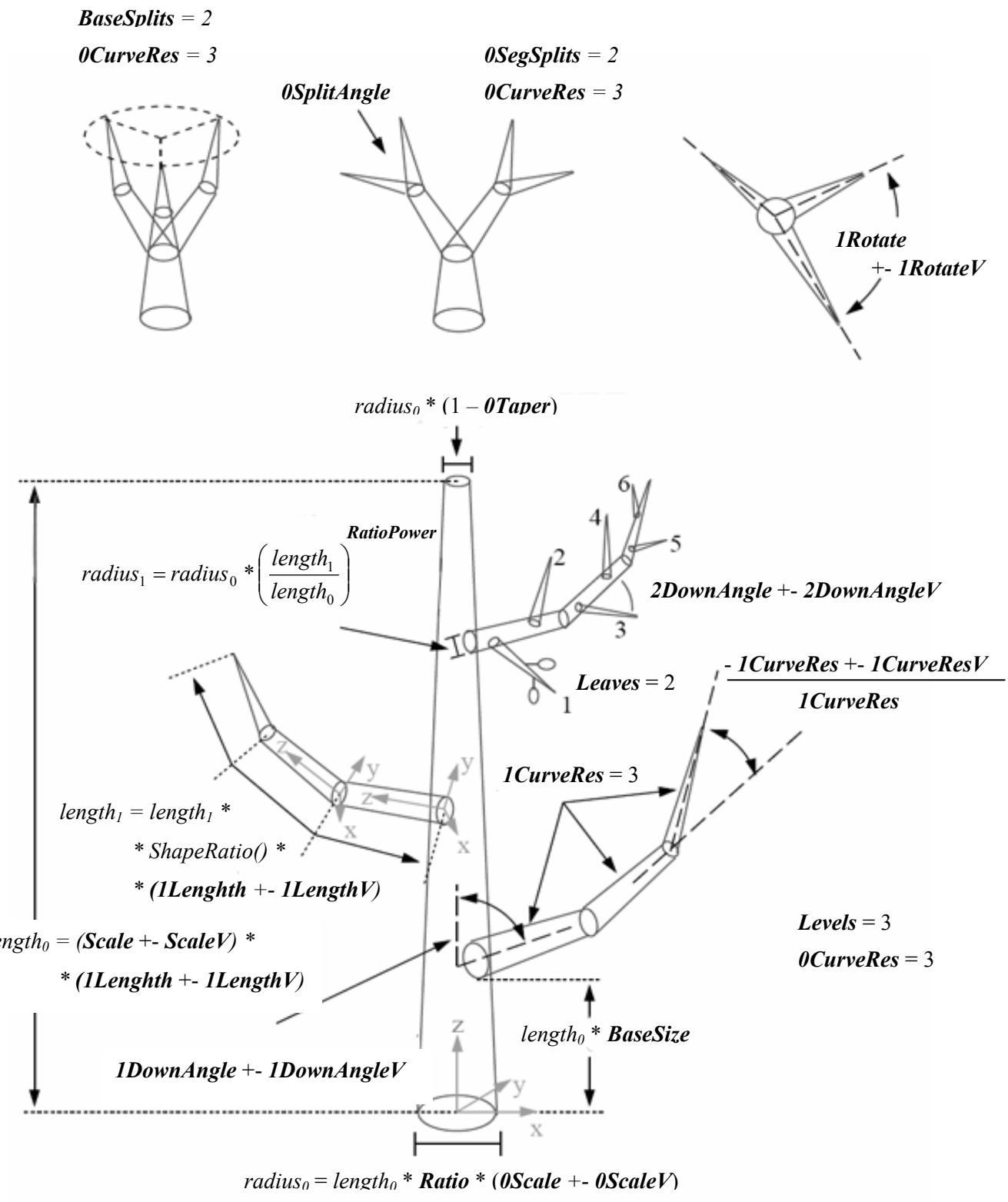
V nasledujúcich kapitolách bude metóda (model) Jasona Webera a Josepha Perna predstavená podrobnejšie.

2.1.1 Kmeň a štruktúra vetiev

Základom celého modelu sú dva komponenty, a to vetva a list. Pod pojmom „vetva" sa myslí vetva na akejkoľvek rekurzívnej úrovni, teda aj úrovni 0. Z geometrického hľadiska je vetva reprezentovaná úzkym, takmer kónickým valcom ktorého os *z* je súhlasná s jeho pozdĺžnou osou, pričom každá vetva má svoj vlastný relatívny súradnicový systém. Telo vetvy ***n***-tej úrovne je rozdelené na niekoľko takmer valcovitých segmentov, ktorých počet je definovaný parametrom ***nCurveRes***. Každý takýto segment je uložený ako približne kruhový prierez. Tieto prierezy sú neskôr spojené dohromady a na ich základe je generovaná trojuholníková sieť.

Vetva sa obvykle rozprestiera smerom k periférii stromu, pričom sa môže potencionálne rozchádzať na niekoľko paralelných vetiev. Tieto sú na rovnakej rekurzívnej úrovni ako pôvodná vetva a dedia všetky jej parametre. Frekvencia takéhoto paralelného rozvetvovania je definovaná parametrom ***nSegSplits***. Zároveň sa na konci prvého segmentu kmeňa stromu uplatňuje parameter ***nBaseSplits***, ktorý je ekvivalentom ***nSegSplits***, a umožňuje generovať strom s niekoľkonásobným kmeňom.

Na obr. 1.1 sú naznačené najdôležitejšie parametre. Ak je ***nCurveBack*** je rovný 0, *os z* každého segmentu sa odkláňa od *z-osi* predchádzajúceho segmentu o ***nCurve/nCurveRes*** stupňov okolo *osi x*. Ak je ***nCurveBack*** nenulový, prvá polovica každého segmentu natočená o ***nCurve/(nCurveRes/2)*** stupňov a druhá polovica o ***nCurveBack/(nCurveRes/2)*** stupňov. Takto je možné dosiahnuť zakrivenie vetvy do tvaru písmena S. V oboch prípadoch je ku každému pripočítaná náhodná rotácia o veľkosti ***nCurveV/nCurveRes***. Os *z* každej vetvi je odklonená od rodičovskej osi *z* o uhol ***nDownAngle+nDownAngleV***. Synovské vetvy sú usporiadané špirálovitým rozložením okolo rodičovskej osi *z*, pričom osi dvoch susedných synovských vetiev zvierajú uhol ***nRotate+nRotateV***. Parametre ***Ratio***, ***nScale*** a ***RatioPower*** slúžia na definovanie priemeru kmeňa a vetiev. Ich význam je zrejmý z obr. 1.1. Priemer akejkoľvek vetvy nie je samozrejme nikdy v väčší ako priemer jej rodičovskej vetvy v mieste kde vyrastá. Parameter ***nTaper*** udáva mieru kužeľovitosti kmeňa alebo vetiev, pričom môže nadobúdať hodnoty z intervalu 0 až 1.



Obr. 1.1: Štruktúra stromu

Funkcia ***ShapeRatio(shape, ratio)*** definuje tvar vetvy. Parameter ***shape*** je index do tabuľky kriviek určujúcich profil, zatiaľ čo ***ratio*** je normalizovaná pozícia na osi vetvy v intervale od 0,0 do 1,0.

2.1.2 Listy

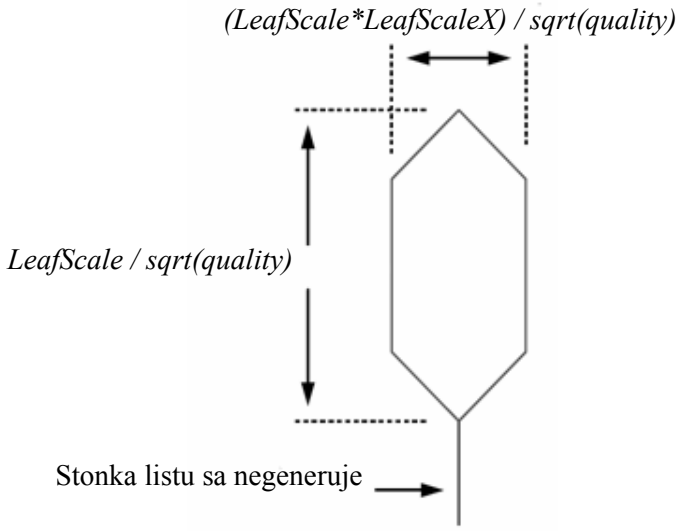
Rekurzívne vetvenie je limitované parametrom ***Levels***. Ten špecifikuje maximálny počet rekurzívnych úrovní, obyčajne má hodnotu 3 alebo 4. Ak sa parameter ***Leaves*** nerovná 0, potom na poslednej úrovni sa generujú miesto vetiev listy. Rovnako sa vyžívajú parametre ***nRotate***, ***nRotateV***, ***nDownAngle*** a ***nDownAngleV***.

Listy môžu mať niekoľko rôznych tvarov, ktoré sú preddefinované a uložené v zozname. Parameter ***LeafShape*** je index do tohto zoznamu. Mierku a veľkosť každého z týchto tvarov je možné definovať pomocou parametrov ***LeafScale*** a ***LeafScaleX*** tak, ako je to znázornené na obr. 1.2. Počet listov na vetve je daný nasledujúcim vzorcom:

$$pocet\_listov\_na\_vetve = Leaves \cdot ShapeRatio(4, \frac{offset_{child}}{length_{parent}}) \cdot quality$$

Parameter ***quality*** máva obyčajne hodnotu blízku 1 a okrem počtu listov ovplyvňuje aj ich veľkosť, aby bolo zachované konštantné pokrytie stromu. To znamená, že strom z nižšou hodnotou ***quality*** bude mat síce menej listov, tie však budú väčšie.

Ak je hodnota ***Leaves*** záporná, využíva sa špeciálna metóda, pri ktorej sú listy umiestnené ako vejár na konci vetvy. Takýmto spôsobom je možné generovať napríklad palmy.

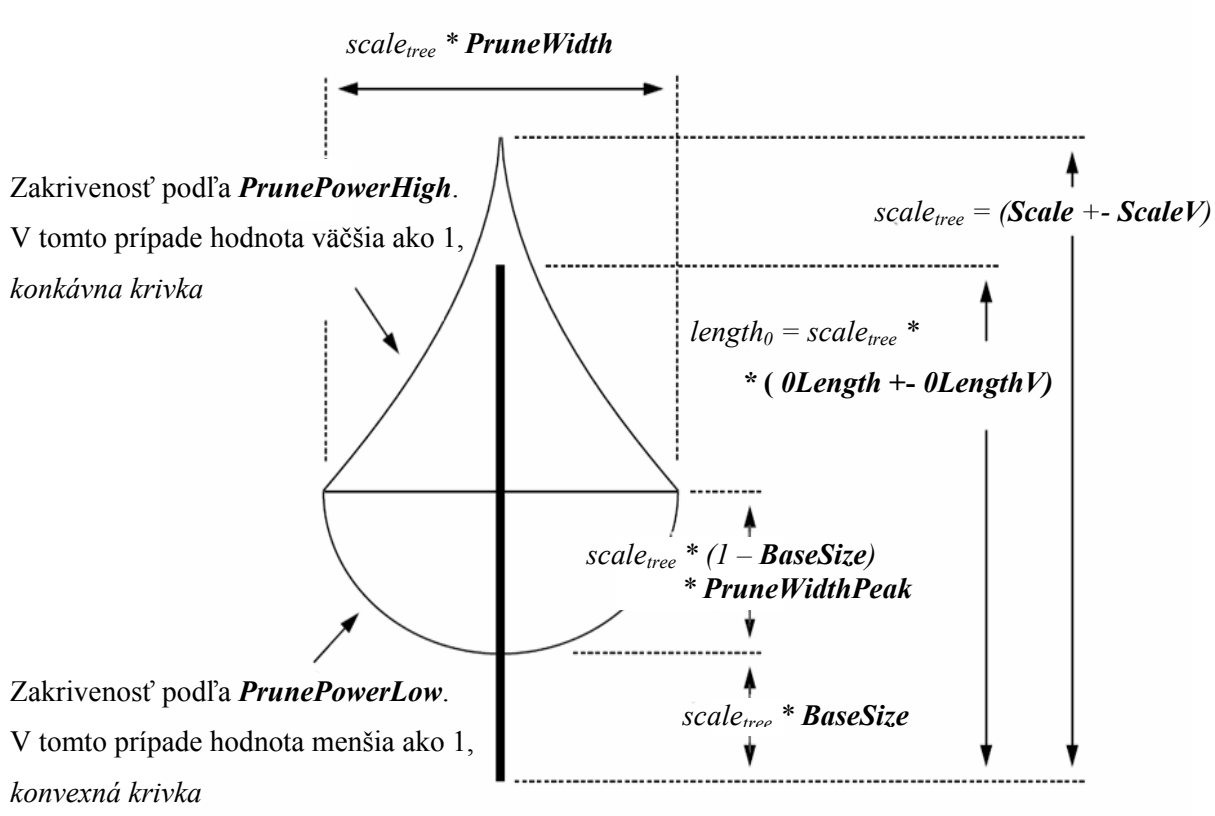


Obr. 1.2: Parametre listu

2.1.3 Tvar stromu

Tvar stromu je definovaný pomocou tzv. obálky, čo je v podstate niekoľkými parametrami definovaný priestor, ktorý vymedzuje hranice stromu (obr. 1.3). Dĺžka každej vetvy je upravená tak, aby nepresahovala túto hranicu.

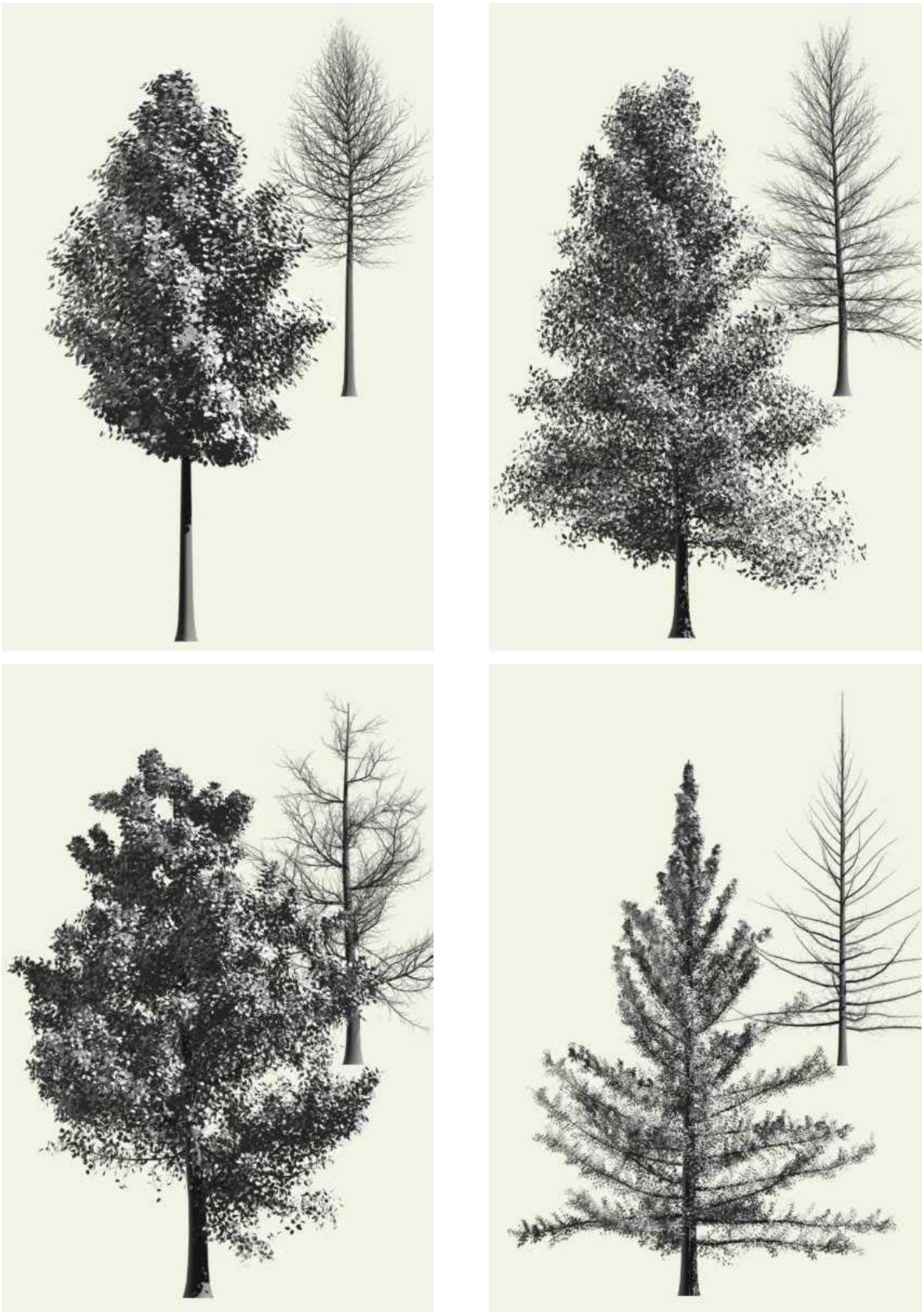
Maximálnu šíku obálky určuje parameter ***PruneWidth***, pričom miesto, kde obálka nadobúda toto maximum, je definované pomocou parametra ***PruneWidthPeak***. Toto miesto je zároveň hranicou, ktorá vertikálne rozdeľuje obálku na dve časti. Zakrivenosť je možné definovať nezávisle pre obe tieto časti pomocou parametrov ***PrunePowerHigh*** a ***PrunePowerLow***.



Obr. 1.3: Obálka definujúca tvar stromu

U reálnych stromov je možné pozorovať snahu mladých vetiev vplyvom slnečného žiarenia vyrastať smerom hore, pričom miera tohoto javu je pri rozličných druhoch stromov rôzna. Vo všeobecnosti však nejde o veľmi výrazný efekt a je ho možné namodelovať pomocou jemných odchýlok každého segmentu vetvy od jeho pôvodnej orientácie. Toto zakrivenie smerom hore sa uplatňuje až pri vetvách druhej a ďalších generácií a je definované parametrom ***AttractionUp***. Záporné hodnoty sa prejavia tým, že sa vetvy budú skláňať smerom dolu, ako to môžeme pozorovať pri niektorých druhoch vŕb.

Nasledujúce obrázky predstavujú ukážky niekoľkých stromov vygenerovaných na základe algoritmu predstaveného v tejto kapitole. Stromy boli vytvorené pomocou voľne dostupnej aplikácie *Arbaro* [2].



Obr. 1.4: Stromy vygenerované aplikáciou Arbaro, na základe algoritmu J. Webera a J. Perna

2.2 L – Systémy

Kľúčová práca v oblasti generovania realistických rastlín je metóda L-systémov, ktorú navrhol Arstid Lindenmayer [3]. L-systém pozostáva z množiny pravidiel, na základe ktorých sú prepisované reťazce. Tieto pravidlá sú opakovane aplikované na prvotný reťazec – axióm, a takto vzniknutý reťazec je následne interpretovaný, čím sa vytvorí jeho geometrická reprezentácia.

2.2.1 Klasifikácia L-systémov

Základný typ L-systému je tzv. ***D0L-systém***. Ide o bezkontextový L-systém, čo znamená že symboly nachádzajúce sa napravo a naľavo od skúmaného reťazca sa pri aplikácií pravidla neuvažujú. Ďalej je tento systém deterministický, čiže pre každý symbol existuje práve jedno pravidlo.

Dôležitým rozšírením D0L-systémov sú takzvané ***zátvorkové L-systémy***, ktoré sa využívajú pri vytváraní stromov a rôznych iných rastlín so stromovou štruktúrou (obr. 1.5). Interpret takéhoto L-systému je vybavený zásobníkom do ktorého ukladá svoj aktuálny stav, pričom pri vetvení sa k tomuto stavu vracia.

Všetky rastliny vytvorené určitým deterministickým L-systémom sú úplne identické, čo nieje v praxi veľmi výhodné. Preto je potrebné pridať do procesu prepisovania určitú mieru náhodnosti, čím vzniká tzv. ***stochastický L-systém***. Pravidlá sú v tomto systéme doplnené o hodnotu, ktorá určuje s akou pravdepodobnosťou bude určitý symbol prepísaný práve týmto pravidlom. Príklad stochastického L-systému je na obr. 1.6.

Ďalším L-systémom je ***kontextový L-systém***, ktorý ma pravidlá rozšírené tak, že pri prepisovaní symbolov uvažuje postupnosť symbolov na pravej a ľavej strane prepisovaného symbolu.

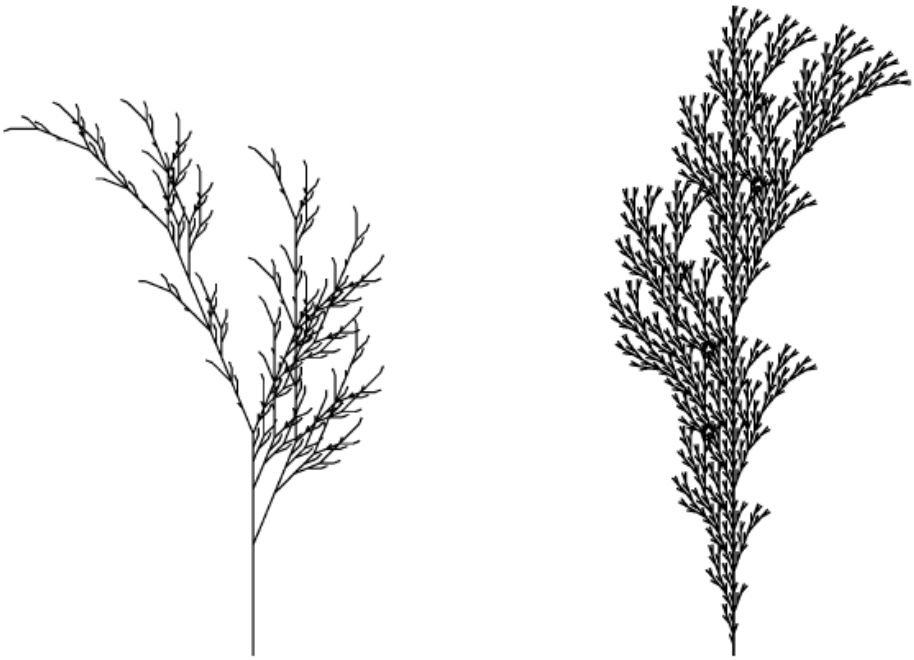
Poslednými a z hľadiska tejto práce najdôležitejšími L-systémami sú tzv. ***parametrické L-systémy***, ktorým bude venovaná samostatná kapitola.

2.2.2 Interpretácia L-systémov

Každému znaku v reťazci vytvorenom pomocou L-systému môžeme priradiť nejaký geometrický význam. Przemysel Prusinkiewicz zaviedol metódu interpretácie pomocou „korytnačky“ (turtle interpretation of the symbols). Každý symbol je preložený ako príkaz pomyselnej korytnačke, reprezentujúcej kresliace zariadenie, ktorá na základe toho putuje rovinou alebo priestorom a pritom kreslí. Na ovládanie korytnačky v priestore sa využíva nasledujúca sada symbolov:

- \+ otočí korytnačku doľava o uhol δ
- \- otočí korytnačku doprava o uhol δ

&	otočí korytnačku dolu o uhol δ
^	otočí korytnačku nahor o uhol δ
\	nakloní korytnačku doľava o uhol δ
/	nakloní korytnačku doprava o uhol δ
|	otočí korytnačku o 180°
F	pohyb korytnačky dopredu o krok *d* s kreslením
f	pohyb korytnačky dopredu o krok *d* bez kreslenia



n = 5, δ = 22,5°
ω: X
p1: X→F-[[X]+X]+F+[+FX]-X
p2: F→FF

n = 5, δ = 20°
ω: F
p1: F→F[+F]F[-F][F]

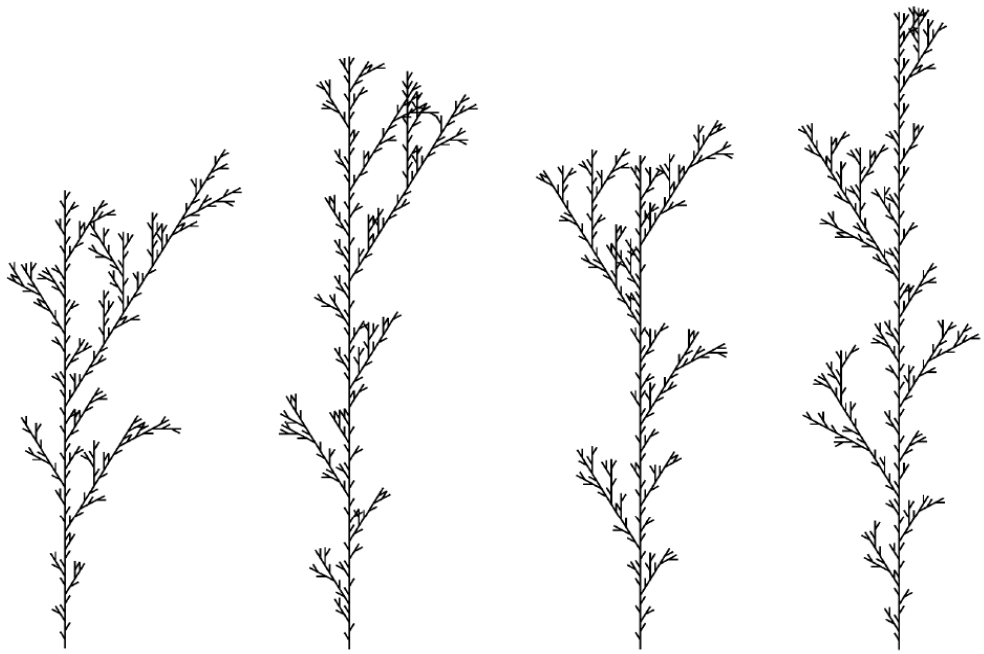
Obr. 1.5: Stromová štruktúra vytvorená zátvorkovým L-systémom

2.2.3 Parametrické L-systémy

Jedným z problémov jednoduchých L-systémov (ako bezkontextových, tak aj kontextových) je nemožnosť ovládať veľkosť kroku korytnačky pri interpretácií. Rovnako je nevýhodná absencia možnosti upravovať v priebehu interpretácie uhol alebo napr. hrúbku čiary, či iné parametre.

Tento problém vyriešili Lindenmayer a Prusinkiewicz [3] vytvorením tzv. parametrických L-systémov, alebo skrátene PL–systémov. Jednoduchšia verzia týchto systémov umožňuje symboly pre pohyb a natočenie korytnačky doplniť o číselný parameter udávajúci hodnotu posunu, resp. natočenia. Okrem toho sú doplnené nové symboly s číselným parametrom, napríklad pre zmenu hrúbky alebo farby čiary.

Omnoho širšie možnosti ponúka použitie symbolických parametrov a doplnenie pravidiel o podmienkovú časť, tvorenú logickým výrazom obsahujúcim formálny parameter. Jednému nahradzovanému symbolu potom zodpovedá viac pravidiel s disjunktnými podmienkovými časťami. Výber pravidla je determinovaný splnením jeho podmienkovej časti po dosadení skutočnej hodnoty parametra nahradzovaného symbolu (obr. 1.7).



n = 5, δ = 30°

ω: F

p1: $F \xrightarrow{33} F[+F]F[-F]F$

p2: $F \xrightarrow{33} F[+F]F$

p3: $F \xrightarrow{34} F[-F]F$

Obr. 1.6: Rôzne variácie rastliny vytvorenej stochastickým L-systémom

2.2.4 Využitie L-systémov

V súčasnosti sa L-systémy osvedčujú ako vhodný simulačný nástroj, napr. pri vizualizácii rastu a interakcie rastlín s okolím, nakoľko ich formalizmus je konzistentný s procesmi prebiehajúcimi v skutočných rastlinách. Naproti tomu majú L-systémy v porovnaní s inými metódami niekoľko ne-

výhod. Napr. exponenciálny nárast výpočtového času a pamäte s rastúcim počtom vykonaných krokov.

L-systémy nachádzajú viacero možností využitia hlavne v oblasti generovania „nedrevených“ rastlín. Dôvodom je že vývin takejto rastliny je vo veľkej miere ovplyvnený vnútornými procesmi, zatiaľ čo pri stromoch prevažujú vonkajšie vplyvy. Napriek tomu je možné jednoduchými metódami generovať pomerne realistické stromové štruktúry, ktoré sú ale dosť všeobecné a väčšinou nemodelujú konkrétny biologický druh.

$n = 10$
$\#define\ r_1\ 0.9$
$\#define\ r_2\ 0.6$
$\#define\ a_0\ 45$
$\#define\ a_2\ 45$
$\#define\ d\ 137.5$
$\#define\ w_r\ 0.707$

ω: A(1, 10)
p1: A(l, w):* → !(w)F(l)[&(a_0)B(l*r_2,w*w_r)]/(d)A(l*r1,w*w_r)
p2: B(l, w):* → !(w)F(l)[-(a_2)$C(l*$r_2$,w*$w_r$)]C(l*r1,w*$w_r$)
p3: C(l, w):* → !(w)F(l)[+(a_2)$B(l*$r_2$,w*$w_r$)]B(l*r1,w*$w_r$)

Obr. 1.7: Stromová štruktúra vygenerovaná parametrickým L-systémom

3 Metódy vizualizácie stromov v reálnom čase

Vzhľadom k obrovskej zložitosti reálnych stromov, je pre zachovanie určitej vizuálnej kvality nevyhnutné, aby boli ich modely vytvorené zo značného množstva polygónov. Dôsledkom je, že pri vykresľovaní väčšieho množstva takýchto modelov je nemožné dosiahnuť také hodnoty FPS (*frames per second*) , aké si vyžaduje interaktívna vizualizácia. Z toho dôvodu bolo vyvinuté značné množstvo akceleračných techník, ktoré tento problém dokážu do určitej miery úspešne riešiť.

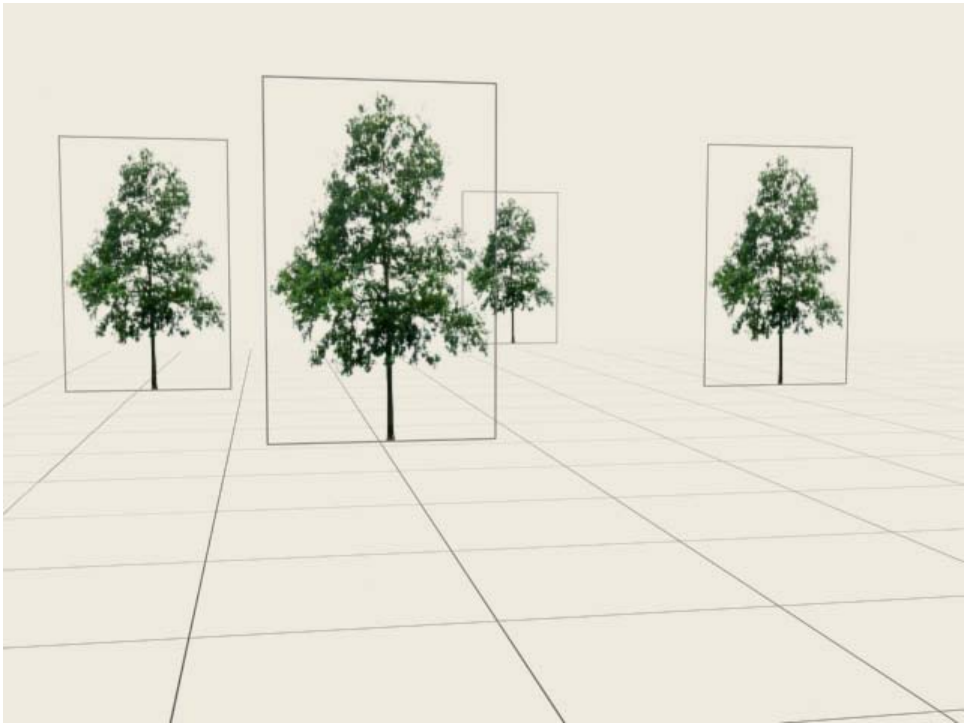
V nasledujúcich kapitolách bude predstavených niekoľko najznámejších a najpoužívanejších techník v tejto oblasti. Niektoré pracujú na princípe zjednodušeného 3D modelu, iné zachádzajú so zjednodušovaním ešte ďalej a pôvodné trojrozmerné modely reprezentujú v 2D a niektoré oba tieto prístupy kombinujú. Všetky však majú spoločnú tú vlastnosť, že akékoľvek zjednodušenie a urýchlenie vykresľovania sa deje na úkor vizuálnej kvality.

3.1 Billboardy

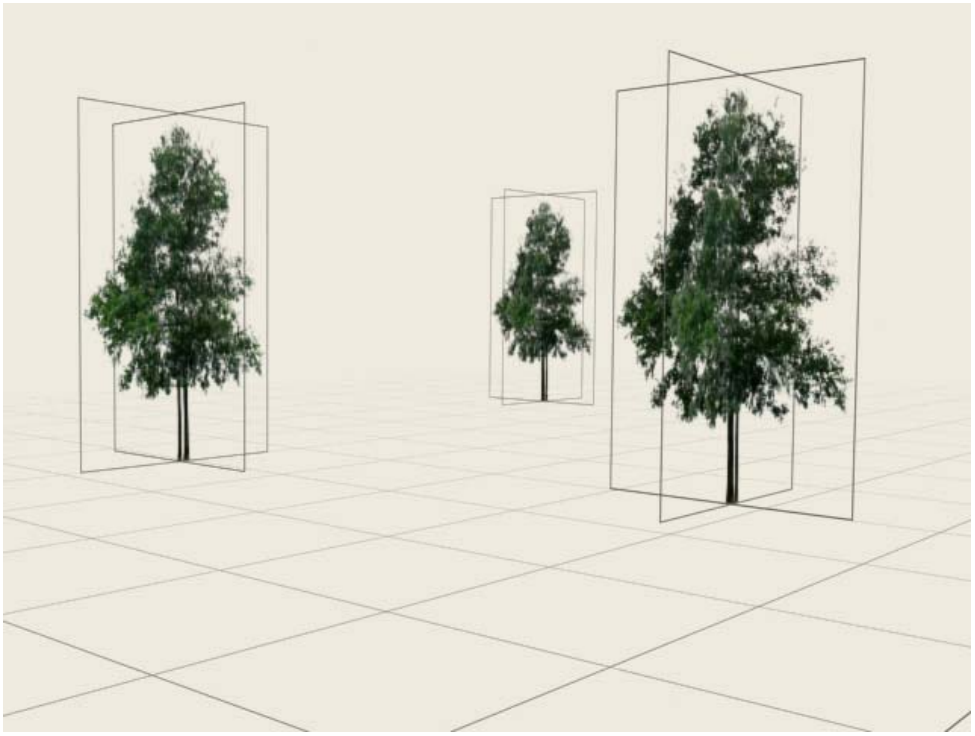
Billboardy sú najbežnejším nástrojom pri vykresľovaní väčších skupín stromov alebo lesov v reálnom čase. Vďaka svojim extrémne nízkym výpočtovým nárokom sú stále považované za najlepšiu voľbu hlavne pri rôznych architektonických vizualizáciách a tam kde je potrebné zobrazovať rozsiahle časti krajiny.

Existuje niekoľko spôsobov vyžitia billboardov. Klasické billboardy (obr. 2.1) predstavujú v prípade stromov najčastejšie obdĺžniky, ktoré sú neustále otočené čelnou stranou ku kamere. Na nich býva namapovaná textúra – obrázok stromu s definovaným alfa-kanálom. Ďalšou možnosťou je použiť na zobrazenie jedného stromu niekoľko billboardov (obyčajne 2 až 4), ktoré sú navzájom prekrížené (obr. 2.2) a ich orientácia nieje viazaná na kameru.

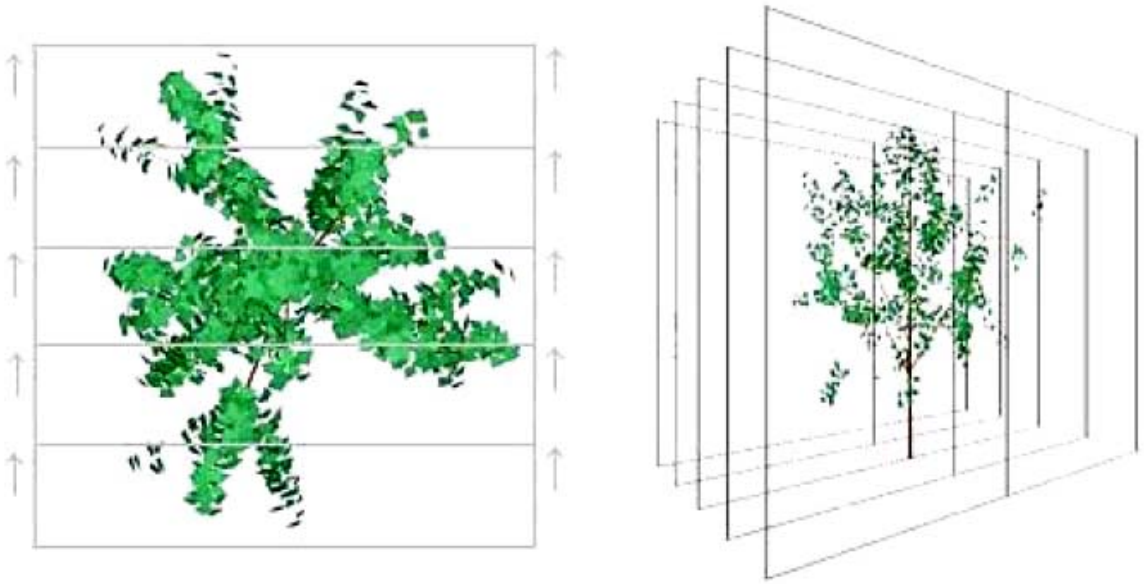
Les zostavený zo spomínaných typov billboardov je samozrejme po vizuálnej stránke neveľmi kvalitný, čo je jasne viditeľné hlavne pri pohybe takýmto lesom. Lepšie výsledky dosiahol v svojej práci Aleks Jakulin [4], ktorý zvolil hybridný prístup. Kmeň a hlavné vetvy sú reprezentované pomocou obyčajnej trojuholníkovej siete, zatiaľ čo kmeň stromu a koruna sú reprezentované pomocou niekoľkých billboardov. Tieto billboardy tvoria skupinu paralelných a rovnako vzdialených štvoruholníkov, ktoré sú individuálne otexturované (obr. 2.3).



Obr. 2.1: Klasické billboardy



Obr. 2.2: Prekrížené billboardy

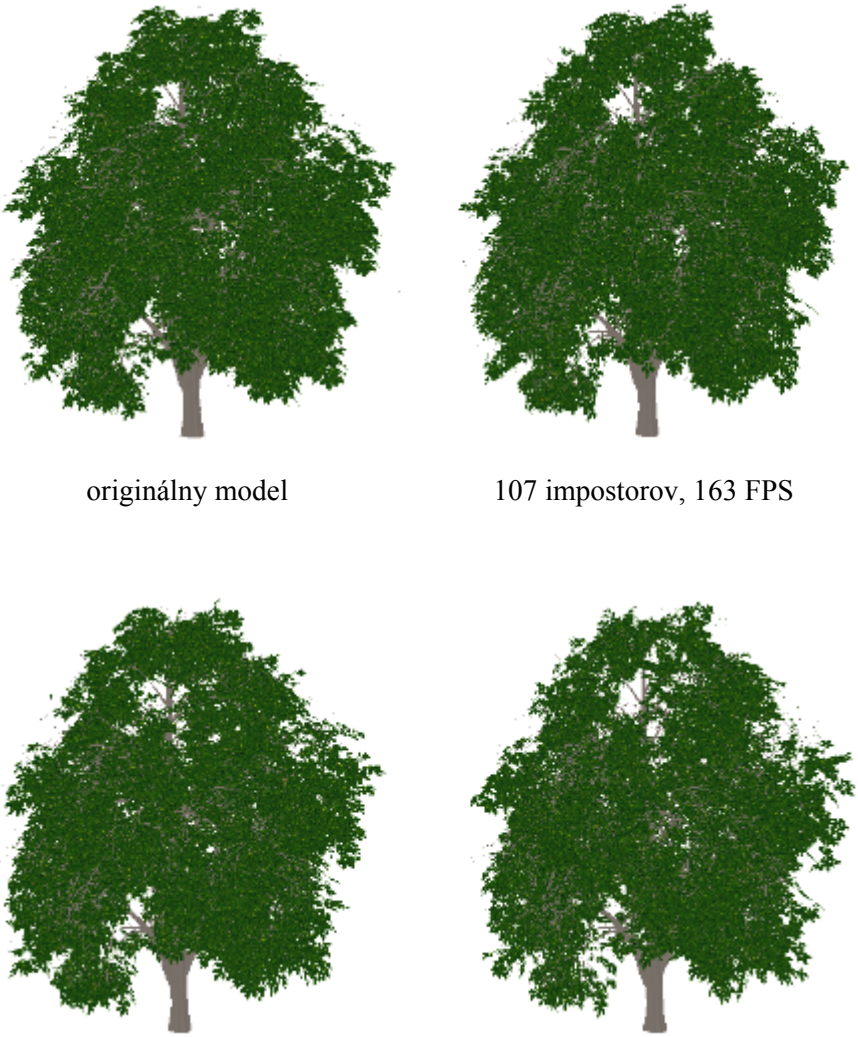


Obr. 2.3: Súbor billboardov podľa Jakulina [4]

3.2 Impostory

Impostory nachádzajú uplatnenie hlavne v scénach, ktoré obsahujú veľké množstvo malých a nehybných objektov. Základný princíp tejto techniky spočíva, tak ako v prípade billboardov, v použití dvojrozmerného obrazu ako náhrady za zložitý 3D objekt. Na rozdiel od billboardov, ktoré sú statické a pri pohybe scénou sa nemenia, impostory sú pri pohybe kamery v určitých intervaloch aktualizované aby sa čo najlepšie prispôsobili uhlu pohľadu. Pri aktualizácií sa pôvodný 3D objekt opäť vykreslí do textúry a táto sa použije pri vykreslení impostoru. Interval aktualizácie býva obyčajne 5 až 50 snímok.

V práci [5] je predstavený prístup k vykresľovaniu stromov, pri ktorom sú zhluky listov reprezentované pomocou polo-transparentných impostorov. Na základe algoritmu *K-means clustering* sa určia jednotlivé plochy tak, aby čo najoptimálnejšie pretínali zhluky listov. Tieto sa následne vykreslia do textúry impostoru. Takáto textúra má však vysokú mieru redundancie, keďže strom je tvorený veľkým množstvom rovnakých listov. Preto je upravená tak, aby niesla len informácie o súradniciach a orientácií jednotlivých listov, čo zredukuje jej veľkosť a čas potrebný na vykreslenie. Na obr. 2.4 sú zobrazené výsledky dosiahnuté pomocou spomínanej metódy.



Obr. 2.4: Porovnanie vizuálnej kvality a rýchlosti vykresľovania originálneho modelu stromu jeho reprezentácií s rôznym počtom impostorov

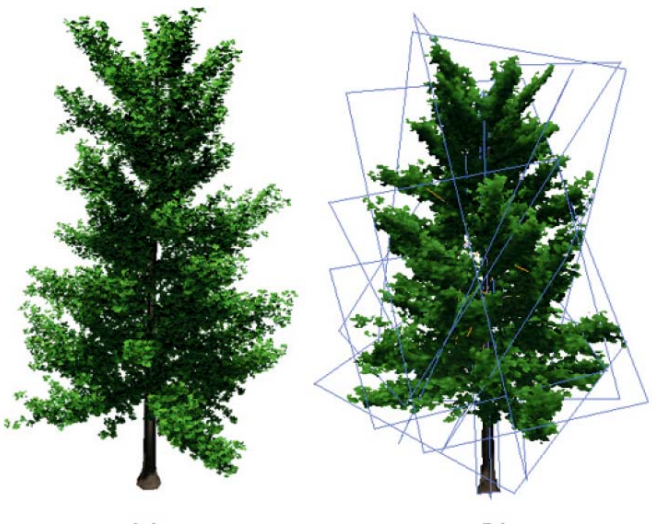
3.3 Billboard clouds

Ďalšia zaujímavá metóda vizualizácie veľkého množstva stromov v reálnom čase je popísaná v [6]. Vychádza z algoritmu označovaného ako *billboard clouds* [7], ktorý je určený na extrémne zjednodušovanie trojrozmerných modelov.

Princíp zjednodušovania spočíva v projekcii modelu, pozostávajúceho z veľkého množstva polygónov, na súbor rovín (obr. 2.5), ktoré aproximujú pôvodnú geometriu v rámci definovanej tolerancie. Najťažšiu a najviac výpočtovo náročnú časť algoritmu predstavuje určenie týchto rovín.

Kvalita zjednodušenia je determinovaná mierou chybovosti ε, ktorá je vyjadrená v percentách polomeru najmenšej možnej gule obklopujúcej daný model. Vrchol môže byť reprezentovaný príslušnou rovinou len v tom prípade, ak je od nej vzdialený menej ako ε. Guľovitá oblasť s stredom v tomto vrchole a s polomerom ε sa nazýva *doména platnosti*. Ak rovina prechádza všetkými doménami platnosti určitého polygónu, je uznaná ako platná pre tento polygón.

Proces zjednodušovania potom spočíva vo vyhľadávaní čo najoptimálnejšieho súboru rovín, tak aby boli platné pre čo najviac polygónov. Poslednou fázou je vygenerovanie textúr pomocou ortogonálnej projekcie polygónov na príslušné platné roviny.



Obr. 2.5: Metóda *billboard clouds* [6]

Obr. 2.6: Porovnanie pôvodného modelu (vľavo) a reprezentácie pomocou *billboard clouds* (vpravo), 21 billboardov

3.4 LOD

Zhruba od sedemdesiatych rokov sa vyžívajú techniky LOD – *Level Of Detail* za účelom zvýšenia výkonu a kvality grafických aplikácií. Tento prístup spočíva v udržiavaní niekoľkých reprezentácií každého modelu na scéne, pričom tieto majú rôznu úroveň zložitosti povrchovej siete a teda aj rôzny počet polygónov. Počas behu aplikácie sú potom menej dôležité objekty vykreslené pomocou jednoduchšej reprezentácie, a naopak dôležité modely v popredí sú zobrazené so zložitejšou povrchovou sieťou a väčším počtom polygónov. V súčasnosti sa využívajú v rámci techník LOD štyri rôzne prístupy [8].

Diskrétny LOD je tradičným prístupom a vyžaduje pre každý objekt niekoľko dopredu pripravených reprezentácií s rôznou úrovňou detailov. Počas vykresľovania sa potom vyberá pre každý objekt príslušná reprezentácia na základe presne stanovených kritérií. Tento prístup nieje náročný na implementáciu a je veľmi dobre podporovaný modernými grafickými kartami. Bez problémov je možné každú úroveň detailov skompilovať do *poľa vertexov*, *display listu* alebo *triangle stripu*, čo dokáže výraze urýchliť proces vykresľovania. Na druhej strane, diskrétny LOD nieje vhodný pre obzvlášť výrazné zjednodušovanie a pre veľké objekty.

Spojitý LOD je odchýlka od tradičného diskrétneho prístupu. Pre každý model vytvára špeciálnu dátovú štruktúru, z ktorej je za behu možné získať reprezentáciu s požadovanou úrovňou detailov. Úroveň detailov býva špecifikovaná presne, nie voľbou z niekoľkých dopredu pripravených možností. To umožňuje lepšie využitie zdrojov a hlavne hladké prepínanie medzi jednotlivými úrovňami.

LOD závislý na pohľade vytvára najvhodnejšiu reprezentáciu za behu, na základe parametrov aktuálneho pohľadu. Ide o vylepšenie predchádzajúcej techniky. Jednotlivé objekty takto môžu obsahovať súčasne niekoľko úrovní zjednodušenia, pričom časti položené bližšie ku kamere majú vyššie rozlíšenie ako vzdialenejšie časti objektu. Taktiež obrysové časti majú vyššiu úroveň detailov. Vďaka spomínaným vlastnostiam umožňuje tento prístup aj výrazné zjednodušenie rozsiahlych objektov.

Predchádzajúci prístup síce úspešne rieši zjednodušovanie veľkých objektov, no čelí ťažkostiam pri zobrazovaní malých objektov. Z tohto dôvodu vznikol *hierarchický LOD*, ktorý zlučuje objekty do zostáv. Na základe vzdialenosti potom zjednodušuje tieto zostavy, nie individuálne objekty. Hierarchický prístup sa dobre dopĺňa s LOD závislým na pohľade vďaka tomu, že zaobchádza s celou scénou ako s jedným objektom, ktorý sa zjednodušovaný v závislosti na pohľade.

Veľmi dôležitým procesom je výber najvhodnejšej úrovne detailov pre príslušný objekt. Tradične sa tento výber uskutočňuje na základe vzdialenosti od kamery, ale v sofistikovanejších systémoch sa využívajú aj ďalšie kritériá:

a. Veľkosť

Úroveň detailov objektu je závislá na počte pixelov ktorými je reprezentovaný na displeji.

b. Excentricita

Vo všeobecnosti sa predpokladá, že divák sa bude pozerať priamo do stredu obrazovky. Preto úroveň detailov môže smerom k okrajom obrazovky klesať.

c. Rýchlosť

LOD objektu je založené na relatívnej rýchlosti vzhľadom k divákovi. Predpokladá sa, že pohybujúce sa objekty sa budú javiť rozmazané, alebo sa objavia len na krátky čas a preto ich divák nebude môcť vidieť jasne. Na druhej strane treba brať do úvahy, že pohyb priťahuje pozornosť.

d. Nemenná hodnota FPS

Úroveň detailov sa volí tak, aby bol dosiahnutý a udržiavaný určitý počet snímok vykreslených za sekundu.

e. Podmienky prostredia

Efekty ako hmla, oblaky, dym, atď. dovoľujú objektom použiť jednoduchšiu reprezentáciu, pretože scéna sa javí rozmazaná a nieje možné postrehnúť jemné detaily.

f. Dôležitosť

Úroveň detailov je založená na ich vizuálnej dôležitosti. Tá je závislá od ich veľkosti, pozície, pohybu a osvetlenia.

Obr. 2.7: Strom s rôznou úrovňou detailov

4 Vlastný generujúci algoritmus

V nasledujúcej časti bude predstavený môj vlastný algoritmus určený na generovanie vysoko realistických modelov stromov, zároveň však rešpektujúci podmienky vykresľovania v reálnom čase. Je postavený tak, aby dokázal generovať širokú paletu listnatých aj ihličnatých stromov a taktiež napodobniť konkrétne biologické druhy. Algoritmus pracuje na báze parametrického generovania, pričom nemodeluje biologické princípy rastu stromu ale snaží sa napodobniť výhradne jeho geometrickú štruktúru. Zároveň pracuje s určitou mierou náhodnosti, takže zakaždým generuje mierne odlišné stromy. Vstupom algoritmu je 40 parametrov, textúra kôry a ľubovoľný počet textúr listu. Samotná metóda a proces generovania bude podrobne popísaný v nasledujúcich kapitolách. Súčasťou práce je demonštračná aplikácia ktorá na základe tohto algoritmu generuje niekoľko druhov stromov. Aplikácia je vytvorená s využitím voľne dostupného súboru knižníc *Open Scene Graph* [9].

4.1 Open Scene Graph

Open Scene Graph (ďalej len OSG) je prenosný grafický toolkit určený na vývoj vysokovýkonných grafických aplikácií ako sú letové simulátory, hry, virtuálna realita alebo rôzne vedecké vizualizácie. Poskytujúc objektovo orientovanú nadstavbu nad OpenGL oslobodzuje programátora od implementácie a optimalizácie nízko-úrovňových funkcií a poskytuje veľké množstvo užitočných pomôcok pre vývoj grafických aplikácií.

OSG má stromovú štruktúru, typicky kreslenú s koreňom hore a s listami dolu. Počiatok stromu tvorí najvrchnejší – koreňový uzol, ktorý zahrňuje celú virtuálnu scénu. Strom sa smerom nadol rozvetvuje na hierarchiu uzlov reprezentujúcich buď zoskupenia objektov, pozície objektov, animácie objektov, alebo definície logických vzťahov medzi objektmi. Listy grafu reprezentujú samotné fyzické objekty, ich geometriu a materiál.

Aj keď OSG nieje kompletný herný alebo simulačný „engine“, býva jeho hlavným komponentom, pričom je primárne určený na reprezentáciu virtuálnej scény a jej efektívne vykresľovanie. Fyzikálny model, detekcia kolízií alebo audio sú prenechané iným knižniciam, ktoré je možné s OSG integrovať.

4.1.1 Hlavné výhody OSG

Výkonnosť

OSG podporuje techniky ako *frustum culling*, *occlusion culling*, *smal feature culling*, LOD, impostory, *vertexové polia* alebo *display listy*. Taktiež umožňuje jednoduché prispôsobenie procesu vykresľovania, takže je možné implementovať techniky ako *spojitý LOD* a tak interaktívne vizualizovať rozsiahle terény.

Produktivita

Jadro OSG zapúzdruje veľkú väčšinu funkcií OpenGL, vrátane najnovších rozšírení, poskytuje rôzne optimalizácie vykresľovania a veľké množstvo prídavných knižníc, vďaka čomu umožňuje veľmi rapídny vývoj vysokovýkonných grafických aplikácií. Umožňuje programátorovi sústrediť sa výhradne sa obsah a funkčnosť namiesto nízko-úrovňového programovania.

Knižnica *osgDB*, určená na zapisovanie a čítanie databáz, podporuje široké spektrum formátov na základe mechanizmu rozširujúcich prídavných modulov – distribúcia v súčasnosti zahŕňa 33 pluginov na načítanie 3D a obrázkových formátov: OpenFlight (.flt), TerraPage (.txp) vrátane podpory multi-threadingu, LightWave (.lwo), Alias Wavefront (.obj), Carbon Graphics GEO (.geo), 3D Studio MAX (.3ds), Peformer (.pfb), Quake Character Models (.md2). Direct X (.x), Inventor Ascii 2.0 (.iv)/ VRML 1.0 (.wrl), Designer Workshop (.dw), AC3D (.ac) a vlastný .osg ASCII formát. Ďalej moduly na načítanie .rgb, .gif, .jpg, .png, .tiff, .pic, .bmp, .dds (vrátane komprimovaných mip-map bitmáp), .tga a qucktime (pod OSX).

OSG taktiež obsahuje sadu tzv. *Node Kits*, čo sú samostatné knižnice, ktoré môžu byť skompilované súčasne s vyvíjanou aplikáciou, alebo načítané za behu, poskytujúce podporu časticových systémov (osgParticle), vysokokvalitného textu (osgText) a vizuálnej simulácie (osgSim).

Prenositeľnosť

OSG bol navrhnutý tak, aby bol minimálne závislý na nejakej špecifickej platforme. Pôvodne bol navrhnutý na IRIX, neskôr prenesený na Linux, potom na Windows a nakoniec na FreeBSD, Mac OSX, Solaris, HP-UX a dokonca aj na PlayStation 2.

4.2 Geometrická štruktúra stromu

Algoritmus umožňuje generovať stromy s niekoľkými úrovňami vetvenia, maximálne však do úrovne 3. Na tejto úrovni už vyzerajú stromy dostatočne vierohodne a ďalšie úrovne by zbytočne zvyšovali zložitosť a znemožňovali by vykresľovanie stromu v reálnom čase. Hlavná štruktúra a princíp vetvenia s príslušnými parametrami je načrtnutý na obr. 3.1.

Samotný proces generovania je rozdelený na niekoľko fáz, pričom v prvej fáze sa generujú tzv. kontrolné body, ktoré definujú geometrickú štruktúru a zložitosť stromu. Počet týchto bodov vo vetve alebo kmeni priamo podmieňuje detailnosť a mieru vizuálnej kvality výsledného stromu. Ako prvé sa vypočítajú súradnice kontrolných bodov kmeňa. Počet týchto bodov je daný parametrom ***SEG_KMEN*** a výškou stromu podľa vzťahu:

$$pocet\_kontr.\_bodov\_kmena = round\left(SEG\_KMEN \cdot VYSKA\_STROMU\right)$$

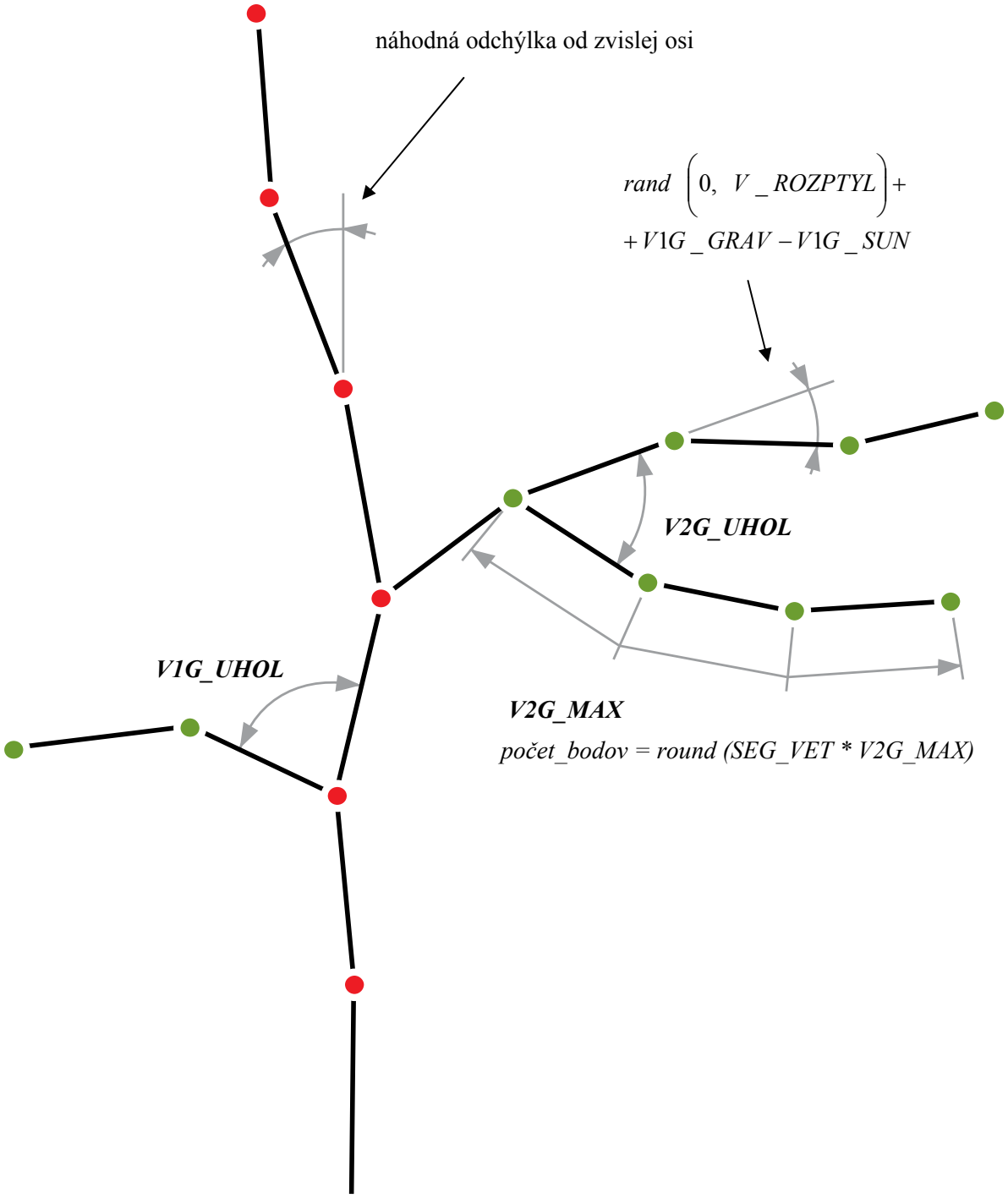
Parameter ***MAX_ROZPTYL_KMEN*** udáva maximálnu možnú odchýlku posledného kontrolného bodu od zvislej osi. Táto odchýlka má náhodnú veľkosť a je aplikovaná na všetky kontrolné body s tým, že sa jej maximálna hodnota smerom k základni stromu lineárne zmenšuje.

Vetva 1. generácie vyrastá z kontrolného bodu kmeňa pod uhlom ***V1G_UHOL***. Tento uhol môže byť po celej výške stromu konštantný, alebo sa môže smerom k vrcholu postupne zmenšovať alebo zväčšovať podľa hodnoty parametra ***UHP***. Pri pohľade zhora sú vetvy špirálovito rozložené s tým, že každá vetva leží v samostatnom kvadrante. Uhol v rovine *XY* pod ktorým vetva vyrastá je náhodný a pre 1. kvadrant je daný nasledujúcim vzťahom, pričom medze sú upravené parametrom ***ALFA_KOR***:

$$\alpha = rand\ \left(0 + ALFA\_KOR, \frac{\pi}{2} - ALFA\_KOR\right)$$

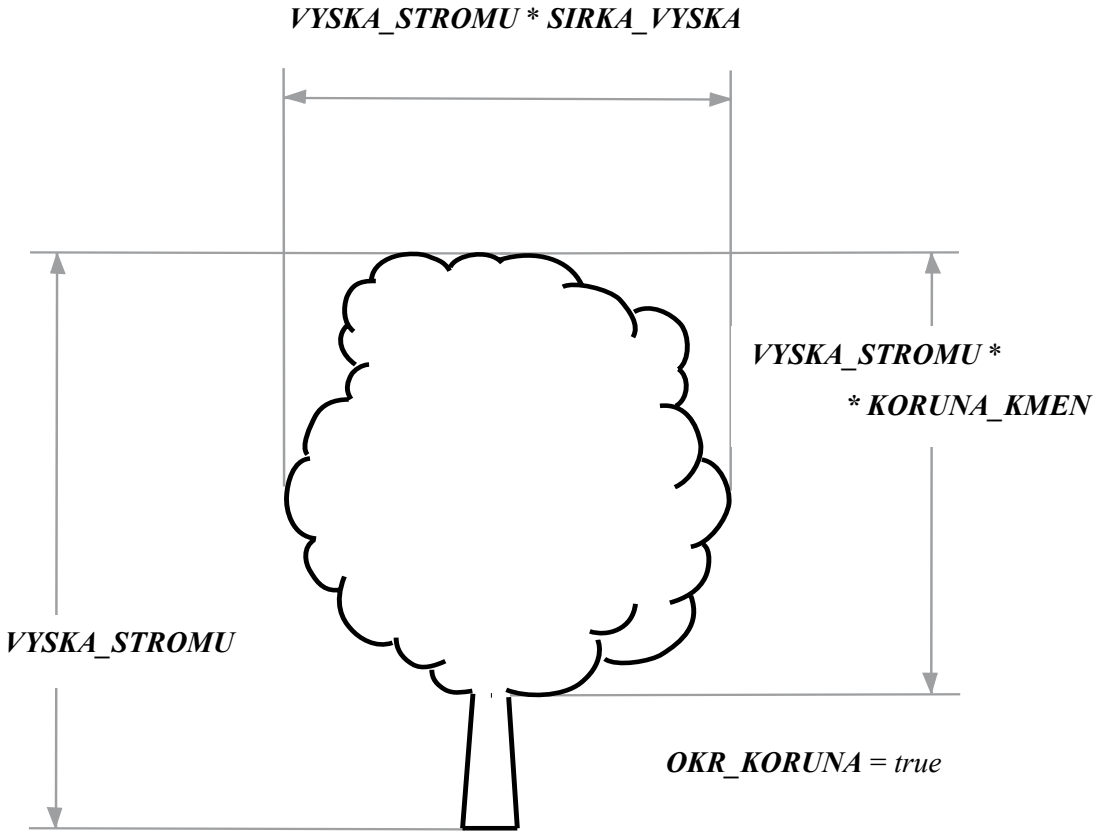
Pre ostatné kvadranty sa uhol počíta analogicky. Funkcia *rand(a, b)* generuje náhodné reálne číslo s presnosťou typu *float* a s rovnomerným rozložením v intervale od *a* po *b*. Počet kontrolných bodov akejkoľvek vetvy podmieňuje parameter ***SEG_VET*** na základe podobného vzťahu ako v prípade kmeňa, pričom parameter ***VYSKA_STROMU*** je nahradený dĺžkou vetvy. Maximálna dĺžka vetvy 1. generácie je daná parametrom ***SIRKA_VYSKA*** čo je vlastne pomer max. polomeru koruny (a teda aj dĺžky vetvy 1. generácie) a výšky stromu. Dĺžky vetiev úrovne 1 sú ďalej dané tvarom koruny, pričom je k dispozícii okrúhla a kužeľovitá koruna. Parameter ***KORUNA_KMEN*** určuje pomer výšky koruny k výške stromu (obr. 3.2).

Parameter ***NEXTG_POMER*** definuje normalizovanú veľkosť tej časti vetvy 1. generácie na ktorej budú vyrastať vetvy ďalšej úrovne. Jeho význam je zrejmý z obr. 3.3.



Obr. 3.1: Hlavná štruktúra stromu – kontrolné body

Vplyv gravitácie na vetvu vyjadruje parameter ***VnG_GRAV***, kde ***n*** je číslo generácie. Tento vplyv je modelovaný pomocou odchýlky každého segmentu vetvy smerom nadol. Veľkosť tejto odchýlky je rovná hodnote ***VnG_GRAV***. Algoritmus zároveň dokáže pomocou parametra ***VnG_SUN*** modelovať zakrivenie koncov vetiev nahor vplyvom slnečného žiarenia. Hodnota ***V_ROZPTYL*** udáva limit náhodných odchýliek kontrolných bodov vetvy od pôvodného smeru, určeného uhlom ***VnG_UHOL*** .



Obr. 3.2: Parametre koruny

Maximálna dĺžka vetiev druhej a tretej generácie sú definované parametrami ***V2G_MAX*** a ***V3G_MAX***. Skutočná dĺžka vetvy je však závislá aj na tom z ktorého miesta na rodičovskej vetve vyrastá, pričom platí, že vetvy vyrastajúce bližšie ku koncu rodičovskej vetvy sú kratšie.

Obr. 3.3: vľavo: ***NEXTG_POMER*** = *1,0* vpravo: ***NEXTG_POMER*** = *0,5*

4.3 Povrch stromu

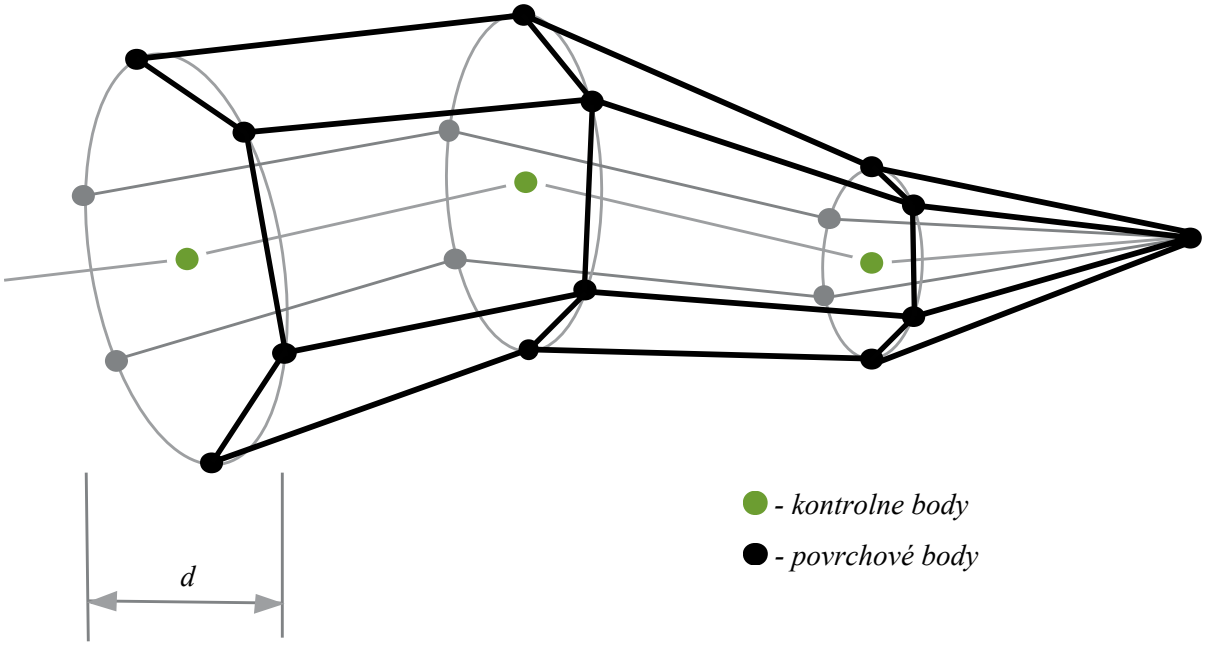
Povrch kmeňa a vetiev je generovaný ako sieť štvoruholníkov a trojuholníkov, ktorých počet je možné ovplyvniť parametrom ***DETAIL***. Generovaniu tejto povrchovej siete predchádza výpočet polomeru pre každý kontrolný bod a určenie prstenca povrchových bodov, taktiež pre každý kontrolný bod. Tento proces je načrtnutý na obr. 3.4.

Priemer základne vetvy je možné nastaviť parametrom ***VRV_POMER*** podľa nasledujúceho vzťahu:

$$priemer = dlzka\_vetvy \cdot VRV\_POMER$$

Tento priemer sa smerom ku koncu vetvy lineárne zmenšuje až na hodnotu *0*, ktorú nadobúda posledný kontrolný bod. Zároveň platí, že priemer základne synovskej vetvy nieje nikdy väčší ako priemer rodičovskej vetvy resp. kmeňa v bode, v ktorom táto synovská vetva vyrastá. Priemer základne kmeňa sa počíta analogicky – ako súčin parametrov ***VYSKA_STROMU*** a ***KRV_POMER***.

Ďalším krokom je výpočet povrchových bodov. Povrchové body tvoria prstence s príslušným priemerom *d* okolo kontrolných bodov, tak ako to znázorňuje obrázok 3.4. Počet povrchových bodov v prstenci je daný parametrom ***DETAIL*** podľa tabuľky T.1. Tieto body predstavujú vrcholy trojuholníkov a štvoruholníkov reprezentujúcich povrch stromu.



Obr. 3.4: Generovanie povrchu

Na takto vzniknuté povrchové segmenty je aplikované tzv. Gourandovo tieňovanie, čo má za následok, že hrany prestanú byť viditeľné a povrch sa javí hladký.

DETAIL	počet povrchových bodov v prstenci			
	kmeň	**generácia 1**	**generácia 2**	**generácia 3**
1	6	5	3	3
2	8	6	5	3
3	12	8	6	5

T.1: Počet povrchových bodov v závislosti na hodnote parametra *DETAIL*

4.3.1 Textúra povrchu

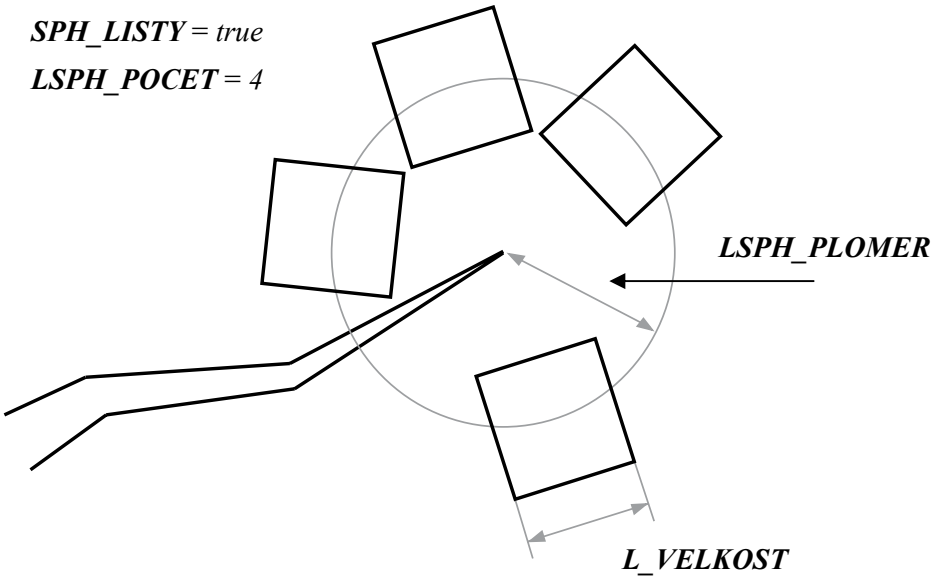
Názov súboru s textúrou kmeňa a vetiev nesie parameter ***K_SUBOR***. Využívajú sa tzv. bezšvové textúry, ktoré nespôsobujú dlaždicový efekt pri opakovaní. Táto textúra je namapovaná na povrch stromu nasledujúcim spôsobom: Ak majú parametre ***H_TEX_MUL*** a ***V_TEX_MUL*** hodnotu 1, textúra sa po obvode vetvy neopakuje. Po dĺžke vetvy sa textúra opakuje toľkokrát, aby pokryla celú vetvu a zostal pri tom zachovaný pomer strán pôvodného obrázku. Pri hodnotách vyšších ako 1, parameter ***H_TEX_MUL*** udáva počet opakovaní textúry po obvode a parameter ***V_TEX_MUL*** násobí počet opakovaní textúry po dĺžke vetvy.

4.4 Listy

Algoritmus ponúka dva spôsoby vytvárania listov, tzv. sférickú metódu (obr. 3.6) a billboardovú metódu (obr. 3.8), pričom každá z nich je vhodná pre iné druhy stromov. Obe tieto metódy využívajú pri zobrazovaní tzv. *alfa blending*, preto je potrebné aby textúry listov mali definované hodnoty alfa-kanálu a boli uložené vo formáte ktorý tento kanál podporuje.

4.4.1 Sférická metóda

Pri tomto spôsobe generovania sú listy zoskupené do zhlukov na konci každej vetvy resp. kmeňa (obr. 3.5). Tieto zhluky majú tvar gule s polomerom definovaným pomocou parametra ***LSPH_POLOMER***. Počet listov v zhluku udáva parameter ***LSPH_POCET***. Listy majú tvar štvorca, ktorého dĺžka strany je rovná hodnote ***L_VELKOST***. Na tento štvorec je namapovaná textúra listu, pričom názov súboru s touto textúrou špecifikuje parameter ***TEX_SUBORY***. Algoritmus dovoľuje zadať niekoľko súborov s textúrami. Tieto sa potom na listy aplikujú náhodne s rovnomerným rozložením. Takto je možné vytvoriť strom s rôznymi listami, prípadne s listami a kvetmi zároveň. Aby boli listy generované sférickou metódou, je potrebné nastaviť hodnotu parametra ***SPH_LISTY*** na *true*.

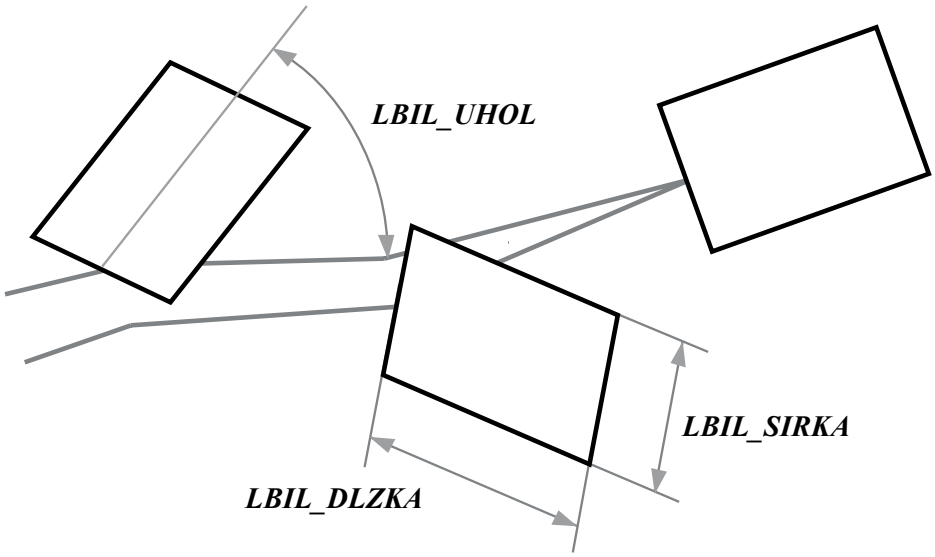


Obr. 3.5: Sférická metóda

Obr. 3.6: Listy vygenerované sférickou metódou,
s textúrou (vľavo) a bez textúry (vpravo)

4.4.2 Billboardová metóda

V tomto prípade listy vyrastajú z kontrolných bodov vetiev poslednej úrovne, tak ako je to naznačené na obr. 3.7. Z pohľadu geometrie sú listy tvorené obdĺžnikmi so šírkou určenou parametrom ***LBIL_SIRKA*** a dĺžkou parametrom ***LBIL_DLZKA***. Uhol, ktorý tento obdĺžnik zviera s vetvou je vyjadrený parametrom ***LBIL_UHOL***. Ostatné uhly sú náhodné. Rovnako ako pri listoch generovaných sférickou metódou, súbory s textúrami listov špecifikuje parameter ***TEX_SUBORY***. Hodnota ***BIL_TRANSLUC*** vyjadruje priesvitnosť (nie priehľadnosť) listu v rozmedzí *0,0* až *1,0*.



Obr. 3.7: Billboardová metóda

Obr. 3.7: Listy (ihličie) vygenerované pomocou billboardovej metódy

4.5 Možnosti výkonnostných optimalizácií

Algoritmus ponúka niekoľko možností ako optimalizovať proces generovania a výsledný model stromu za účelom dosiahnutia vyššieho výkonu pri vykresľovaní. Najjednoduchšou možnosťou je zjednodušenie povrchovej siete modelu znížením hodnoty parametra ***DETAIL***. Ak uvažujeme, že strom zaberá maximálne 70% výšky obrazovky, čo reprezentuje veľkú väčšinu prípadov, je zmena z ***DETAIL*** = *3* na ***DETAIL*** = *1* takmer nepostrehnuteľná. Pri strome s priemernou hustotou vetiev, aké zobrazuje napr. demonštračná aplikácia na priloženom CD, môže táto zmena znamenať zníženie počtu bodov povrchovej siete v priemere o 13 000, čo predstavuje zhruba 17%. Toto zníženie však nemá zásadný vplyv na čas potrebný na vykreslenie stromu, pretože prevažná časť výkonu je spotrebovaná na vykresľovanie listov.

Ďalšou možnosťou je zníženie počtu listov a zároveň zväčšenie ich rozmerov, aby sa zachovalo pokrytie stromu. Táto zmena je však v porovnaní s predchádzajúcou o niečo viac nápadná, spôsobuje totiž, že stromy vypadajú mladšie. Prináša však dosť významné zvýšenie počtu snímok vykreslených za sekundu. Ako príklad môže poslúžiť jeden zo stromov z demonštračnej aplikácie (obr. 3.8), pri ktorom bola znížená hodnota parametra ***LSPH_POCET*** z 10 na 5 pri súčasnom zvýšení hodnoty ***L_VELKOST*** z 0,2 na 0,5. Táto zmena priniesla nárast hodnoty FPS zo 14 na 24, čo je dosť výrazné zlepšenie.

Obr. 3.8: Porovnanie vizuálnej kvality:
vľavo: ***LSPH_POCET*** = *10* ***L_VELKOST*** = *0,2* vpravo: ***LSPH_POCET*** = *5* ***L_VELKOST*** = *0,5*

Efekt „mladého stromu“ je možné eliminovať tým, že sa namiesto textúry samostatného listu použije textúra so skupinou listov. Takto sa počet listov opticky zvýši, pričom sa zmenšia ich rozmery.

Ďalšou možnou výkonnostnou optimalizáciou je využiť billboardovú metódu na nahradenie poslednej úrovne (prípadne dvoch posledných úrovní) vetvenia veľkými obdĺžnikmi s textúrou celej vetvy.

4.6 Demonštračná aplikácia

Metódu generovania stromov predstavenú v tejto kapitole demonštruje jednoduchá aplikácia spustiteľná pod WinXP, ktorú je možné nájsť na priloženom CD. Aplikácia vygeneruje sedem stromov rôzneho druhu a umožňuje užívateľovi medzi týmito stromami prepínať a jednotlivo prezerať. Taktiež je možné zmeniť parametre v konfiguračných súboroch jednotlivých stromov a vytvárať tak vlastné stromy.

Aplikácia bola napísaná v jazyku C++ v prostredí *Visual Studio 2003*, s využitím knižníc *OpenSceneGraph*, pričom zdrojový kód je dostupný taktiež na priloženom CD. Tento zdrojový text je rozdelený do niekoľkých modulov podľa funkčnosti. Ide o nasledujúce moduly:

geometria.cpp

Realizuje matematické a geometrické výpočty, taktiež transformácie vektorov a bodov, pre vytvorenie základnej štruktúry stromu a výpočet súradníc kontrolných a povrchových bodov.

mersenne.cpp

V tomto module je implementovaný generátor náhodných čísel typu *mersenne twister*. Súbor je prevzatý z [11] s tým, že v ňom bola doplnená funkcia na výpočet náhodného čísla typu *float* v ľubovoľnom intervale.

povrch.cpp

Definuje funkcie pre generovanie povrchovej siete stromu a výpočet koordinačných súradníc pre textúru povrchu.

listy.cpp

V tomto module sú definované funkcie generujúce listy.

tex.cpp

Modul preberá správu textúr a aplikáciu textúr, načítanie obrázka zo súboru, zostavenie a výber textúry, nastavenie *alfa blendingu* pri textúrach listu, atď.

ovladanie.cpp

Definuje ovládanie aplikácie užívateľom.

hud.cpp

Realizuje vytvorenie a zobrazenie panelu s informáciami v rohu obrazovky.

parser.cpp

Úlohou funkcii definovaných v tomto súbore je spracovať konfiguračný súbor stromu a získať hodnoty jednotlivých parametrov.

Aplikácia vytvára stromy na základe jednoduchých konfiguračných súborov. Každý takýto súbor obsahuje hodnoty parametrov určitého stromu, pričom formát je nasledovný:

```
parameter1=hodnota1
parameter2=hodnota2
parameter3=hodnota3
...
```

Na poradí parametrov nezáleží. Ak definícia akéhokoľvek parametra v súbore chýba, alebo ak má nepovolenú hodnotu, je tento parameter nahradený preddefinovanou hodnotou.

4.6.1 Ovládanie aplikácie

Demonštračná aplikácia sa ovláda klávesnicou a myšou nasledujúcim spôsobom:

KLÁVESA	**VÝZNAM**
←	predchádzajúci strom
→	nasledujúci strom
'DEL'	skryť / zobraziť informačný panel
'ESC'	ukončenie programu
'**1**'	**TrackBall manipulátor**
	ľavé tlačidlo myši – otáčanie
	stredné tlačidlo – posúvanie

pravé tlačidlo – približovanie / odďaľovanie

’**2**’	**“flight camera” manipulátor**
	ľavé tlačidlo myši – zrýchlenie
	stredné tlačidlo – stop
	pravé tlačidlo – spomalenie, cúvanie
	pohyb myšou – ovládanie letu
’**3**’	**“flight camera” manipulátor**
	*mód Q – stlačením ’**q**’*
	pohyb myšou dopredu / dozadu – pohyb na scéne dopredu / dozadu
	pohyb myšou doprava / doľava – otočenie doprava / doľava
	*mód A – stlačením ’**a**’*
	ľavé tlačidlo myši – zrýchlenie
	stredné tlačidlo – stop
	pravé tlačidlo – spomalenie, cúvanie
	pohyb myšou doprava / doľava – otočenie doprava / doľava
’**f**’	režim zobrazenia: celá obrazovka / okno
’**l**’	vypnutie / zapnutie osvetlenia
’**o**’	export do formátu *.osg*
’**s**’	informácie o vykresľovaní
’**t**’	vypnutie / zapnutie textúr
’**SPACE**’	reset kamery

5 Záver

Táto práca pojednáva o súčasných metódach generovania a vizualizácie realistických stromov, a zároveň prezentuje môj vlastný prístup k tejto problematike. Ide o algoritmus, ktorý nemodeluje princípy rastu a vývinu stromu z biologického hľadiska, namiesto toho sa snaží napodobniť výhradne jeho geometrickú štruktúru. A to na takej úrovni zjednodušenia, aby výsledný strom bol dostatočne realistický, no zároveň aby bolo možné využiť ho v interaktívnych vizualizáciách. To znamená, že algoritmus reprezentuje akýsi kompromis medzi mierou realistickosti stromu a jednoduchosťou jeho štruktúry. Na základe pozorovania skutočných stromov som určil približne 40 parametrov, pomocou ktorých je možné definovať široké spektrum listnatých aj ihličnatých stromov, pričom je možné priamo modelovať konkrétne biologické druhy.

Ďalším cieľom, ktorý som sa snažil vo svojej práci splniť, je jednoduchosť spomínaného algoritmu. Tento bol vytvorený tak, aby umožňoval bežnému užívateľovi vytvárať stromy na základe pozorovania alebo fotografie, bez toho aby poznal nejaké zložité matematické alebo biologické princípy.

Aj keď je tento algoritmus určený primárne na generovanie stromov, ktoré by vyhovovali podmienkam vizualizácie v reálnom čase, jeho možnosti tu nekončia. Umožňuje generovať stromy aj na omnoho vyššej úrovni zložitosti a teda aj realistickosti. Vývoj grafického hardvéru totiž napreduje vysokým tempom a preto v blízkej budúcnosti už určite nebude problém zobrazovať v reálnom čase aj takéto stromy, tvorené obrovským množstvom polygónov.

Ďalšou súčasťou tejto práce je demonštračná aplikácia, ktorá na základe spomínaného algoritmu generuje niekoľko stromov. Textúry, ktoré aplikácia využíva, som buď vytvoril ja sám, alebo sú voľne dostupné na internete. Parametre vygenerovaných stromov sú dané jednoduchými konfiguračnými súbormi, takže užívateľ ich môže ľahko meniť.

Práca ponúka niekoľko možností pokračovania v budúcnosti. Jednou z nich je napríklad vytvorenie jednoduchého užívateľského rozhrania a aplikácie ktorá by umožňovala interaktívne generovať modely stromov a tieto exportovať do známych 3D formátov, ako napr. *.3ds*, *.lwo*, *.obj*, atď. Demonštračná aplikácia zatiaľ ponúka export do formátu *.osg*. Ďalšou možnosťou by bolo pokračovať v generovaní rozsiahlych lesných scenérií a ich vizualizácie v reálnom čase, alebo vytvoriť na základe prezentovaného algoritmu *plug-in* modul do niektorého z rozšírených 3D modelovacích prostredí ako je napr. 3Ds MAX, CINEMA 4D alebo Blender.

Literatúra

[1] WEBER, J., PENN, J. *Creation and rendering of realistic trees*. [online]. 1995, [cit. 2007-05-05]. Dostupné z: <http://portal.acm.org>.

[2] *Arbaro, release 1.9.7 – 3D tree generator*, 2003.
Dostupné z: <http://sourceforge.net/projects/arbaro>.

[3] PRUSINKIEWICZ, P., LYNDENMAYER, A. *The Algorithmic Beauty Of Plants*. [online]. Springer-Verlag, New York, 1st edition, 1996.
Dostupné z: <http://algorithmicbotany.org/papers/abop/abop.pdf.>.

[4] JAKULIN, A. *Interactive vegetation rendering with slicing and blending*. [online]. 2000, [cit. 2007-05-05]. Dostupné z: <ai.fri.uni-lj.si/~aleks/slicing-and-blending/trees-electronic.pdf>.

[5] GARCIA, I., SBERT, M., SZIRMAY, L. *Leaf cluster impostors for tree rendering with parallax*. [online]. 2005, [cit. 2007-05-05]. Dostupné z: <www.iit.bme.hu/~szirmay/tree2.pdf>.

[6] FUHRMANN, A., UMLAUF, E., MANTLER, S. *Extreme model simplification for forest rendering*. [online]. 2004, [cit. 2007-05-05].
Dostupné z: <http://www.vrvis.at/vr/billboardclouds/>.

[7] DECORET, X., DURAND, F., DORSEY, J. *Billboard clouds for extreme model simplification*. [online]. 2003, [cit. 2007-05-05].
Dostupné z: <http://www.cs.utah.edu/classes/cs5610/handouts/billboard-clouds.pdf>.

[8] DAMAN, D., HEOK, T. *A review on level of detail*. [online]. 2004, [cit. 2007-05-05].
Dostupné z: < http://portal.acm.org>.

[9] *Open Scene Graph*, *release 1.2 – graphic toolkit*, 2006
Dostupné z: <http://www.openscenegraph.com>.

[10] *Mersenne Twister – very fast random number generator*, 2002
Dostupné z: <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>.

Zoznam príloh

Príloha 1. Zoznam parametrov generujúceho algoritmu.

Príloha 2. Ukážky vygenerovaných stromov.

Príloha 3. CD s demonštračnou aplikáciou a zdrojovými textami.

Zoznam parametrov generujúceho algoritmu

VYSKA_STROMU - výška stromu, reálne číslo
MAX_ROZPTYL_KMEN - max. odchýlka najvyššieho bodu kmeňa od zvislej osi, reálne číslo
SIRKA_VYSKA - pomer šírka stromu / výška stromu, reálne číslo
KORUNA_KMEN - pomer výška stromu / výška koruny, reálne číslo
SEG_VET - segmentácia vetvy, celé číslo
SEG_KMEN - segmentácia kmeňa, celé číslo
ALFA_KOR - korekcia náhodného uhlu pod kt. vyrastá vetva, reálne číslo
V_ROZPTYL - miera náhodného rozptylu kontrol. bodov vetvy, reálne číslo
KRV_POMER - pomer priemeru kmeňa k výške stromu, reálne číslo
VRV_POMER - pomer priemeru vetvy k dĺžke vetvy, reálne číslo
LSPH_POLOMER - vzdialenosť listov od koncového bodu vetvy, reálne číslo
LSPH_POCET - počet listov v zhluku, sférická metóda, reálne číslo
LBIL_DLZKA - dĺžka billboardu, billboardová metóda, reálne číslo
LBIL_SIRKA - šírka billboardu, billboardová metóda, reálne číslo
LBIL_UHOL - uhol billboardu k vetve, reálne číslo
L_VELKOST - veľkosť listu, sférická metóda, reálne číslo
LTEX_POCET - počet textúr listu
SPH_LISTY - generovať listy sférickou metódou, *false / true*
BIL_LISTY - generovať listy billboardovou metódou, *false / true*
OKR_KORUNA - tvar koruny: *true* – okrúhla, *false* - kužeľová
V1G_POCET - počet vetiev 1. generácie vyrastajúcich z jedného bodu, celé číslo
BIL_TRANSLUC - priesvitnosť listu, billboardová metóda, reálne číslo: 0,0 – 1,0
K_SUBOR - názov súboru s textúrou kôry
H_TEX_MUL - násobiteľ textúry kôry po obvode vetvy, celé číslo
V_TEX_MUL - násobiteľ textúry kôry po dĺžke vetvy, celé číslo
V1G_UHOL - uhol vetvy 1. generácie ku kmeňu, reálne číslo
UHP - prírastok uhlu vetvy 1. generácie ku kmeňu, reálne číslo
NEXTG_POMER - miera rozvetvovania na prvej úrovni, reálne číslo: 0,0 – 1,0
V1G_GRAV - vplyv gravitácie na vetvu 1. generácie, reálne číslo
V1G_SUN - zakrivenie vetvy 1. generácie smerom hore, reálne číslo
V2GENER - generovať druhú generáciu, *false / true*
V2G_UHOL - uhol vetvy 2. generácie ku kmeňu, reálne číslo
V2G_GRAV - vplyv gravitácie na vetvu 2. generácie, reálne číslo
V2G_SUN - zakrivenie vetvy 2. generácie smerom hore, reálne číslo
V2G_MAX - maximálna dĺžka vetvy druhej generácie, reálne číslo
V3GENER - generovať tretiu generáciu, *false / true*
V3G_UHOL - uhol vetvy 3. generácie ku kmeňu, reálne číslo
V3G_GRAV - vplyv gravitácie na vetvu 3. generácie, reálne číslo
V3G_SUN - zakrivenie vetvy 3. generácie smerom hore, reálne číslo
V3G_MAX - maximálna dĺžka vetvy tretej generácie, reálne číslo
DETAIL - úroveň detailov, celé číslo: 1 - 3
TEX_SUBORY - názvy súborov s textúrami listov

Ukážky vygenerovaných stromov