VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

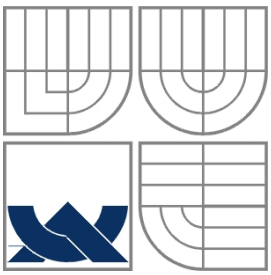
MODELOVACÍ PLUGIN PRE BLENDER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

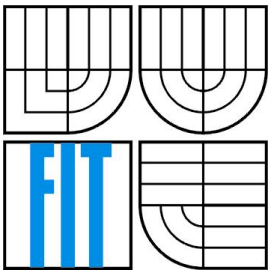
AUTOR PRÁCE
AUTHOR

MICHAL ČERNÁK

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MODELOVACÍ PLUGIN PRO BLENDER

MESH GENERATING PLUGIN FOR BLENDER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ČERNÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL ŽÁK

BRNO 2009

Abstrakt

Tato práce se zabývá automatizovaným generováním modelů stromů v prostředí open-source 3D modelovacího a animačního nástroje Blender. Program byl vytvořen v rozhraní pro programování aplikací Blenderu v jazyce Python. Práce rozebírá nejběžnější techniky generování rostlin a stromů a postup, který byl použit v programu.

Abstract

This work describes automatic generation of trees in open-source 3D graphics application Blender. It also concerns different techniques for generating plants and method.

Klíčová slova

3D, objekty, Blender, generování, stromy, modelování

Keywords

3D, objects, Blender, generation, trees, modeling

Citace

Michal Černák: Modelovací plugin pre Blender, bakalárska práca, Brno, FIT VUT v Brně, 2009

Modelovací plugin pre Blender

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením...
Další informace mi poskytli...
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Černák
2.5.2009

Poděkování

Chcel by som poďakovať môjmu školiteľovi Ing. Pavlovi Žákovi, Romanovi Kantorovi, Tomášovi Kaňokovi, Michalovi Krupovi, Milošovi Vyletelovi, Petrovi Sagálovi, Anke Chmurovej, mojim rodičom a bratovi.

Obsah

1 Úvod

V dnešnej dobe môžeme intenzívne vnímať vzostup multimédií a počítačovej grafiky v našom okolí. To sa obzvlášť týka oblasti počítačovej 3D grafiky, ako aj modelovania a animácií. So zvyšujúcou sa úrovňou počítačovej grafiky sa však zvyšujú nároky nielen na detailnosť a vierohodnosť, ale aj na efektivitu vytvárania grafiky a grafických prvkov. Na zvýšenie tejto efektivity sa využívajú rôzne generátory objektov, ktoré grafikom ušetria čas pri ich modelovaní.

Jedným z dôležitých prvkov v oblasti počítačových hier a animácií je zobrazenie krajiny a okolia, ktoré dotvára atmosféru výsledného produktu. Súčasťou prostredia bývajú stromy a rastliny. Tie dodávajú scénam vierohodnosť a prirodzenosť.

Stromy a inú vegetáciu je vo všeobecnosti možné vytvárať ručne, alebo generovať automaticky pomocou rôznych algoritmov. Automatická tvorba prináša veľké výhody. Ich vytvorenie trvá nepomerne kratšiu dobu oproti ručnému vytváraniu, čím sa zvyšuje efektivita tvorby scenérie. Generátory zvyknú byť veľmi dobre parametrizované, čo umožnuje nastaviť množstvo vlastností generovanej vegetácie. To nám dáva možnosť vytvárať konkrétne druhy rastlín, či dokonca iné špecifické vlastnosti ako je ich vek, mohutnosť a podobne. Generovanie obvykle ovplyvňuje aj náhodná veličina. Vďaka tomu sa dá vyprodukovať veľké množstvo stromov presne podľa požiadaviek, ktoré budú vyzerať rozlične a dodávať tak hodnovernosť celej scéne.

Práca sa zaoberá predstavením programu Blender a metódami používanými na generovanie vegetácie. Následne je prezentovný implementovaný generátor a ukážky výstupov.

2 O programe Blender

Blender je 3D grafická open-source aplikácia slúžiaca na vytváranie modelov a animácií. Je distribuovaný pod licenciou GNU GPL. Využíva grafickú knižnicu OpenGL, vďaka čomu je prenositeľný na množstvo platforiem – FreeBSD, IRIX, GNU/Linux, Microsoft Windows, Mac OS X, Solaris, a iné.

Podporuje vytváranie doplnkov v jazyku Python prostredníctvom aplikačného programovacieho rozhrania (API). Keďže samotný program je prenositeľný na rôznych platformách, nie je ani veľký problém s prenositeľnosťou skriptov vytvorených pre Blender. Pomocou API Blenderu bol implementovaný plugin pre generáciu stromov, ktorý je náplňou tejto bakalárskej práce. API má prístup k širokému spektru objektov a metód a ponúka množstvo užitočných funkcií. Na internete sa nachádza rozsiahla dokumentácia [6], bez ktorej sa nezaobíde žiaden programátor používajúci API Blenderu.

Aj keď interpretované jazyky, akým je napríklad Python, sú znateľne pomalšie než jazyky kompilované, výkon je pre účely generovania objektov dostačujúci a prednosti sú prevažujúce nad nedostatkami. V prípade, že rýchlosť skriptu nie je uspokojivá, je možné niektoré výkonovo náročnejšie časti kódu vytvoriť v jazyku C++ a ten pomocou istých mechanizmov zahrnúť do skriptu.

Blender obsahuje niekoľko typov objektov, napríklad krivky, plochy, meshe, meta objekty, osvetlenia, kamery, a iné. Medzi najpoužívanejšie sa radí objekt typu mesh. Je to polygónová sieť, ktorá sa skladá z vrcholov (vertexov), ktoré sú pospájané hranami (edges) a môžu tvoriť plochy (faces). Využívajú sa najmä na definovanie tvaru mnohostenov.

Nemenej používaným typom objektu sú krivky a plochy (iné plochy ako tvoria vrcholy meshu). Na rozdiel od meshov nie sú tvorené vrcholmi, ale sú vyjadrené matematickými funkciami. Práca s nimi sa odlišuje od práce s meshmi. Sú určené riadiacimi bodmi, čo prináša iné možnosti modelovania. Krivky a plochy sa jednoducho dajú previesť na objekt typu mesh. V skripte boli použité práve objekty typu mesh a krivky.

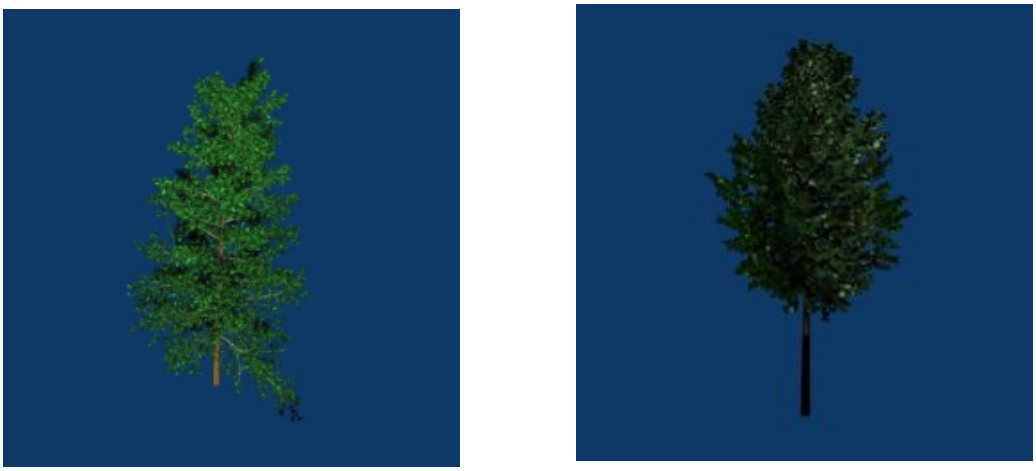
Blender používa vlastné jednotky, ktoré sú interpretované užívateľom podľa uváženia. To znamená že 1 jednotka v Blenderi sa dá chápať ako 1 centimeter aj ako 1 meter. Je vhodné, pokiaľ skripty túto relatívnosť zachovávajú, aby nebol užívateľ skriptu obmedzený pri ich použití v už existujúcich scénach. Ak by v scéne užívateľa boli objekty, ktoré by interpretovali jednotky Blenderu inak ako skript, musel by ich užívateľ transformovať na potrebnú veľkosť.

2.1 Existujúce pluginy

Pluginov vytvorených pre program Blender je veľmi veľké množstvo. Niekoľko z nich, a nie je ich málo, sa venuje generovaniu stromov. V texte budú uvedené dva z nich, ktorých generátory pracujú odlišnými metódami.
Jeden z kvalitných generátorov stromov a rastlín pre Blender ponúka skript zvaný L-System [8]. Pre generáciu využíva metódu L-systém, ktorá je detailnejšie osvetlená nižšie. Umožňuje vytvoriť mnoho druhov stromov vďaka väčšiemu množstvu nastavení generácie. Zoznámenie sa s ich významom nie je triviálnou záležitosťou, avšak po ich pochopení skript umožňuje vygenerovať realistické stromy aj so špecifickými prvkami konkrétnych druhov. Príklad vygenerovaných stromov je na obrázku 1.

Obrázok 1: Stromy vygenerované pluginom L-System [8]

Ďalším z pluginov, ktorý umožňuje generovať stromy je skript s názvom Gen3 [9]. Generátor využíva metódu zvanú parametrické generovanie. Tá je založená na modeli generovania popísanom v práci J.Webera a J. Penna [2]. Obsahuje až okolo 80 parametrov, čo umožňuje vytvoriť stromy presne podľa predstáv. Keďže pochopenie ich významu je bez znalostí metódy Webera a Penna náročnou úlohou, obsahuje aj niekoľko prednastavení, ktoré generujú niektoré konkrétne druhy stromov. Príklady vygenerovaných stromov sú na obrázku 2.



Obrázok 2: Stromy vygenerované pluginom Gen3

3 Metódy generovania stromov

Pri generovaní vegetácie ako súčasti scény v 3D priestore vytvárajú komplexný dojem vlastnosti, ktoré dotvárajú objekt ako celok. Detaily ako napríklad podrobná stavba kôry alebo uchytenie listov nehrajú takú dôležitú úlohu v porovnaní s rozmiestnením a tvarom vetiev, ktoré vytvárajú korunu stromu. Preto je vhodné, pokiaľ môžme ovplyvniť najmä významné parametre.

Pomocou parametrov metódy generovania by sme mali byť schopní vytvoriť rozličné druhy stromov a kríkov, listnatých aj ihličnatých. Počet parametrov v konečnom programe by však nemal byť príliš veľký a ich význam príliš špecifický, pretože tým sa generátor stáva zložitým pre užívateľa, ktorý tým môže byť odradený.

Metódy by mali pracovať aj s náhodnou veličinou, ktorá dodáva každému vygenerovanému stromu alebo rastline jedinečnosť. Súbor stromov, ktoré vyzerajú úplne rovnako by na vierohodnosti scény nepridával. Rozsahom vplyvu náhodnej veličiny na generovanie sa potom dá jednoducho určiť miera náhodnosti procesu generácie.

Čo sa týka výpočtovej náročnosti metód, rýchlosť nie je prioritnou vlastnosťou generátorov. Vytvorenie rastliny je jednorázový úkon a rýchlosť generácie existujúcich programov sa pohybuje približne v rozsahu niekoľkých sekúnd.

Neskoršie kapitoly sa venujú najpoužívanejším metódam generovania vegetácie. Prvou sú L-systémy, známe aj ako Lindenmeyerové systémy, ktoré využívajú formálne gramatiky. Inou technikou je parametrizované generovanie rastlín.

3.1 Vývoj metód

Metódy generovania stromov sa začali vyvíjať už pred mnohými desaťročiami. Prirodzene sa postupne dospelo k viacerým metódam, menej či viac úspešným [2].

Honda[3] predstavil v roku 1971 jeden z prvých modelov generácie pomocou parametrov. V diele jasne opísal rozdiel medzi monopodickým a dichotomickým vetvením. Monopodické vetvenie je spôsob, pri ktorom jedna vetva pokračuje v raste v pôvodnom smere a druhá vetva sa oddeľuje iným smerom. V dichotomickom vetvení sa obe vetvy odkláňajú od pôvodneho smeru rastu.

S úplne iným prístupom, ktorý dostal názov L-systémy, prišiel Aristid Lindenmeyer[1]. Využil pri tom formálne gramatiky generujúce reťazce, ktoré sa dajú interpretovať ako stromy a rastliny. Neskôr metódu spolu s Prusinkiewiczom rozšíril o kotextové gramatiky, náhodné veličiny a iné. Metóda bude detailnejšie opísaná v následujúcej kapitole.

Oppenheimer[4] vzal v úvahu princípy fraktálov pre generovanie stromov. Využil parametre ako uhol vetvenia, pomer veľkostí rodičovských a dcérskych vetiev, veľkosť úkosu kmeňa a podobne. Po vzore fraktálov používal rovnaké parametre pre každú úroveň vetiev.

Reeves a Blau[5] použili pri modelovaní časticové systémy. Zamerali sa hlavne na celkový vzhľad a prostredie lesov, kríkov a tráv, ale tiež na dostatočne detailné vyobrazenie jednotlivých stromov. Taktiež kládli väčší dôraz na vizuálne výsledky, než na presné zachovanie botanických detailov jednotlivých druhov.

De Reffye a spol.[6] dosiahli pôsobivé výsledky pri dodržaní presných botanických dát a princípov. Ich model simuluje rast vegetácie do určitého veku s využitím pravdepodobností smrti, zastavenia rastu, vetvenia a pod. Nevýhodou je nutná znalosť dosť podrobných botanických poznatkov a taktiež ich modelu generovania.

Weber a Penn[2] sa vo svojej metóde opierajú o podobné princípy ako Reeves a Blau. Pri generovaní využívajú niekoľko desiatok parametrov ktorými možno opísať strom. Na detaily, ktoré je možné si všimnúť iba z blízka, nekladú veľký dôraz. Metóde bude venovaná samostatná kapitola.

3.2 L-systémy

L-systémy, známe tiež pod pojmom Lindenmeyerove systémy, navrhol v roku 1968 maďarký biológ Aristid Lindenmeyer. Boli vytvorené za účelom modelovania vývoja primitívnych viacbunkových organizmov. Zjednodušene možno L-systémy nazvať gramatikou založenou na prepisovaní celého vstupného reťazca naraz v jednom derivačnom kroku podľa istých pravidiel. Rekurzívnym prepisom reťazca dojdeme k výslednému slovu, ktoré sa dá reprezentovať graficky so zaujimavými výsledkami. Existuje niekoľko druhov L-systémov. Bližšie budú opísané v následujúcich kapitolách.

Slová generované L-systémami sú výsledkom rekurzívneho aplikovania pravidiel. Preto majú tieto slová vlastnosť sebepodobnosti, ktorá je typická nielen pre fraktály, ale do istej miery aj pre štruktúry nachádzajúce sa v prírode. To umožňuje pri použití správnej gramatiky nájsť predpis reprezentujúci konkrétne organizmy v prírode. L-systémy sa úspešne využívajú aj pre jednoduché simulácie rastu rastlín a stromov.

Formálne sa L-systémy dajú definovať ako usporiadaná trojica $G = < V, \omega, P>$, kde V je abeceda systému, $\omega \in V^+$ je počiatočný symbol (axióm) - slovo z množiny neprázdnych slov V^+ nad abecedou V a $P \subset V^* \times V$ je konečná množina pravidiel, kde V^* je množina všetkých slov nad abecedou V. Pravidlo $(a, \chi) \in P$ sa zapisuje ako $a \rightarrow \chi$. Pri aplikácii pravidla sa symbol a prepíše na symbol χ. Predpokladá sa, že pre každé $a \in V$ existuje aspoň jedno slovo $\chi \in V^*$ také, že $a \rightarrow \chi$. Pokiaľ pre niektorý symbol neexistuje explicitné pravidlo, uvažujeme implicitné pravidlo $a \rightarrow a$ [1].

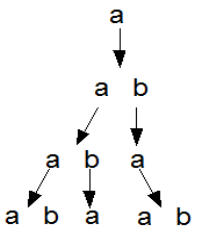
3.2.1 Reprezentácia L-systémov

Výsledkom L-systémov sú reťazce, ktoré sa dajú interpretovať rôznymi spôsobmi. Najčastejšie sa chápu ako postupnosť príkazov pre tzv. „korytnačku“. Korytnačka je spôsob vykresľovania v programovacom jazyku LOGO a môže sa pohybovať v 2D alebo 3D priestore. Obvykle má 3 vlastnosti: pozícia, orientácia a stav pera (farba pera, hrúbka, pero zapnuté/vypnuté a pod.). Ovláda sa príkazmi, ktoré určujú jej pohyb a otočenie. Pri použití iterácií v LOGu korytnačka začne vytvárať zaujímavé obrazce, pripomínajúce fraktály.

3.2.2 Typy L-systémov

Najjednoduchšou triedou L-systémov sú **DOL-systémy**. Sú bezkontextové, takže pri prepise znaku nie je podstatné, aké znaky sa nachádzajú pred alebo za prepisovaným znakom. Majú tiež vlastnosť determinizmu, z čoho plynie, že pre každý symbol existuje práve jedno pravidlo prepisu na iný symbol.

Príklad takého systému môže byť jednoduchá gramatika G = <V, ω, P>, kde V = {a, b}, ω = {a}, P = {a→ab, b→a}. Po troch iteráciách dostaneme zo vstupného slova a reťazec $abaab$ [Obrázok 3].



Obrázok 3: Jednoduchý príklad DOL-systému.

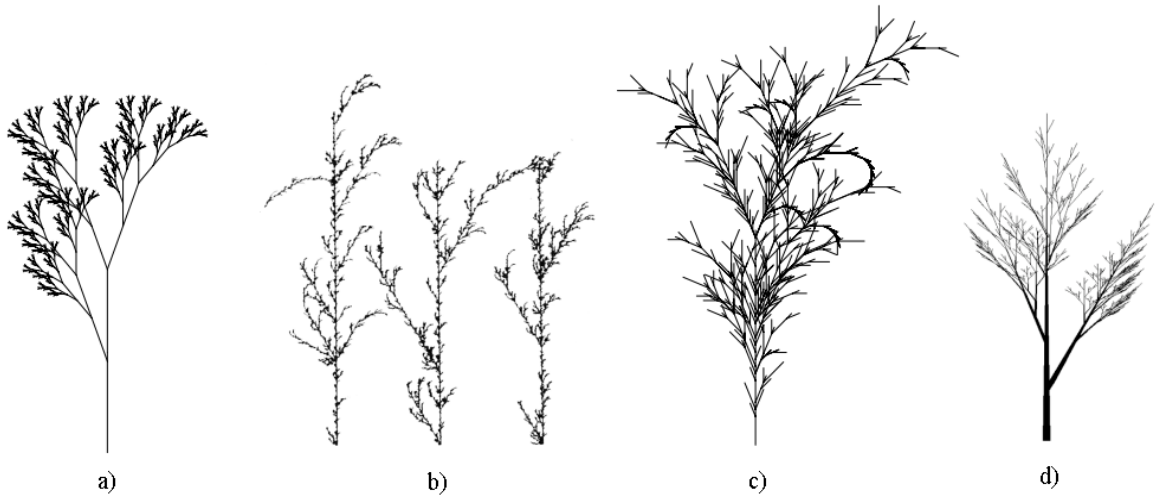
Ďalšou z tried sú **kontextové** L-systémy. Ich použitie záleží na kontexte, v ktorom sa nachádza prepisovaný symbol. Sú popísané gramatikami, v ktorých je na ľavej strane pravidla 1 alebo viac symbolov. Podľa tvaru pravidiel sa delia na 1L-systémy a 2L-systémy. 1L-systémy obsahujú pravidlá v tvare $a_l < a \rightarrow \chi$ alebo $a > a_r \rightarrow \chi$ a prepisovaným symbolom je písmeno a. Pri 2L-systémoch záleží na symboloch na obidvoch stranách prepisovaného písmena, takže pravidlá majú tvar $a_l < a > a_r \rightarrow \chi$. Kontextové L-systémy sú vhodné napríklad pre simuláciu propagácie prvkov v štruktúrach.

Dôležitým rozšírením sú **zátvorkové** L-systémy. Obsahujú naviac symbol [a symbol]. Ich význam je následujúci - [značí vloženie aktuálneho stavu na zásobník a] značí výber stavu z vrcholu zásobníka a použitie ako aktuálneho stavu. Bez použitia zátvoriek dovoľujú L-systémy vykresľovať iba lineárnu čiaru. K vetveniu je nutné využiť zátvorky; tie umožnujú vrátit sa k predchádzajúcemu stavu a vykresliť tak objekty so stromovou štruktúrou.

Pre generovanie realistických stromov a rastlín je kľúčová istá miera náhodnosti. Doteraz predstavené triedy L-systémov boli deterministické. To znamená, že stromy vygenerované pomocou jedného L-systému sú identické. **Stochastické** L-systémy prinášajú do procesu generovania náhodnú veličinu, ktorá zachováva typické rysy vygenerovaného slova, avšak generované slová sa líšia v detailoch. Dajú sa charakterizovať ako štvorica $G = <V, \omega, P, \pi>$. Gramatika zavádza oproti deterministickým L-systémom navyše parameter π nazývaný pravdepodobnostné rozloženie. π môže nadobúdať hodnôt $(0, 1>$ pre každé pravidlo P. Táto hodnota π pravidla P značí pravdepodobnosť, že bude požité. Platí, že pre každý symbol $a \in V$ je súčet pravdepodobností pravidiel, pri ktorých je prepisovaným symbolom a rovný 1.

Vykresľovacie možnosti korytnačky sú značne obmedzené, pokiaľ neexistuje spôsob ovládania veľkosti jej kroku alebo otočenia. **Parametrické** L-systémy prinášajú nové možnosti tým, že pracujú so symbolmi, ktoré obsahujú parametre. Gramatika má tvar $G = <V, \Sigma, \omega, P>$. Zavádza naviac Σ – zoznam formálnych parametrov. Parametrické systémy naviac rozširujú pravidlá o podmienkovú časť. Príklad pravidla parametrického L-systému s podmienkou môže byť následujúci:
$A(t) : t > 5 \rightarrow B(t + 1)CD(t \wedge 0.5, t - 2)$. Pravidlo sa aplikuje, iba pokiaľ parameter t písmena A je väčší ako 5.

L-systémy sú vhodným prostriedkom pre generovanie mnohých organizmov a simuláciu procesov vývinu (napríklad simulácia rastu rastliny v závislosti na príjme vody). Lepšie výsledky sa dosahujú v oblasti generovania bylín. Stromy sú v dôsledku ich nepravidelnosti ťažšie realisticky generovateľné. Príklady vygenerovaných rastlín zobrazuje Obrázok 4.



Obrázok 4: Príklady L-systémov: a)zátvorkový b)stochastický c)kontextový d)paramterický [1]

3.3 Parametrizované generovanie

Parametrizované generovanie je spolu s L-systémami jednou z najviac používaných metód generovania stromov. V porovnaní s L-systémami je však jeho použitie značne jednoduchšie a intuitívnejšie. Stromy a rastliny sú generované na základe nastavenia niekoľkých parametrov, na ktorých závisí ich výsledný vzhľad. Ich zmenou je možné dosiahnuť podobu rôznych druhov stromov a rastlín v priebehu krátkeho časového intervalu.

Rozborom v tejto oblasti sa zaoberali vo svojej významnej práci Jason Weber a Joseph Penn [2]. Rozbor bol použitý ako základ generátora, ktorý je náplňou tejto bakalárskej práce, preto bude podrobnejšie opísaný v nasledujúcich kapitolách.

3.3.1 Koncepcia stromu

Koncepcia stromu Webera a Penna je založená na tom, že sa skladá z hlavného, rôzne zakriveného kmeňa, ktorý má tvar kužeľovitého valca. Ten sa môže rozdeľovať na viaceré, rovnako významné časti, ktoré sa môžu následne tiež rozdeľovať. Všetky časti kmeňa majú rovnaké vlastnosti a parametre. To predstavuje dichotomické delenie kmeňa. Vetvy stromu sú rozdelené na úrovne, kmeň má číslo 0, vetvy vyrastajúce z kmeňa úroveň 1, atď.

Časti kmeňa sa okrem dichotomického delenia delia aj monopodicky. V tomto prípade sa od kmeňa odkláňa dcérska vetva, ktorá nemusí mať rovnaké vlastnosti ako materská vetva. Avšak parametre, ako napríklad dĺžka alebo priemer, sú často ovplyvňované parametrami materskej vetvy. Tieto dcérske vetvy sa ďalej monopodicky aj dichotomicky delia. Pre obvyklé potreby stačia obyčajne tri až štyri úrovne vetiev. Modelovanie prebieha postupne, najskôr sa vygeneruje prvá úroveň, ak vyhovuje našim predstavám, pokračuje sa ďalšou úrovňou.

Tvar stromov sa líši v závislosti od mnohých atribútov, ako je druh, vek, prostredie a podobne. Jedným zo spôsobov, ako vytvoriť prirodzený tvar stromu, je precízne zachovať všetky vlastnosti. Avšak jednoduchším a často lepším spôsobom je definovať tvar stromu explicitne. Weber a Penn sa rozhodli pre využitie tzv. neviditelnej obálky. Tá sa nachádza okolo stromu a má definované určité rozmery, ktoré vetvy nemôžu presiahnuť. Pokiaľ sa tak stane, vetvy sú skrátené tak, aby nepresahovali obálku.

Metóda využíva množstvo parametrov, z ktorých sa značný počet opakuje pre všetky úrovne vetiev stromu. Preto sa pred generalizovanými parametrami používa prefix **n**. Napríklad pre všeobecný parameter značiaci úkos všetkých úrovní je to **nTaper**, avšak pre konkrétnu 2. úroveň má parameter názov **2Taper**. Parametre môžu mať aj suffix **V**, značiaci maximálnu hodnotu parametra, v ktorom je použitý.

3.3.2 Špecifikácia vetiev

Model je založený na dvoch základných elementoch, vetve a liste. Vetvou sa myslí vetva na akejkoľvek úrovni, vrátane kmeňa. Vetva má tvar kužeľovitého valca. Každá vetva má vlastný súradnicový systém, pričom os z je spoločná s centrálnou osou valca tvoriaceho vetvu. Pre vetvy vyrastajúce z kmeňa smeruje os z kolmo ku kmeňu, os y smerom k oblohe a os x je rovnobežná so zemou.

Vetva je rozdelená na **n** segmentov podľa parametru **nCurveRes**. Segmenty majú približný tvar kruhu, ktoré sú spojené sieťou trojuholníkov. Vetva sa môže pozdĺžne rozdeľovať na viacero vetiev, oddeľujúce sa vetvy v tom prípade dedia všetky vlastnosti originálnej vetvy. Frekvenciu delenia definuje parameter **nSegSplits**. Parameter má obvykle hodnotu medzi 0,0 a 1,0. 1 značí dichitomické delenie na každom segmente. **nSegSplits** rovné 2 znamená delenie na 3 rovnocenné vetvy na každom segmente. Dodatočný parameter je **nBaseSplits**, ktorý má význam ako **nSegSplits**, ale je uplatňovaný len na konci prvého segmentu kmeňa. Umožňuje to generovať stromy, ktoré sa

majú tendenciu deliť sa iba na začiatku kmeňa. Každá vetva, ktorá už je dichotomicky rozdelená má zníženú pravdepodobnosť ďalšieho delenia na polovicu.

Odklon segmentov je regulovaný parametrom **nCurveBack**. Ak je rovný 0, os z každého segmentu je odklonená od osi z predchádzjúceho segmentu o **(nCurve/nCurveRes)** stupňov okolo osi x. Ak je **nCurveBack** nenulový, prvá polovica segmentov vetvy je odklonená o (**nCurve/ (nCurve/2))** stupňov a druhá polovica segmentov vetvy o **(nCurveBack/(nCurve/2))**. To dovoľuje formovať vetvy v tvare písmena S. Každý segment je naviac vždy odklonený o **(nCurveV/nCurveRes)**. Špeciálnym prípadom je, keď je parameter **nCurveV** rovný 0. Vtedy je vetva formovaná v tvare špirály s maximálnym uhlom odklonu **nCurveVary**.

Maximálny počet dcérskych vetiev vetvy jednej úrovne je definovaný parametrom **nBranches**. Skutočný počet vetvy prvej úrovne je výsledok vzorca (1):

$$stems = stems_{max} * (0.2 + \frac{0.8 * (length_{child} / length_{parent})}{length_{child,max}}) \quad (1)$$

Skutočný počet vetvy ďalších úrovní určuje vzorec (2):

$$stems = stems_{max} * (1.0 - \frac{0.5 * offset_{child}}{length_{parent}}) \quad (2)$$

kde $offset_{child}$ je pozícia v metroch od začiatku materskej vetvy. Tvar vetvy určuje funkcia **ShapeRatio(shape, ratio),** ktorú je možné nájsť v práci Webera a Penna. **Shape** udáva index v tabuľke možných prierezov, **ratio** je parameter od 0 do 1 určujúci pozíciu dcérskej vetvy na materskej od jej začiatku. Maximálna dĺžka dcérskej vetvy je súčinom skutočnej dĺžky materskej vetvy a parametru $\mathbf{length_{child,max}}$. Detailný vzorec je možné nájsť v [2].

Odklon vetiev od materskej vetvy určuje súčet parametrov **nDownAngle** a **nDownAngleV**. Umiestnenie vetiev je špirálovité, uhol medzi jednotlivými dcérskymi vetvami je **(nRotate+nRotateV)**. Pri zápornej hodnote **nRotate** sú vetvy odklonené naviac o uhol 180° vzhľadom na os y materskej vetvy. To dovoľuje umiestnenie vetiev takmer v jednej rovine. Parameter **AttractionUp** ovplyňuje tendenciu vetiev rastu smerom hore k slnku, podľa vzoru prirodzenej vlastnosti rastlín.

Priemer vetvy nemôže byť nikdy väčší, než je priemer miesta, odkiaľ vetva vyrastá. Úkos vetiev je definovaný parametrom **nTaper**, podľa tabuľky úkosu, ktorý môže mať hodnoty od 0 do 3.

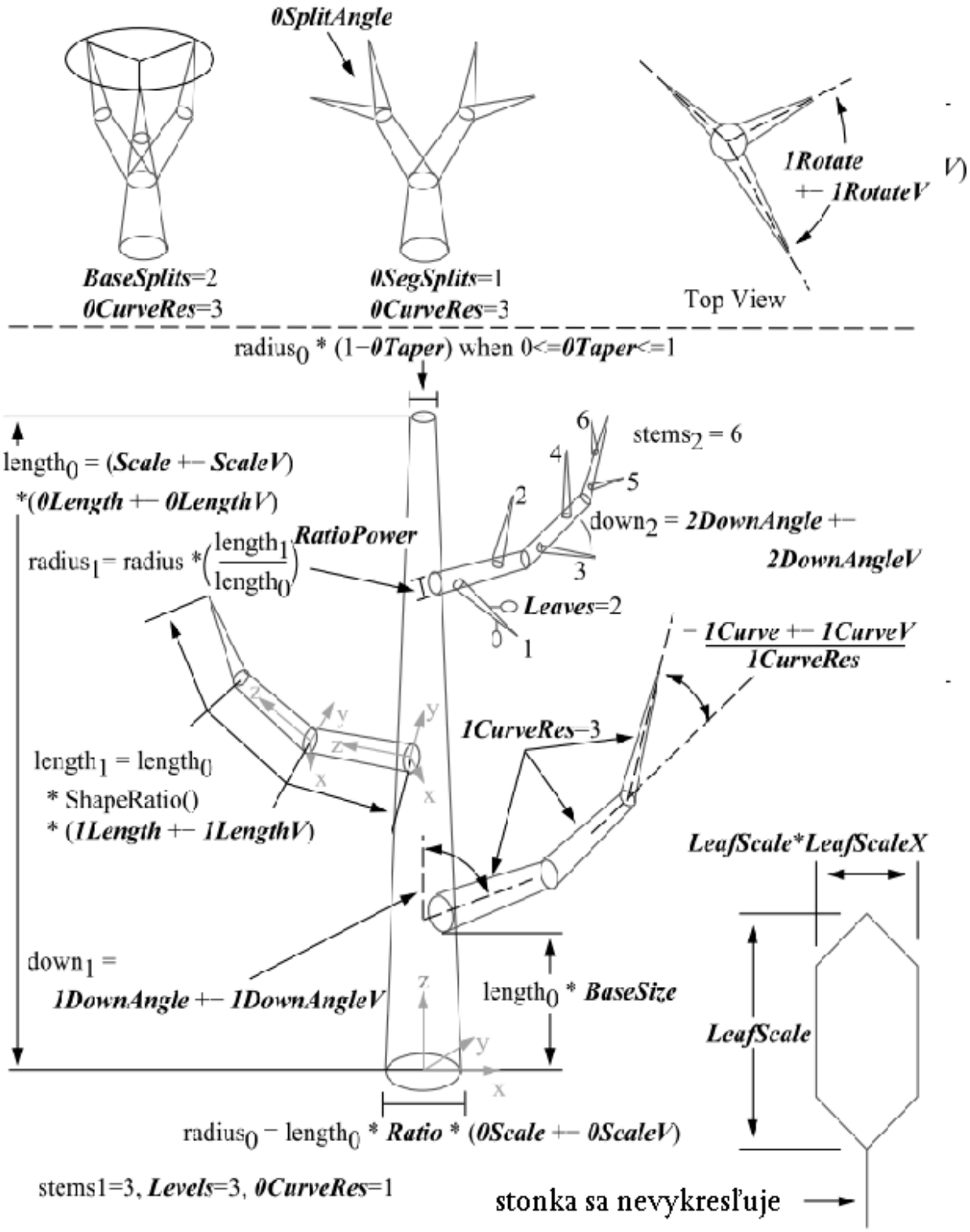
Pre názornejšie vyobrazenie parametrov slúži [Obrázok 5].

3.3.3 Špecifikácia listov

Poslednou úrovňou z obvykle 3 až 4 úrovní sú listy. Listy sa generujú ak parameter **Leaves** nie je nulový. Listy sú ovplyvnené parametrami **nDownAngle, nDownAngleV, nRotate a nRotateV** z tej úrovne, v ktorej sa nachádzajú. Všetky vetvy a listy, ktoré sú na úrovni 3 a viac využívajú parametre tretej úrovne. Parameter **Leaves** podobne ako **nBranches** určuje hustotu listov. Počet listov je daný súčinom parametrov **Leaves**, výsledkom funkcie **ShapeRatio** a **quality**. **Quality** zadáva užívateľ, obvykle je približne 1. Listy sú umiestňované prednostne na okraji stromu tak, ako je to aj v prírode. Špeciálne umiestnenie listov v tvare vejára na konci vetvy sa aplikuje pri zápornej hodnote **Leaves**.

Tvar listov je preddefinovaný v tabuľke listov. **LeafShape** je indexom v tejto tabuľke. Možné tvarmi listov sú napríklad oválny, trojuholníkový, 3-cípy a iné. Šírku a dĺžku listov udáva parameter **LeafScale** a v osi x **LeafScaleX**. Ich orientácia je v preddefinovanom nastavení náhodná.

Na obrázku sú názorne zobrazené parametre listu [Obrázok 5].



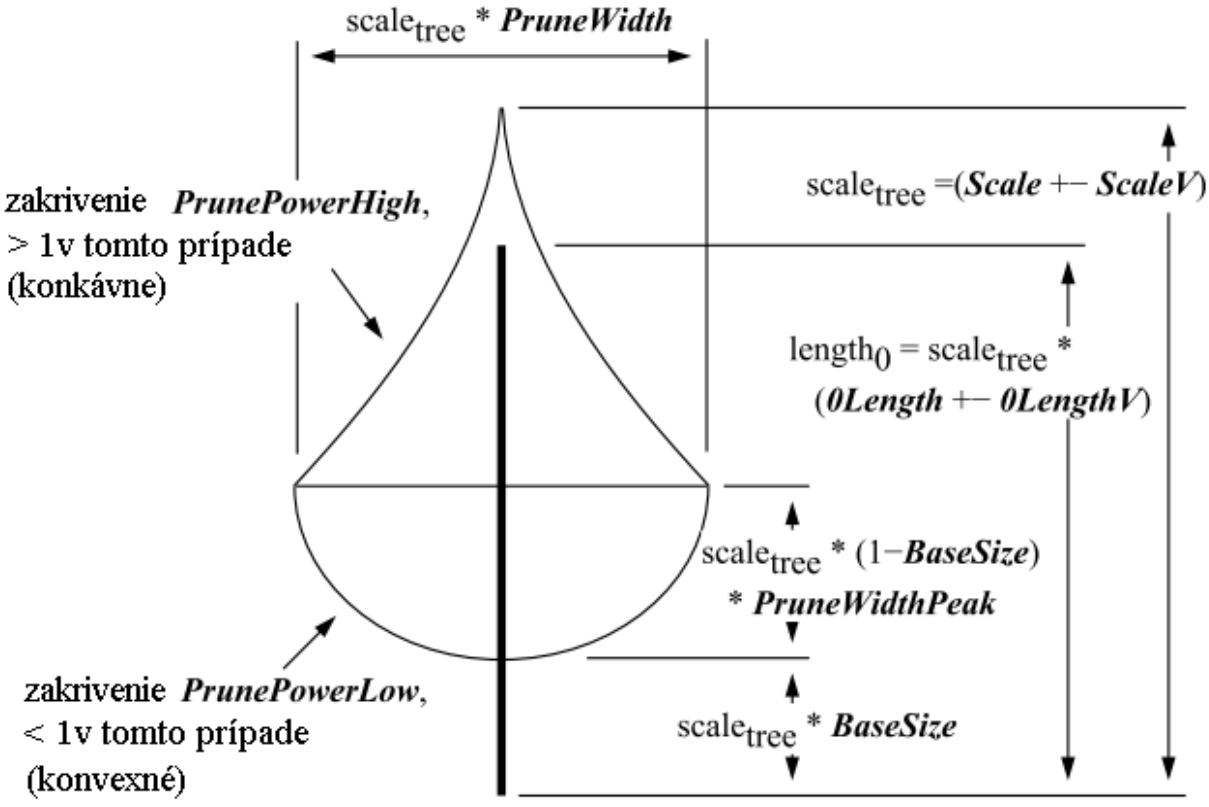
Obrázok 5: Zobrazenie parametrov vetiev a listu [2]

3.3.4 Špecifikácia obálky

Obálka stromu slúži na definíciu tvaru stromu. Táto metóda je veľmi nápomocná pokiaľ potrebujeme vygenerovať strom špecifického tvaru. Generovaná a skrátená vetva musí prepočítať všetky svoje parametre skôr, než sa z nej vygenerujú nové vetvy, keďže parametre sú navzájom závislé. Pokiaľ vetva presahuje obálku, je iteratívne skracovná, až kým sa nenachádza iba vo vnútri obálky.

Tvar obálky je pseudo-elipsovitý, definovaný niekoľkými parametrami. Maximálnu šírku obálky definuje parameter (**PruneWidth*scale$_{tree}$**). Umiestnenie na osi z kmeňa definuje **PruneWidthPeak**. Obálka sa delí na 2 časti, ktorých zakrivenie je definované parametrami **PrunePowerHigh** a **PrunePowerLow**. Tvar častí môže byť konvexný aj konkávny. Mieru orezávania vetiev definuje **PruneRatio**. Hodnota 1 nastavuje orezávanie všetkých vetiev v mieste okraja obálky, 0 potláča orezávanie. Vhodným nastavením sa dá vyhnúť geometricky presnému tvaru stromu, ktorý tak pôsobí neprirodzene.

K lepšiemu porozumeniu parametrov poslúži [Obrázok 6].



Obrázok 6: Názorné zobrazenie parametrov obálky [2]

4 Návrh riešenia

V tejto časti bude objasnený môj algoritmus na generovanie stromov v programe Blender. Bola použitá metóda parametrizovaného generovania vegetácie. Program bol napísaný v jazyku Python s využitím aplikačného programovacieho rozhrania Blenderu. Použitie pluginu predpokladá aspoň základné znalosti Blenderu. Ako už bolo spomenuté v predchádzajúcej kapitole, plugin je teoreticky prenositeľný na všetky platformy ktoré sú podporované Blenderom. Bol testovaný na operačných systémoch Windows a Debian GNU/Linux. Plugin bol vytvorený pre aktuálnu verziu Blenderu 2.48a (10.5.2009) s použitím jazyka Python vo verzii 2.5.4.

Oblasť generácie stromov v Blenderi je pomerne rozšírená. Existuje mnoho nástrojov pre automatickú generáciu vegetácie, ktoré umožňujú vytvorenie dostatočne verných modelov konkrétnych druhov stromov. Ich použitie však nie je obvykle príliš triviálne a zvyčajne obsahuje mnoho parametrov, ktorých význam sa bežnému užívateľovi nemusí zdať na prvý pohľad zjavný. Niektoré z nich predpokladajú základné znalosti z oblasti generácie stromov. Môj program si kladie za úlohu čo najviac zjednodušiť prácu potenciálnemu užívateľovi a umožniť jednoducho generovať stromy v čo najkratšom čase a bez špecifických vedomostí. Program nekladie dôraz na možnosť modelovania konkrétnych druhov stromov, ani ich niektorých špecifických vlastností. V nasledujúcich kapitolách bude predstavený postup pri tvorbe jednotlivých častí generátora a užívateľské rozhranie.

4.1 Základný rozbor

Za vzor bola vzatá metóda Webera a Penna [2]. Strom vždy pozostáva zo 4 úrovní, vrátane kmeňa a listov. Nultou úrovňou je kmeň, treťou sú listy. Tento počet bol zvolený vzhľadom na dostačujúce rozlíšenie aj pri bližšom pohľade na strom. Vyšší počet by nebol o veľa prínosnejší a výpočtové a pamäťové nároky programu by prudko vzrástli pri každej ďalšej úrovni. Program vyžaduje na vstupe niekoľko parametrov generácie, pomocou ktorých je ovplyvnené vytváranie vetiev a listov. Množstvo parametrov je minimalizované na takú úroveň, aby bolo možné ovplyvniť hlavné rysy vetiev a lisov. Ich význam je na dostatočne abstraktnej úrovni a je zrejmý na prvý pohľad.

Hlavná myšlienka programu je dať možnosť užívateľovi vytvoriť jednoduchý strom požadovaného tvaru. Preto program žiada na vstupe okrem nastavených parametrov aj krivku reprezentujúcu tvar kmeňa stromu a mesh reprezentujúci obálku. Nie sú potrebné žiadne doplňujúce nastavenia a úpravy týchto objektov ako je napríklad ručné vytváranie profilu kmeňa a podobne. Účelom spomínaných objektov je určiť základný tvar generovaného stromu. Program využíva objekty typu beziérová krivka pre modelovanie vetiev a objekty typu mesh pre obálku ohraničujúcu strom a listy.

4.2 Generovanie kmeňa a vetiev

Generovanie vetiev má 3 fázy: vygenerovanie kmeňa z krivky určenej užívateľom, vygenerovanie prvej úrovne vetiev a vygenerovanie druhej úrovne vetiev. Modelovanie kmeňa a prvej úrovne vetiev je obsluhované spoločným tlačidlom užívateľského rozhrania (GUI), druhá úroveň má samostatné tlačidlo. Vetvy sú reprezentované beziérovými krivkami, čo dáva možnosť jednoduchej úpravy tvaru vetvy posunom riadiacich bodov krivky. Pri posune jedného riadiaceho bodu sa automaticky prepočítava celkový tvar krivky. Keby boli vetvy reprezentované sieťou polygónov, úprava ich tvaru by nebola triviálnou záležitosťou a aby vyzerali prirodzene by bolo nutné ručne posúvať veľké množstvo vrcholov polygónov.

4.2.1 Kmeň

Krivku reprezentujúcu kmeň musí vložiť do prostredia Blenderu užívateľ. Hrúbku kmeňa užívateľ nastavuje parametrom **trunkWidth**. Ten môže nadobúdať hodnoty od 0,0 do 1,0. Reálna hrúbka kmeňa je závislá od reálnej výšky kmeňa, a je výsledkom rovnice (3):

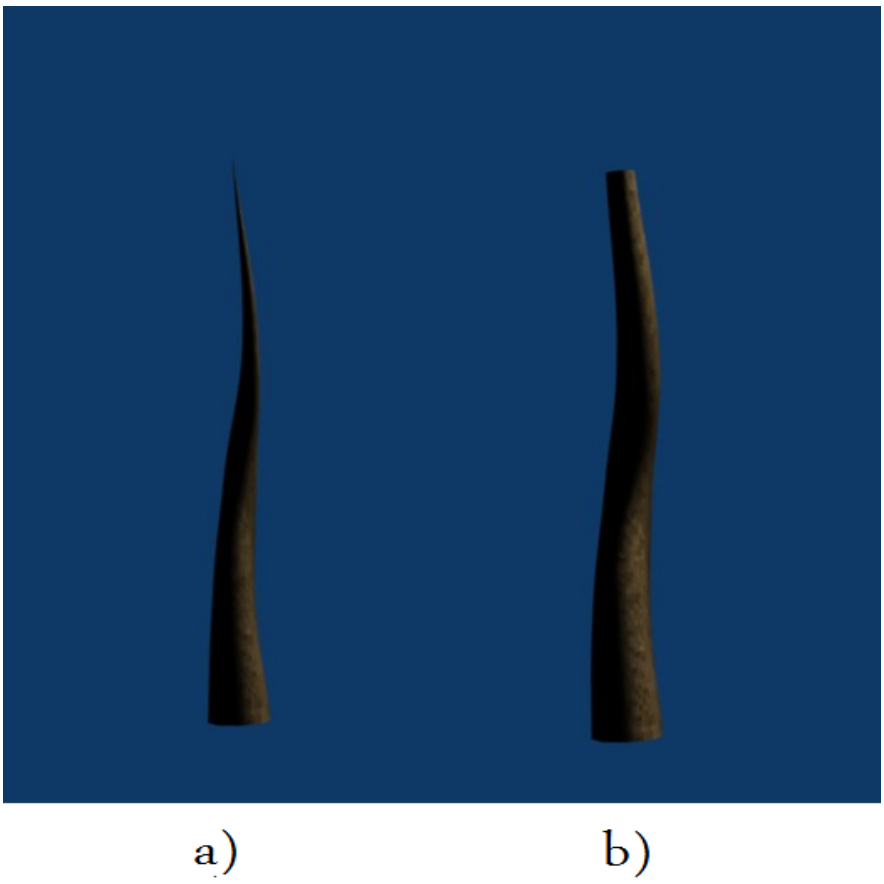
$$width = (\frac{trunkLength}{10}) * trunkWidth \quad (3)$$

kde width je reálna šírka a trunkLength je reálna výška kmeňa. Z rovnice vyplýva, že maximálna šírka kmeňa môže byť rovná desatine výšky. Z pozorovania reality bolo usúdené, že väčší priemer vzhľadom na výšku nie je potrebný.

V pokročilých voľbách GUI je možné nastaviť aj hrúbku koncového bodu parametrom **trunkEndWidth**, ktorého rozmedzie je v hodnotách 0,0 až 1,0. Jeho základné nastavenie je 0. Rozdiel v nastavení parametra je vidieť na obrázku 7.

Tvar kmeňa je kužeľovitý, úkos však nie je rovnomerný. Nazvime šírku v riadiacom bode pri rovnomernom úkose lineárna šírka. Reálna šírka je potom daná náhodne Gaussovým rozložením so stredom v lineárnej šírke a smerodajnou odchýlkou rovnou (0,1*lineárna šírka). Tým sa dosiahnú jemné variácie pri úkose generovaných kmeňov.

Tvar prierezu kmeňa je tvorený krivkou v tvare kružnice, ktorej rozlíšenie je dané parametrom **trunkRes**. Môže nadobúdať hodnoty od 1 do 5, základné nastavenie je 2. Pri nastavení 1 je tvar prierezu v tvare štvorca, pri zvyšovaní rozlíšenia sa tvar prierezu približuje tvaru kružnice. Zvyšovaním rozlíšenia prirodzene stúpajú aj výpočtové nároky pre prácu s objektom.



Obrázok 7: Znázornenie kmeňa s parametrom trunkEndWidth a) 0 b) 0,5

4.2.2 Vetvy

Vetvy prvej úrovne sú generované po vytvorení kmeňa. Program podporuje tvorbu iba monopodických vetiev. Vytváranie dichotomických vetiev by bolo s použitím beziérových kriviek náročnou úlohou, keďže beziérové krivky nie je možné pozdĺžne rozdvojiť. Vetvy sa skladajú zo segmentov. Ich počet je pre vetvy prvej úrovne závislý na veľkosti materskej vetvy, ako aj na priestore, ktorý majú k dispozícii pre svoj rast. Vetvy druhej úrovne sa skladajú z maximálne 4 segmentov. Pri vyššom počte vetiev by viac segmentov na vetvu mohlo byť pamäťovo náročné.

Princípy generácie vetiev prvej aj druhej úrovne sú veľmi podobné, preto je následný opis platný pre obe úrovne. Pokiaľ sa v niečom budú odlišovať, bude to explicitne uvedené.

Keďže jednotky v Blenderi sú relatívne, ich interpretácia závisí na užívateľovi. Veľkosť krivky 1 v jednotkách Blenderu môže byť chápaná ako 1 meter, ale aj 50 metrov. Pre zachovanie tejto relatívnosti nebolo možné určiť počet vetiev vzhľadom na veľkosť kmeňa. Preto je základný počet vetiev daný pevne a užívateľ ho môže ovplyvniť parametrom **branch1Density** pre prvú úroveň a **branch2Density** pre druhú úroveň. Reálny počet vetiev je daný vzorcom (4):

$$branchesAmount = 10 * branchDensity^2 \tag{4}$$

pre najdlhšiu materskú vetvu, a pre ostatné vetvy je počet vetiev zmenšený pomerom veľkosti k najdlhšej vetve.

Miesta, z ktorých vyrastajú vetvy sú rozložené po časti materskej vetvy, ktorá sa nachádza vo vnútri obálky. Odstup miesta, z ktorého vyrastá vetva od miesta rastu predchádzajúcej vetvy, je daný Gaussovým rozložením so stredom v priemernej vzdialenosti vetiev a smerodajnou odchýlkou (0,1*priemerná vzdialenosť). Ich rozloženie po obvode materskej vetvy je špirálovité s určitou odchýlkou. Pre každú vetvu sa generuje náhodný uhol v rozmedzí 0 – 90°. Vetva vyrastá vždy v nasledujúcom kvadrante ako predchádzajúca vetva a je otočená o vygenerovaný uhol.

Šírka vetvy nesmie byť väčšia, než šírka miesta odkiaľ vyrastá. Je ovplyvnená parametrom **branch1Width** a **branch2Width**. Parameter môže nadobúdať hodnôt 0,0 až 2,0. Reálna šírka vetvy sa odvíja od miesta odkiaľ vyrastá. Upravená je Gaussovým rozložením. Nazvime šírku miesta kde sa pripája vetva parentWidth. Potom výpočet reálnej šírky je daný vzorcom (5)

$$width = gauss\left(\frac{parrentWidth * branchWidth}{2}, \frac{0{,}1 * parrentWidth * branchWidth}{2}\right) \tag{5}$$

kde funkcia gauss je funkcia Gaussovho rozloženia pravdepodobnosti.

Pre úkos vetiev prvej aj druhej úrovne platia rovnaké princípy ako pre kmeň, avšak šírka koncového bodu je vždy rovná 0. Rovnako profil vetiev je tvorený krivkou v tvare kruhu a rozlíšenie je regulovateľné parametrom **branchRes** v rozmedzí 1 až 5. Ten je spoločný pre obe úrovne vetiev a jeho základná hodnota po spustení programu je 1 (najmenšie rozlíšenie). Bez detailnejšieho pohľadu je táto hodnota postačujúca. V prípade potreby je zvýšením hodnoty parametra možné zvýšiť rozlíšenie.

Dĺžka vetiev prvej úrovne nie je nijak upravovaná parametrami. K tomuto účelu je použitá obálka. Vetva, ktorá „vyrastie“ mimo obálku, je skrátená, až kým sa nenachádza vo vnútri obálky. Vetvy druhej úrovne sú tiež orezané pokiaľ presiahnu okraj obálky, avšak môžu byť aj kratšie. K tomu slúži parameter **branch2Len** s rozmedzím hodnôt od 0,0 do 2,0. Tento parameter priamo ovplyvňuje veľkosť segmentov druhej úrovne.

Uhol, pod ktorým vyrastá dcérska vetva od materskej vetvy je daný parametrom **branchAngle**. Môže nadobúdať hodnoty od -1,0 do 1,0. Pri hodnote 0 vyrastajú vetvy približne kolmo od vetvy materskej. Kladné hodnoty parametra znamenajú, že vetvy vyrastajú smerom hore a záporné hodnoty naopak značia rast smerom dole. Maximálny uhol, ktorý môže zvierať dcérska vetva s materskou, je pri hodnote parametra **branchAngle** 1 približne 72°. Uhol je naviac ovplyvnený Gaussovým

rozložením so stredom v danom uhle zodpovedajúcom nastaveniu parametra branchAngle a odchýlkou veľkosti 1/10 daného uhla.

Miera zakrivenia vetvy pri raste je daná parametrom **branchCurvature**. Ten môže nadobúdať hodnoty od 0,0 do 1,0. Pri hodnote 0,0 majú jednotlivé segmenty vetvy rovnakú orientáciu a vetvy nie sú zakrivené vôbec. Pri hodnote 1,0 môže dosahovať uhol medzi 2 segmentami vetvy až 45°. Uhol daný parametrom je naviac náhodne upravený Gaussovým rozložením, podobne ako u ostatných parametrov so stredom v danom uhle a odchýlkou veľkosti 1/10 daného uhla.

4.3 Obálka

Obálka slúži na explicitnú definíciu tvaru stromu. Je potrebné, aby ju užívateľ vytvoril a označil pred spustením generátora. Vetvy sú generované len na tej časti kmeňa, ktorá je vo vnútri obálky a nikdy nevyrastajú mimo nej. Obálka musí byť typu mesh. Jej tvar a rozmery môžu byť akékoľvek, podmienkou ale je, že musí byť konvexná. V prípade jej nekonvexnosti môže dôjsť k chybnému určeniu polohy vetvy vzhľadom na obálku. Zložitosť obálky však priamo vplýva na rýchlosť generácie, preto by mala byť čo najjednoduchšia. Pokiaľ sa krivka definujúca tvar kmeňa nachádza mimo obálky, nie sú generované žiadne vetvy. Funkciu obálky demonštruje [Obrázok 8].

Obrázok 8: Demonštrácia funkcie konvexnej obálky

4.4 Generovanie listov

Listy sú generované na vetvách prvej a druhej úrovne. Výskyt listov sa prirodzene zhusťuje smerom ku koncu vetiev. Ich množstvo ovplyvňuje parameter **leavesDensity** s hodnotou v rozmedzí 0,0 – 2,0. Listy sú na vetvu umiestňované nasledujúcim spôsobom. Pre každú vetvu je stanovená základná vzdialenosť step vzorcom

$$step = \frac{length/10}{leavesDensity} \tag{6}$$

kde length je dĺžka príslušnej vetvy. Následne sa na vetve umiestňujú listy vo vzdialenosti step. Po každom umiestnení listu je vzdialenosť step zmenšená koeficientom 19/20. Tak je dosiahnutá väčšia frekvencia výskytu listov v smere ku koncu vetvy. Aby nedošlo k zacykleniu, minimálna veľkosť parametru step je stanovená na length/30.

Veľkosť listu je nastaviteľná parametrom **leavesSize** v rozmedzí 0 - 2. Skutočná veľkosť listu je odvodená od polomeru kmeňa. Šírku listu určuje vzorec

$$leavesWidth = \frac{trunkRadius}{2} * leavesSize^2 \tag{7}$$

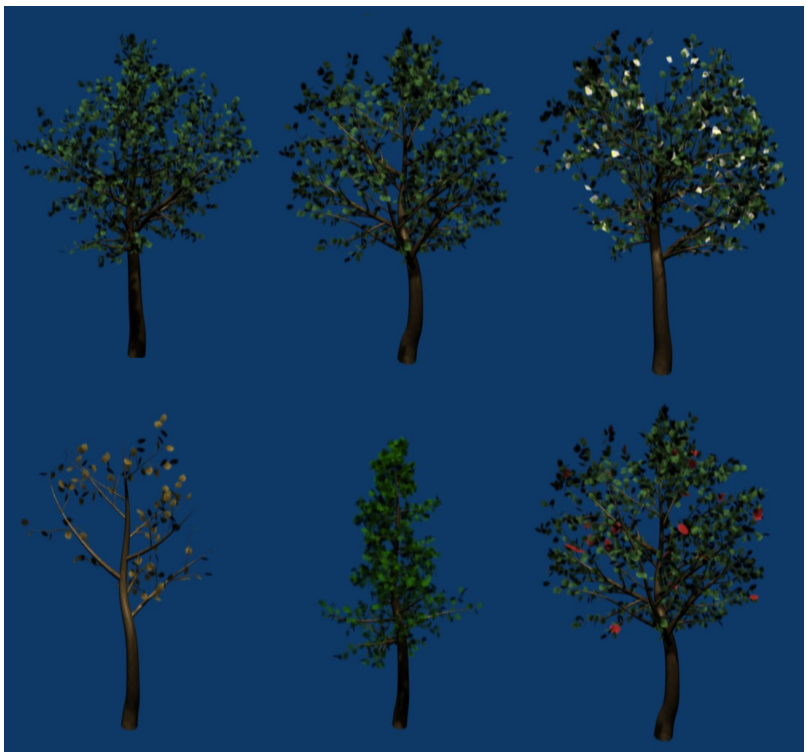
kde trunkRadius je skutočný polomer šírky kmeňa v jeho najširšej časti. Dĺžka listu je dopočítaná podľa pomeru strán obrázka, ktorý je použitý ako textúra listu.

Umiestnenie listu na obvode vetvy je náhodné. Rovnako náhodná je aj rotácia listu podľa osi z. Uhol medzi vetvou a listom je daný rovnomerným náhodným rozdelením v rozmedzí 18° - 72°, takže sa nemôže stať, že by sa list nachádzal vo vnútri vetvy.

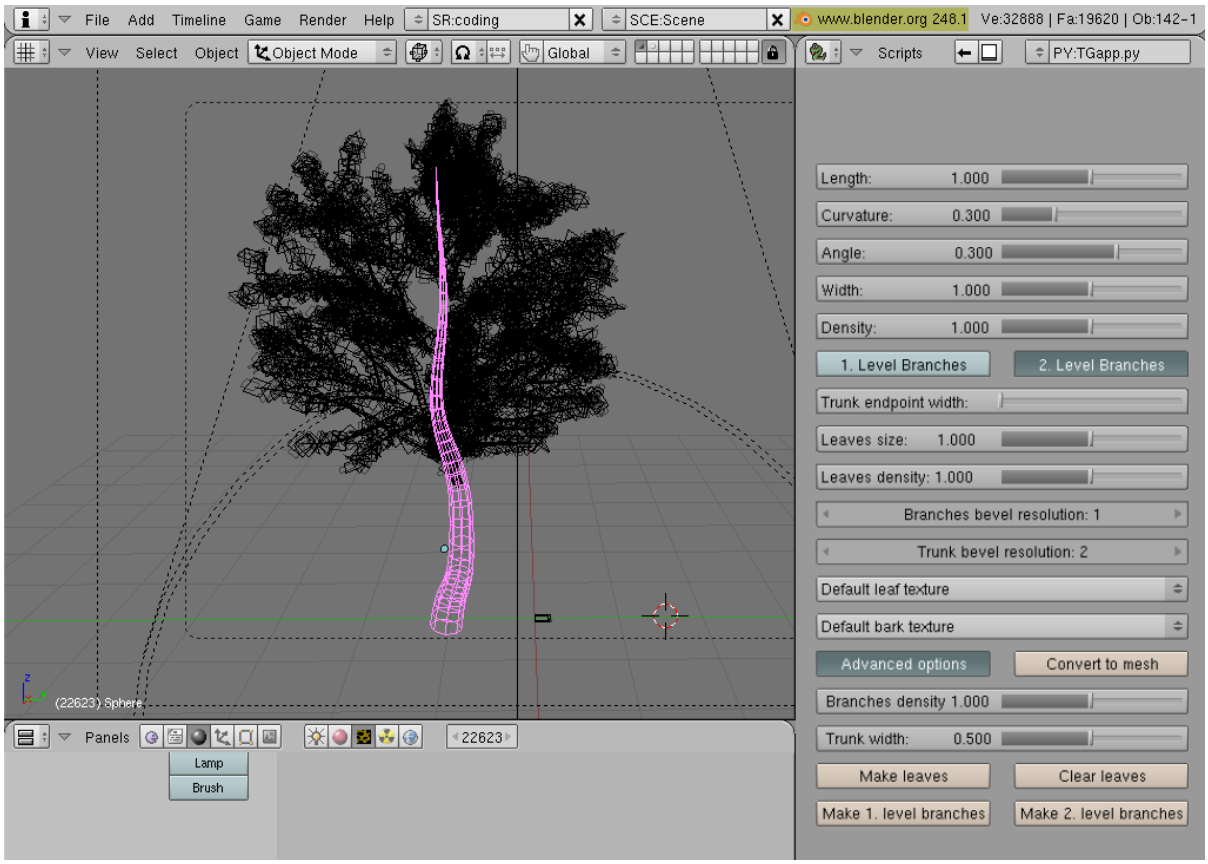
Keďže vykreslenie každého listu ako samostatného objektu by bolo pamäťovo neúnosné, je použitá metóda Blenderu nazývaná DupliFaces. Využíva duplikovanie objektu – vykreslený je iba jeden list, ktorý obsahuje všetky nastavenia a textúru. Ten je ďalej duplikovaný na miesta, kde sa nachádzajú ostatné listy. Stonky listov by mali byť súčasťou textúry listu. Ukážka uchytenia listu k vetve je na obrázku [Obrázok 9].

Obrázok 9: Ukážka uchytenia listu k vetve

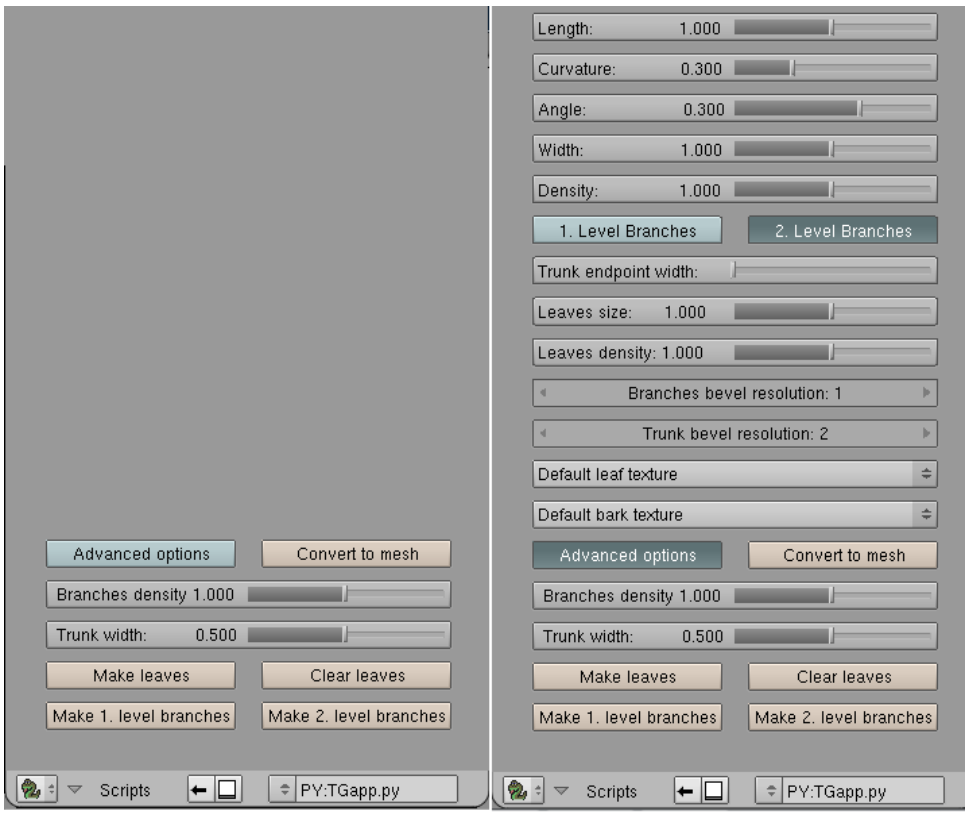
4.5 Ukážky z programu



Obrázok 10: Ukážka vygenerovaných stromov



Obrázok 11: Prostredie Blenderu so spusteným pluginom



a) b)

Obrázok 12: Ukážka GUI pluginu : a) základné možnosti b) pokročilé možnosti

Obrázok 13: Ukážka vygenerovaných stromov pomocou pluginu

5 Záver

Zadaním práce bolo implementovať generátor 3D modelov v programe Blender. Ako triedu objektov som si zvolil stromy. Práca sa zaoberá metódami generovania stromov a stručnou históriou ich vývoja. Detailnejšie sa venuje L-systémom a modelu parametrického generovania stromov podľa Webera a Penna [2].

Ďalej je predstavená parametrická metóda, ktorá bola implementovaná v Blenderi verzie 2.48a s využitím Pythona verzie 2.5.4. Za základ je vzatý zjednodušený model Webera a Penna, prispôsobený možnostiam a potrebám API Blenderu. Plugin umožňuje generovať jednoduché stromy s využitím beziérových kriviek. Vytvorené stromy sa skladajú zo 4 úrovní; kmeňa, vetiev prvej a druhej úrovne a listov. Generovanie sa dá ovplyvniť parametrami ako šírka vetvy, rozlíšenie, hustota vetiev, uhol oddelenia od materskej vetvy a iné. Tie sa nachádzajú v GUI vytvorenom pomocou modulu Draw, ktorý je súčasťou API Blenderu. Veľkosť a množstvo listov je tiež ovplyvniteľné parametrami. Užívateľ má možnosť jednoduchého nastavenia textúr pre vetvy aj listy pomocou GUI. Tvar stromu ohraničuje konvexná obálka definovaná užívateľom.

Jedným zo zámerov pluginu bolo umožniť jednoducho a rýchlo generovať stromy žiadaného tvaru bez špecifických znalostí z botaniky alebo problematiky generovania stromov. To sa snaží dosiahnuť nízkym počtom parametrov, automatickým nastavením čo najväčšieho počtu potrebných vlastností objektov a jednoduchým GUI.

V budúcnosti by sa program dal rozšíriť niekoľkými spôsobmi. Jedným z nich je rozšíriť ho tak, aby umožnil tvorbu konkrétnych a botanicky presných druhov stromov.

Inou možnosťou je simulovať vplyv vonkajšieho prostredia na rast stromu, ako je svetlo, množstvo vody a podobne.

Treťou možnosťou by bolo vytvoriť podklad pre animácie stromu a umožniť užívateľovi jednoduchšie vytvoriť animácie, ktorých súčasťou sú stromy. Mohol by sa simulovať vplyv vetra na strom v závislosti na nastavení jeho sily a smeru.

Literatúra

[1] PRUSINKIEWICZ, Przemyslaw, LYNDENMAYER, Aristid. *The Algorithmic Beauty Of Plants*. [online]. Springer-Verlag, New York, 1996.
Dostupné z WWW: <http://algorithmicbotany.org/papers/>.

[2] WEBER, Jason, PENN, Joseph. *Creation and Rendering of Realistic Trees.* [online].1995 [cit. 2009-05-04]. Dostupné z WWW: <portal.acm.org>.

[3] HONDA, H. Description of the Form of Trees by the Parameters of the Tree-like Body: Effects of the Branching Angle and the Branch Length on the Shape of the Tree-like Body. Journal of Theoretical Biology. 1971, pp. 331-338.

[4] OPPENHEIMER P. Real Time Design and Animation of Fractal Plants and Trees. Proceedings of SIGGRAPH '86 (Dallas, Texas, August 18-22, 1986). In Computer Graphics Proceedings, Annual Conference Series, 1986, ACM SIGGRAPH, pp. 55-64.

[5] DE REFFYE, EDELIN C., FRANCON J., JAEGER M., PUECH C. Plant models faithful to botanical structure and development. Proceedings of SIGGRAPH 88 (Atlanta, Georgia, August 1-5,1988). In Computer Graphics Proceedings, Annual Conference Series, 1988, ACM SIGGRAPH, pp. 151-158.

[6] The Blender Python API Reference, Blender 2.48a, 2009
Dostupné z WWW: <http://www.blender.org/documentation/248PythonDoc/>

[7] POKORNÝ, Pavel. Blender : naučte se 3D grafiku. 1. vyd. Praha : BEN, 2006.

[8] L-System Script,
Dostupné z WWW: <http://www.geocities.com/blenderdungeon/lsystem/index.html>

[9] Gen3, Blender tree generator
Dostupné z WWW: <http://www.geocities.com/bgen3/>

Zoznam príloh

Príloha 1. CD so zdrojovými kódmi