TECHNICKÁ UNIVERZITA V LIBERCI

FAKULTA MECHATRONIKY A MEZIOBOROVÝCH INŽENÝRSKÝCH STUDIÍ

Studijní program: B 2612 – Elektrotechnika a informatika
Studijní obor: 1802R022 Informatika a logistika

Rozšiřující Python skript pro Blender

Extension Python script for Blender

BAKALÁŘSKÁ PRÁCE

autor bakalářské práce: **Radek Hátle**

vedoucí práce: Ing. Jiří Hnídek
konzultant: Ing. Richard Charvát

Datum: 18.5.2007

zde vložte zadání

P r o h l á š e n í

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č.121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **souhlasím** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.)
Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 19.5.2007
Podpis ……
Radek Hátle

Poděkování

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Jiřímu Hnídkovi a celé Katedře aplikované informatiky za rady a připomínky a za celkový velmi kladný profesní i lidský přístup. Dále bych rád poděkoval rodině za podporu.

Abstrakt

Cílem bakalářské práce je podrobněji se seznámit s profesionálním 3D modelovacím a animačním programem Blender a jeho Python API. Na základě získaných informací, vytvořit rozšiřující skript pro vytváření krajiny v jazyce Python s několika funkcemi a GUI prostředím pro Blender.

V první části bakalářské práce se seznámíme s fraktální geometrií a několika základními pojmy. Dále je zde rozebrán L-systém, pomocí kterého můžeme vytvářet fraktály podobné vegetaci. Také se zmiňuji o želví grafice. Ve druhé části je charakterizován programovací jazyk Python. Ve třetí části je stručně popsáno prostředí Blender. Poslední část je zaměřena prakticky. Jsou zde uvedeny některé funkce skriptu a také návrh grafického prostředí. Nakonec jsem zde uvedl program, kterým byla vygenerována vegetace. Ta se následně použila v našem programu.

Abstract

The aim of this project is to gain understanding about professional 3D modeling and animation program Blender and its Python API in more details. On basic of gained information to create extended script for modeling of landscape in Python language with several functions and graphical user interfaces for Blender.

First part of this bachelor work introduces to the fractal geometry and provides several basic ideas. Moreover, L-system is also described in this part. Using of this system give the opportunity to create fractal similar to vegetation. There are also few sentences about turtle graphic. Second part characterizes programming language Python. Third part briefly describes setting of Blender. The last part of this work is focused on practical usage of above mentioned techniques. This chapter shows description of several functions of the script and also design of the graphic interface.

Finally, the work provides the program that is able to create vegetation, which is used in main program of this project.

Obsah

Poděkování 4
Abstrakt 5
Abstract 5
Seznam použitých zkratek 8
Úvod 9
1 Procedurální modelování 10
1.1 Fraktální geometrie 10
1.2 Soběpodobnost 10
1.2.1 Soběpodobnost přesná 10
1.2.2 Soběpodobnost statická 10
1.3 Fraktální dimenze, fraktál 11
1.3.1 Topologická dimenze 11
1.3.2 Hausdorffova dimenze 12
1.3.3 Fraktál 12
1.3.4 Výpočet fraktální dimenze 12
1.4 Dělení Fraktálů 14
1.5 L-systémy 15
1.5.1 Přepisovací pravidla 15
1.5.2 Úprava gramatik pro deterministické L-systémy 16
1.6 Částicový systém 17
2 Python 19
2.1 Výhody jazyka Python 19
2.1.1 Python je zdarma 19
2.1.2 Přenositelný 19
2.1.3 Objektově orientovaný jazyk 19
2.1.4 Jednoduchá a přehledná syntaxe 20
2.1.5 Dobré použití 20
2.1.6 Výkonný Python 21
2.1.7 Spolupráce 21
2.1.8 Široké využití Python 22
2.2 Slabiny jazyka Python 22
3 Charakteristika Blenderu 23
3.1 Historie 23
3.2 Architektura Blender 23
3.3 Uživatelské prostředí 25
3.4 První spuštění 25
3.5 Pomoc a podpora 26
4 Praktická část 27
4.1 Blender Python API 27
4.2 Spouštění skriptu 28
4.3 Návrh grafického prostředí 28
4.4 Popis tlačítek v grafickém prostředí 28
4.5 Mód váha vertexu 30
4.5.1 Přiřazení váhy vertexů modelu krajiny 30
4.5.2 Částicový systém a váha vertexů 31
4.6 Low a High detail 34
4.6.1 High detail 34

4.6.2Low detail..... 34
4.7Generování stromů pomocí skriptu L-systém..... 36
4.7.1Popis funkce tlačítek..... 37
4.8Části programu..... 40
4.9Volání Funkce přepočti..... 40
4.9.1Použití registrů..... 41
Závěr..... 42
Seznam použité literatury..... 43
Seznam obrázků..... 44
Seznam příloh..... 44
Příloha A: Obsah přiloženého CD..... 45
Příloha B: Vyrenderované obrázky..... 46

Seznam použitých zkratek

GUI	graphical user interface
API	application programming interface
GNU/GPL	general public license
FTP	file transfer protocol
HTML	HyperText Markup Language
XML	Extensible Markup Language
CGI	Common Gateway Interface
IRC	Internet Relay Chat

Úvod

Profesionální 3D modelovací a animační program Blender je charakterizován tím, že je svobodný. Znamená to, že kdokoliv se může podílet na jeho vývoji a tudíž je i zdarma. V mnoha ohledech konkuruje i komerčním programům, které se pořizují za nemalou částku. Na vývoji programu se můžeme podílet několika způsoby. Ti, co ovládají C/C++, si můžou upravovat stávající verze ke svým potřebám, nebo je nějakým způsobem vylepšovat. Dalším způsobem, jak lze rozšířit funkcionalitu Blenderu, je napsání Python skriptu. Tento způsob rozšíření Blenderu je podstatně jednodušší z několika hledisek. Prvním je ten fakt, že Python je velice jednoduchý programovací jazyk a pokud máme základy alespoň nějakého jiného jazyka, můžeme se jej naučit během několika dní, hodin. Další výhodou pro začátečníka je i to, že pro napsání jednoduchého skriptu nemusíme nic instalovat kromě samotného Blenderu.

1 Procedurální modelování

Procedurální modelovaní je získávání modelů pomocí algoritmu. Největší výhodou tohoto způsobu je jeho parametrizovatelnot. Znamená to, že při změně vstupních dat simulace můžeme dostat širokou škálu výstupních dat.

1.1 Fraktální geometrie

Teoretický aparát pro simulaci objektů podobných tvarově reálného světa představuje tzv. fraktální geometrie. Je to vědní disciplína, která je intenzivně rozvíjena od šedesátých let dvacátého století. Za objevitele je považován vědec polského původu Benoit B. Mandelbrot, který žil a pracoval v USA.

Zatímco klasická geometrie popisuje invariance vzhledem k transformacím, jako je otočení nebo posunutí, fraktální geometrie je oblast zabývající se invariatními množinami ke změně měřítka. Tato vlastnost je typická pro mnoho reálních objektů. Například malý kámen vypadá jako celá hora, nebo větev stromu je podobná malé větévce a podobně.

1.2 Soběpodobnost

Ústřední pojem fraktální geometrie je soběpodobnost. Rozlišujeme dva druhy soběpodobnosti, soběpodobnost přesnou a statickou.

1.2.1 Soběpodobnost přesná

Množina A je přesně soběpodobná, pokud je sjednocením konečného počtu nepřekrývajících se transformovaných kopii sebe samé.

$$A = \bigcup_{i=1}^{n} \phi_i(A) \qquad [1.2.1]$$

V tomto vztahu jsou transformace Φi posunutí a otáčení. Každá z těchto transformací je zároveň změnou měřítka, které má koeficienty S,x,y,z. Přesně soběpodobný objekt je například sněhová vločka Kochové.

1.2.2 Soběpodobnost statická

Množina A bude staticky soběpodobná, jestliže bude sjednocením konečného počtu nepřekrývajících se transformovaných kopií sebe samé, dle [1.2.1], a každá z kopií Φ i(A) bude mít stejné statické charakteristiky jako množina A. Říkáme že, Φi(A) a A jsou

statický nerozlišitelné. Na transformace Φi nebude kladen žádný další požadavek kromě změny měřítka, které má koeficienty (Φi ∈ (0,1)). Jsou-li aplikované transformace Φi lineární, bude také soběpodobná množina lineární. Za zachování podmínky statické soběpodobnosti v praxi obyčejně považujeme shodu směrodatné odchylky a průměru a ne všech statických momentů.

Příkladem statické soběpodobnosti může být kámen a hora. Pokud budeme vhodně porovnávat tyto dva objekty na fotografii bude pro nás obtížné poznat, co je kámen a co je celá hora. Dalším příkladem může být například šum z rádia. Jestliže šum budeme přehrávat libovolnou rychlostí (budeme měnit měřítko), ale bude znít stále stejně.

Obrázek 1: Soběpodobný fraktál, list kapradiny

1.3 Fraktální dimenze, fraktál

K tomu, abychom mohli kvantifikovat vlastnosti přírodních objektů, jako je členitost hor, košatost stromů, obrysy korálových útesů nebo drsnost povrchu planet, musíme zavést pojem fraktál a fraktální dimenze. Proto je nezbytné alespoň zjednodušeně definovat některé základní pojmy týkající se míry a dimenze objektů. [1]

1.3.1 Topologická dimenze

Geometrické objekty, které je možné popsat klasickou Euklidovskou geometrií, mají celočíselnou dimenzi, nazývanou také topologická dimenze. Pokud si celou problematiku zjednodušíme, můžeme říci, že topologická dimenze určuje počet parametrů, kterým lze dané těleso popsat. Například bod má nulovou dimenzi, jelikož je sám popsán

vztahem $P=X$ (tj. konstantním vektorem) a úsečka má dimenzi rovnu jedné, neboť ji lze popsat vztahem $y_t=y_0+kt$, kde t je jediný parametr. Pozici každého bodu ležícího na úsečce lze vyjádřit výše uvedeným vztahem. Jakákoliv hladká plocha (kruh, trojúhelník, n- úhelník) má dimenzi rovnu dvěma, to znamená, že poloha bodu musí být určena pomocí dvou parametrů. Krychle, (vyplněná) koule, válec nebo celý běžný prostor kolem nás mají dimenzi rovnu třem, protože poloha jakéhokoli bodu je v nich jednoznačně určena třemi parametry.

1.3.2 Hausdorffova dimenze

Objekty popisované fraktální geometrií mají dimenzi neceločíselnou. Dimenzi fraktálních objektů nazýváme fraktální dimenzí či Hausdorffovou dimenzí. Hodnota této dimenze (resp. míra rozdílu mezi fraktální dimenzí a dimenzí topologickou) potom udává úroveň členitosti daného objektu.

Měřením délky geometricky hladké křivky, která má topologickou dimenzi rovnu jedné, dostaneme při pohledu v různých měřítkách vždy stejné konečné číslo. Měřením délky břehu ostrova (což je opět křivka s topologickou dimenzí rovnou jedné) se při zmenšování měřítka toto číslo stává nekonečně velkým. Pobřeží tedy v rovině zabírá více místa než hladká křivka. Nezabírá však všechno místo (přesněji řečeno, nevyplňuje celou rovinu). Jeho skutečná dimenze je tedy větší než topologická dimenze křivky (ta je rovna jedné) a současně je menší než topologická dimenze roviny (ta je rovna dvěma). Z toho jasně vyplývá, že dimenze takového útvaru není celočíselná. Toto neceločíselné číslo se nazývá Hausdorffovou dimenzí. [2]

1.3.3 Fraktál

Rozdílu mezi topologickou dimenzí a Hausdorffovou dimenzí použijeme i při následující definici fraktálu. Tuto definici formuloval matematik Benoit B.Mandelbrot.

"Fraktál je množina či geometrický útvar, jejíž Hausdorffova dimenze je (ostře) větší než dimenze topologická."

1.3.4 Výpočet fraktální dimenze

1.3.4.1 Úsečka

Mějme úsečku o jednotné délce. Nyní tuto úsečku rozdělíme na N dílů. To

odpovídá tomu, jako bychom se na úsečku podívali s *N*-násobným zvětšením. Měřítko nové úsečky se tedy vypočítá následujícím způsobem:

$s=1/N,$

kde *s* značí měřítko a *N* je počet dílů, na které se úsečka rozdělí. Pro Hausdorffovu dimenzi obecně platí následující podmínka:

$Ns^D=1.$

Z výše uvedeného vztahu vyplývá, že Hausdorffova dimenze se pro dané dělení *N* a dané měřítko *s* vypočítá pomocí postupu:

$Ns^D=1,$

$\log Ns^D=\log 1,$

$\log N+\log s^D=0,$

$\log N+D \log s=0,$

$D \log s=-\log N,$

$D=(-\log N)/\log s,$

$D=\log N/\log (1/s)$.

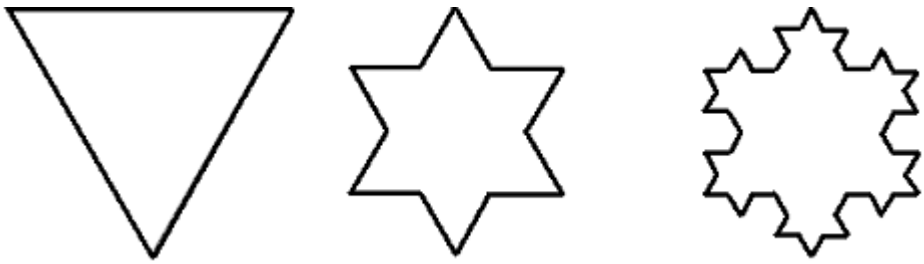
Po dosazení výše uvedeného vztahu $s=1/N$ do vzorce $D=\log N/\log (1/s)$ získáme výsledek:

$D=\log N/\log (1/s)=\log N/\log N=1$

Topologickou dimenzi úsečky samozřejmě známe z Euklidovské geometrie: je rovna jedné. Hausdorffovu dimenzi jsme nyní vypočítali a je také rovna jedné. Hausdorffova dimenze se tedy rovná dimenzi topologické. Z výše uvedené definice fraktálu tedy vyplývá, že úsečka není fraktál (pro fraktál musí být Hausdorffova dimenze ostře větší než dimenze topologická).

1.3.4.2 Sněhová vločka Kochové

Nyní zkusíme vypočítat Hausdorffovu dimenzi útvaru, jehož zjemnění o jeden krok spočívá v tom, že se každá úsečka předchozího útvaru nahradí dvěma úsečkami se třetinovou délkou a rovnostranným trojúhelníkem sestrojeným uprostřed mezi dvěma novými úsečkami (viz obrázek 2). Tento objekt se nazývá podle svého objevitele vločka či křivka Helge von Kocha.



Obrázek 2: křivka Helge von Kocha ve třech intracích.

Při trojnásobném zjemnění se délka křivky zvětší čtyřikrát, proto Hausdorffova dimenze není celé číslo. Pro *N=4* se tedy měřítko musí zmenšit na třetinu:

s=1/3,

N=4.

Hausdorffova dimenze křivky Helge von Kocha se vypočítá jako:

D=log N/log (1/s)=log 4/log 3=1,2618595.

Topologická dimenze této křivky je rovna jedné, Hausdorffova dimenze je však větší než jedna. Z toho vyplývá, že křivka Helge von Kocha je fraktálem.

1.3.4.3 Hausdorffova dimenze vybraných přírodních útvarů

V následující tabulce je uveden odhad Hausdorffovy dimenze některých přírodních útvarů.

přírodní objekt	odhad fraktální dimenze
pobřeží	1.26
povrch mozku člověka	2.76
neerodované skály	2.2 - 2.3
obvod 2D - průmětu oblaku	1.33

Tabulka 1: Dimenze některých útvarů.

[2]

1.4 Dělení Fraktálů

Podle nejhrubší dělení rozlišujeme následující typy fraktálů:

- dynamické systémy s fraktální strukturou,
- l-systémy,
- systémy iterovaných funkcí IFS,
- stochastické fraktály (nepravidelné fraktály).

1.5 L-systémy

L-systémy navrhl a zkoumal Aristid Lindenmayer. Jsou to fraktály především vhodné pro simulaci rostlin. Název této skupiny fraktálů pochází ze zkráceniny anglického sousloví LOGO-*like* turtle. LOGO je programovací jazyk, vycházející z funkcionálního programovacího jazyka LISP, ve kterém se kromě snadného zpracování datových struktur, dají s využitím jednoduchých příkazů pomocí takzvané želví grafiky kreslit různé obrazce.

L-systémy jsou matematické algoritmy založené na přepisování řetězců podle určitých pravidel, která jsou buď předem zadaná množinou přepisovacích pravidel (*gramatiky*), nebo se mění v průběhu generování fraktálního obrazce, například na základě zpětné vazby či na podněty okolního prostředí (gravitace, dopadající světlo apod.). Řetězce tvoří terminální a neterminální symboly. Pod neterminálním symbolem si můžeme představit znak abecedy, na který lze aplikovat pravidlo, kdežto na terminální symbol, neboli koncový symbol, pravidlo aplikovat nelze. Je mu přiřazen geometrický význam pro želvu v jazyce LOGO. Jedná se o želvu, která umí interpretovat základní příkazy. Je to pohyb dopředu, otočení se doprava o daný úhel a uložení stavů do zásobníku. Případně svůj stav může ze zásobníku vyzvednout a zorientovat se podle něj. Celá operace začíná tzv. startovním symbolem. Startovním symbolem rozumíme neterminální symbol, kterým přepisování začíná. Posloupností terminálních a neterminálních symbolů je tvořeno pravidlo. Pravidlo budeme přepisovat tak dlouho dokud obsahuje neterminální symboly.

1.5.1 Přepisovací pravidla

Gramatiku je možné popsat uspořádanou čtveřicí:

$G=[N,\Sigma,P,S],$

kde:

- N je konečná abeceda nonterminálních symbolů,
- Σ je konečná abeceda terminálních symbolů,
- P je konečná množina přepisovacích pravidel tvaru $A \rightarrow B,$
- S je axiom: neprázdná posloupnost symbolů $S \in (N \cup \Sigma)^{+}.$

Přepisovací pravidla jsou tvaru $S \rightarrow aBC,\ B \rightarrow XY,\ C \rightarrow a,\ C \rightarrow D$ (dokonce i $XYZ=DB$), kde velkými písmeny jsou označeny nonterminální symboly, písmeny malými zase symboly terminální (ty se již dále nerozepisují). [3]

1.5.2 Úprava gramatik pro deterministické L-systémy

V L-systémech je použita zjednodušená forma gramatik, ve kterých splývají terminální a nonterminální symboly. Determinismus tohoto L-systému vychází z podmínky, kde v množině *P* nesmějí být dvě pravidla se stejnou levou stranou. Při přepisování tedy vždy přesně víme, které pravidlo pro přepis vybrat.

Deterministický bezkontextový L-systém je tvořen uspořádanou trojicí:

G=[V,P,S],

kde:

- *V* je konečná abeceda symbolů (zde nerozlišujeme symboly terminální a nonterminální),
- *P* je konečná množina pravidel tvaru $A \rightarrow B;\ A \in V;\ B \in V^*$,
- *S* je axiom: neprázdná posloupnost symbolů $S \in V^+$. [3]

1.5.2.1 Příklad dl-systému, želví grafika

Například první dvě derivace L-systému *G*:

G=[{A,F,-,+}, {A → F - - F - - F,F → F + F - - F + F,+ → +,- → -},A]

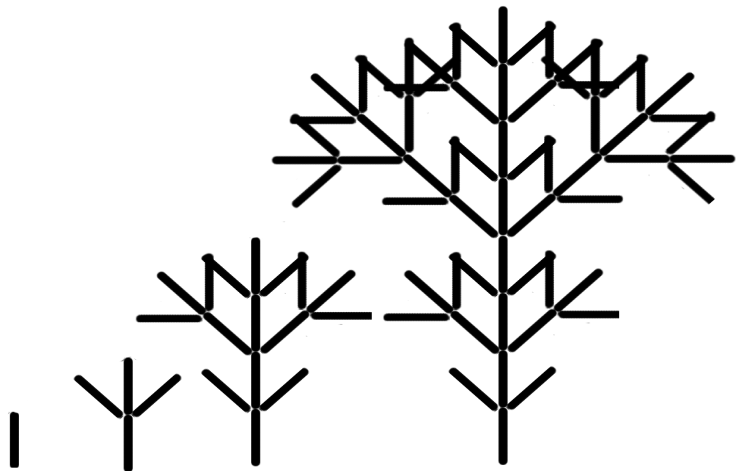
se startovním symbolem *A* mají podobu:

A → F - - F - - F → F + F - - F + F - - F + F - - F + F - - F + F - - F + F.

Posloupnost symbolů získaná v poslední derivaci se ve druhém kroku geometricky interpretuje pomocí již zmiňované želví grafiky. Želva je definována svým stavem a tabulkou akcí. Stav se skládá ze dvou prvků, z polohy želvy bodu a její orientace. Orientace ve trojrozměrném prostoru se skládá ze tří vektorů H,U,L, které spolu s polohou tvoří lokální souřadnicový systém želvy. H (heading) určuje směr, kterým se želva pohybuje kupředu, U (up) určuje směr, kterým má krunýř a konečně L(left) nám říká, na které straně má levou přední i zadní nožičku.

Pokud naučíme želvu ukládat svůj stav do zásobníku případně ho z něho vyzvedávat nebo se podle něj zorientovat, můžeme pomocí něj generovat rozvětvené struktury jako jsou keře a stromy. Viz následující příklad:

G=[{F,+,-,],[},{F→F[+F][-F]F},F].



Obrázek 3: Obrázek získaný dl-systémem.

Želvu můžeme vybavit dalšími gramatickými symboly, jako je například návrat na počátek souřadnicového systému, pootočení dle určitého vektoru v prostoru a další. Přidáním těchto symbolů získáme věrohodněji vypadající porost, ať už je to strom nebo nějaký keř.

1.6 Částicový systém

Částicový systém je velice silná modelovací a animační technika. Byl objeven v roce 1983. Tento systém se používá zejména pro modelování objektů, jejichž tvar není reprezentován jako povrch. Může to být například oheň, déšť, hejna ptáků, tráva, les a podobně. Systém částic je reprezentován jako soubor částic, jejichž vlastnosti se mění v čase. Mezi tyto vlastnosti si můžeme například uvést tvar, směr, barvu atd. Tyto části mohou v jistém okamžiku vznikat a následně zanikat. Můžou se také srážet nebo generovat další částice. Částice bývají nejčastěji reprezentovány jako body. Mohou to být ale i třeba malé míčky, kapky deště, sněhové vločky, bublinky v limonádě, hvězdy na obloze...

Vlastnosti, které částice mají, jsou charakterizovány jednoduchým vztahem.

$X = s + rand * v,$

kde *s* je střední hodnota hustoty pravděpodobnosti náhodného generátoru, *v* je jeho rozptyl, *rand* je náhodné číslo a konečně *x* reprezentuje vlastnost částice, jako je například barva, životnost atp.

Dynamický částicový systém byl poprvé použit ve filmu Star Trek v roce 1985 na

scénu výbuchu planety. Kdežto statický částicový systém byl úplně poprvé použit pro tvorbu lesa, který se skládal z velkého množství objektů, které nebylo možno vytvořit ručně. Tvorba takovéto scény byla následující. Plocha, na níchž měly být umístěny stromy se rozdělila na menší čtverce. Do prostřed každého z nich se umístil jeden bod, reprezentující rostlinu. Ve druhém kroku se tento bod posune o náhodnou hodnotu tak, aby neopustil svůj čtverec. Následně se určité procento bodů ze scény vypustí a ze zbylých bodů vyroste strom.

Na podobném principu je založen náš skript. Blender má také implementovaný částicový systém. Částice mohou být vystřelovány buď z vertexu a nebo z plochy. Pokud budou vystřelovány z plochy, můžeme také nastavit náhodnou pozici, která je vztažená k celkové ploše. Z těchto části pak roste námi zvolená vegetace. Více se dozvíme v poslední části této práce.

2 Python

Vývoj Pythonu začal v roce1990 na Stichtig Mathematisch Centrum v Nizozemí. U jeho zrodu stál Guido vam Rossum. Programovací jazyk dostal jméno podle pořadu BBC Monty Python's Flying circus. Python je volně šířitelný, objektově orientovaný skriptovací jazyk, kde se v jeho návrhu nejen mísí prvky tradičních programovacích jazyků a postupů vývoje, ale také i jednoduchost použití skriptovacích jazyků. Ideálně se zde snoubí značná výkonnost s nesmírně jednoduchou syntaxí, protože tvůrci využili předností dosavadních programovacích jazyků a jejich nejčastější neduhy přizpůsobili dnešním potřebám.

2.1 Výhody jazyka Python

2.1.1 Python je zdarma

Python je volně šířitelný programovací jazyk. To znamená, že nejsme nijak omezeni v jeho používání, kopírování, nebo jeho další distribuci. Můžeme ho také prodávat. Licence je kompatibilní s GPL. Tím, že je svobodný, není nijak omezený a to přispívá k jeho rychlému vývoji. Z rychlostí odezev on-line komunity okolo Pythonu a následné rychlé vyřešení problému by se mohl poučit leckterý dodavatel komerčního softwaru.

2.1.2 Přenositelný

Přenositelností programu se myslí možnost jejich spouštění na různých operačních systémech a různých hardwarových architekturách. Samotný Python je napsán v přenositelném ANSI C a je možné jej zkompilovat na téměř každé platformě. Můžeme se s ním například setkat na Unixech a Linuxu, kde je součástí instalace ve většině distribucí, dále se s ním můžeme setkat na Microsoft Windows, DOSu, Macintoshi, OS/2, Amiga, BeOS, VMS, PocketPC a další. Také zde můžeme vytvářet aplikace, které budou mít přenositelné grafické uživatelské rozhraní tzv. GUI. Tyto aplikace budou vypadat na všech platformách stejně.

2.1.3 Objektově orientovaný jazyk

Objektově orientované programovaní je mocným nástrojem pro strukturování a znovu používání kódu, činí jej ideálním skriptovacím jazykem pro tradiční OO jazyky,

jako jsou C++ a Java.

Python je od základu navržen jako objektově orientovaný jazyk. Podporuje nejen polymorfizmus tříd dědících od jisté třídy jako ve statických typových jazycích, ale i plný polymorfizmus pro všechny objekty. Samozřejmě také podporuje dědičnost zahrnující vícenásobné dědičnosti a přetěžování operátorů. Díky tomu, že Python je dynamický jazyk, tak i třída samotná je v něm objekt, tedy datový prvek. To umožňuje vytváření nových tříd přímo za běhu programu a podobně.

2.1.4 Jednoduchá a přehledná syntaxe

Tvůrci Pythonu považují syntaxi za velmi důležitou. Jejich cílem je jednoduchá, přehledná, snadno srozumitelná a zároveň velice schopná syntaxe, která poskytuje co nejméně příležitostí k chybám. Někdy bývá označován jako „spustitelný pseudokód“. Python se snaží odstraňovat všechny složitosti, z čehož plyne, že délka kódu v Pythonu je jen zlomkem délky odpovídajícího programu v C, C++ nebo v Javě. A z toho se sám nabízí fakt, že máme menší šanci udělat někde chybu.

Dalším znakem syntaxe, kterého si můžeme všimnout na první pohled, je odsazování. Odsazování je povinné a tvoří strukturu programu, bloky příkazů. V ostatních programovacích jazycích se k tomu používají složené závorky, jako je tomu v C, Java a PHP. Jiné jazyky používají klíčová slova například v Pascalu nebo VB. Syntaxi, a to jak vypadá, odsazení programu můžeme vidět na následujícím přikladu. Jedná se o skript, který slouží k vykreslení křivky Helge von Kocha.

```
from turtle import *;
a=60;b=120
def f(_):
    if _: f(_-1);left(a);f(_-1);right(b);f(_-1);left(a);f(_-1)
     else:forward(10)
reset();up();backward(b);down();f(3);raw_input(' ')
```

2.1.5 Dobré použití

Programování v Pythonu se stává zábavou a ne prací. Pokud chceme vytvořit aplikaci v Pythonu, stačí ji napsat a okamžitě spustit. Odpadá zde kompilování a linkovaní

jako je tomu například v C nebo C++. Díky tomu se značně zrychluje vývoj aplikací a ihned můžeme vidět dopad změn v programu.

2.1.6 Výkonný Python

Python se řadí mezi hybridní jazyky, kombinuje přednosti skriptovacích a kompilovaných jazyků. Programuje se v něm stejně dobře jako ve skriptovacích jazycích, jako je například Perl nebo Tcl, ale na rozdíl od nich obsahuje mocné nástroje kompilovaných jazyku a díky tomu je možné vytváret i velké projekty.

2.1.6.1 Dynamické typování

Vzhledem k tomu, že Python využívá dynamické typování, na rozdíl od statického nemusíme na začátku kódu uvádět typ a velikost proměnné. Vše to za Vás udělá virtuální stroj za běhu. Pokud dojde ke změně rozměru seznamu popřípadě jiného typu, Python sám automaticky alokuje nový prostor, zkopíruje do něj jednotlivé položky a předchozí prostor uvolní. Na to je ovšem potřeba brát zřetel, aby byl program efektivní.

2.1.6.2 Standardní datové typy a práce s nimi

Součástí standardních datových typů jazyka Python je například seznam, slovník a řetězec. Tyto datové typy jsou velice užitečné. Pro snadnou práci s nimi můžeme využít mocné nástroje jako je skládání, třídění, výběr částí posloupností, vyhledávání a další. Můžeme s nimi velice jednoduše vyřešit řadu algoritmických úkonů.

2.1.6.3 Automatická správa paměti

Python obstarává všechny nízko úrovňové detaily okolo správy paměti. Když datové objekty už nejsou potřeba, pak automaticky zanikají. Znamená to, že jsou automaticky vymazány z paměti počítače.

2.1.7 Spolupráce

Programy v Pythonu lze snadno rozšiřovat dalšími částmi napsanými v jiných jazycích, což je jeden z předpokladů pro jazyk, ve kterém se mají psát velké projekty.

Například můžeme projekt psát v jazyce Python a části náročné na výpočet vytvořit v C nebo C++. Díky knihovnám, která se jmenují Python/C API, lze z programů v Pythonu

volat kód v C nebo z C volat kód v Pythonu.

2.1.8 Široké využití Python

Python se používá ve skutečných ziskových projektech jako je například NASA, Google, Yahoo. Dále jej také používá Red Hat ve svých nástrojích pro instalaci Linuxu.

Standardní knihovny Pythonu podporují práci se systémem, proto je Python ideální nástroj pro psaní přenositelných, snadno spravovatelných skriptů.

Pomocí GUI můžeme vytvářet rychle a jednoduše uživatelská rozhraní. Slouží nám k tomu například knihovna Tk. Pokud nebudeme vyžadovat přenositelnost, můžeme použít knihovnu MFC.

Python je vhodný pro rychlý vývoj aplikace. Můžeme vyvinout prototyp programu a následně určité nebo celé části programu přepsat do jazyka C. Části, které nepotřebujeme urychlit necháme v Pythonu pro snadnější správu.

Z tradičních prostředků pro Python nabízí spolupráci s databázemi firem Sybase, Oracle, IBM, nebo ZODB, která je napsaná přímo v Pythonu.

Python exceluje s řadou funkcí určených pro práci s internetem. Podporuje komunikaci přes FTP, parsuje HTML a XML, programovaní v CGI skriptech.

Pro práci s matematickými funkcemi je tu knihovna NumPy, která mimo jiné nabízí některé další datové typy např. N-dimenzionální pole a dále umožňuje přístup k běžným velkým matematickým knihovnám. Použitím těchto rychlých knihoven získáme pokročilý nástroj.

2.2 Slabiny jazyka Python

Žádný programovací jazyk není ideální, tedy i Python má své slabiny. Jedním z nedostatků je rychlost. Zpomalení způsobuje to, že u Pythonu předem nedeklarujeme datové typy jako například v C. V Pythonu se velikost objektu může měnit za běhu. Pokud tedy do datového objektu, jako je například seznam, přidáme další prvky, které se již do vyhrazené paměti nevejdou, musí být překopírován celý seznam na volné místo v paměti, což způsobuje ono zpomalení. Tuto slabinu lze odstranit tím, že náročné úlohy lze naprogramovat v C a následně je volat v Pythonu.

3 Charakteristika Blenderu

Blender je multiplatformní open source software zaměřený na vytváření 3D počítačové grafiky a animace. Součástí je rovněž GameEngine a Python API, umožňující tvorbu interaktivních prezentací, vizualizací nebo her. Multiplatformni znamená, že je dostupný na většině operačních systémech jako je Windows, Linux, Mac OS X a další. Je to program, který je založený na grafické knihovně openGL, která umožňuje nejen hardwarovou akceleraci vykreslování 2D a 3D modelů, ale i onu již zmiňovanou multiplatformost. Blender je šířený pod licencí GNU/GPL, která dovoluje program volně používat, šířit a modifikovat.

3.1 Historie

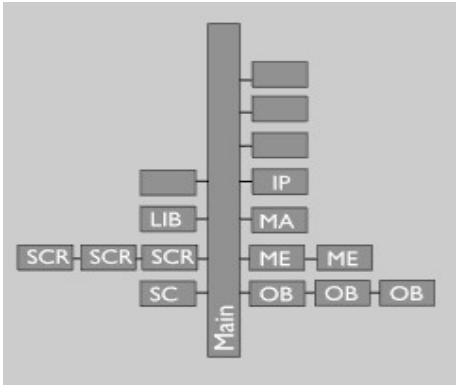
Vývoj Blenderu započala holandská firma NeoGeo v roce 1995. Po třech letech vznikla z původní firmy nová Not a Number (NoN), která měla hlavní úkol, a to zdokonalit Blender. Hlavním zdrojem přijmu měly tvořit produkty Blenderu, nikoli prodej softwaru samotného. Cílem bylo vytvořit volně šířitelný profesionální modelovací a animační program, který by mohl konkurovat podobným komerčním produktům. NoN v roce 2000 získala od investorů finanční podporu ve výší 4,5 mil Eur, což umožnilo velký rozmach firmy i rozšíření pracovních míst. Hned na to společnost uvedla novou verzi programu Blender 2.0, která byla rozšířena o herní rozhraní. V roce 2001 byl uveden na trh první komerční produkt společnosti NoN, jmenoval se Blener Publisher a byl zaměřen na nově vzniklý trh s interaktivními 3D medii pro WWW. Zklamání z prodeje a těžká situace vedla investory, aby pozastavili všechny práce NoN. Ton Roosendaal, zakladatel společnosti NeoGeo, založil nekomerční organizaci, která se jmenovala Blender Foundation. Následně na to, v roce 2002, byla vykoupena práva na zdrojové kódy Blenderu za 100 000 Eur. K tomuto aktu přispěly bývalí nadšení zaměstnanci společnosti NoN , kteří uspořádali kampaň „Free Blender“. Od října roku 2002 je Blender šířen pod licencí GNU General Public License (GPL)

3.2 Architektura Blender

Blender má striktně ogranizovanou datovou strukturu připomínajíci databázi, pouze s tím rozdílem, že datové bloky mají i některé vlastnosti Objektů. Všechna data v paměti

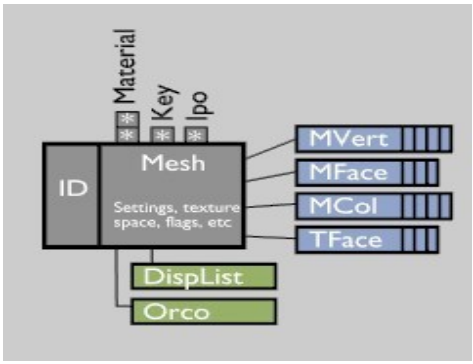
jsou uložena v hlavním stromu, který je ve skutečnosti tvořen seznamem bloků. Ilustruje to obrázek 4.

Datové bloky Blenderu mohou být modifikovány, kopírovány a spojovány s jinými. Každý takový blok má své ID nesoucí unikátní jméno a v určitých případech také jméno knihovny, ze které blok pochází.



Obrázek 4: Uspořádání dat v paměti.

Na obrázku 5 vidíme datový blok typu Mesh. Jsou zde uchovány informace o vertexech, plochách a nebo například informace o použitém materiálu (modré bloky na obrázku 5). Tyto data se označují jako *Direct data*. Data se čtou, nebo jsou ukládány přímo s objektem do souboru.



Obrázek 5: Podrobný náhled datového bloku typu Mesh.

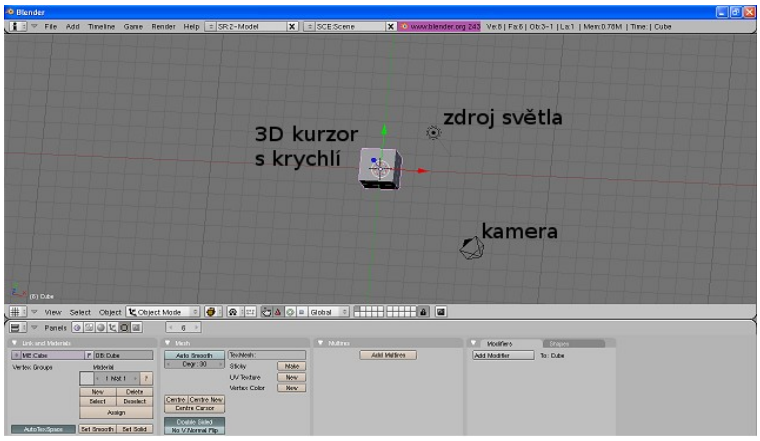
3.3 Uživatelské prostředí

Blender používá tzv. „sub window“, což je systém oken, který se nikterak nepřekrývá. Stávající okna v programu se můžou rozdělit na další okna různého typu dle potřeby. Také můžeme nastavit velikost tohoto okna. Je to však na úkor ostatních oken. Změnit můžeme dokonce i funkci. Je to například základní okno 3D view, textové okno, které se může použít pro vkládání poznámek, nebo skriptování pomocí jazyku Python. Dále to může být například okno typu Outliner, ve kterém se zobrazují datové bloky. V Blenderu je něco přes patnáct typů oken a nebudeme si je zde všechny vypisovat, protože to není cílem bakalářské práce.

3.4 První spuštění

Po spuštění programu v systém Microsoft Windows by se měla otevřít dvě okna. Jedno je textové, slouží pro vypisováno funkcí programu, zobrazování případných chyb v Python skriptu a podobně. Také zde zjistíme, zda-li máme nainstalovaný Python a případně i jeho verzi. Druhé okno je grafické.

Jak je vidět na obrázku 6 ve standardním nastavení jsou 3 typy oken. Ve střední části je to okno typu *3D View*, slouží pro zobrazení scény. Vidíme zde základní objekt krychli s 3D kurzorem, který slouží mimo jiné pro polohu umisťování dalších objektů do scény. Umístěn je zde také zdroj světla typu lampa, který je symbolizován jako dva kruhy s tečkou uprostřed. Posledním objektem ve scéně je kamera, která připomíná čtyřboký jehlan. Určuje směr a polohu pozorovatele. V horní části je umístěno okno typu *uživatelské nastavení*, které má skryté vlastnosti, jako je nastavení jazyka a další užitečných věcí. Konečně v dolní části obrazovky je oknu typu *Button*. Zde jsou umístěna různá tlačítka spojena s vybraným objektem.



Obrázek 6: Základní rozdělení oken a popis objektů ve scéně.

3.5 Pomoc a podpora

Stejně jako většina open source programů má i Blender svoji uživatelskou podporu. Patří k ní různá diskuzní forá. Na internetu jich je několik. K těm hlavním lze zařadit [9] a [10]. Dále můžeme také použít IRC chatovací program. Je to program, ve kterém chatuje více uživatelů najednou. Na serveru irc.frenode.net je místnost #blenderpython , která slouží k debatám o programovacím rozhraní pro Python a vývoji skriptů. Dalším zdrojem informací může být wiki, je to open sours projekt na tvorbu webových stránek. Nejznámější je jako otevřená encyklopedie wikipedia.org. Wiki pro Blender naleznema na [11].

4 Praktická část

Blender je rozšiřitelný prostřednictvím Python skriptů nebo použitím materiálových nebo sekvenčních (post-produkčních) plugin modulů, dodávaných ve formě souborů knihoven (např. dll). Co se týče Python skriptů, je jich celá řada. Některé z nich můžeme nalézt přímo na oficiálních stránkách Blenderu [5]. Mohou to být skripty, které nám pomáhají při animaci postav, nebo to může být skript pro tvorbu ozubených kol, dále to jsou různé skripty pro editaci objektu. Zajímavý je skript, pomocí něhož lze vytvořit lidskou postavu, a nebo skript, určený pro vytváření stromů tzv. L-systém. V následující části je popsán vlastní skript, pomocí něhož můžeme vytvářet les nebo park. Architekti by ho mohli využít při tvorbě vizualizace nové zástavby.

4.1 Blender Python API

Pomocí Python API můžeme manipulovat s daty, které náleží objektům v Blenderu. Pokud je změníme, změna se projeví na objektu.

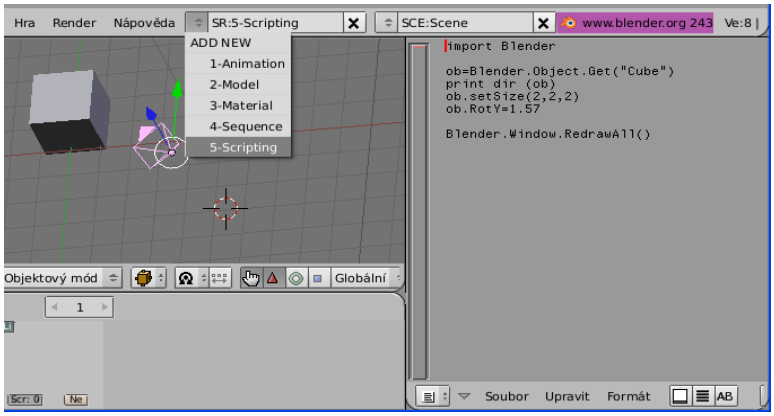
Základním modulem je modul Blender. Abychom mohli metody a objekty modulu používat musíme je načíst. To provedeme pomocí klíčového slova *import*. Na ukázku je uveden malý skript, pomocí něhož změníme velikost objektu a následně jej otočíme.

```
1. import Blender
2. from Blender import Object
3. ob=Blender.Object.Get("Cube")
4. ob.setSize(2,2,2)
5. ob.RotY=1.57
```

Každý modul obsahuje další submoduly, jimiž mohou být například Lamp, Object, Camera, Mesh a podobně. Celý přehled submodulů a jejich parametry najdeme na adrese [4]. Pomocí druhého řádku v programu načteme submodul *Object*. Abychom s objektem mohli pracovat musíme ho načíst. To provedeme pomocí funkce *Get*, jak je vidět ve třetím řádku. Ve čtvrtém a pátém řádku manipulujeme s objektem, nejdříve jej zvětšíme ve všech směrech na dvě jednotky a následně otočíme o daný úhel. Hodnota je zapsána v radiánech.

4.2 Spouštění skriptu

Začneme tím, že spustíme Blender. Okno si rozdělíme svisle na dvě podokna. Můžeme k tomu použít i předem přednastavitelné obrazovky, jak lze vidět na obrázku 7 . Do pravé části pak píšeme skript. Ten lze spustit pomocí zkratky *ALT+P*.



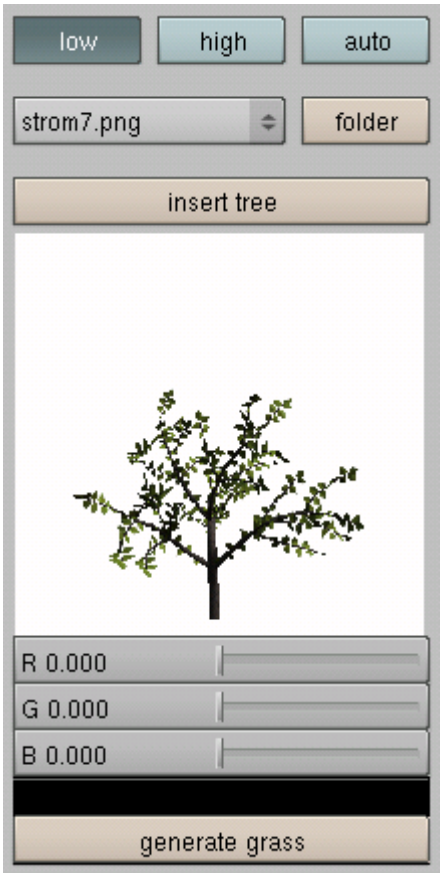
Obrázek 7: Výřez s ukázkou změny obrazovky a místem, kam se píše skript.

4.3 Návrh grafického prostředí

Funkcionalitu Blenderu jsme rozšířili o skript pomocí něhož lze umisťovat rostliny do krajiny. Skript má vlatní grafické prostředí (GUI), které bylo vygenerováno pomocí programu Blender-Python GUI, ten je k dostání na adrese [6]. V tomto programu lze navrhnout rozmístění potřebných komponent, ať už je to button, slider, menu nebo obrázek. Výhodou je, že se nemusí pracně vypočítávat umístění komponenty, ale vizuálně se umístí zvolený objekt na plochu. Tento způsob je velice rychlý a ušetří spousty času.

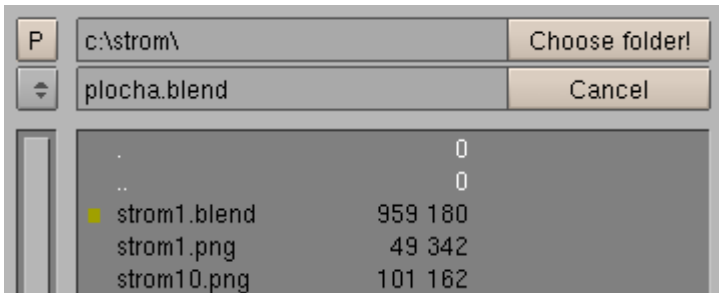
4.4 Popis tlačítek v grafickém prostředí

Grafické prostředí je vidět na obrázku 8. Program má několik funkcí. Lze si zde vybrat, zda-li má být krajina generována ve vysokém, nebo nízkém rozlišení. To se navolí zamáčknutím tlačítka *low* nebo *high*. Dále je tu zamačkávací tlačítko s názvem *auto*. *S*louží pro generování hustoty vegetace dle strmosti krajiny. Tedy pokud strmost krajiny odpovídá určitému úhlu, nebo jej překročí, stromy zde budou růst minimálně.



Obrázek 8: Frafické prostředí vlastního skriptu.

V přírodě je to dáno tím, že semínka stromů, až na některé druhy, se jen těžko uchytí pod velkým spádem. Po zmáčknutí tlačitka *folder*, se otevře *File Browser* (prohlížeč složek) obrázek 9.



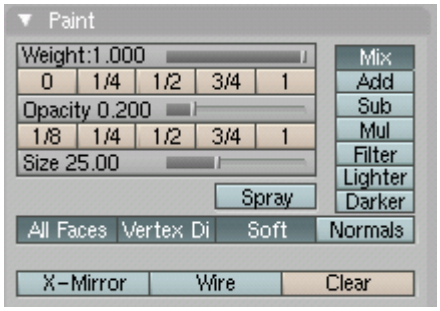
Obrázek 9: Výřez okna typu File Browser.

Zde si navolíme adresář, kde máme uloženy stromy. Po stisku tlačítka *Choose folder* se soubory načtou do rozbalovácího menu, ve kterém si následně vybereme porost pro tvorbu krajiny. Abychom měli představu o zvoleném porostu, náhled se zobrazí pod tímto

tlačítkem. Po zmáčknutí tlačítka *insert tree* se vybraný objekt naimportuje na plochu. Pokud bude zamáčknuté tlačítko *auto*, stromů bude několik, a to dle členitosti krajiny a velikosti celkové plochy. Tlačítka, která jsou umístěna pod obrázkem, slouží pro generování trávy. Pomocí slideru *R,G,B* si namícháme barvu trávy, která je okamžitě vidět pod touto trojicí tlačítek. Poslední tlačitko s popiskem *generate grass* je pro vložení trávy. Umístění trávy se opět vygeneruje pomocí módu váha vertexů.

4.5 Mód váha vertexu

Mód Weight Paint (váha vertexů) slouží pro udělování váhy vertexu. Váha vertexu se pohybuje v rozmezí hodnot od 0 do 1. Její přiřazena barva, která postupně přechází z modré do červené. Modrá barva zobrazuje nejnižší váhu vertexu, které odpovídá hodnota 0 a naopak červená barva je maximální hodnotou váhy, a to 1. Váhu vertexům lze přidělit pohybem myši po ploše. Ve vlastnostech, které jsou vidět na obrázku 10, si lze nastavit například velikost oblasti nebo maximální přidělenou váhu. Je zde i vlastnost *spray*, která při aktivaci způsobuje to, že se myš chová jako sprej a při delším držení levého tlačítka se váha mění od současné hodnoty do maximální. Pokud by toto tlačítko nebylo zamáčknuto, váha by se přidělovala skokově, a to při každém stisku levého tlačítka myši o nastavenou hodnotu. Všechny vlastnosti související s váhou vertexů jsou ilustrovány na obrázku 10.

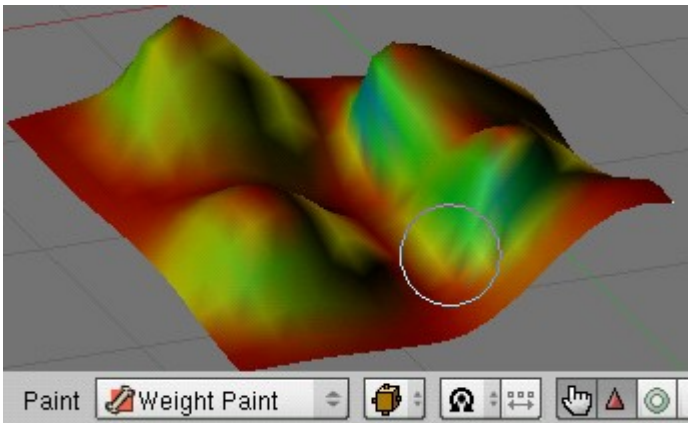


Obrázek 10: Vlastnosti módu weight paint.

4.5.1 Přiřazení váhy vertexů modelu krajiny

Na obrázku 11 je vidět model krajiny. Pomocí skriptu, kde bylo použito tlačítko *auto,* mu byla přiděla skupina vertexů s různou váhou. Jednotlivá váha vertexu se vypočítala ze souřadnice normály každého vertexu a dle stanovené podmínky mu byla přidělena hodnota. Pokud by takto přidělená váha vertexů nebyla vyhovující, můžeme si umístění a hustotu stromů následně upravit pomocí módu váha vertexů. Nebo v jiném případě

bychom neměli zamáčknuté tlačítko *auto* a v módu váha vertexu si určíme, kde stromy a s jakou hustotou porostou.



Obrázek 11: Model krajiny s váhou vertexu dle strmosti.

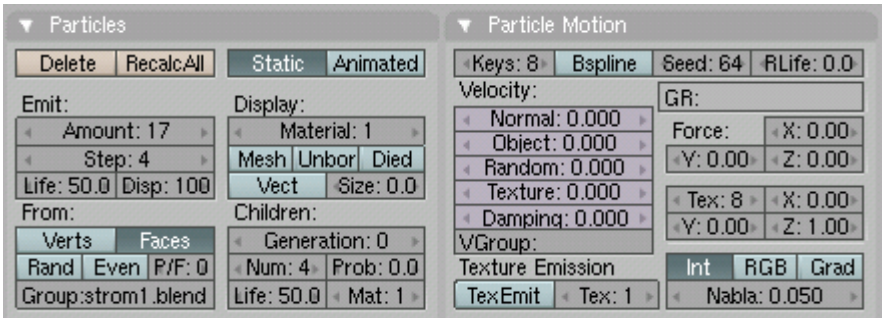
4.5.2 Částicový systém a váha vertexů

V teoretické části již byla zmínka o částicovém systému, proto se již nemusíme tímto modelovacím systémem zabývat obecně.

Pokud na objektu, který je na obrázku 11, vytvoříme statický částicový systém a nastavíme určité vlastnosti, dostaneme model krajiny s očekávanou vegetací. Obrázek 12 ilustruje nastavení částicového systému krajiny ve vlastním skriptu. Ke klíčovým vlastnostem patří *Amount.* Určuje celkový počet částic, v našem případě počet stromů, nebo jiného porostu. Dále je tu skupina tlačítek, která určuje, z čeho se budou částice generovat. Mohou to být vertexy (Verts) nebo plochy (Faces). Tyto dva emitory lze vybrat současně. Pokud se budou částice generovat z plochy, tlačítkem *Rand* se nastavuje náhodné generování. Tlačítkem *Even* se určuje závislost generování na velikosti celé plochy.

V mém experimentu nebyla tato tlačítka použita, protože tlačítko *Rand* sice generuje náhodné rozmístění částic, ale při vkládání dvou druhů porostu a zapnuté funkci *auto*, se náhodné rozmístění částic sice provedlo, jenže rozmístění bylo stejné jako u jiného druhu porostu. Důsledek to mělo ten, že dva různé porosty byly vygenerovány ze stejného místa. Proto náhodné rozmístění bylo vytvořeno pomocí tlačítka *Seed*, které má tu funkci, že lze náhodně posunout částici v tabulce hodnot, čímž je docílen efekt, že žádné dvě částice nejsou vystřelovány ze stejného místa.

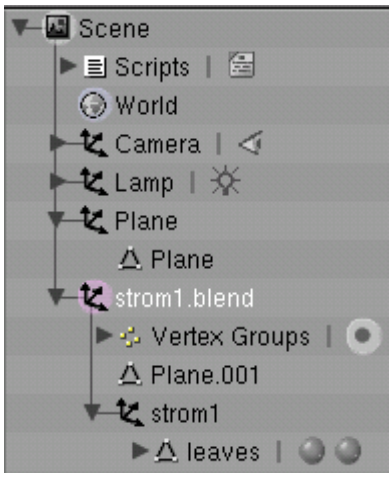
Do textového pole *Vgroup* je nutné napsat jméno skupiny vertexu, která byla vytvořena na ploše. Tím je zaručeno to, že částice budou jen na těch místech, kde byla přiřazena nebo automaticky vygenerována váha vertexů.



Obrázek 12: Nastavení částicového systému.

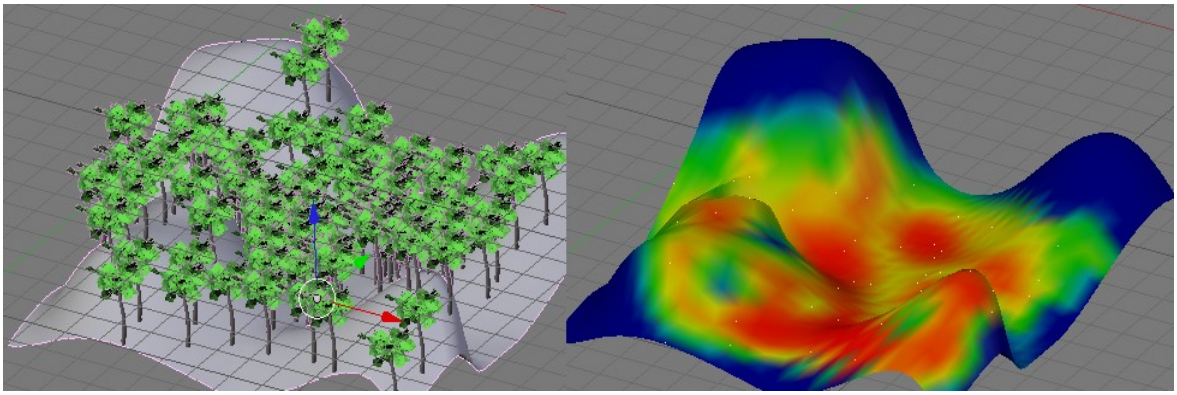
Možná by nebylo uplně pochopitelné, proč jsou dvě částice generovány ze stejného místa. Je to způsobeno tím, že celý skript funguje na následujícím principu:

Při vybrání porostu v grafickém prostředí a nasledném zmáčknutí tlačítka *insert tree*, se hlavní plocha zduplikuje a přiřadí se jí jméno stromu s příponou blend. Je to patrné na obrázku 13, který znázorňuje okno typu Outliner. Je zde vidět hierarchie objektů.

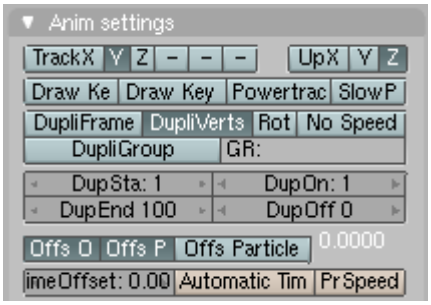


Obrázek 13: Okno typu Outliner zobrazující objekty ve scéně.

Dále se vybraný strom nastaví jako potomek zduplikované plochy. To způsobí, že na místě, kde se částice vyskytují, se zobrazí porost a nejen čístice reprezentovaná bílou tečkou jako je tomu na obrázku 14 vpravo . Aby se toto provedlo, je třeba ještě nastavit v panelu *object button* na záložce, která je vidět na obrázku 15, vlastnost *DupliVerts*.



Obrázek 14: Vlevo je krajina po aplikování tlačítka DupliVerts, vpravo statický částicový systém a zobrazení váhy vertextexů téže krajiny.



Obrázek 15: Nastavení Anim setting.

4.5.2.1 Funkce pro vytvoření a nastavení částicového sysému pomocí Python API

Nejprve je nutné časticový systém na objektu vytvořit pomocí příkazu *Blender.Effect.Particle.New( )*. Funkce má povinný parametr, a tím je jméno objektu. Dále do nějakého identifikátoru nastavit data částicového systému: *cast=Blender.Effect.Get('Objekt')*. Na jeden objekt je možné vytvořit několik částicových systémů, proto je třeba do dalšího identifikátoru načíst částicový systém, který se vytvořil jako první: *eff=cast[0]*. Nyní je možné měnit parametry pomocí identifikátoru *eff*.

Fuknce *eff.vGroup='Group'* nastaví skupinu vertexů, ve které se májí částice tvořit. Nastavení celkového počtu částic se provede funkcí *eff.setTotpart(cislo)*. Parametr v závorce je celkový počet částic. Většina vlastností se nastavuje pomocí tzv. Flagu (příznak). Například nastavení emitoru, konkrétně plochy, ze kterého se mají částice generovat se provede příkazem *eff.setFlag(Blender.Effect.Flags.FACES)*. Na flagy je aplikován logický součet, to znamená, pokud bychom chtěli nastavit další vlastnost pomocí flagu, je nutné to udělat takto:

eff.setFlag(Blender.Effect.Flags.STATIC | Blender.Effect.Flags.EVENDIST). Pokud bychom se pokusili nastavit funkce, které se ovládají pomocí flagů jednotlivě, zůstala by nastavena pouze poslední funkce ve skriptu, ostatní by se přepisovali vždy tou následující.

Všechny flagy a jejich popis pro nastavení částicového systému pomocí Python API najdeme v dokumentaci [8].

4.6 Low a High detail

4.6.1 High detail

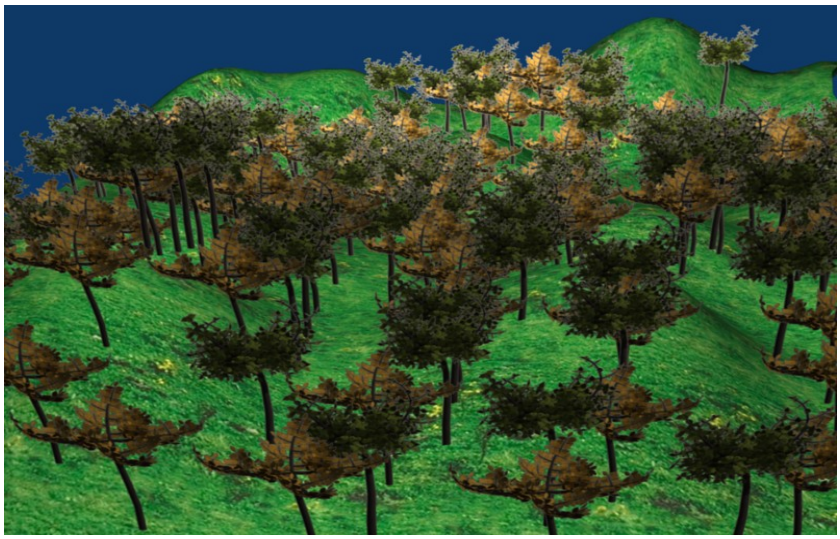
Pokud se bude generovat krajina, je možné si vybrat ze dvou druhů detailů. První je *High*. Ten zpočívá v tom, že použitý objekt – porost (strom, keř) je naimportovaný 3D objekt. Toto rozlišení je doporučeno používat, pokud je k dispozici výkonný počítač. Jinak program nebude plynule spolupracovat. Je to způsobeno tím, že importvaný objekt v *High* detailu je složitý komplexní objekt, který může mít i 10 000 vertexů. Krajina s 3D objekty je vidět na obrázku 16.

Obrázek 16: Vygenerovaná krajina se stromy v high detailu a vygenerovanou trávou.

4.6.2 Low detail

Generování krajiny v *Low* detailu má značně nižší nároky na hardware než v *High* detailu. Je to tím, že objekt je pouze textura namapovaná na plochu, která má pouze 4 vertexy. Je to tedy čtverec s texturou stromu nebo keře. Vygenerovaná krajina s použitím

nižšího detailu je vidět na obrázku 17.



Obrázek 17: Vygenerovaná krajina v low detailu

Aby byla docílena co největší realističnost, je nutné udělat několik zásadních kroků: Prvním je použít texturu s alfa kanálem. Alfa kanál je průhlednost textury. Alfa kanál způsobí to, že objekt při renderingu bude mít stejný tvar jako obrázek. Použitý obrázek s alfa kanálem lze načíst například z formátu *png*. Dalším způsobem, jak tohoto zobrazení docílit, je použitím dvou map. První bude textura v jakémkoliv formátu bez alfa kanálů. A druhým obrázkem je samotný alfa kanál k prvnímu obrázku. Výsledek bude totožný jako při použití textury s alfa kanálem. V našem skriptu byl použit první způsob. Druhá metoda texturování je použita v *High* detailu pro zobrazení listu. Další nutnou podmínkou k věrohodně vypadajícímu porostu je vytvoření vazeb (constraint) mezi kamerou a objektem. Je to proto, aby se plocha vždy při pohybu kamery otáčela společně s kamerou a směřovala k ní čelem, tedy celou texturou stromu.

4.6.2.1 Funkce pro vytvoření plochy pomocí Python API

V následující části si popíšeme vytvoření čtverce pomocí Python API. Tento čtverec je základem pro vkládání objektů v low detailu. Je to plocha, na kterou je namapována textura.

Nejprve si vytvoříme nový objekt typu mesh. Provede se to příkazem

me=NMesh.New( ), v závorce může být jeden parametr udávající jméno objektu. *Me* je identifikátor mesh objektu.

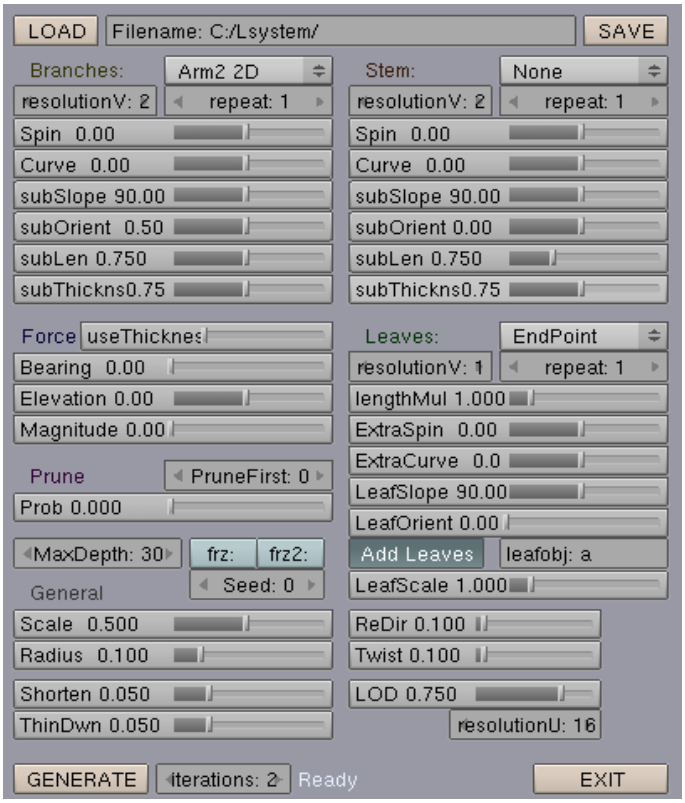
Dále vytvoříme čtyři vertexy. Vertexu přiřadíme identifikátor příkazem *v=NMesh.Vert(0,0,0), v* závorce jsou tři parametry, které určují lokální souřadnicový systém. Ve druhém kroku příslušný identifikátor reprezentující vertex přidáme do mesh objektu příkazem *me.verts.append(v).* Je nutné dodat, že vkládáni vertexů do mesh musí dodržet určitou posloupnost. Nejčastěji se vertexy vkládají po směru, nebo proti směru hodinových ručiček. Pokud bychom tento předpis nedodrželi, výsledná plocha by byla jen spletí hran a nikoli čtyřúhelník. Posloupnost vertexů také určuje směr normály.

Dále je nutné vytvořit plochu (face), což se provede příkazem *f=NMesh.Face( ).* Následuje přikaz *f.v.append(me.verts[x]) ,* parametrem je vertex, který jsme již vytvořili. *X* určuje pozici uloženého vertexu v seznamu. V našem případě to je hodnota nula až tři, protože seznam se indexuje právě od nuly. Příkazem *me.faces.append(f)* přidáme plochu do mesh objektu.

Na závěr přidáme vytvořený mesh objekt do scény. Provedeme to příkazem *NMesh.PutRaw(me, " Plocha").* První parametr je identifikátor mesh objektu a druhý je jméno objektu, textový řetězec.

4.7 Generování stromů pomocí skriptu L-systém

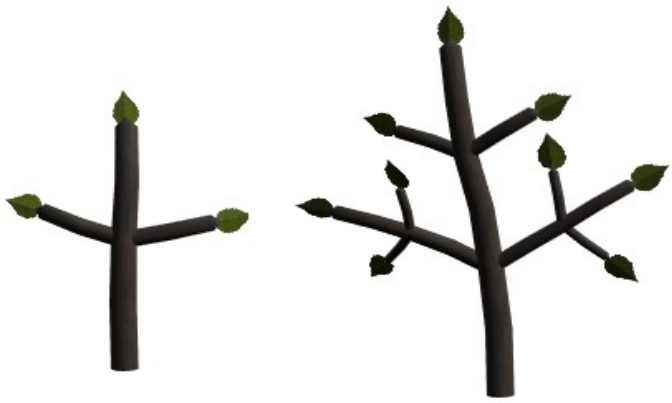
Stromy byly vygenerovány pomocí skriptu L-systém. Tento skript je možné bezplatně stáhnout na adrese [7]. Jeho grafické rozhraní je vidět na obrázku 18



Obrázek 18: L-system.

4.7.1 Popis funkce tlačítek

Ve spodní části obrázku je umístěno tlačítko *Generate*. To přepočítává hodnoty nastavené v celém skriptu a následně generuje strom. Vedle něho je umístěno tlačítko *iterations* a číselná hodnota, udávající počet interací. Jaký je rozdíl mezi interací 2 a 3 je vidět na obrázku 19. V horní části obrázku 18 jsou tlačítka *LOAD* a *SAVE*, sloužící pro načtení, případné uložení proměnných hodnot na disk do souboru podle zadání v textovém okně, které se nachází mezi nimi.Všechny parametry, které můžeme měnit, jsou seskupeny v odpovídajícím příslušnému bloku. Je to *Branches* (větve), *Stem* (kmen), *Force* (rozšířené vlastnosti), *Prune*(prořezávání), *Leaves*(listy) a konečně *General* (globální nastavení).



Obrázek 19: Generování stromu, vlevo je druhá interace, vpravo třetí.

4.7.1.1 Branches

Tlačítka *Branches* nastavují jednotlivé vlastnosti větví. V případě, že není aktivován *Stem*, platí toto nastavení i pro hlavní kmen. Další tlačítka mají následující význam:

ResolutionV - udává počet částí, ze kterých se každá větev skládá. Pokud bude hodnota rovna 1, je to obyčejný válec. Hodnota 4 bude znamenat to, že se větev skládá ze 4 válců.

Spin - je úhel ve stupních, udávající vzájemné natočení podstav válcovitých částí.

Curve - udává úhel ve stupních vzájemného natočení sousedních válcovitých částí.

subSlope -je úhel ve stupních udávající odklon větviček od nosné větve, standandardní nastavení je 90 stupnů.

SubOrient - nastavuje se v rozsahu 0 – 1, udává, v jaké míře má vetvička kopírovat tvar nosné větve.

SubLen - je poměr délky větvičky vzhledem k délce nosné větve.

SubThickns **-** udává poměr tloušťky větvičky vzhledem k tloušťce nosné větve.

4.7.1.2 Force

Skupina tlačítek s názvem *Force* má následující vlastnosti:

useThickness - udává možnost ovlivňování velikostí a tvaru větviček vzhledem k nosné větvi. Položky *Bearing, Elevation* a *Magnitude* určují směr a velikost větviček.

4.7.1.3 Prune

Pod skupinou tlačítek s názvem *Prune* se skrývají vlastnosti spojené s prořezáváním stromů.

PruneFirst obsahuje celočíselnou hodnotu. Tato hodnota určuje kolik pater spodních větví se nemá generovat.

Prob nabývá hodnoty 0 – 0,5. Hodnota 0 znamená, že se žádné větně nebudou prořezávát a naopak hodnota 0,5 udává, že téměř všechny větve budou prořezány.

4.7.1.4 Leaves

Volby spojené s nábídkou *Leaves* ovlivňují tvorbu listů nebo jehličí. První rozbalovací nabídka určuje počet a způsob, jakým budou rozmístěny listy na každé větvi. Popis dalších parametrů je následující:

repeat - udává počet opakujících se listů na jednom segmentu.

lengthMul – je délka segmentu větve, a tím se určuje hustota listů na větvi.

ExtraSpin - úhel ve stupních natočení listů vzhledem k ose segmentu větve.

ExtraCurve - úhel vzájemného natočení jednotlivých segmentů větví, kterým se udává i natočení listů, které se na větvi nacházejí.

LeafSlope - úhel listů ve stupních, které svírají vzhledek k ose segmentu větvě.

SubOrient - orientace listů ve stupních závisející na druhu listu.

leafobj - formolářové pole obsahuje jedno písmeno. Písmen je celkem 16 od *a* do *o*. Každé písmeno odpovídá jednomu druhu listu. Vedle je tlačítlo *Add Leaves*, při stisknutí se budou renderovat listy. Pokud stisknuto nebude, vyrenderuje se pouze kmen stromu.

LeafScale - určuje relativní velikost listu.

4.7.1.5 Generals

Skupina tlačítek s názvem *Generals* slouží pro celkové nastavení stromu.

Scale je měřítko pro celý strom.

Radius - udává hlavní poloměr kmenu.

Shorten - udává velikost segmentu, který vyrůstá z nosného kmenu.

ThinDwn udává, jak moc se větev ztenčuje.

reDir je míra faktoru pro orientaci větví z nosné větve.

twist - míra náhodného faktoru pro zakřivení větví.

resolutionU - udává celočíselnou hodnotu s počtem vrcholů na obvodu větví.

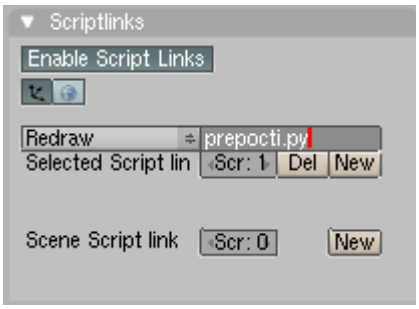
LOD - má vliv na snížení počtu bodů na obvodu větví. [12]

4.8 Části programu

Program se skládá ze dvou souborů. V prvním z nich jsou definovány funkce a grafické rozhraní. Soubor se jmenuje *hlavni.py.* Ve druhém souboru je uložený přepočet pro generování stromů. Soubor se jmenuje *prepocet.py.*

4.9 Volání Funkce přepočti

V Blenderu je možné vytvořený skript volat pomocí funkce *Render, Redraw, FrameChanged, OnSave, OnLoad.* Nejprve je nutné vytvořit vazbu (link) na volaný skript. Provede se to tím, že zmáčkneme v *Button menu* tlačítko *Script.* Objeví se nabídka, která je vidět na obrázku 20.



Obrázek 20: Nastavení skriptu na volání funkce Redraw.

Je nutné zaškrtnout tlačítko *Enable Script Links.* Na obrázku 20 je vidět, že pomocný skript pro přepočet objektů je volán funkcí *Redraw.* Jméno skriptu se napíše do prvního text boxu. Tímto způsobem je skript nalinkovaný a bude se provádět automaticky při volání funkce Redraw.

V našem případě jsme skript nalinkovali pomocí Python API takto:

Funkce *Text.Load(cesta*) nahraje soubor do okna typu *text editor,* má jeden parametr, a to cestu na soubor, kde je uložený skript. Dále pomocí indetifikátoru *scene* je nutné vybrat scénu, pro kterou se má skript použít: *scene = Scene.Get('Scene').* Parametr v závorce je jméno scény. Poslední funkce nalinkuje samotný skript. Prvním parametrem je jméno skriptu a druhým je událost, na kterou se bude volat. *scene.addScriptLink("prepocet.py", "Redraw")* .

4.9.1 Použití registrů

Každé volání funkce program značně zatěžuje, přepočet se provede po padesátem zavolání. Uchování proměnné o počtu volání skriptu se provedlo pomocí registrů.

Při spuštění skriptu si Blender vytváří vlastní knihovnu proměnných. Ta zaniká při zkončení programu. To je způsob, jak se vyhnout problémům s případnými jmennými kolizemi a garbage collectorem. Registry lze například použít pro uchovávání konfiguračních dat, které budou použity později, nebo také pro uchování hodnot při stisku tlačítek v grafickém prostředí.

4.9.1.1 Funkce pro použití registrů

Funkce pro uložení registrů je následující: *Blender.Registry.SetKey(key, dict, cache=False)* , má tři parametry. První udává jméno nových dat, nejčastěji se sem píše jméno skriptu, dalším parametrem je jméno knihovny, ve které jsou data uloženy, a konečně posledním je parametr udávající, zda-li se budou data ukládat na disk.

Pro opětovné použití dat lze použít funkci *Registry.GetKey(key, cached=False),* použití je podobné. Má dva parametry. Prvním je opět jméno skriptu, a druhý napovídá o tom, jestli se data mají hledat v souboru.

Úplný zdrojový kód pro práci s registry je použitý v souboru *prepocti.py*, který je uvedený na přiloženém CD.

Závěr

Na základě seznámení s fraktální geometrií byla rozšířena funkcionalita profesoinálního modelovacího a animačního programu Blenderu o skript, pomocí něhož lze generovat krajinu.

Pomocí částicového systému, což je silný modelovací nástroj, který má Blender standardně implementovaný, byl požadavek na vytvoření skriptu splněn.

Generování krajiny lze provést ve dvou rozlišení. Ve vysokém a nízkém. Vysoké rozlišení se generuje z 3D objektů vytořených pomocí skriptu L-systém a nízké z plošek, na které jsou namapovány textury stromů.

Skriptu navíc byla přidána funkce automatické generování stromů dle členitosti terénu. Přepočet se provedl pomocí normálových vertexů kde ze z-ové souřadnice vektoru se následně vypočítala váha vertexu, která určuje hustotu porostu.

Dále byl skript rozšířen o modelování trávy, která se vytváří rovněž pomocí částicového systému.

Během času stráveným nad bakalářkou prací se Blender stal mým novým koníčkem, proto budu na skriptu nadále pokračovat i po odevzdání bakalářské práce. V dubnu roku 2007 jsem se také účastnil Blender konference, která se konala v Praze na AVU.

Seznam použité literatury

[1] ŽÁRA, Jiří, BENEŠ, Bedřich, FELKEL, Petr. *Moderní počítačová grafika*. 1998. vyd. Martina Mojsezová. [s.l.] : Computer Press, c1998. s. 212-213.

[2] TIŠNOVSKÝ , Pavel. Http://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-ii/. *Fraktaly v počítačové grafice* [online]. 2005 [cit. 2007-04-10].

[3] TIŠNOVSKÝ , Pavel. http://www.root.cz/clanky/l-systemy-prirodni-objekty-i-umele-artefakty/. *Fraktaly v počítačové grafice* [online]. 2006 [cit. 2007-04-14].

[4] http://www.blender.org/documentation/243PythonDoc/API_intro-module.html

[5]http://www.blender.org/download/python-scripts/

[6]http://oregonstate.edu/~dennisa/Blender/BPG/

[7] http://jmsoler.free.fr/util/blenderfile/images/lsystem/lsystem.htm

[8] http://www.blender.org/documentation/243PythonDoc/Effect-module.html

[9] www.blender.org

[10] www.elysiun.com

[11]wiki.blender.org

[12]POKORNÝ , Pavel. *Stromy v Blenderu snadno a rychle* [online]. 2005 [cit. 2007-05-06]. Dostupný z WWW: <http://www.blender3d.cz/?q=lsystem_tutorial>.

[13] HARMS, Daryl, KENNETH, McDonald. *Začínáme programovat v jazyce Python*. Ivo Fořt, Lubomír Škápa. 2003. vyd. [s.l.] : Computer Press, a.s, c2003. 451 s. ISBN 80-7226-799-X.

[14] LUTZ, Mark, ASCHER, David. *Naučte se python : Pohotová příručka*. [s.l.] : GRADA Publishing, 2003. 360 s. ISBN 80-247-0367-X.

Seznam obrázků

Obrázek 1: Soběpodobný fraktál, list kapradiny.....11
Obrázek 2: křivka Helge von Kocha ve třech intracích.....14
Obrázek 3: Obrázek získaný dl-systémem.....17
Obrázek 4: Uspořádání dat v paměti.....24
Obrázek 5: Podrobný náhled datového bloku typu Mesh.....24
Obrázek 6: Základní rozdělení oken a popis objektů ve scéně.....26
Obrázek 7: Výřez s ukázkou změny obrazovky a místem, kam se píše skript.....28
Obrázek 8: Frafické prostředí vlastního skriptu.....29
Obrázek 9: Výřez okna typu File Browser.....29
Obrázek 10: Vlastnosti módu weight paint.....30
Obrázek 11: Model krajiny s váhou vertexu dle strmosti.....31
Obrázek 12: Nastavení částicového systému.....32
Obrázek 13: Okno typu Outliner zobrazující objekty ve scéně.....32
Obrázek 14: Vlevo je krajina po aplikování tlačítka DupliVerts, vpravo statický částicový systém a zobrazení váhy vertextexů téže krajiny.....33
Obrázek 15: Nastavení Anim setting.....33
Obrázek 16: Vygenerovaná krajina se stromy v high detailu a vygenerovanou trávou.....34
Obrázek 17: Vygenerovaná krajina v low detailu.....35
Obrázek 18: L-system.....37
Obrázek 19: Generování stromu, vlevo je druhá interace, vpravo třetí.....38
Obrázek 20: Nastavení skriptu na volání funkce Redraw.....40

Seznam tabulek

Tabulka 1: dimenze některých útvarů.....14

Seznam příloh

Příloha A: Obsah přiloženého CD

Příloha B: Vyrenderované obrázky

Příloha A: Obsah přiloženého CD

/BP/BP_Radek_Hatle2007.pdf

tento dokument ve formátu pdf

/STROMY

adresář se stromy ve formátu png a blend

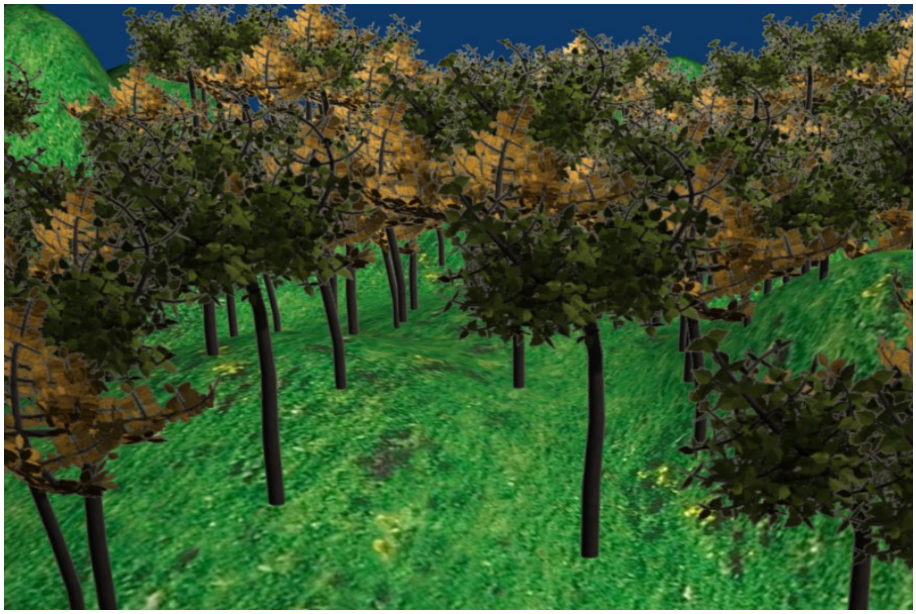
/SKRIPT

obsahuje program, který má dva soubory: *hlavni.py, prepocti.py*

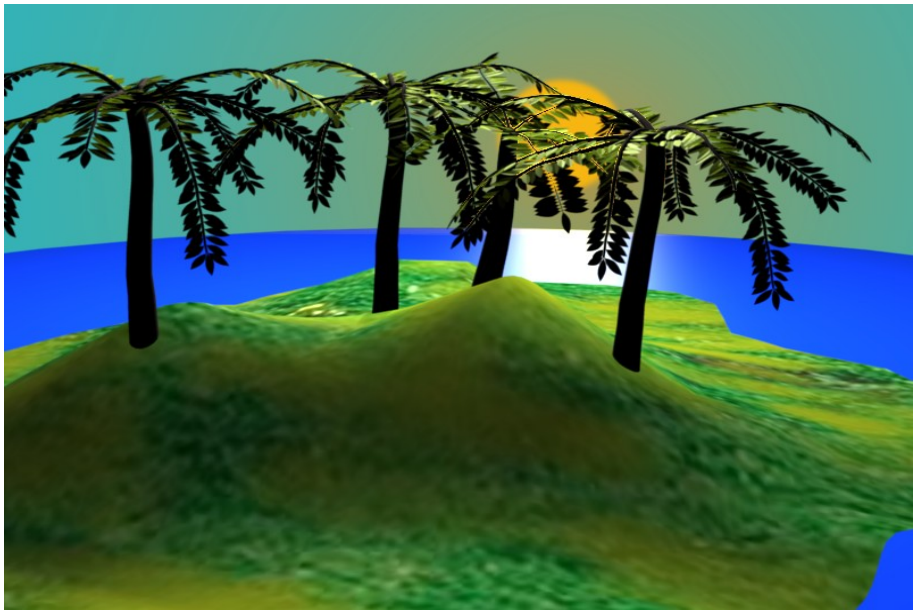
/BLENDER

obsahuje Blender ve verzi 2.43

Příloha B: Vyrenderované obrázky

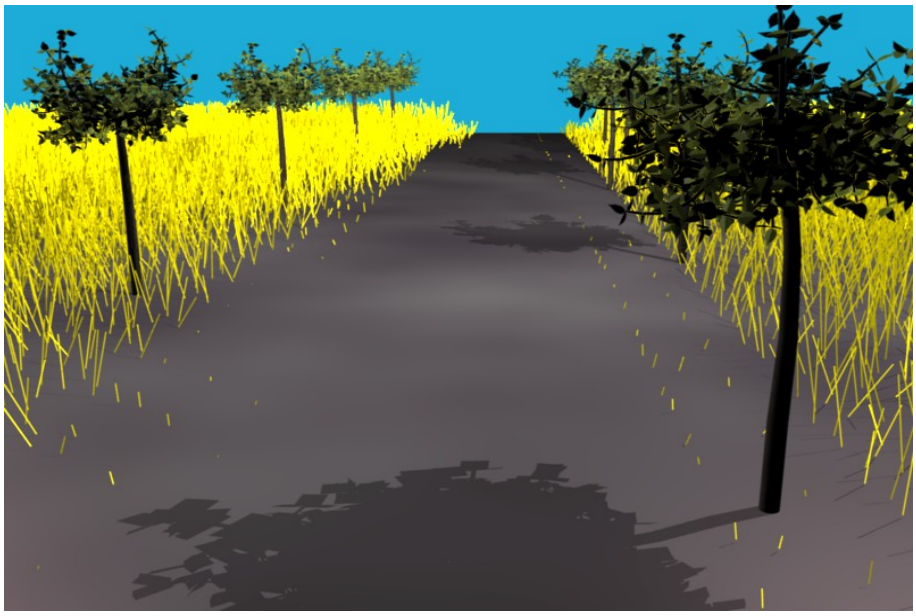


výsledek skriptu s nastavením LOW detailu



výsledek skriptu s nastavením HIGH detailu

výsledek skriptu s nastavením High detailu a vytvořením trávy



výsledek skriptu s nastavením High detailu a vytvořením trávy