Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Vizualizace L-systémů

Plzeň, 2012 Barbora Staffová

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 1. května 2012

Barbora Staffová

Abstract

The Bachelor thesis deals with visualisation of plants via L-Systems. In the first part individual types of L-Systems and programs for visualization of L-Systems are described.

The main task of the second part is to create a 2D and 3D projection program for plants represented by L-Systems. This is achieved through the use of knowledge gained in the first part of the thesis. The application is written in Java programming language with the usage of Java 3D graphic library. A simple Graphical User Interface (GUI) for user interaction is created.

Obsah

1 Úvod

Problém popsat přesně a věrně nejrůznější geometrické tvary vyskytující se v přírodě provází vědu celou historií, kdy se lidé začali zajímat o přírodní jevy. Rozvoj vědy umožnil poskytnout vědcům adekvátní nástroje pro realistickou reprezentaci objektů vyskytujících se v přírodě až s rozvojem matematiky, informatiky a biologie. V roce 1968 představil maďarský biolog Aristid Lindenmayer formální gramatiky pro popis jednoduchých mnohobuněčných organismů. Při řešení tohoto problému vycházel z výsledků, které byly v tuto dobu známy v oblasti teorie gramatik a automatů. Dnes se nazývají výše uvedené gramatiky Lindenmayerovými systémy, zkráceně L-systémy. Spolu s kolegy začal Lindenmayer své poznatky sepisovat do knihy *The Algorithmic Beauty of Plants* [PL90]. Kniha byla vydána až po Lindenemayerově smrti Przemysławem Prusinkiewiczem, který je dnes považován v oblasti modelování rostlin pomocí gramatik za jednu z největších osobností.

L-systémy se uplatňují při simulaci rostlin, ale dají se používat i při modelování deterministických fraktálů, mušlí, toků řek a jiných útvarů nacházejících se v přírodě. Při realizaci konkrétních úkolů, ve kterých se používají L-systémy, se musí vycházet na jedné straně z teoretických výsledků a požadavků a na druhé straně z nástrojů a prostředků, které jsou k dispozici.

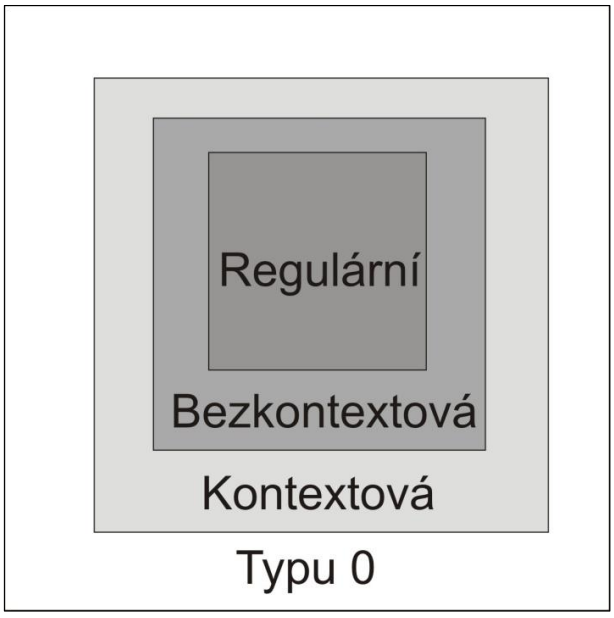
Úkolem této práce bylo seznámení se L-systémy a s nástroji pro jejich vizualizaci. V návaznosti na to navrhnout program pro 2D i 3D vizualizaci rostlin reprezentovaných L-systémy. Pro realizaci daných úkolů byl zvolen programovací jazyk Java. Výsledný program by měl umožňovat zobrazování rozmanitých typů rostlin tak, jak to umožňují ve své podstatě L-systémy.

2 Gramatiky

Americký lingvista Noam Chomsky v roce 1956 vytvořil formální gramatiky jako teoretický nástroj pro popis přirozených jazyků. Ve svých teoretických pracích vycházel z výsledků bádání lingvisty Z. Harrrise, který používal matematické nástroje k popisu jazykových transformací. Chomského práce byla posléze využita nejen pro přirozené jazyky. Dnes se formální gramatiky a s tím spojená teorie automatů využívají v oblasti teoretické informatiky.

2.1 Chomského hierarchie

Chomsky zavedl klasifikaci gramatik a jazyků tak, že gramatiky rozdělil do čtyř tříd [KŠCH89], viz Obrázek 1. Název hierarchie se používá proto, že třída jazyků s nižším označením obsahuje jako vlastní podtřídu třídu s vyšším označením. Hierarchie zachycuje formální složitost jednotlivých typů gramatik. Od nejobecnějšího typu 0 až ke gramatice typu 3.



Obrázek 1: Chomského hierarchie tříd jazyků.

2.1.1 Gramatika typu 0

Gramatika typu 0 je uspořádaná čtveřice $G = \langle N, T, P, S \rangle$, kde N je množina neterminálních symbolů. T je abeceda terminálů taková, že $N \cap T = \emptyset$, P je množina přepisovacích pravidel ve tvaru

$$P \subseteq (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$$

a S je počáteční symbol, kde $S \in N$. Přepisovací pravidla se obvykle zapisují ve tvaru $\alpha \rightarrow \beta$.

V celé hierarchii zaujímá nejobecnější místo. Na přepisovací pravidla nejsou v podstatě kladena žádná omezující kritéria, kromě těch, která vyplývají z postavení jednotlivých členů čtveřice G. Zahrnuje v sobě všechny ostatní třídy Chomského hierarchie.

2.1.2 Gramatika typu 1 – kontextová gramatika

Kontextová gramatika je uspořádaná čtveřice $G = \langle N, T, P, S \rangle$ a obsahuje přepisovací pravidla z P ve tvaru $\alpha X \beta \rightarrow \alpha \gamma \beta$, kde $X \in N$ a $\alpha, \beta, \gamma \in (N \cup T)^*$ a současně $\gamma \neq \varepsilon$, ε označuje prázdný řetězec.

Jak naznačuje sám název - kontextová gramatika, jedná se o gramatiku, která je již usměrňována určitými pravidly a není tak obecná jako gramatika typu 0. Přepisování se realizuje v závislosti na kontextu, ve kterém se nachází příslušný neterminální symbol. Jednodušeji řečeno, neterminální symbol se přepíše, pokud před a za ním se vyskytují určité řetězce nebo řetězce generované určitým způsobem.

2.1.3 Gramatika typu 2 – bezkontextová gramatika

Bezkontextová gramatika je uspořádaná čtveřice $G = \langle N, T, P, S \rangle$ a obsahuje přepisovací pravidla z P ve tvaru $X \rightarrow \gamma$, kde $X \in N$ a $\gamma \in (N \cup T)^*$.

Na rozdíl od kontextové gramatiky, u bezkontextové dochází k přepisu neterminálního symbolu bez ohledu na to, v jakém kontextu se daný neterminální symbol právě vyskytuje. To znamená, že levá strana přepisovacího pravidla obsahuje vždy pouze jeden neterminální symbol.

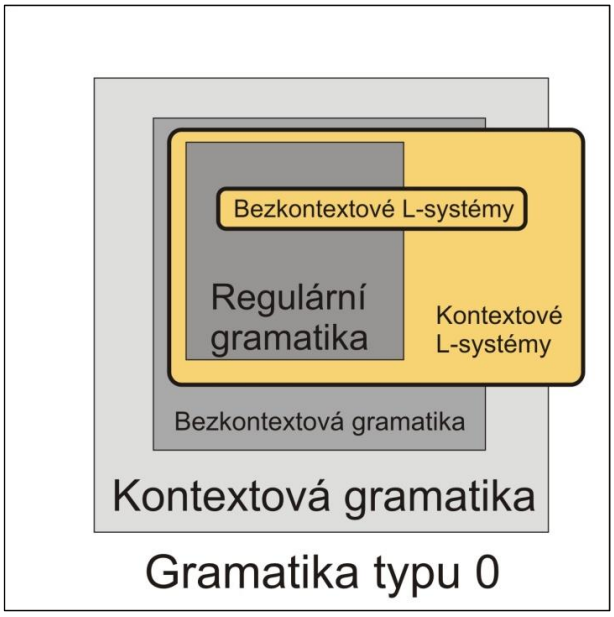
2.1.4 Gramatika typu 3 – regulární gramatika

Regulární gramatika je uspořádaná čtveřice $G = \langle N, T, P, S \rangle$ a obsahuje přepisovací pravidla z P buď ve tvaru $X \rightarrow wY$, kde $X, Y \in N$ a $w \in T^*$ nebo $X \rightarrow w$, kde $X \in N$ a $w \in T^*$.

Podobně jako bezkontextová gramatika, tak i regulární gramatika má na levé straně přepisovacího pravidla pouze jeden neterminální symbol. Na rozdíl od ní však regulární gramatika má na pravé straně omezení. Pravou stranu může tvořit buď tvar wY (resp. Yw), nebo pouze w, kde w je řetězec složený z terminálních symbolů a Y je neterminální symbol. Podle umístění neterminálního symbolu se rozlišuje pravostranná wY (resp. levostranná Yw) lineární gramatika.

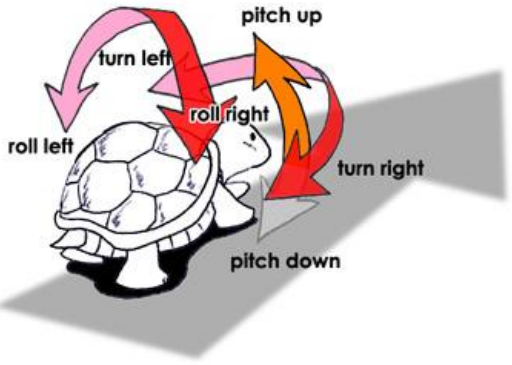
3 L-systémy

L-systémy navazují svým pojetím na definici formálních gramatik, které byly popsány v předchozí kapitole, viz Obrázek 2. Podstatou mechanismů, podle nichž se L-systémy chovají, je paralelní přepisování řetězců podle určitých pravidel. Každý symbol má geometrickou interpretaci, která v celkovém výsledku vytváří geometrický objekt (dvoj nebo trojrozměrný).



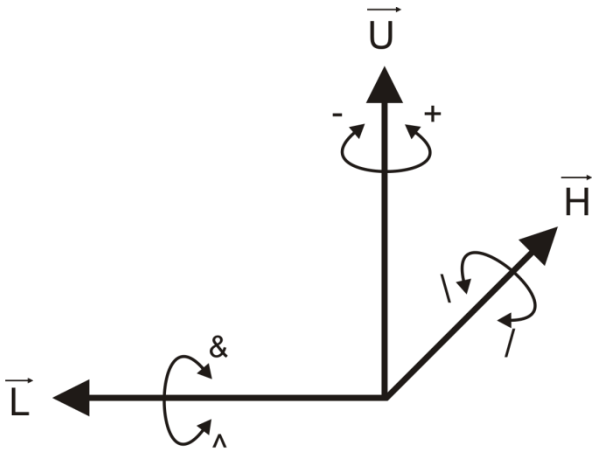
Obrázek 2: Vztahy mezi Chomského gramatikami a třídy jazyků generovaných L-systémy.

Protože je důležitá geometrická interpretace, bylo třeba zavést reprezentaci kreslícího zařízení. K tomu byla použita tzv. želví grafika (*turtle graphics*) [VSV08], viz Obrázek 3. Samotná pomyslná želva je vždy určena svojí polohou a orientací.



Obrázek 3: Želva cestující v trojrozměrném prostoru [VK06].

Podle popisu akce se želva přemisťuje v prostoru (2D nebo 3D). Ve dvourozměrném prostoru je stav želvy reprezentován trojicí (x, y, alfa), kde x, y jsou souřadnice želvy a alfa (*heading*) je úhel orientace pohledu želvy. V trojrozměrném prostoru, viz Obrázek 4, je okamžitá orientace želvy reprezentovaná třemi navzájem kolmými jednotkovými vektory. Tyto vektory jsou $\vec{H}, \vec{L}, \vec{U}$.



Obrázek 4: Orientace želvy v trojrozměrném prostoru.

$\vec{H}$ (heading)	směr hlavičky, tzn. směr pohybu dopředu
$\vec{L}$ (left)	směr levých nožiček
$\vec{U}$ (up)	směr krunýře (směr nahoru vzhledem k želvě)

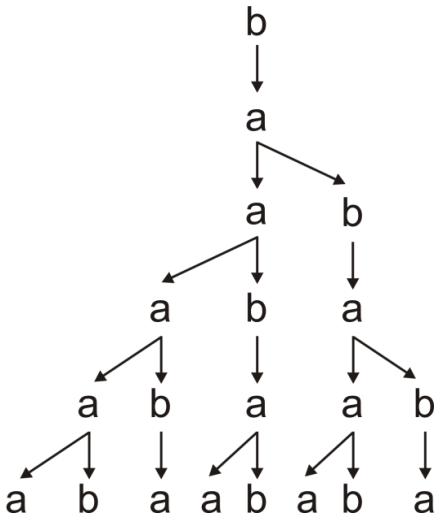
Dle složitosti a charakteristiky pravidel dělíme L-systémy do několika skupin, které budou popsány blíže v následujícím textu.

3.1 D0L-systémy

D0L-systém je deterministický bezkontextový L-systém. Je to uspořádaná trojice $G = \langle V, P, \omega \rangle$, kde V je konečná abeceda symbolů, P je konečná množina pravidel a ω, neboli axiom, je neprázdná posloupnost symbolů, pro kterou platí $\omega \in V^+$. D0L-systém je deterministický, proto nesmí mít dvě a více pravidel stejnou levou stranou. Bezkontextovost spočívá v tom, že u přepisovacích pravidel je na levé straně pouze jeden symbol, nikoli řetězec. Na pravé straně může být jeden symbol i řetězec [PL90].

$$A \rightarrow B, A \in V, B \in V^*$$

Například pravidlo $a \rightarrow ab$ znamená, že písmeno a je nahrazeno řetězcem ab, a pravidlo $b \rightarrow a$ znamená nahrazení písmena b písmenem a. Přepisovat se začíná řetězec, který je nazýván axiom. Pokud bude pro výše uvedená pravidla axiom b, bude při prvním kroku nahrazen písmenem a. V dalším kroku vznikne ab, poté aba, atd. Postup je graficky znázorněn, viz Obrázek 5. D0L-systémy umožňují zobrazovat pouze spojitou čáru bez větvení. Pro popis pohybu želvy se v přepisovacích pravidlech používají základní symboly.



Obrázek 5: Příklad odvozování řetězce D0Lsystému.

F	posun želvy dopředu o krok d, stav želvy se změní $(x + d\cos\alpha, y + d\sin\alpha, \alpha)$ a současně se namaluje úsečka
f	posun želvy dopředu o krok d bez kreslení úsečky
$+$	otočení o úhel δ v protisměru hodinových ručiček, následující pozice želvy je $(x, y, \alpha + \delta)$
$-$	otočení o úhel δ ve směru hodinových ručiček, následující pozice želvy je $(x, y, \alpha - \delta)$

Tyto čtyři základní symboly jsou schopny generovat např. Kochovu křivku, viz následující předpis a Obrázek 6.

$\omega : F$
$P : F \rightarrow F + F - -F + F$
a úhel $\alpha = 60°$

Obrázek 6: Kochova křivka.

Pro trojrozměrné modelování se zavádějí další symboly.

& rotace dolů podle vektoru $\vec{L}$

^ rotace nahoru podle vektoru $\vec{L}$

/ rotace doleva podle vektoru $\vec{H}$

\ rotace doprava podle vektoru $\vec{H}$

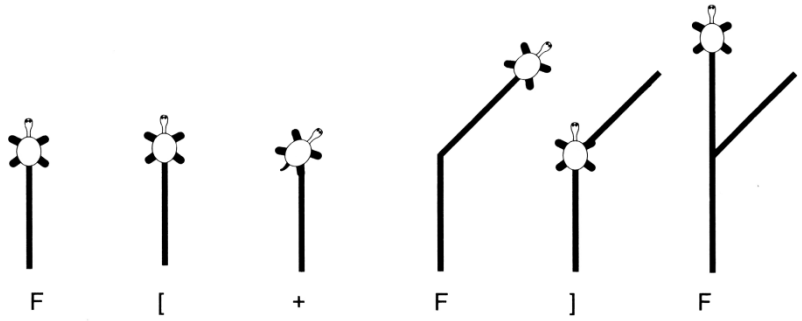
3.2 Závorkové L-systémy

Závorkové L-systémy umožňují generovat rozvětvené objekty, jako jsou stromy a keře. Pro větvící se struktury byla želva doplněna o zásobník, který funguje jako paměť želvy. Do zásobníku si želva může uložit svoji aktuální pozici nebo uloženou pozici vybrat a nastavit se podle ní. Do abecedy V byly přidány dva symboly pro zásobník [ŽBS04].

[uložení stavu do zásobníku

] vybrání posledního uloženého stavu z vrcholu zásobníku a aktualizace polohy želvy

Obrázek 7 znázorňuje interpretaci řetězce F[+F]F, kdy se želva nejprve posune dopředu, zapamatuje si aktuální stav, otočí se doprava a následně se posune dopředu. Pak se vrátí do uložené pozice vybrané ze zásobníku a pokračuje v původním směru.



Obrázek 7: Implementace řetězce F[+F]F prostřednictvím želví grafiky [ŽBS04].

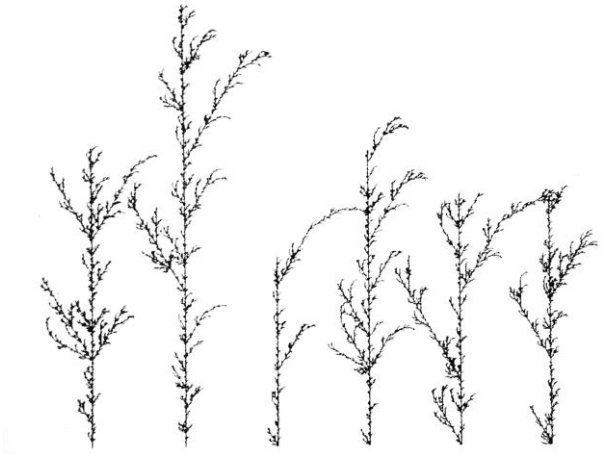
3.3 Stochastické L-systémy

Stochastické L-systémy patří společně s kontextovými a parametrickými do skupiny tzv. otevřených L-systémů [MP96]. Tyto nedeterministické kontextové parametrické L-systémy byly vyvinuty pro interakci simulace růstu rostlin a jejich prostředí. Otevřené L-systémy informují o detekcích kolizí větví rostlin, o množství dopadajícího světla nebo popisují soutěž mezi rostlinami o životní prostor. Zachycují i interakci mezi kořeny a živinami, jako jsou např. voda a sůl v půdě.

Jednotlivé rostliny stejného druhu nejsou ve skutečném světě vždy identické. Často se od sebe liší např. tvarem, ale přitom stále odpovídají charakteristikám svého druhu. Proto se zavedly stochastické L-systémy, které používají ve svých pravidlech určitou náhodnost. Umožňují vytvářet rozvětvené rostliny, které se od sebe liší, i když mají stejný předpis.

Stochastický L-systém je definován jako čtveřice $G = \langle V, \omega, P, \pi \rangle$, kde $\pi : P \rightarrow (0,1]$ nazývána *probability distribution* určuje, s jakou pravděpodobností se přepíše symbol daným pravidlem. Součet pravděpodobností jednotlivých pravidel se stejnou levou stranou musí být roven jedné. Pro názornost je uveden příklad z [PL90], viz Obrázek 8.

$\omega : F$
$p_1 : F \rightarrow F[+F]F[-F]F \quad : 0{,}33$
$p_2 : F \rightarrow F[+F]F \quad : 0{,}33$
$p_3 : F \rightarrow F[-F]F \quad : 0{,}34$



Obrázek 8: Stochastické L-systémy [PL90].

3.4 Kontextové L-systémy

U těchto L-systémů jsou pravidla pro přepisování symbolů rozšířena o kontext. Umožňuje se tím napodobit mechanismus, pomocí kterého si rostliny regulují svůj vývoj. Jednotlivé sousedící buňky rostliny si pak pomocí živin dokážou vyměňovat informace např. o množství světla nebo ročním období.

Symboly se přepisují v závislosti na svém okolí. Okolím se myslí řetězce symbolů vlevo a vpravo od přepisovaného symbolu (obdoba kontextových gramatik, jak jsou známy z teorie formálních jazyků). Mohou nastat dva případy [PL90]. První označován jako 2L-systém bere při přepisování v úvahu jak řetězce (kontext) z levé strany tak z pravé. Druhým je 1L-systém, který má kontext pouze z jedné strany. 2L-systémy jsou obecnější a zahrnují v sobě i 1L-systémy. Formálně vypadají pravidla kontextového 2L-systému následovně

$$a_l < a > a_r \rightarrow x$$

$a_l \in V^*$ je levý kontext, $a_r \in V^*$ je pravý kontext, $a \in V$ je přepisovaný symbol (předchůdce) a $x \in V^*$ je pravá strana přepisovacího pravidla.

3.5 Parametrické L-systémy

Parametrický L-systém pracuje s moduly (písmena abecedy rozšířena o parametry). Posloupnost modulů pak tvoří tzv. parametrická slova. Parametry ovlivňují velikost a orientaci želvího kroku v průběhu generování, tzn., že želva může svůj krok

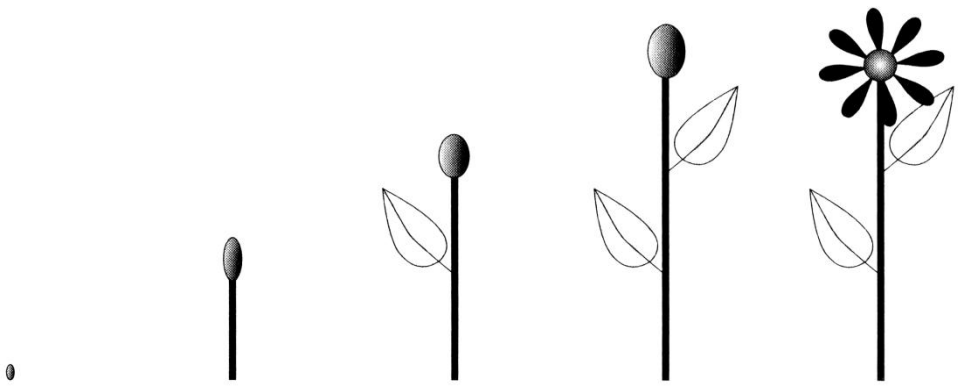
zmenšovat nebo zvětšovat, případně měnit velikost úhlu pohybu. To například umožňuje vykreslovat zmenšující se fraktály pomocí L-systémů. Parametrické L-systémy jsou důležité pro modelování růstu rostliny v čase, viz Obrázek 9.

Parametrický L-systém je definován jako čtveřice $G = \langle V, \Sigma, \omega, P \rangle$, kde V je abeceda, Σ je množina formálních parametrů, P je konečná množina pravidel a $\omega \in (V \times R^*)^+$ je neprázdné parametrické slovo, axiom. Symboly : a → rozdělují pravidlo na tři části: předchůdce (*predecessor*), podmínku (*condition*) a následníka (*successor*). Pravidlo parametrických L-systémů má poté tvar

$$id : \ levý\ kontext < pred > pravý\ kontext : cond \rightarrow succ$$

id je číslo pravidla, cond (podmínka) je logický výraz, který je pravdivý nebo nepravdivý, a succ je pravá strana pravidla [MP96].

$$\begin{aligned}
&\omega \ : A(0) \\
&p_1 : A(x) && : x == 0 \rightarrow FA(x+1) \\
&p_2 : F < A(x) && : x == 1 \rightarrow [-L]FA(x+1) \\
&p_3 : F < A(x) && : x == 2 \rightarrow [+L]FA(x+1) \\
&p_4 : A(x) && : x == 3 \rightarrow B
\end{aligned}$$



Obrázek 9: Znázornění růstu rostliny pomocí parametrických L-systémů [ŽBS04].

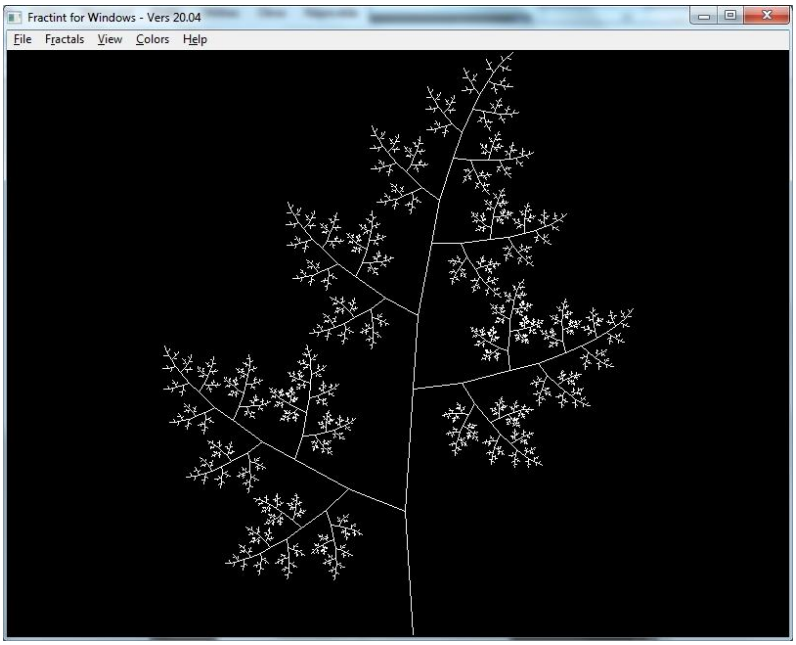
4 Programy pro vizualizaci L-systémů

Pro zobrazování L-systémů bylo vytvořeno mnoho různých aplikací. Některé jsou placené nebo jsou distribuované jako moduly do komerčních programů. Současně jsou ale také k dispozici volně šiřitelné programy, které jsou na kvalitní úrovni. Vybrala jsem několik zástupců, ze kterých jsem se inspirovala pro svou následující práci.

4.1 Fractint

Fractint [FractInt] je oblíbený program pro generování fraktálů. Byl vyvinut pro operační systém DOS, dnes jsou i verze pro systémy Windows a Linux, viz Obrázek 10.

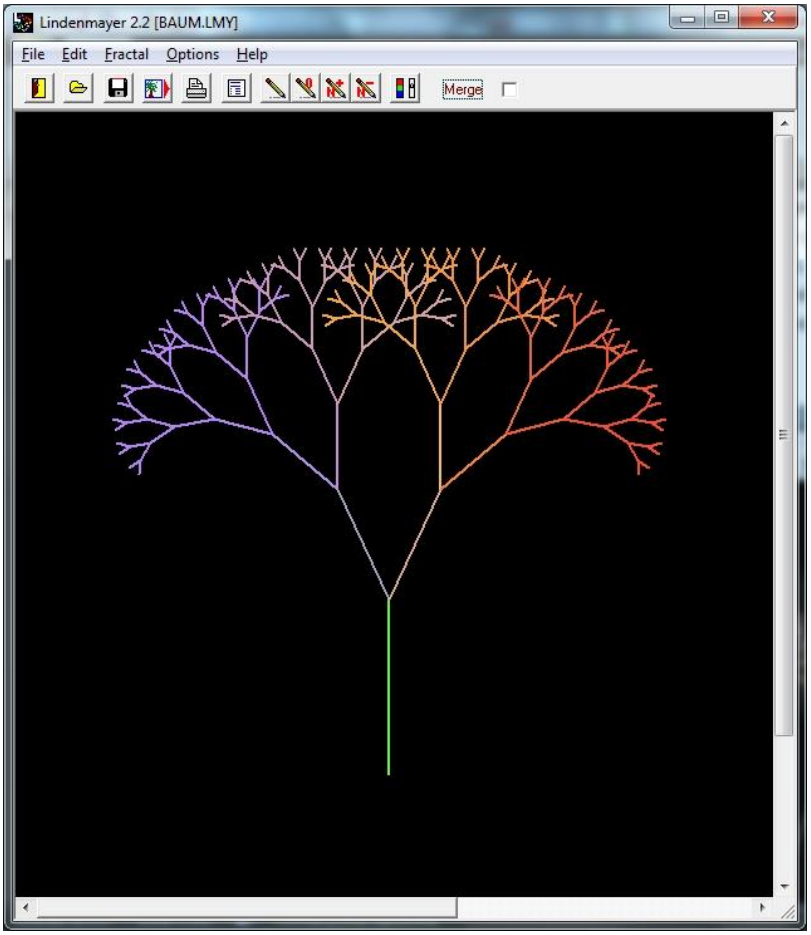
Fracint dovoluje zobrazovat různé geometrické útvary, jako jsou fraktály nebo L-systémy. Objekty mohou být zobrazeny v různých barevných variantách, ale pouze ve dvourozměrném prostoru. Fractint umožňuje zobrazit základní D0L-systémy. Velkou nevýhodou u zobrazení L-systémů je nepraktické načítání vstupních souborů s gramatikou, které jsou jinak velmi přehledné a srozumitelné. Jednotlivá pravidla pro L-systémy se totiž nedají zadat do programu manuálně, ale musí se pokaždé složitě načítat z externího souboru. Výsledný obrázek lze uložit do souboru BMP nebo GIF.



Obrázek 10: List vytvořen pomocí programu Fractint.

4.2 L-Mayer2

Lindenmayer-Fractals [Lmayer] je německý program napsaný v jazyce Delphi 4.0, viz Obrázek 11. Na domovských stránkách programu je k dispozici zjednodušený applet. Je to opět velmi jednoduchý program pro kreslení dvourozměrných L-systémů. Výhodou v této aplikaci oproti Fractint je možnost manuálního zadávání přepisovacích pravidel a symbolů. Velmi jednoduše se dají nastavit i základní vlastnosti objektu, jako je délka kroku, úhel nebo úroveň. Podporuje i načítání již dříve vytvořených gramatik a také umožňuje export do BMP nebo JPG.



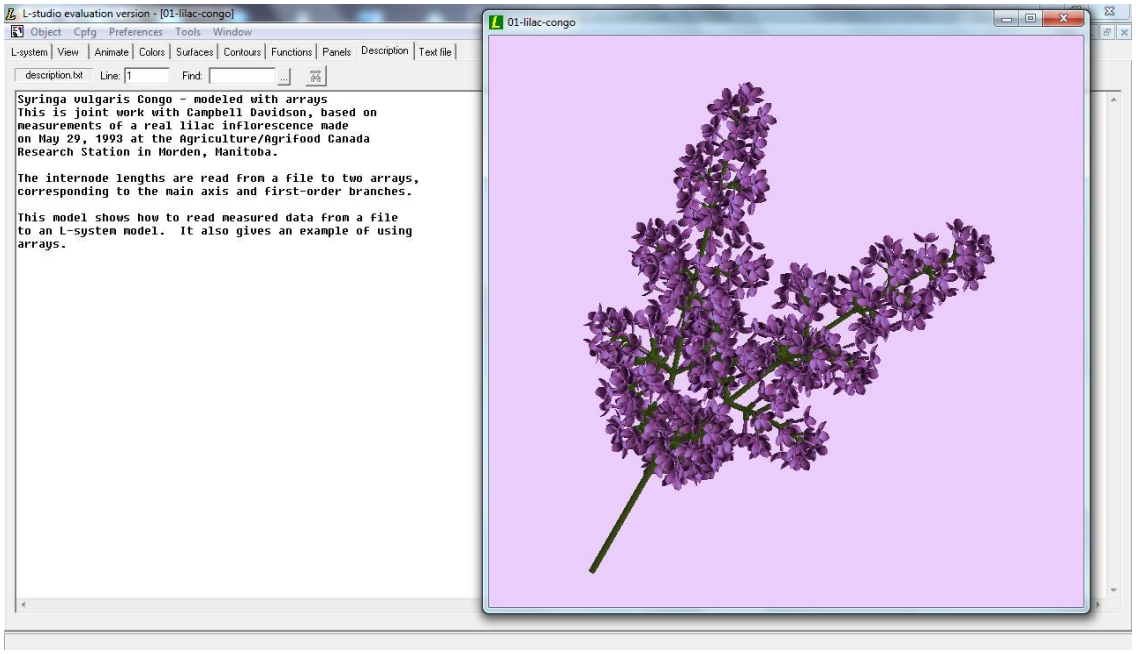
Obrázek 11: Ukázka uživatelského rozhraní programu L-Mayer2.

4.3 L-Studio

Program L-Studio [LStudio] patří mezi nejlepší programy pro tvorbu L-systémů. Jedná se o komerční aplikaci, která je ale k dispozici jako demo verze. Po uplynutí třiceti dnů

je však omezeno tisknutí obrázků. Demoverzi lze stáhnout po bezplatné registraci na domovských stránkách. L-Studio (Virtual Laboratory pro Linux) byl vyvinut na univerzitě v Calgary profesorem Przemysławem Prusinkiewiczem a jeho týmem.

Aplikace je velmi kvalitní, umožňuje zobrazovat všechny výše popsané L-systémy a disponuje mnoha dalšími užitečnými rozšířeními, viz Obrázek 12. Dokáže zobrazovat modely rostlin v dvourozměrné i trojrozměrné formě. Jednotlivé rostliny si může uživatel pomocí myši natočit do požadované pozice a případně spustit animaci růstu dané rostliny, pokud je u zvoleného modelu podporována. Je tedy možno sledovat jednotlivé fáze rostliny, kdy např. postupně ze stonku vyrostou listy a pupeny, z nichž se později vytvoří květy. L-Studio dokáže dokonce nasimulovat přírodní jevy, jako je vítr, gravitace, slunce nebo brouka, který konzumuje rostlinu. Výsledek je možno exportovat do formátu BMP, TGA, PostScript nebo OBJ. Základní příslušenství programu obsahuje mnoho ukázkových L-systémů, doplněné o animace nebo kvalitní rendering. L-Studio je nejkvalitnější program pro tvorbu L-systémů, se kterým jsem se při této práci setkala.

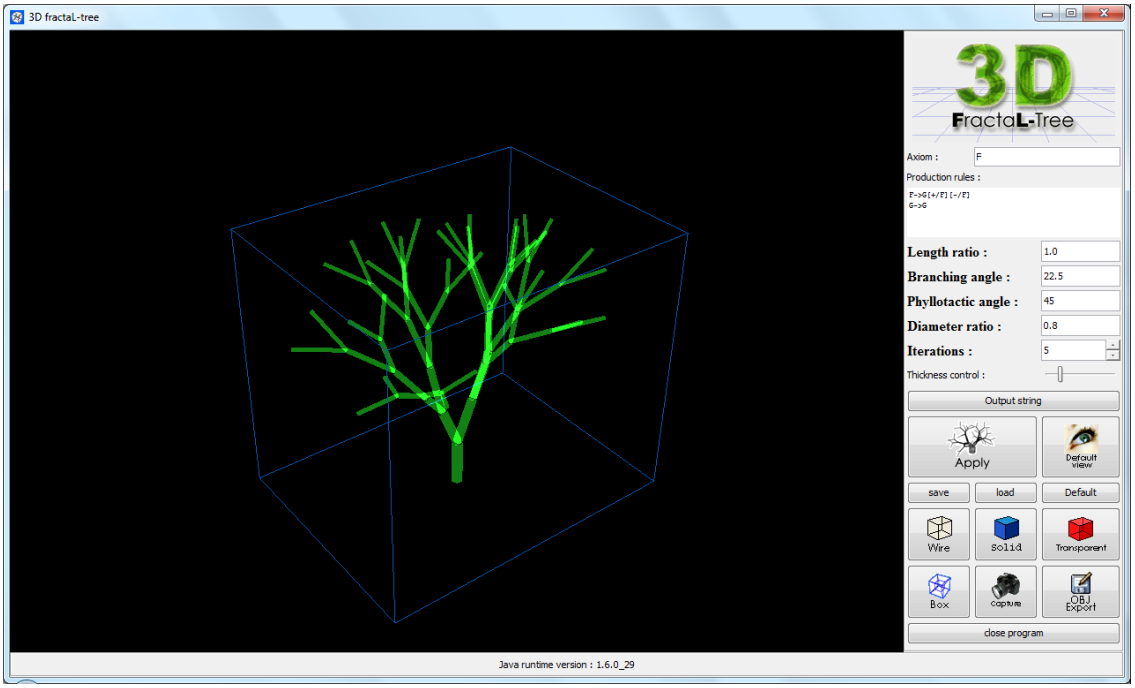


Obrázek 12: Ukázka uživatelského rozhraní programu L-Studio.

4.4 3D FractaL-Tree

Jednoduchý program, viz Obrázek 13, 3D FractaL-Tree [3DFract] používá pro tvorbu L-systémů knihovnu Java3D. K dispozici je v různých verzích po bezplatné registraci.

FractaL-Tree je rendrovací a vizualizační nástroj. Podporuje libovolné bezkontextové L-systémy. Výhodou je jednoduchost v nastavování jednotlivých parametrů vykreslování, jako je například délka kroku nebo úhel. Program také umožňuje otáčet, přibližovat a posunovat výsledný objekt pomocí myši. Další výhodou je, že dovoluje uložit výsledný obrázek ve formátu JPEG a exportovat 3D objekty pro ostatní 3D softwary ve formátu OBJ.



Obrázek 13: Ukázka vygenerovaného L-systému programem 3D FractaL-Tree.

4.5 Specifikace funkčnosti aplikace

Jedním z úkolů této bakalářské práce tak, jak je to uvedeno v jejím zadání, bylo určit množinu funkcí potřebných pro detailní vizualizaci gramatik a přitom vycházet z dříve prostudovaných programů, které se touto problematikou zabývají. Z této skutečnosti také vyplýval výběr funkcí, které by bylo vhodné do programu implementovat. Některé

z funkcí, které přicházely v úvahu, byly společné pro více aplikací, některé byly specifické pouze pro jednu.

Univerzální požadavkem každého programu pro vizualizaci rostlin prostřednictvím L-systémů je zadávání vstupních parametrů. Ve Fractintu se vstupní pravidla musí nejprve zapsat do textového souboru v přesně daném formátu, který je vytvořen některým z textových editorů. Teprve takto připravený vstup je možno načíst do programu a následné zpracovat a vytvořit výsledný grafický model. Na rozdíl od tohoto programu všechny ostatní si vstupy zajišťují pomocí vlastních nástrojů. L-Mayer2 pro vstup používá formulář, do kterého se zadá axiom a maximálně deset přepisovacích pravidel. Na záložce tohoto formuláře je umožněno zadat iteraci, velikost úhlu, barvu a některé další vlastnosti L-systému. Program L-Studio nabízí možnost zadávat vstup do předem definovaného okna, kam se předpisy zapisují v obdobné podobě jako v textovém editoru pro Fractint. Textový vstup má přesně danou strukturu. V aplikaci 3D FractaL-Tree je v pravé části umístěn přehledný panel, na kterém jsou nástroje pro nastavování jednotlivých parametrů. Prostředí je vytvořeno tak, aby i uživatel, který nemá hlubší znalosti z problematiky L-systémů, mohl poměrně jednoduchým způsobem zadávat požadavky na tvorbu modelu rostliny. Společnou vlastností u výše uvedených programů je možnost archivovat data tak, aby byla vždy znovu k dispozici. Každý systém má svůj specifický způsob ukládání archivních dat. Může se jednat o textové nebo binární soubory. Návrh programu MyLsystem byl nejvíce inspirován v oblasti zadávání vstupních dat programem 3D FractaL-Tree. Základním prvkem návrhu je okno programu, které je rozděleno na dvě části. V levé části se nacházejí komponenty pro jednoduché intuitivní nastavování parametrů a přepisovacích pravidel. Na pravé straně je pak prostor pro grafické znázornění vygenerovaného modelu rostliny.

Na samotné zpracování jsou kladeny nejrůznější požadavky vyplývající z konkrétního účelu, pro který byl program sestaven. První dva uvedené programy, Fractint a L-Mayer2, vytvářejí pouze dvourozměrné modely. Proto nepodporují znaky (/, \, ^, &), které se využívají pro znázorňování ve 3D. O to více je kladen důraz na ostatní možnosti grafického ztvárnění, jako je různobarevnost nebo univerzálnost grafických 2D prvků. 3D FractaL-Tree je postaven tak, aby mohl zobrazovat 3D modely nejrůznějších rostlin. Ve svém pojetí vůbec nezabývá 2D problematikou, výsledný model je vždy složen z 3D prostorových prvků. Samozřejmě lze bez použití znaků /, \, ^, & nasimulovat i 2D rostlinu. L-Studio je z uvedených programových

prostředků nejuniverzálnější. Podporuje jak 2D, tak 3D modely. Výsledný program má za úkol zobrazovat 2D a 3D modely rostlin. Do návrhu byl tedy zohledněn i tento požadavek. Za základní stavební prvek, ze kterého byly vytvářeny L-systémy, byl zvolen rotační válec. Tento prvek by bylo možné nahradit i jinými grafickými tělesy, v úvahu přichází např. komolý kužel. Z uživatelského hlediska by měl návrh programu obsahovat možnost náhledu na model rostliny z různých úhlů a vzdáleností. Tyto pozice by pak uživatel mohl průběžně měnit.

Dalšími požadavky při realizaci by mohly být funkce znázorňující za pomocí animace růst rostlin, zobrazování vývoje květů a listů nebo přidávání různých textur pro větší realističnost. Tyto funkce zahrnuje ze zmíněných programů pouze L-Studio. Jako jediný umí zobrazit květy v různých stádiích vývoje. V návrhu programu by měla být zahrnuta možnost tvorby květů. Jednodušším řešením může být vhodné otexturování koule nebo ve složitějším vytvoření modelu květu za použití jiných grafických útvarů. Podobně by mělo být možné vytvářet jednotlivé listy rostliny. Jak již bylo řečeno, v některých systémech je zahrnuta možnost zachycovat vývoj rostlin. Děje se to především pomocí využití technických prostředků, které používá animace. Většinou tento úkol bývá poměrně náročný a ne vždy se jej podaří uspokojivě implementovat.

Z tohoto návrhu množiny funkcí vycházela i samotná realizace výsledného programu. Většina navrhovaných funkcí je inspirována funkcemi ze zmíněných nástrojů pro vizualizaci L-systémů.

5 Technologie 3D zobrazení

K vytvoření nástroje pro vizualizaci trojrozměrných objektů bylo potřeba si zvolit vhodný nástroj. V dnešní době patří mezi nejčastěji používaná grafická API (Application Programming Interface) knihovna DirectX od společnosti Microsoft a konkurenční grafická knihovna OpenGL, původně vyvinutá společností Silicon Graphics Inc (SGI). Grafické API je rozhraní pro komunikaci mezi programem a grafickou kartou. Určuje, jak budou volány jednotlivé funkce knihovny ze zdrojového kódu programu. Obě dvě knihovny se stále vyvíjejí pro potřeby dnešních programátorů a nových technologií.

5.1 OpenGL

Knihovna OpenGL (Open Graphics Library) se používá pro 2D ale především pro 3D počítačovou grafiku. Na dodržování specifikace dohlíží konsorcium ARB (Architecture Review Board), do něhož patří firmy NVIDIA, SGI, IBM, Intel a do roku 2003 k nim patřila i společnost Microsoft. Vlastní OpenGL si může navrhnout kdokoliv, pokud jeho implementace splňuje standard OpenGL a projde testy. Největší klad této knihovny je nezávislost na platformě. Bez závažnějších problémů se aplikace s OpenGL mohou přeložit a spustit na operačním systému Windows, Linux a dalších. Kvůli přenositelnosti je knihovna OpenGL omezena pouze na vykreslování grafiky. Ostatní části, např. zvuky, zpracování událostí a další, jsou proto součástí jiných knihoven [OpGLDX].

5.2 DirectX

Další grafickou API je Microsoft DirectX. Tato sada knihoven je vyvíjena od roku 1995 společností Microsoft. Její největší nevýhodou je podpora pouze systému Microsoft Windows. Součástí DirectX jsou na rozdíl od OpenGL komponenty např. pro komunikaci přes počítačovou síť, vykreslování 2D grafiky, přehrávání zvuků a jiné. Mezi ně patří také Direct3D, který nabízí aplikacím velké množství funkcí pro práci s 3D grafikou. Rozhraní je objektově orientované, což je pro mnoho aplikací výhodné.

5.3 Java a 3D grafika

Pro tvorbu počítačové grafiky si většina programátorů zvolí jazyk C/C++ nebo C#. V době, kdy se začaly programovat první trojrozměrné počítačové hry, ještě Java nebyla na světě. Většina zastánců C/C++ ale i dnes stále tvrdí, že Java, jako zástupce interpretovaných programovacích jazyků, je příliš pomalá a nehospodaří dobře s pamětí. Tyto předsudky vznikly, když byla Java ještě mladý jazyk a oproti aplikacím napsaným v C/C++ byla skutečně pomalá. Od té doby se Java ale vyvíjela a dnes je na stejné úrovni [DK06].

Za zrychlením také stojí tzv. technologie HotSpot, která je zavedená od verze Java 2 Standard Edition 1.3. Tato technologie funguje tak, že pomáhá identifikovat a následně optimalizovat kritické sekce programu, které zatěžují výpočetní výkon procesoru. HotSpot tedy sleduje po spuštění programu nejčastěji volané metody a ty následně optimalizuje. Z toho vyplývá, že po nějakou dobu běží program pomaleji, než postupně optimalizuje potřebné funkce [HotS].

Jak již bylo zmíněno, programátoři v jazyce C/C++ se domnívají, že Java plýtvá pamětí. Java totiž pracuje s pamětí jinak, nepracuje s ukazateli jako jazyk C. Tím nemohou nastat obvyklé chyby se správou paměti, např. přístup do cizí paměti. V Javě jsou tyto typické chyby zachyceny překladačem a uvolňování paměti obstarává Garbage collector. K řádnému odstranění objektu z paměti může dojít jen, pokud jsou veškeré vazby na objekt přerušené, neboli v daném momentě již na objekt nikdo jiný neodkazuje.

Existuje mnoho prostředků pro programování 3D grafiky pomocí Javy. Knihovny pro práci s 3D grafikou v Javě jsou rozděleny do několika kategorií [DK06].

5.3.1 Java a vazba na OpenGL

Knihoven, které využívají vazbu Javy na OpenGL, je mnoho. Dále je popsáno několik nejznámějších knihoven s touto vazbou.

OpenGL for Java Technology, označována také jako **GL4Java**, byla velmi populární vazba Javy na OpenGL, dokud se neobjevilo rozhraní JOGL. GL4Java lze použít s knihovnami Swing a AWT. Také používá rozhraní Java Native Interface.

Knihovna Lightweight Java Game Library (**LWJGL**) umožňuje programovat kvalitní hry v Javě profesionálním i amatérským programátorům. LWJGL využívá New I/O, knihovna pro náročné I/O operace. Také dovoluje používání game padů, joysticků a dalších herních zařízení. Aplikační rozhraní je malé, a tak se hlavně používá pro zařízení s omezenými zdroji. Na rozdíl od GL4Java nepodporuje LWJGL knihovny Swing ani AWT.

API Java OpenGL (**JOGL**) paří mezi nejnovější vazby Javy na OpenGL. Byla vyvinuta Kennethem B. Russellem a Christopherem J. Klinem ze skupiny Game Technologies Group při společnosti Sun Microsystems. JOGL je velmi podobná LWJGL, ale liší se v tom, že JOGL je integrována do knihoven Swing a AWT. Od roku 2010 se jedná o nezávislý open source projekt pod BSD licencí[1].

5.3.2 API pro práci s grafem scén

Tato skupina knihoven pracuje s 3D grafikou na vysoké úrovni. Používá proto nízkoúrovňovou vazbu na OpenGL, viz předešlá podkapitola. Uvedené knihovny pracují s grafem scény. Graf scény je stromová struktura, která se skládá z uzlů reprezentujících objekty ve scéně.

Java3D je knihovna od společnosti Sun Microsystems na vysoké úrovni, která podporuje OpenGL a DirectX. Umožňuje práci s grafem scény složeného z geometrických útvarů, materiálů, světel a dalších prvků. Obsahuje metody pro práci s modely, nastavování světel nebo pro komunikaci po síti. Při programování na operačním systému Windows si programátor může od verze 1.3.2. zvolit, jestli použije OpenGL nebo DirectX.

Na stejném základu jako Java3D používá graf scény knihovna **Xith3D**, která dokáže přímo zavolat operace rozhraní OpenGL. Podporuje např. implementaci vlastních shaderů nebo výpočet objemu stínů. API Xith3D a Java3D jsou si velmi podobné, proto lze programy mezi těmito knihovnami jednoduše převádět. Xith3D může fungovat s rozhraním knihoven JOGL nebo LWJGL.

[1] Dovoluje volné šíření obsahu pod touto licencí. Vyžaduje uvedení autora, informace o licenci a upozornění o zřeknutí odpovědnosti za dané dílo.

V API **OpenMind** se nacházejí veškeré prvky, které jsou nutné pro práci s grafem scény. Umožňuje transformaci objektů, hierarchicky řídit scény či nastavování kamery a světel. Dříve bývala knihovna OpenMind implementována nad GL4Java, dnes je realizována nad rozhraním JOGL.

5.4 Výběr prostředku pro realizaci programu

Původně jsem se pro psaní programu rozhodovala mezi jazykem C# v prostředí .NET a Javou s knihovnou Java3D. Oba dva patří mezi vysokoúrovňové objektově orientované programovací jazyky. Java je multiplatformní jazyk tzn., že funguje na všech běžných operačních systémech. Oproti tomu C# podporuje pouze operační systém Windows. Oba jazyky mají sice odlišnou syntaxi, která je ale v základě velmi podobná. Jak u Javy, tak u C#, je možnost používat knihovnu OpenGL nebo DirectX. Pro vývoj aplikace jsem si nakonec zvolila multiplatformní programovací jazyk Java a knihovnu Java3D. Výsledný program pro vizualizaci rostlin L-systémy byl vytvořen pomocí vývojového prostředí Eclipse Indigo.

6 Aplikace pro vizualizaci L-systémů

Cílem této bakalářské práce je vytvoření aplikace, která umožňuje zobrazovat modely rostlin reprezentovaných L-systémy. V kapitole 4 je uvedeno několik programů, které umí zobrazovat různé L-systémy. Z nich jsem si vytvořila množinu funkcí, které by měla ideální aplikace pro vizualizaci L-systémů obsahovat (viz podkapitola 4.5), a následně jsem se snažila tyto funkce implementovat. Pro práci bylo vytvořeno přehledné grafické uživatelské rozhraní, které pomáhá uživateli obsluhovat program. Další část programu je vykreslení daného L-systému, složeného z válců, pomocí knihovny Java3D.

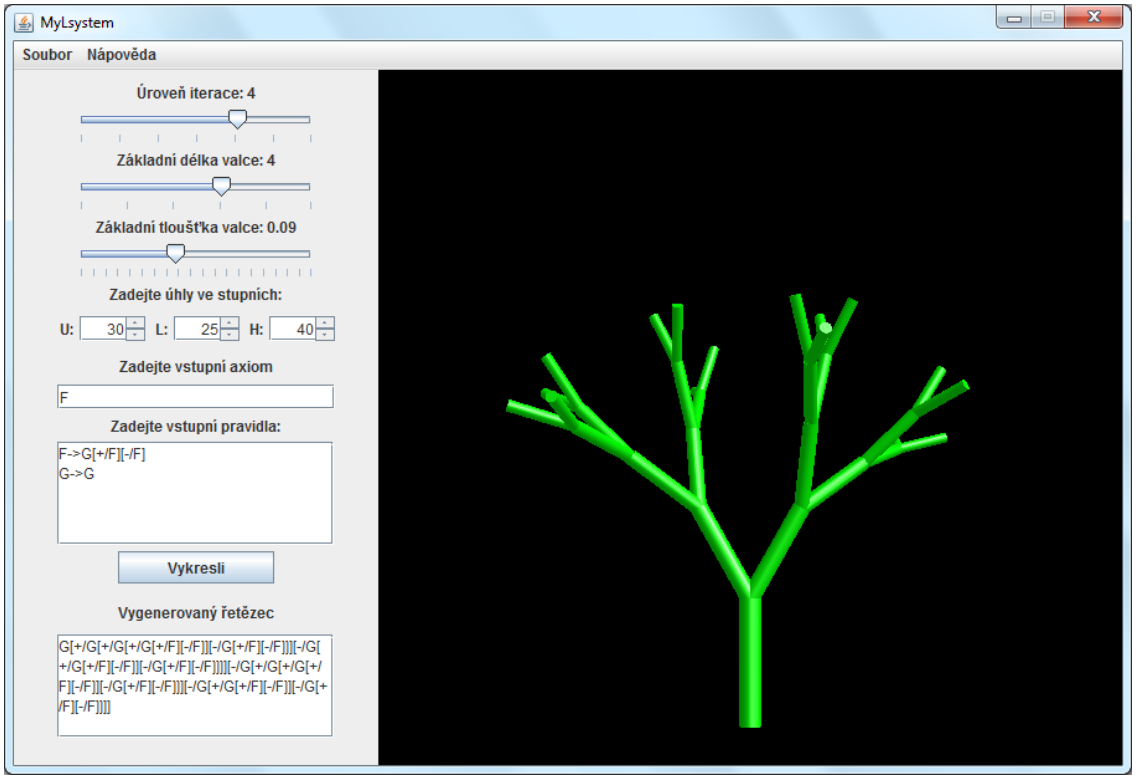
Prvotní návrh aplikace vycházel z enginu pro L-systémy, LSystemLab [LSLab], napsaného s pomocí knihovny Java3D. Navrhl jej profesor Chris Buckalew jako cvičný příklad pro své studenty a v knize [DK06] je uveden jako jeden z příkladů programů pro zobrazení L-systémů. Jeho předností, kvůli které ho uvádí autor knihy Andrew Davison, bylo použití rekurze. LSystemLab není uveden v kapitole 4, neboť se jedná o velmi primitivní applet, který nemá téměř žádné uživatelské funkce. Umí vytvořit jednoduchou scénu a podporuje pouze znaky pro dvourozměrné zobrazení. Výsledný L-systém je vytvořen z jednotlivých válců. Jediná podporovaná funkce je možnost natáčení vytvořeného modelu rostliny.

Výsledný program MyLsystem podporuje zobrazení D0L-systémů a závorkových L-systémů. Program podporuje základní symboly *F*, *G*, *[*, *]*, kde symbol *F* a *G* mají stejný geometrický význam. Symbol *f* není podporován, neboť ho není zapotřebí pro vizualizaci stavby rostliny. Pro zobrazení dvourozměrného L-systému musí být použity pro rotaci pouze znaky + a -, v případě trojrozměrného zobrazení mohou být využity následující symboly +, -, /, \, ^, &.

6.1 GUI

Po vzoru ostatních programů, které zobrazují L-systémy, byl v této práci vytvořen GUI program, viz Obrázek 14, jehož úkolem je tvorba požadovaného L-systému podle předem zvolených parametrů. Model vytvořené rostliny se skládá z jednotlivých rotačních válců různé délky a tloušťky. Program je navržen tak, aby uživateli

poskytoval co největší škálu nastavitelných parametrů a poté umožnil zobrazit výsledný L-systém.



Obrázek 14: Ukázka vytvořeného interaktivního GUI programu.

GUI je rozděleno do základních tří částí, které jsou skládány do výsledného okna ve třídě *WindowGUI*, která dědí od instance JFrame. *WindowGUI* nastavuje základní vlastnosti hlavního okna aplikace MyLsystem, jako jsou rozměry, název okna nebo přidání a umístění jednotlivých komponent. Nejprve se přidá panel pro zobrazení rostliny a následně se dodají další komponenty umožňující nastavení. Velikost hlavního okna má pevně dané rozměry a počáteční pozice je nastavena na střed obrazovky.

6.2 PanelSetting

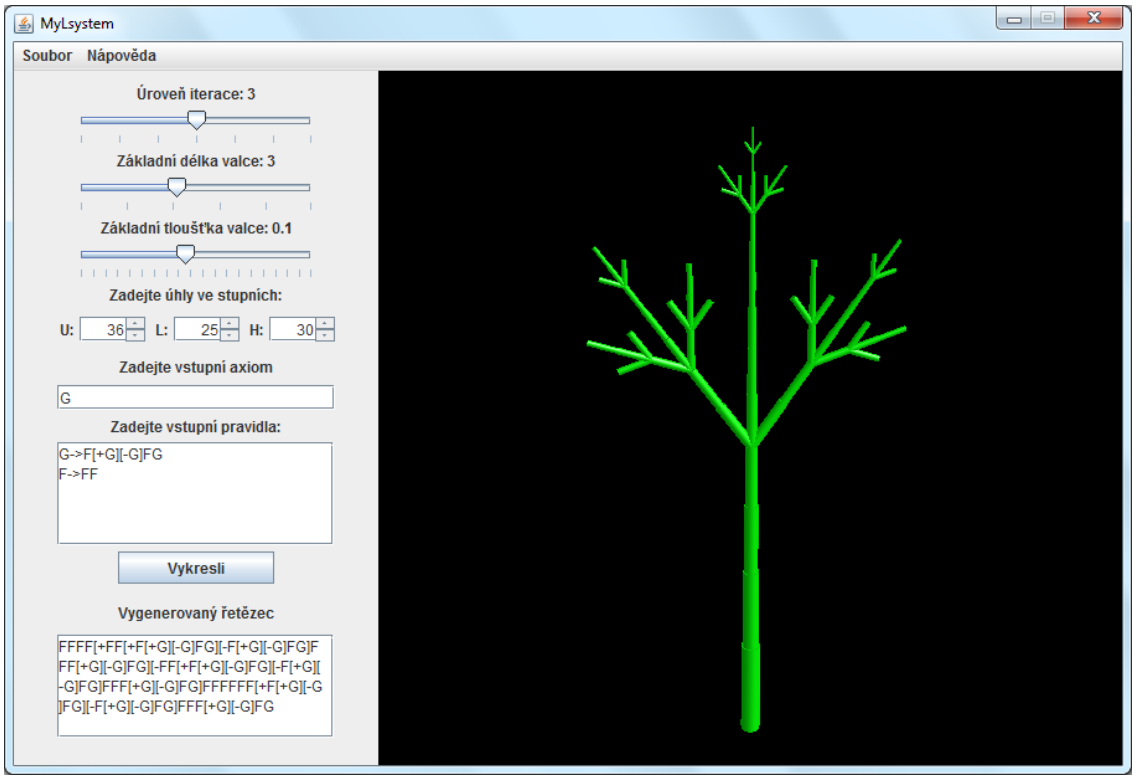
Levou část hlavního okna zabírá panel pro nastavení jednotlivých hodnot, který je sestavován ve třídě *PanelSetting*. Jednotlivé komponenty tohoto panelu jsou pro přehlednost rozděleny do metod. V úvodní volané metodě *createSliders()* se nacházejí posuvníky. První z nich stanovuje hodnotu iterace, neboli určuje, kolikrát bude voláno přepisování řetězce podle daných pravidel tak, aby vznikl výsledný

předpis. Maximální iterace je nastavena na hodnotu šest, aby nedocházelo k zbytečnému zahlcení programu, který složitější předpisy nedokáže při větších iterací zobrazit. Následující dva posuvníky se věnují vlastnostem jednotlivých válců, které vytvářejí výsledný model L-systému. Posuvník pro nastavení základní délky, určuje délku základního válce (lze si jej představit jako kmen stromu). Poslední posuvník nastavuje tloušťku tohoto základního válce. Maximální a minimální hodnoty posuvníků jsou zvoleny tak, aby umožňovaly nastavení rozmanitých L-systémů a současně zachovaly zobrazitelné rozměry.

Dále se na panelu nacházejí tři tzv. spinnery pro nastavení úhlů, které jsou vytvořeny metodou *createAngles()*. Každý ze tří spinnerů může mít nastavenou hodnotu od 0° až do 360°. Popisky určující jednotlivé osy jsou zvoleny podle definice tzv. želvy, viz 3. kapitola Obrázek 4. Osa U označuje v klasické kartézské soustavě souřadnic osu y, obdobně osa L odpovídá ose $-x$ a H je totožná s osu $-z$. Jejich změnou se pak při použití daných symbolů mění úhly a následná pozice dalšího válce.

Pro zadání axiomu slouží komponenta JTextField, která je vytvářena v metodě *createAxiom()*. Následuje metoda *createRules()*, která přidává komponentu JTextArea pro přepisovací pravidla. Přepisovací pravidla mají danou strukturu zápisu. Každé pravidlo musí být zapsáno na novou řádku. Dále musí být zapsáno ve tvaru, kdy symbol na levé straně je oddělen od pravé strany znakem „->“. V pravidlu se mohou libovolně vyskytovat bílé znaky, které jsou pro další zpracovaní ořezány. Pokud není zadáno žádné pravidlo, vykreslí se pouze axiom. Ukázka předpisu a výsledného L-systému je uvedena níže, viz Obrázek 15.

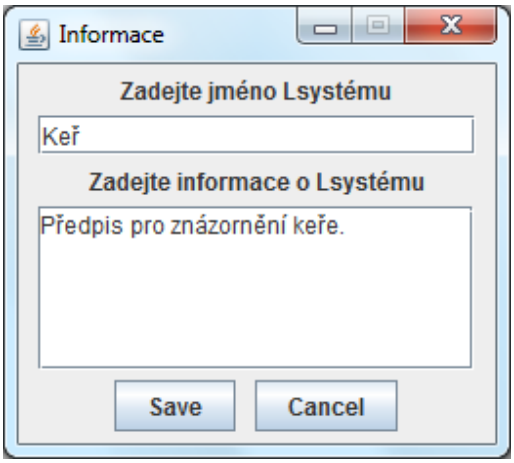
Po komponentách umožňující zadávat vstupní pravidla, následuje tlačítko „Vykresli“. Po zmáčknutí tlačítka, se nejdříve generuje výsledný řetězec pro želví grafiku ve třídě *CreateOutputString*. Tato třída bude popsána později. Vygenerovaný řetězec se uloží do textového pole, které se nachází pod tlačítkem. Při složitějším řetězci může docházet u zápisu do tohoto pole ke zpoždění v rámci několika milisekund. Pokud nastane situace, že řetězec je příliš komplikovaný pro znázornění, program automaticky zjednodušíš řetězec na symbol „F“, upozorní uživatele o složitosti předpisu a vykreslí pouze základní stonek. Pokud je však výsledný řetězec korektní, vytvoří se požadovaný L-systém v pravé části hlavního okna.



Obrázek 15: Ukázka L-systému vzniklého z přepisovacích pravidel $G -> F[+G][-G]FG$ a $F -> FF$.

6.2.1 Ukládání a načítání dat

Další výraznou komponentou v hlavním okně je lišta. V této liště, která je opět tvořena ve třídě *PanelSetting*, se nachází dvě základní nabídky. První nabídka „Soubor“ umožňuje uživateli uložit parametry L-systému, načíst již dříve uložený L-systém nebo ukončit celou aplikaci, obdobně jako při zavření okna křížkem v pravém horním rohu aplikace. Data jsou ukládaná do formátu XML (Extensible Markup Language). Pokud uživatel zvolí nabídku „Uložit“, zobrazí se mu nejprve okno nazvané Informace, viz Obrázek 16. Toto okno je objekt třídy *InforamtionForSaving*, kde dochází k vytvoření výsledného XML dokumentu. Pokud uživatel chce zadat název a podrobnější informace o L-systému, může tyto informace vyplnit. Následně musí uživatel zadat umístění dokumentu. Nezadá-li k názvu dokumentu koncovku .xml, je tato koncovka automaticky doplněna. Po uložení se dá opět dokument načíst z nabídky „Soubor → Otevřít“. Parametry aplikace se nastaví na hodnoty podle XML dokumentu a zároveň se současně vykreslí uložený L-systém.



Obrázek 16: Okno Informace.

Pro přehlednou strukturu uloženého dokumentu byl zvolen právě formát XML. Výhodou XML je čitelnost v různých textových editorech. Nejprve je v dokumentu uvedena deklarace XML, viz příklad níže.

```
<?xml version="1.0" encoding="windows-1250" standalone="no"?>
```

Kvůli podpoře češtiny v názvu nebo popisu L-systému je nastaveno kódování dokumentu na `windows-1250`. Následuje kořenový element `<lsystem>`, do něhož jsou vloženy ostatní elementy, které obsahují hodnoty jednotlivých nastavení. Dále je uveden příklad uloženého XML dokumentu.

```
<?xml version="1.0" encoding="WINDOWS-1250"?>
<lsystem>
	<name>Rostlina</name>
	<iteration>2</iteration>
	<length>4</length>
	<thickness>5</thickness>
	<angle>
		<U>30</U>
		<L>30</L>
		<H>30</H>
	</angle>
	<axiom>F</axiom>
	<rule>
		<formula>F->F[+F][-F][\F][/F][^F][&F]</formula>
	</rule>
	<information/>
</lsystem>
```

Další záložka na liště se nazývá „Nápověda“. Uvnitř jsou dvě nabídky. Pokud byly v načteném XML dokumentu vyplněny nepovinné údaje o jménu a popisu L-systému, jsou k dispozici pod nabídkou „O L-systému“. Druhá nabídka „O programu“ obsahuje informace o autorovi a aplikaci.

6.3 CreateOutputString

Ve třídě *CreateOutputString* se generuje řetězec pro želví grafiku. Po zmáčknutí tlačítka „Vykresli“ nebo načtení uloženého XML dokumentu, se zavolá metoda *splitRules()*. Nejdříve se kontroluje, zda byl zadán axiom a přepisovací pravidla. Poté jsou jednotlivá pravidla rozdělena podle řádků do připraveného pole. Následně jsou rozdělena znovu do dvou polí podle symbolu „->“ na levé a pravé strany. K výpočtu výsledného předpisu je použit cyklus, který volá metodu *substitutionOfString()* pro přepisování. Metoda je volána několikrát, počet cyklů určuje hodnota iterace.

Metoda má tři vstupní parametry. První je řetězec, ve kterém budou nahrazovány symboly levých stran pravidel odpovídajícími pravými stranami. Další dva parametry jsou pole, ve kterých jsou uloženy levé a pravé strany. Vstupní řetězec je rozdělen do pomocného pole *signs* podle jednotlivých symbolů. K substituci dochází uvnitř dvou for cyklů, kde se porovnává symbol z pole *signs* se symboly z levé strany. Pokud se symboly shodují, je do pomocného řetězce přidána odpovídající pravá strana přepisovacího pravidla a je nastavena indikace substituce na nenulovou hodnotu. Pokud nedošlo ke shodě a následné substituci, tzn., že indikace substituce má nulovou hodnotu, je do pomocného řetězce přidán symbol z pole *signs*. Výstupem je řetězec, kde došlo k nahrazení symbolů podle daných přepisovacích pravidel.

6.4 SpaceForPlant

V této třídě se vytváří scéna sestávající se z černého pozadí a vygenerovaného L-systému nasvíceného světlem, kde může uživatel pohybem myší scénou procházet. V konstruktoru se tvoří graf scény. Volají se v něm metody pro nastavení osvětlení nebo metody řešící pozici pozorovatele. Při tvorbě této třídy jsem vycházela především z ukázkových programů ke knize [DK06] a z tutoriálu ke knihovně Java 3D [J3DTuto].

Pozice pozorovatele se nastavuje v metodě *initUserPosition()*. Pomocí metody *lookAt()* se určí poloha pozorovatele ve virtuálním světě, bod, na který se pozorovatel dívá, a vektor, který určuje směr vzhůru. Pohyb pozorovatele ve scéně je umožněn třídou *OrbitBehavior* z knihovny Java 3D. Otáčení a posun uživatele je umožněn stiskem tlačítka myši a jejím následným pohybem. Toto chování je nastaveno v metodě *orbitControls()*.

V metodě *createSceneGraph()* se nastaví světlo, pozadí a L-systém. Metoda *lightScene*() do scény přidá směrové a ambientní světlo. Ambientní světlo přichází ze všech směrů se stejnou intenzitou. Směrové světlo vychází z jednoho vzdáleného zdroje a dopadá na povrch pod určitým úhlem. Pro definici směrového světla se musí, na rozdíl od ambientního, nastavit vektor udávající směr záření. Pozadí scény je nastaveno v metodě *addBackground()* na černou barvu.

L-systém se do scény přidává v metodě *generateAnotherBranch()* pomocí rekurze. Vstupními parametry metody jsou měřítko a řetězec popisující L-systém. Řetězec se čte po znaku, který je následné porovnáván s jednotlivými symboly. Nejdůležitější symboly jsou F a G, které reprezentují posun želvy a vytvoření válce. Každý přidaný válec má určité vlastnosti. *Cylinder* je třída z knihovny Java3D, která je odvozená od třídy *Primitive* a která vytváří válec požadovaných rozměrů. Pro vzájemné působení mezi barvou a světlem se používá komponenta Material, která určuje, v jaké barvě se bude jevit válec při nasvícení různými světly.

```
Material material =
          new Material(ambientColor, emissiveColor,
                       diffuseColor, specularColor, shininess);
```

První argument určuje barvu při ambientním osvětlením. Další, `emissiveColor`, určuje barvu, kterou vyzařuje sám válec. Protože válec žádnou barvu nevyzařuje, je tento parametr nastaven na černou barvu. Argument `diffuseColor` udává barvu při osvětlení směrovým světlem. `Shininess` nastavuje sílu odlesků a `specularColor` určuje odrazivost povrchu válce. Všechna tato nastavení společně s uzlem typu TransformGroup vytvářejí válec na určité pozici a s danými vlastnostmi.

Pokud byl přečten jeden ze znaků +, -, /, \, ^, &, otočí se pozice želvy na požadované ose o daný úhel, která se projeví po přidání dalšího válce. Při přečtení symbolu [se prohledává zbytek vstupního řetězce, dokud se nenajde příslušná pravá hranatá závorka]. Poté se řetězec rozdělí na řetězec uvnitř závorky a na zbytek řetězce za závorkou. Pro oba vzniklé řetězce se následně volá rekurze. Při každém přečtení validního symbolu se kontroluje, zda to není poslední symbol ze vstupního řetězce. Pokud následují další symboly, je rekurzivně volaná metoda *generateAnotherBranch()*.

Program MyLsystem se od výchozího programu LSystemLab liší především v nástrojích, které umožňují zadávání parametrů a jejich zpracování, načítání nebo ukládání dat. Také se liší v podporovaných typech L-systémů, kdy MyLsystem

podporuje závorkové trojrozměrné L-systémy. Rozdíl je i v pohybu ve scéně. LSystemLab pouze natáčel model rostliny. V MyLsystem umožňuje měnit pozici kamery tak, že se lze podívat na model i proti zdroji světla. Podoba mezi těmito programy je jen v použití rekurze pro vytváření jednotlivých válců, které reprezentují rostlinu.

6.5 Animace

Jednou z funkcí programu MyLsystem měla být možnost animace postupného růstu rostliny v čase. Při implementování této funkce jsem narazila na několik problémů, které nakonec neumožňovaly animaci do programu přidat. Původní myšlenka animace byla taková, že se v určitém časovém intervalu obnoví graf scény s objekty tvořící model rostliny. To by bylo možné, pokud by se L-systém vytvářel z jednoho válce, u kterého by se měnila pouze poloha a rozměr pomocí transformačních matic. Jednoduše by se pak pomocí nových transformačních matic, které by zastupovaly další krok animace, změnil graf scény. Jelikož se v programu vytváří mnoho samostatných válců, musela by být zavedena struktura, která by si u každého válce pamatovala jeho polohu a rozměry, aby při dalším kroku animace mohla být poloha válců aktualizována. Do struktury by se také přidaly válce, reprezentující nově přirostlé větve. Tento způsob animace se při použití rekurze ukázal jako nevhodný.

Dalším způsobem pro animaci, který jsem se snažila implementovat, bylo vytvořit pro každý krok animace nový L-systém, který by zcela nahrazoval ten starý. Formalismus L-systému není příliš vhodný pro animaci, protože každá expanze vytváří složitější rostlinu, u které je ale velmi obtížné rozlišit nově vytvořené části od původních. Myšlenka, kdy se pokaždé stará rostlina nahradí novou, se z počátku jevila vhodná. Při testování této funkce se ale objevil problém. Nadměrné použití rekurze způsobuje, příliš velké paměťové nároky. Mezi jednotlivými kroky animace pak vznikají velké časové prodlevy, které jsou způsobeny výpočtem a následným generováním komplikovanějšího L-systému.

Problém animace se nepodařilo úspěšně začlenit do aplikace MyLsystem. Hledání jednotlivých možností animace a následná snaha o jejich implementaci zabrala poměrně výrazný časový úsek při tvorbě realizační části. Proto se z časových důvodů nepodařilo do aplikace zavést podporu otevřených L-systémů nebo znázornění květů

a listů. Nakonec jsem dospěla k závěru, že L-systémy vytvářené rekurzivně nejsou zcela vhodné k animaci. Obdobný výsledek je zmiňován v knize [DK06], kde se autor zabýval tvorbou programu pro rostoucí les tvořený L-systémy. V případě realizace MyLsystemu se ukázalo, že i pro animaci růstu jedné rostliny není rekurze vyhovující.

6.6 Testování

Výsledná aplikace MyLsystem byla důsledně otestována. Při testování se odhalilo několik nedostatků, které byly následně odstraněny nebo případně ošetřeny. Jedním z problémů, objeveným při testu, byla malá vzdálenost zadní ořezávací roviny. To způsobovalo, že u větších modelů docházelo již při malém oddálení k velkému ořezu. Tento problém byl vyřešen změnou defaultní hodnoty 10 metrů na vzdálenost 30 metrů.

Dalším problémem bylo přetečení zásobníku. Problém nastal, pokud se zadal příliš složitý předpis L-systému, který při vytváření modelu rostliny měl velkou hloubku rekurze. Kvůli tomu je omezena maximální úroveň iterace na hodnotu šest. Při zadání komplikovaného předpisu docházelo k přetečení zásobníku i při nižší iteraci. Proto je v programu přidán čítač, který hlídá hloubku rekurze a při překročení určité hranice, přeruší vytváření modelu a upozorní uživatele chybovým hlášením. Zároveň se při každém generování výsledného předpisu pro rostlinu kontroluje počet znaků ve výstupním řetězci, jestli nepřekročil hodnotu 10 000 symbolů.

7 Závěr

Celá bakalářská práce byla rozvržena do několika částí. V úvodní části jsem se zabývala teoretickými základy, ze kterých se dále vycházelo při tvorbě programu. Teoretické základy zahrnovaly Chomského hierarchii a její vztah k L-systémům. Jsou zde také uvedeny základní typy L-systémů a jejich rozdělení s příslušnými ilustrujícími příklady. V souvislosti s generováním L-systémů se také zabývám popisem želví grafiky.

V přechodu mezi teoretickou a realizační částí jsem popsala a zhodnotila dostupné existující programy pro tvorbu L-systémů a z nich jsem následné odvodila základní množinu funkcí pro vytvořenou programovou aplikaci.

V další části práce jsem porovnávala jednotlivé možnosti pro tvorbu aplikací využívající počítačovou grafiku. Z nich jsem si vybrala programovací jazyk Java s knihovnou Java3D, která umožňuje trojrozměrné zobrazování.

Výsledkem je program MyLsystem, jehož pomocí lze zobrazit struktury zadané pro D0L-systémy a závorkové L-systémy. Výsledný objekt si může uživatel prohlížet a nastavovat u něho nejrůznější parametry. K tomu slouží jednoduchý GUI program, který umožňuje zadávání jednotlivých požadavků. Příloha A obsahuje několik ukázkových modelů vytvořených aplikací MyLsystem.

Na rozdíl od původního návrhu řešení se nepodařilo implementovat animaci a zpracování otevřených L-systémů, důvody jsou uvedeny v kapitole 6.5.

7.1 Možné rozšíření projektu

Jedna z funkcí, která by mohla být možným rozšířením programu, je podpora otevřených L-systémů. Tato funkce by dovolovala vytvářet stochastické keře nebo parametrické L-systémy, které by případně umožňovaly zobrazovat růst rostliny. Další rozšíření by mohlo být přidání květů a listů, které by vytvářely realističtější model rostliny.

Seznam zkratek

API	Application Programming Interface – grafické rozhraní pro komunikaci mezi programem a grafickou kartou.
ARB	Architecture Review Board – konsorcium, které spravuje standard OpenGL.
AWT	Abstract Windows Toolkit – umožňuje tvorbu grafického uživatelského rozhraní a je součástí Java Core API.
BMP	Microsoft Windows Bitmap – počítačový formát pro ukládání rastrové grafiky.
BSD	Berkeley Software Distribution – obchodní organizace při University of California v Berkeley.
GL4Java	OpenGL for Java Technology – vazba Javy na OpenGL.
GUI	Graphical User Interface – uživatelské rozhraní, které umožňuje ovládat program pomocí interaktivních grafických prvků.
IBM	International Business Machines corporation – akciová společnost zabývající se informačními technologiemi, především výrobou a prodejem počítačového softwaru a hardwaru.
JOGL	Java OpenGL – nejnovější vazby Javy na OpenGL.
JPEG	Joint Photographic Experts Group – formát souboru, který používá standardní metodu ztrátové komprese pro ukládání počítačových obrázků.
L-systém	Lindenmayerův systém – formální gramatika, vyvinutá pro modelování rostlin.
LWJGL	Lightweight Java Game Library – knihovna s vazbou Javy na OpenGL.
OBJ	Object Files – formát souboru pro popis 3D scén.
OpenGL	Open Graphics Library – knihovna, která se používá pro 2D a 3D počítačovou grafiku.
SGI	Silicon Graphics International Corp. – americká firma, která vyrábí počítačový hardware a software.
TGA	Truevision Advanced Raster Graphics Adapter – formát počítačových souborů pro ukládání rastrové grafiky.
XML	Extensible Markup Language – je obecný značkovací jazyk vyvinutý konsorciem W3C.

Literatura

[3DFract] 3D FractaL-Tree [počítačový program]. 2006-2007. Pro operační systémy: Windows 98 a vyšší. Download dostupný po bezplatné registraci z: http://bioquest.org/BQLibrary/library_details.php?product_id=12992

[DK06] DAVISON, Andrew, KREJČÍ, Lukáš. *Programování dokonalých her v Javě*. Brno: Computer Press, 2006. ISBN 80-7226-944-5.

[FractInt] Fractint [počítačový program]. Verze 20.04. The Stone Soup Group ©1990-2012. Freeware. Domovské stránky dostupné z: http://www.fractint.org/

[HotS] MIKOLÁŠEK, Vašek, HEROUT, Pavel. Java HotSpot. *www.kiv.zcu.cz* [online]. [Cit. 6. 4.2012]. Dostupné z: http://www.kiv.zcu.cz/~herout/pruzkumy/hotspot/hotspot.html

[J3DTuto] Java 3D API Tutorial. *Oracle: Sun Developer Network* [online]. Oracle, ©2010. [Cit. 14. 4. 2012]. Dostupné z: http://java.sun.com/developer/onlineTraining/java3d/

[KŠCH89] KOLÁŘ, J., ŠTĚPÁNKOVÁ, O.; CHYTIL, M. *Logika, algebry a grafy*. Praha: Státní nakladatelství technické literatury, 1989.

[Lmayer] SCHWEBINGHAUS, Ulrich. Lindenmayer-Fractals [počítačový program]. Verze 2.2. ©2001-2003. Dostupný z: http://www.fraktalwelt.de/myhome/lmayer.htm

[LSLab] BUCKALEW, Chris. LSystemsLab.java [zdrojový kód]. ©2002. Dostupný z: http://users.csc.calpoly.edu/~buckalew/474Lab7-W02.html

[LStudio] PRUSINKIEWICZ, Premyslaw. L-studio [počítačový program]. Verze 4.2 ©1998-2009. Po bezplatné registraci dostupný z: http://algorithmicbotany.org/

[MP96] MĚCH, Radomír, PRUSINKIEWICZ, Premyslaw. Visual Models of Plants Interacting with Their Environment. *SIGGRAPH '96 Conference Proceedings*, August 1996, p. 397 – 410.

[OpGLDX] ElCondor. OpenGL vedle DirectX stále žije - vychází openGL 3. In: *DDWorld.cz* [online]. Poslední změna 14. 11. 2007 [Cit. 3. 4. 2012]. Dostupné z: http://www.ddworld.cz/software/linux/opengl-vedle-directx-stale-zije-vychazi-opengl-3.html. Path: Homepage; Software; Linux.

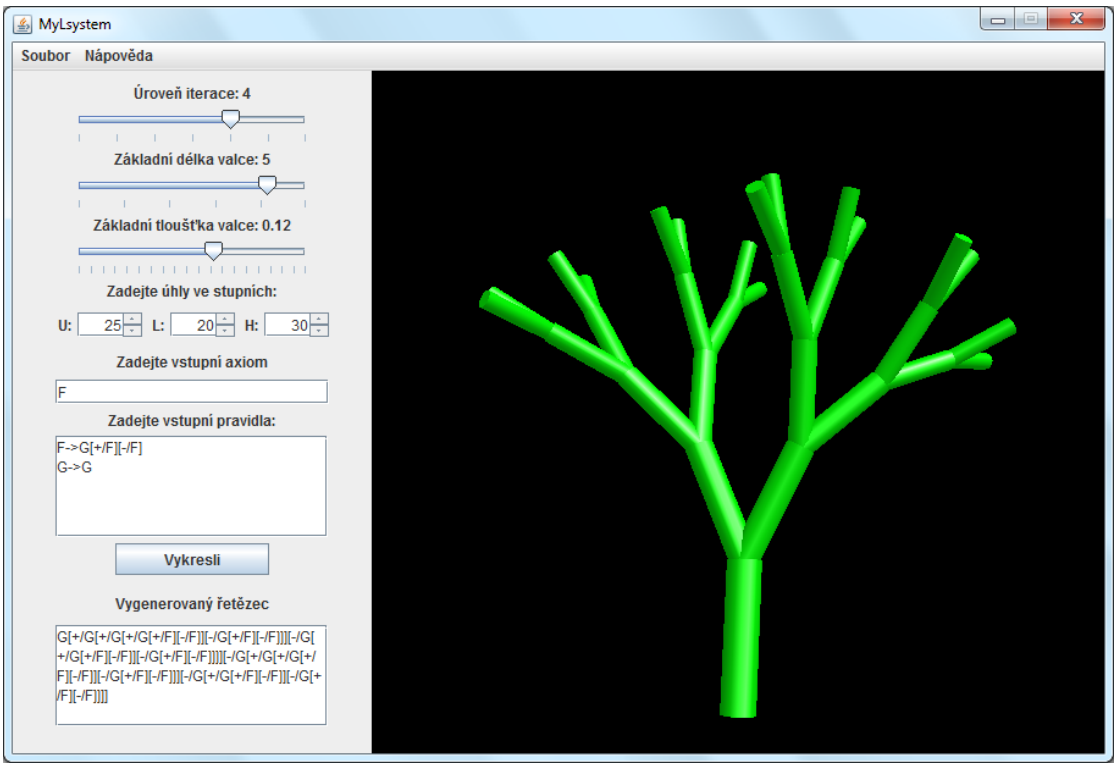
[PL90] PRUSINKIEWICZ, Premyslaw, LINDENMAYER, Aristid. *The Algorithmic Beauty of Plants*. New York: Spronger-Verlag, 1990. ISBN 0-387-94676-4. Dostupné z: http://algorithmicbotany.org/papers/#abop

[VK06] VIRUCHPINTU, Rawin, KHIRIPET, Noppadon. *Real-time 3D Plant Structure Modeling by L-System with Actual Measurement Parameters*. 2006. Dostupný z: http://bioquest.org/esteem/esteem_details.php?product_id=13157

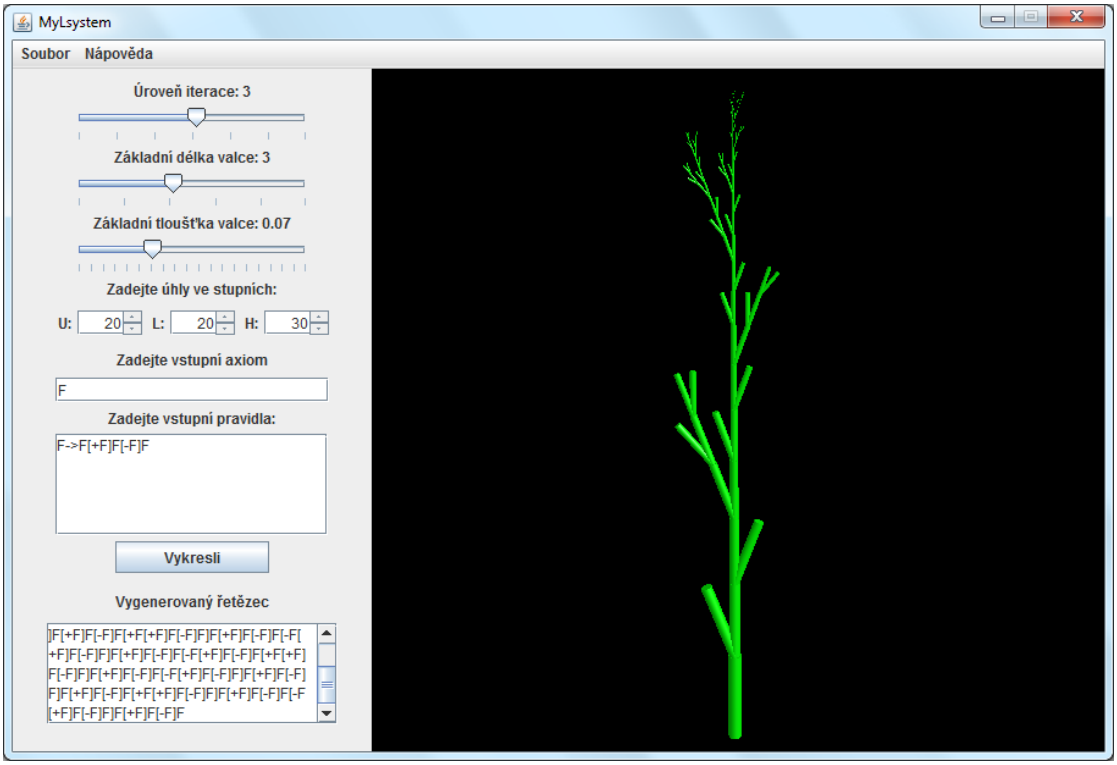
[VSV08] VAHIDIPOUR, M., SABAGHIAN-BIDGOLI, H., VALKILI-NEZHAAD, G. Generation of the Figures of Some FuIIerenes by Using L-Systems. *Croatica chemica acta*, April 2008, vol. 81, no. 2, p. 341 - 345. ISSN-0011-1643.

[ŽBS04] ŽÁRA, Jiří; BENEŠ, Bedřich, SOCHOR, Jiří. *Moderní počítačová grafika*. 2. přepracované vyd. Brno: Computer Press, 2004. ISBN 80-251-0454-0.

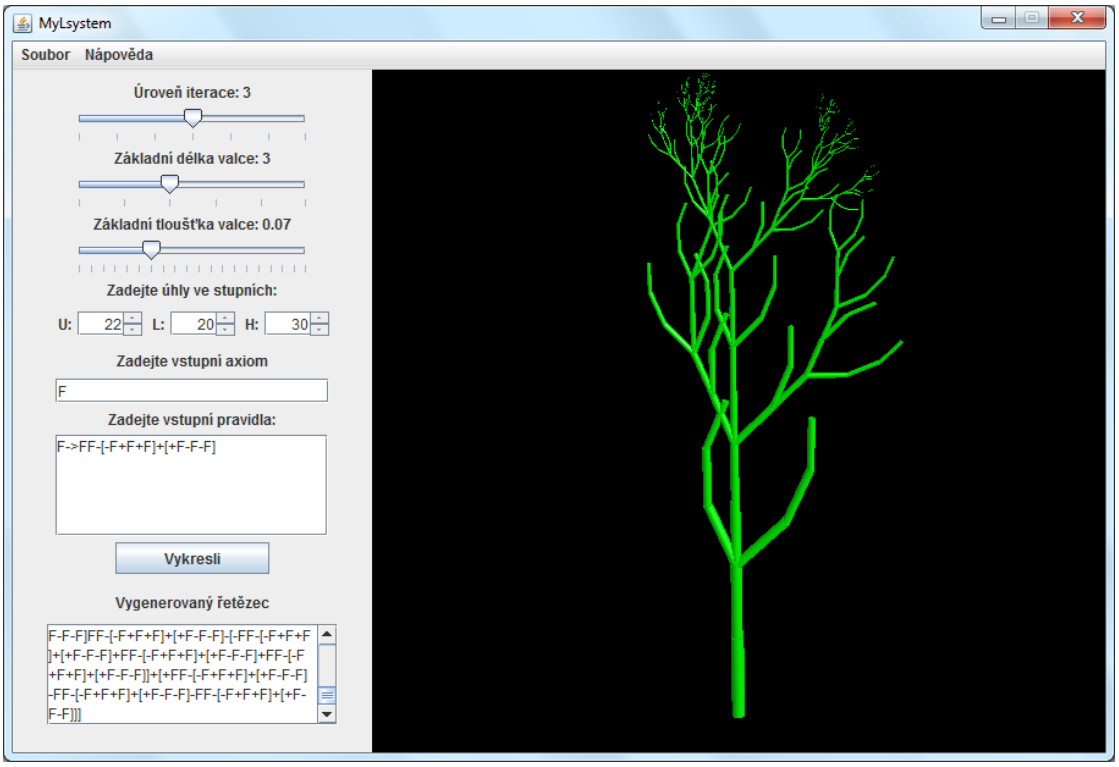
A. Příloha



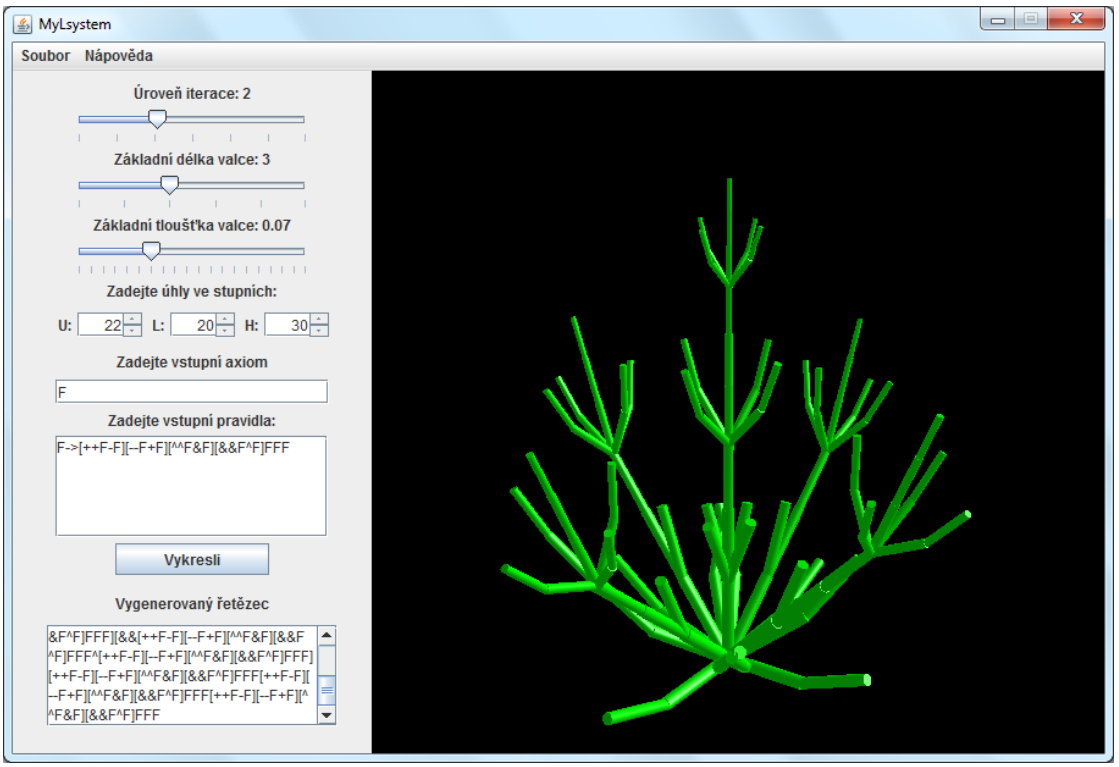
Obr. 1: Ukázka výsledku programu znázorňující trojrozměrný strom.



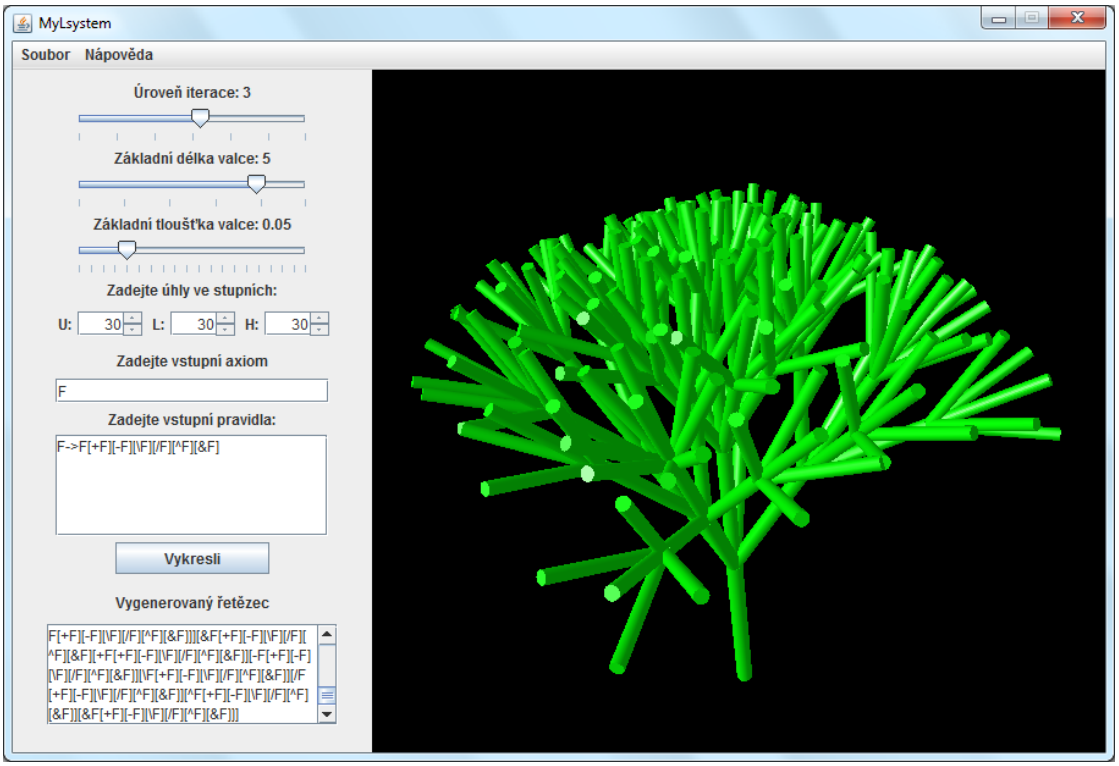
Obr. 2: Ukázka rostliny znázorňující zmenšování a zeštíhlování větví.



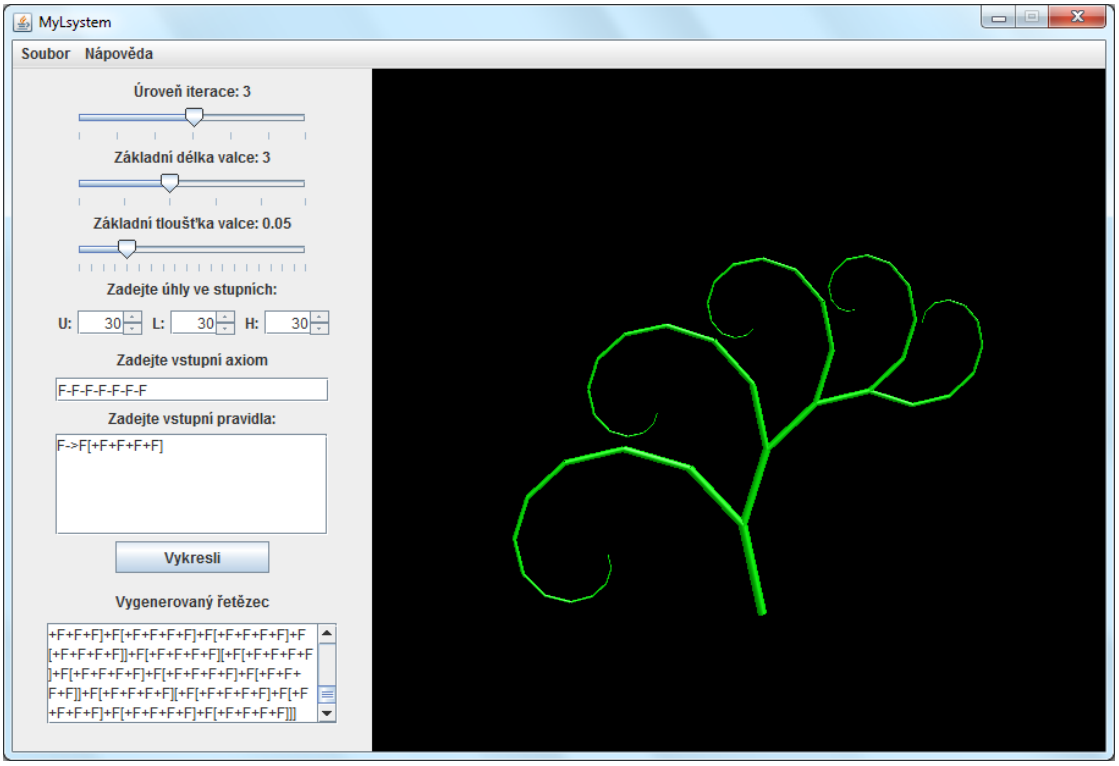
Obr. 3: Ukázka znázorňující dvourozměrnou rostlinu.



Obr. 4: Ukázka výstupu programu znázorňující keř.



Obr. 5: Ukázka rostliny v trojrozměrném prostoru.



Obr. 6: Ukázka znázornění rostliny se spirálovitě zahnutými větvemi.