Gymnázium, Plzeň, Mikulášské nám. 23

Maturitní práce z informatiky a výpočetní techniky

dokumentace

Pavel Šigut

Oktáva E

2015/2016

Obsah

I. Úvod

Jako téma jsem si vybral program vykreslující L-systémy, protože mě zaujala jejich grafická implementace, kdy jednoduchými postupy vznikají krásné složité obrazce.

K programování jsem zvolil programovací jazyk C# s vývojovým prostředím Microsoft Visual Studio Community 2015, neboť s ním mám ze školy zkušenosti a dovoluje snadno vytvářet uživatelsky přívětivá rozhraní.

1.L-systém

1.1 Definice

L-systém (Lindenmayerův systém) se jako typ *formální gramatiky (G = N, T/ **Σ**, P, S)* sestává z následujících prvků:

- *abecedy* – množiny symbolů, které máme v jazyce k dispozici
    - *neterminální **N*** – nahrazují se pravidly tak dlouho, dokud nevznikne slovo daného jazyka, které je zapsáno znaky *terminálními **T, Σ***
    - např. A, B, +, -, [,]
- *semínka (axiomu) **S*** – počátečního stavu soustavy, na který budeme aplikovat pravidla
    - např. A+B
- *slova (řetězce)* – aktuálního stavu soustavy vygenerovaného ze semínka pomocí pravidel; množina všech těchto slov se nazývá formální jazyk
- *přepisovacích pravidel **P***, která určují způsob, jakým se bude slovo vyvíjet
    - přepis symbolů slova ve tvaru A → A
    - přepisy pomocí všech pavidel probíhají paralelně

Toto názvosloví je odvozeno od jazyků přirozených. Jednotlivé vlny aplikace pravidel se nazývají iterace (generace).

1.2 Historie

L-systémy byly poprvé popsány v roce 1968 maďarským biologem Aristidem Lindenmayerem. Ten je používal k popisu chování buněk rostlin, kvasinek, hub atp., ale dnes je jimi možné modelovat struktury větvení a celé vyšší rostliny. Později je polský informatik Przemysław Prusinkiewicz popsal grafickými prvky pomocí želví grafiky a společně s A. Lindenmayerem vydali knihu *The Algorithmic Beauty of Plants*, jež se dá považovat za základ oboru.

1.3 Želví grafika

Želva je pomyslná reprezentace aktuálního bodu vykreslování a uchovává tyto informace:

- *aktuální souřadnice*
- *úhel* – směr, kterým je natočená
- *velikost kroku* – délka vykreslené čáry

Některé znaky slova se nevykreslují, ale ovlivňují vývin křivky. V abecedě existují konvenční symboly k provádění určitých operací

- A, B, F, G, X, Y a jiná velká písmena vykreslí úsečku
- + otočí želvu v kladném směru, - ve směru záporném
- [uloží aktuální pozici,] načte poslední uloženou pozici (pro potřeby větvení)

1.4 Typy

Použití přepisovacích pravidel se dá různě upravovat, což dává za vznik následujícím:

- *stochastické* – zavádějí prvky náhody do velikosti úhlu a kroku
- *parametrické* – umožňují použití číselných hodnot v pravidlech pro změny délek úseček
- *kontextové* – berou ohled na okolí symbolu, různá okolí podmiňují různá pravidla pro ten konkrétní symbol

Všechny zmíněné umožňují přirozeněji vypadající výsledky. Stochastismus je umožněn i v tomto programu.

1.5 Použití

Termín *fraktál* poprvé použil francouzsko-americký matematik Benoît Mandelbrot v roce 1975 pro soběpodobné křivky (při jakémkoliv měřítku vypadají stejně), které mají velmi složitý tvar, protože jsou generovány pomocí určitých pravidel donekonečna. Proto je možné některé z nich sestrojit pomocí L-systémů, např. Dračí křivku, či Sierpińského trojúhelník.

Následuje různé modelování rostlin, měst či krajin pro počítačové hry a filmy. V přírodě má mnoho struktur takovýto charakter, například plíce a vločky.

1.6 Zdroje informací

Wikipedie, otevřená encyklopedie

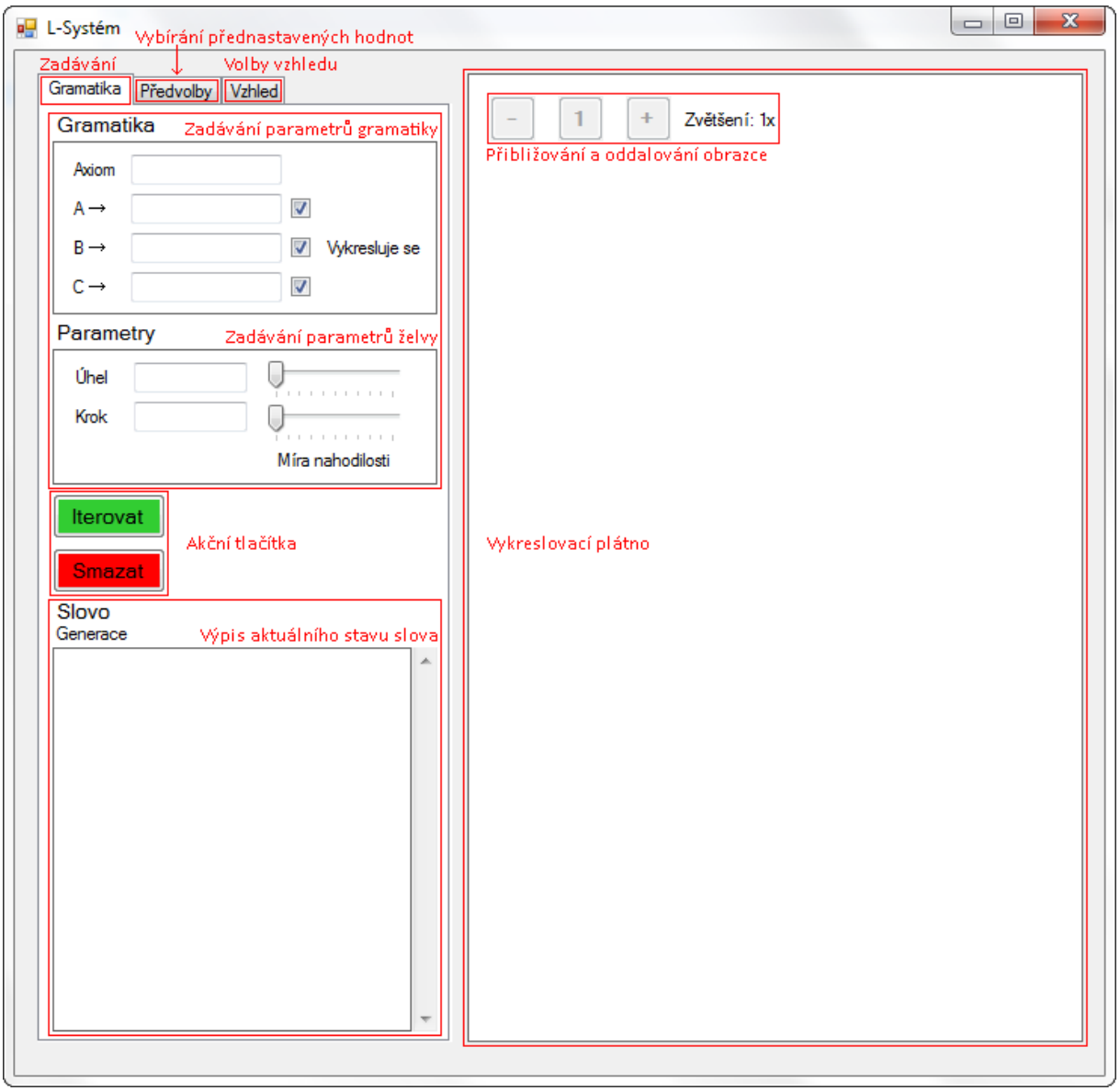
The Algorithmic Beauty of Plants

II. Program

1.Popis uživatelského prostředí

Prostředí programu je složeno z jednoho okna, ve kterém se nacházejí všechny prvky pro jeho obsluhu (až na dialog barvy). Poskytuje uživateli možnost zadat až tři pravidla a rovněž možnost nevykreslování symbolů, na které jsou vázána.

Pro přesnější popis obsluhy vizte příslušné kapitoly.



2.Popis mechanismu

2.1 Vkládání proměnných

Uživatel do pole „Axiom“ vepíše axiom, do polí „A/B/C →“ pravidla, přičemž **jediné** povolené symboly jsou A, B, C pro úsečky, +, - pro otočení o úhel a [,] pro větvení. Tři písmenné symboly bývají pro většinu L-systémů dostačující. Do pole „Úhel“ pak napíše úhel ve stupních, který bude odpovídat otočení, a do pole „Krok“ velikost vykreslovaných úseček v pixelech.

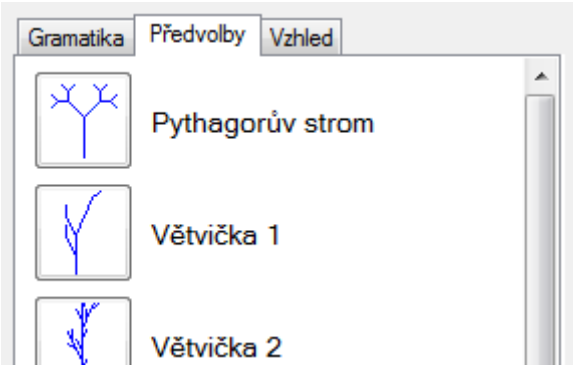
Následně stisknutím tlačítka „Iterovat“ dojde k předání parametrů ke zpracování a znemožnění jakékoliv změny deaktivací ovládacích prvků.

Pokud nebyla vyplněna povinná pole (axiom, úhel a krok – bez nich nemá program žádný smysl; pravidlo nutné není, můžeme chtít vykreslit jen konečný řetězec), tak se podbarví růžově, aby upozornila uživatele na jeho chybu. Nejsou podbarvena od začátku chodu programu, protože takto si uživatel snáz uvědomí, co po něm program chce.

Jedná se o metodu nacteni(), která:

- *převede text textboxů do proměnných (pokud není nic vyplněno, doplní vlastní, aby zabránila chybě)*
- *deaktivuje ovládací prvky vkládání a zpřístupní zvětšování a zmenšování obrazce*

2.1.1 Vkládání pomocí předvolby



Záložka „Předvolby“ hlavního ovládacího panelu umožňuje výběr z předem nadefinovaných gramatik. Uživatel stiskne požadované tlačítko s obrázkem, načež je vrácen zpět na záložku „Gramatika“, kde uvidí nově vložené proměnné.

2.2 Přepisování axiomu pomocí pravidel

Axiom se pak v souladu s filosofií L-systémů každým stisknutím tlačítka „Iterovat“ přepíše pomocí zadaných pravidel. V podstatě se jedná o manipulaci s textovým řetězcem. Aby se nepletly staré prvky slova a ty již nově přepsané (protože přepisování všemi pravidly fakticky neprobíhá naráz, ale postupně), nejdřív jsou písmenné symboly A, B, C nahrazeny symboly X, Y, Z.

O toto se stará metoda stisku tlačítka „Iterovat“, bIterovat_Click(), která se kromě toho:

- *ujistí, zda je vše správně zadáno (popř. zruší chybové podbarvení políček)*
- *zavolá metody implementace(true) a vykresleni(true)*

- *vypíše aktuální stav slova a číslo generace do příslušného pole*
- *navýší číslo generace o 1*

2.3 Zjednodušení slova pro vnitřní potřeby

Jelikož slovo obsahuje zbytečné znaky pro vykreslování (různá písmena pro označení vykreslení úsečky, znaky, které se nevykreslují), dojde k jeho zjednodušení.

Jedná se o metodu zjednoduseniSlova(), která:

- *převede všechny znaky na „A“*
- *pokud není zaškrtnuté příslušné políčko „Vykresluje se“, tak nahradí znak prázdným řetězcem „“*

2.4 Zavedení prvku náhody

Posouváním jezdců na trackbarech „Míra nahodilosti“ od minimální (0) do maximální (10) hodnoty můžeme zavést prvky náhody. Při maximální nahodilosti může velikost úhlu a kroku nabývat hodnot od nuly až do jejich dvojnásobku. Je též zaveden speciální generátor náhodných čísel. Konečná hodnota je vypočtena podle vzorce (uvedeno pouze pro krok; pro úhel obdobně):

```
krok = krok + [krok * (náhodná hodnota od -10 do 10) * (hodnota příslušného trackbaru) / 100]
```

Jelikož hodnota trackbaru nabývá hodnot od 0 do 10 a náhodná čísla jsou od -10 do 10, je třeba součin vydělit stem, aby byl ve tvaru poměru.

Následné velikosti úhlů a kroků se uloží do seznamu. Celý tento krok se provádí jen jednou, aby při transformaci nedošlo k přepočítání (nechceme, aby se třeba při zvětšení vykreslilo něco úplně jiného), čehož je dosaženo argumentem metody implementace(poprve), která buď provede (true), nebo neprovede (false) znáhodnění.

Jedná se o metodu znahodneni(), která kromě toho:

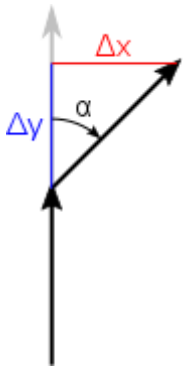
- *vyčistí seznamy pro ukládání náhodných úhlů a kroků pro jejich znovupoužití*

2.5 Vypočítání souřadnic na základě slova

Metoda postupně prochází celé zjednodušené slovo a pomocí příkazu *switch* vyhodnocuje jednotlivé znaky.

- „+“ přičítá hodnotu úhlu zadanou uživatelem k otočení
- „-“ tuto hodnotu odečítá
- „A“ vypočítává nové souřadnice konce orientované úsečky

K vypočítání souřadnic konce orientované úsečky se používá funkce sinus a kosinus (pro vysvětlení vizte obrázek vpravo) na základě vztahu:




```
x = x + x * sin(otočení)

y = y + y * cos(otočení),
```

kde x a y jsou aktuální souřadnice a otočení je „směr“ budoucí želvy (čili odchylka od „aktuálního“ rovného směru). Souřadnice jsou následně ukládány do kolekcí po dvojicích (počáteční a konečná souřadnice úsečky).

2.5.1 Větvení

Pokud příkaz switch narazí na [, uloží aktuální souřadnice, pokud narazí na], načte poslední uložené souřadnice. To umožňuje vytvářet větvené struktury, ne pouze lineární.

Jedná se o metodu implementace(), která kromě toho:

- *zavolá metody zjednoduseniSlova() a poprvé i znahodneni()*
- *vyčistí seznamy pro ukládání souřadnic*
- *přenastaví otočení na jeho původní hodnotu*

2.6 Grafická implementace souřadnic

Metoda pomocí příkazu *for* projde seznamy souřadnic x a y a vykreslí podle nich úsečky.

Jedná se o metodu vykresleni(), která kromě toho:

- *zavede na pictureboxu objekt plátna Graphics a dvě pera, jedno pro vykreslení osového kříže (kde bude počátek obrazce) a druhé pro samotné vykreslování obrazce*
- *vyčistí plátno, aby se jednotlivé iterace nepřekrývaly (pokud je argument false (pro potřeby mazání), tak dojde pouze k vyčištění – k samotnému vykreslování již ne)*
- *nakonec zahodí grafické objekty příkazem Dispose()*

3. Transformace

Odemknou se až po stistknutí tlačítka „Iterovat“.

3.1 Přesouvání myší

Ovládá se pomocí kliknutí a táhnutí v pictureboxu s obrazcem. Funguje na principu vytvoření směrového vektoru na základě rozdílu souřadnic místa, kde jsme stiskli pravé tlačítko myši, a místa, kde jsme pravé tlačítko pustili.

Jedná se o metody pictureBox1_MouseDown() a pictureBox1_MouseUp, která kromě toho:

- *zavolá metody implementace(false) a vykresleni(true)*

3.2 Zvětšování tlačítky „-“, „1“ a „+“

Funguje na principu změny velikosti kroku (po jednom pixelu) a následném překreslení. Zároveň počítá a vypisuje zvětšení, které je dáno podílem:

```
zvětšení = (nový krok) / (starý krok)
```

V záložce vzhled můžeme zvolit variantu „Při zvětšování přepočítávat“, která při zvětšení/zmenšení ztlušťuje/ztenčuje čáru, tak jako při zvětšování v klasických grafických editorech. Jinak čára zůstává pořád stejně tlustá.

Tlačítko „1“ vrátí zvětšení zpět na hodnotu 1 a přesune obrazec na jeho výchozí pozici, protože se mohl uživateli při transformování snadno ztratit z dohledu.

Jedná se o metody bMinus_Click(), b1_Click() a bPlus_Click(), které kromě toho:

- *zavolají metody implementace(false) a vykresleni(true)*

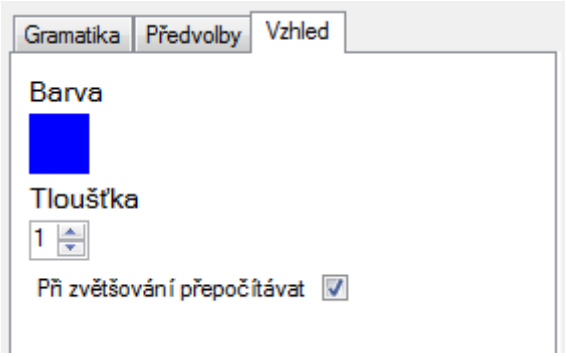
4.Mazání

Ke smazání veškerého obsahu pictureboxu s obrazcem dojde při stisknutí tlačítka „Smazat“. Mimo to se vymažou a zpřístupní pro nové vkládání všechna pole v kolonce „Gramatika“. Program je znovu takový, jako byl při spuštění.

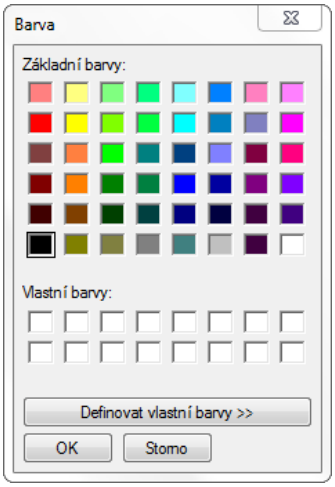
Jedná se o metodu bSmazat_Click(), která kromě toho:

- *vynuluje všechny proměnné*
- *znovu zakáže tlačítka pro zvětšování*

5. Vzhled



5.1 Změna barvy obrazce



Stisknutím barevného políčka pod nápisem „Barva“ dojde k vyvolání dialogu barvy, který nám umožní zvolit si novou barvu obrazce. Můžeme vybírat ze základních barev, nebo po stisknutí tlačítka „Definovat vlastní barvy >>“ zvolit přesnou barvu HSV/RGB modelu. Ta se po stisknutí tlačítka „OK“ dialogového okna uloží.

Jedná se o metodu bBarva_Click(), která kromě toho:

- *vyvolá metodu vykresleni(true)*

← Dialog barvy

5.2 Změna tloušťky čáry

Šipkami nahoru/dolů u kontrolky pod nápisem „Tloušťka“ můžeme měnit tloušťku čáry na hodnoty mezi 1 a 5. Tato hodnota se pak stává novou tloušťkou per, kterými vykreslujeme v metodě vykresleni(). Při zaškrtnutí „Při zvětšování přepočítávat“ se tloušťka per ještě navíc vynásobí zvětšením.

Jedná se o metodu nudTloustka_ValueChanged(), která kromě toho:

- *vykreslí náhled tloušťky vedle kontrolky*
- *vyvolá metodu vykresleni(true)*

6.Jiné metody

6.1 Načtení předvoleb

Podrobněji popsáno v kapitole 3.1.1 Vkládání pomocí předvolby.

Jedná se o více metod typu (název tlačítka předvolby)_Click(), které:

- *vloží hodnoty na příslušná místa v záložce „Gramatika“*
- *pošle uživatele zpět na záložku „Gramatika“*

6.2 Přizpůsobení velikosti pictureboxu

Při změně velikosti okna programu dojde k automatické změně velikosti pictureboxu tak, aby se všechny prvky do okna vešly. Při zmenšení okna je občas třeba obrazec přesunout, aby byl v menším pictureboxu vidět (kapitola 4.1 Přesouvání myší).

Jedná se o metodu Form1_Resize(), která kromě toho:

- *změní výšku oblasti záložek vlevo (šířka je kvůli přehlednosti neměnná)*

III. Závěr

Ačkoliv program začínal jako jednoduché vykreslení obrazce na pozadí formuláře, postupným vylepšováním se z něj stala přívětivá utilita, která může leckomu naplnit i jeho umělecké ambice. Poskytuje prostor si vyhrát s jednotlivými parametry, pozorovat, jak se vše změní díky maličkostem, a proto doufám, že ani po chvíli neomrzí.