Dokumentace k zápočtovému programu

Jakub Levý

10.1.2018

Abstrakt

Tento program implementuje neparametrický 2D L-systém. Je napsán v jazyce C#, využívá knihovnu WinForms a GDI+. Kontroluje vstupy a umožňuje vykreslení více fraktálu najednou. V uživatelském rozhraní lze jednoduše zadat a editovat vstup. Součástí programu je 10 fraktálů, které je možné použít jako testovací data.

Obsah

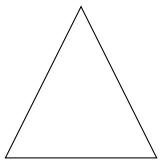
1 Úvod do L-systémů

Tato kapitola popisuje funkčnost jednoduchých 2D L-systémů. Rád bych podotknul, že tento úvod se nesnaží formálně správně definovat L-systémy. Jde spíše o jejich pochopení.

1.1 Historie

Důležitý koncept, který L-systémy používají se nazývá **přepisování**. Jedná se o techniku definování komplexních objektů neustálým přepisováním částí původního objektu pomocí **přepisovacích pravidel**. Klasický příklad takového objektu je *Kochova vločka* navržená roku 1905 Helge von Kochem! Její postupnou konstrukci definoval následovně:

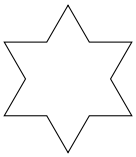
Mějme dva obrázky, první nazveme iniciátor a druhý generátor:



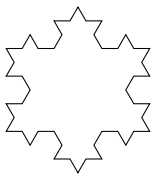
iniciátor

generátor

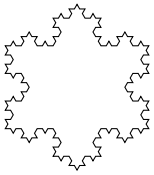
Postupnou aplikací generátoru na iniciátor dostáváme:



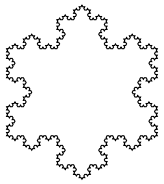
1.



2.



3.



4.

Takovéto přepisovací systémy se začaly studovat už v polovině 20. století. Už v té době se pro reprezentaci používali systémy, kde se přepisovaly znaky textovými řetězci. Až v roce 1968 biolog Aristid **L**indenmayer publikoval nový systém, který my nyní známe pod pojmem **L**-system.

1.2 Popis

Nyní se můžeme podívat jak 2D L-systémy fungují. Definujeme každému znaku, kterému L-systém rozumí nějakou akci:

znak	akce
$A - Z$	vykresli úsečku ve směru aktuální rotace
$a - z$	posuň pozici štětce ve směru aktuální rotace
$+$	otoč se o předem stanovený úhel proti směru hodinových ručiček
$-$	otoč se o předem stanovený úhel ve směru hodinových ručiček
$[$	ulož aktuální stav na zásobník
$]$	nastav aktuální stav na vrchol zásobníku a smaž ho

- stavem se rozumí: aktuální pozice štětce a úhel rotace
- délka jednotlivého kroku tj. délka vykreslené úsečky, případně délka posunu štětce musí být předem dohodnutá
- úhel rotace také musí být předem známý, v případě Kochovy vločky nakreslené výše se jednalo o 60°
- implicitně uvažujeme že úsečky se vykreslují černou barvou, což nemusí být vždy zcela vhodné

L-system na vstupu dostane 3 zásadní informace:

1. iniciátor, kterému se říká **axiom**
2. asociativní pole generátorů, kterému se říká **přepisovací pravidla**
3. úhel, o který se má rotovat

Další informace, které L-systém dostane, nejsou nutné pro teoretickou funkčnost.

1.3 Příklad

Ukažme si jednoduchý příklad vstupu L-systému pro vykreslení již naší známé Kochovy vločky.

- axiom := $F + +F + +F$
- přepisovací pravidla := $\{F \to F - F + +F - F\}$
- úhel rotace := 60°

1.3.1 Vykreslení 0. generace

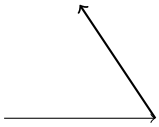
Vykreslením 0. generace se rozumí vykreslení axiomu. Uložme si axiom do proměnné instrukce. Předpokládejme že výchozí orientace štětce je vpravo. Pojďme se podívat na jednotlivé iterace kreslení:

1. iterace instrukce = $\mathbf{F} + +F + +F$, úhel orientace štětce = 0°
(tučně zobrazený znak je právě prováděná instrukce)

Obrázek 4: provedení 1. instrukce

2. a 3. iterace instrukce = $F{+}{+}F++F$, úhel orientace štětce = 120°
Dvakrát změníme orientaci štětce, pokaždé přičtením 60°. Náš obrázek se tedy po provedení těchto instrukcí nezměnil. Stále vypadá stejně:

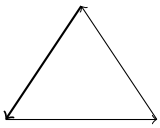
4. iterace instrukce = $F++\mathbf{F}++F$, úhel orientace štětce = 120°
Oproti 1. iterace nyní vykreslíme úsečku pod úhlem 120°.



Obrázek 6: provedení 4. instrukce

5. a 6. iterace instrukce = $F++F{+}{+}F$, úhel orientace štětce = 240°
Obdobně jako v 2. a 3. iteraci, dvakrát přičteme 60°.

7. iterace instrukce = $F++F++\mathbf{F}$, úhel orientace štětce = 240°
V poslední instrukci vykreslíme úsečku pod úhlem 240°. Dostáváme tedy obrázek našeho axiomu, který je stejný jako obrázek iniciátoru.



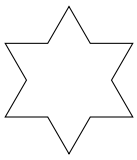
Obrázek 7: provedení 7. instrukce

1.3.2 Vykreslení 1. generace

Nejprve v našem axiomu nahradíme všechny znaky, které jsou klíčem nějaké hodnoty v přepisovacích pravidlech. Znaky, které nejsou klíčem žádné hodnoty v přepisovacích pravidlech necháme být. Dostaneme:

$$\text{instrukce} = F-F++F-F++F-F++F-F++F-F++F-F$$

Po provedení jednotlivých instrukcí:



Obrázek 8: 1. generace Kochovy vločky

1.3.3 Vykreslení n-té generace

Stačí pouze n-krát nahradit všechny znaky stejným způsobem, který je popsán pro vykreslení 1. generace.

2 Využité technologie

Program L-system, kterého se tato dokumentace týká by nemohl vzniknout bez různých technologií, kterým chci věnovat tuto kapitolu.

2.1 C#

Celý program byl napsán v programovacím jazyku C#. Tento programovací jazyk jsme zvolil z důvodu jeho jednoduchosti, výkonnosti a možnosti objektově orientovaného programování.

2.2 WinForms

Pro jednoduché zadání vstupu a jeho případné editace jsem se rozhodl využít grafického rozhraní. Ačkoliv knihovna WPF je modernější, rozhodl jsem se naprogramovat rozhraní v knihovně WinForms ze dvou důvodů:

1. kompatibilita s UNIX / UN*X / Mac OS X / Windows
2. jednoduchá tvorba uživatelského rozhraní

2.3 Visual Studio 2017

U volby vývojové prostředí jsem neváhal. Konkurenční JetBrains Rider a MonoDevelop neobsahují designer pro tvorbu uživatelského prostředí aplikací využívajících knihovnu WinForms.

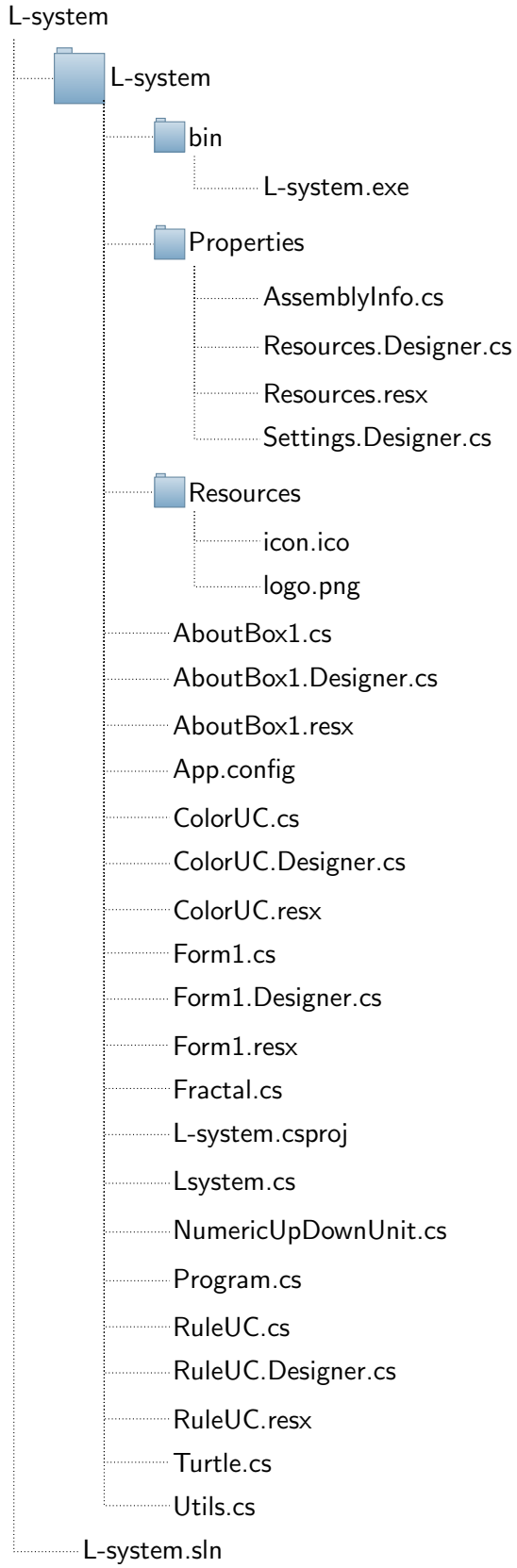
2.4 GDI+

Kvůli jednoduchosti se o vykreslování stará součást API OS Windows. Pro účely tohoto programu je dostačující.

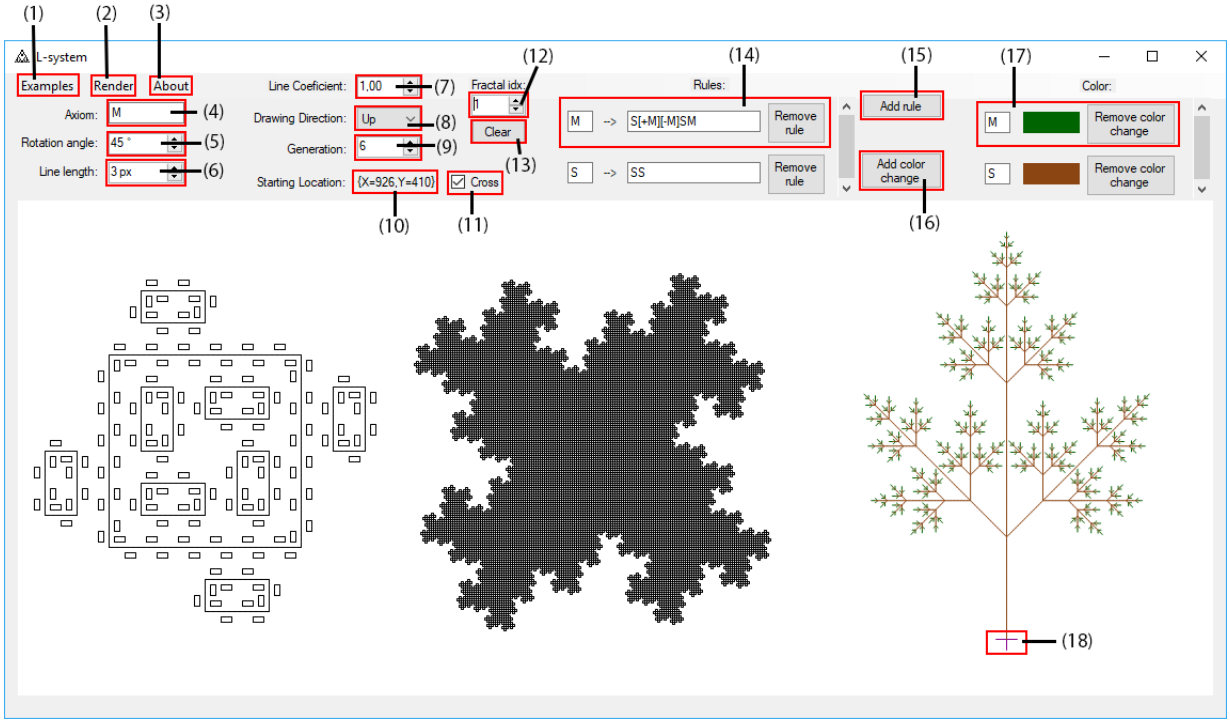
3 Program

V této kapitole popíšu jednotlivé části programu.

3.1 Struktura projektu



3.2 Uživatelské rozhraní



(1) Zobrazí nabídku předem připravených fraktálů k vykreslení.

(2) Vykreslí zadaný fraktál.

(3) Zobrazí informace o programu.

(4) Zadání axiomu.

(5) Zadání úhlu rotace.

(6) Zadání délky kroku (při kreslení úsečky nebo pohybu štětce).

(7) Zadání koeficientu zkracování kroku.

(8) Zadání výchozí orientace štětce.

(9) Zadání požadované generace fraktálu.

(10) Souřadnice počátku kreslení.

(11) Zobraz/skryj křížek zobrazující souřadnice počátku kreslení.

(12) Index právě aktivního fraktálu.

(13) Smaž právě aktivní fraktál.

(14) Komponenta pravidlo obsahující klíč, hodnotu a tlačítko ke smazání.

(15) Přidání pravidla.

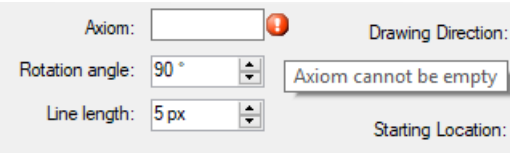
(16) Přidání změny barvy.

(17) Komponenta změny barvy obsahující klíč, hodnotu a tlačítko ke smazání.

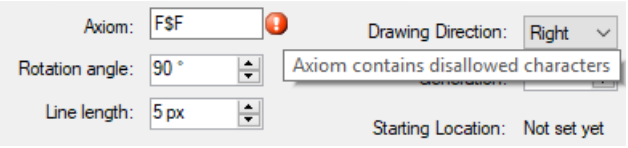
(18) Křížek zobrazující souřadnice počátku kreslení.

3.3 Ověřování vstupu

Program kontroluje vstupy po opuštění textových polí. Následně pomocí třídy `ErrorProvider` upozorní na neplatný vstup a zakáže tlačítko (2).

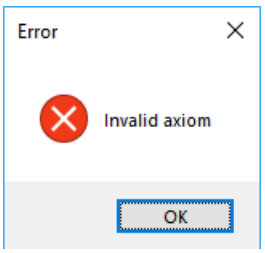


prázdný axiom

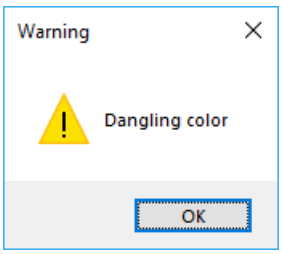


neplatný znak v axiomu

V případě, že uživatel neopustí textové pole a stiskne tlačítko (2), provede se kontrola vstupu ihned po stisku před vykreslením.



prázdný axiom



barva s chybějícím klíčem

3.4 Složky programu

L-system/bin Obsahuje jediný soubor a tím je spustitelný soubor programu.

L-system/Properties Jedná se o několik souborů obsahujících informace, verzi a název programu.

L-system/Resources Obsahuje ikonku aplikace a logo, které se zobrazuje při stisknutí tlačítka (3).

3.5 Lsystem/

V následujícím textu kapitoly **3.5** budeme předpokládat, že se pohybujeme pouze v této složce.

3.5.1 Lsystem.cs a Turtle.cs

Jde o nejzajímavější soubory programu. Nejprve si popíšeme nejdůležitější součásti třídy `Lsystem`, nacházející se v souboru Lsystem.cs.

Pro vytvoření instance nám stačí axiom (`string`), můžeme však předat i přepisovací pravidla (`Dictionary<char, string>`), ty lze však předat později pomocí vlastnosti `Rules`. Po předání přepisovacích pravidel se můžeme používat funkci `void NthGeneration(int generation)`, které do vlastnosti `Sentence` uloží instrukce pro nakreslení fraktálu. Tyto instrukce bude zpracovávat třída `Turtle`, nacházející se v souboru Turtle.cs, která se stará o vykreslení fraktálu.

Tímto končí funkčnost třídy `Lsystem`. Přesuneme se ke třídě `Turtle`. Pro vytvoření instance této třídy už potřebujeme více dat.

```
public Turtle(string sentence, Panel panel, double lineLength,
              double rotationAngle, Point startingPoint,
              Point directionPoint, Dictionary<char, Color> colors,
              double lineLengthCofficient = 1)
```

string sentence Hned prvním parametrem jsou instrukce (vlastnost Sentence) z třídy `Lsystem`.

Panel panel Dalším nezbytným parametrem je panel na který se má kreslit.

double lineLength Délka kroku.

double rotationAngle Úhel, který se přičte/odečte k aktuálnímu úhlu při zpracovávání instrukce $+/-$.

Point startingPoint Souřadnice počátku kreslení.

Point directionPoint Jednotkový vektor určující výchozí orientaci štětce.

Dictionary<char, Color> colors Asociativní pole, klíč je znak (instrukce) a hodnota je barva. Při zpracovávání instrukce kreslení čáry se kreslí barvou takovou, která je asociována s klíčem, pokud klíč neexistuje v poli, kreslí se černou barvou.

double lineLengthCoefficient = 1 Koeficient zkracování délky kroku. Délka kroku se tímto koeficientem vynásobí po každé provedené instrukci kresby nebo posunu. Vzhledem k tomu, že obvykle délku kroku chceme konstantní, je výchozí hodnota nastavena na 1.

Soubor Turtle.cs obsahuje dvě struktury. První je struktura `DrawInfo`. Její implementace vypadá takto:

```
struct DrawInfo
{
        public PointF Start { get; set; }
        public PointF End { get; set; }
        public Pen Pen { get; set; }
}
```

Instance této struktury budou reprezentovat jednotlivé čáry fraktálu, tyto se budou ukládat do kolekce, z které se pak fraktál bude možné překreslit. Druhou strukturou souboru Turtle.cs je `State`, jejíž implementace vypadá následovně:

```
struct State
{
        public PointF Location { get; set; }
        public double CurrentAngle { get; set; }
}
```

Obsahuje tedy aktuální stav, pozici štětce a jeho orientaci vůči té výchozí. Tyto dvě struktury se nám budou hodit pro popsání nejdůležitější metody `void Render()`, která vykreslí fraktál. Funguje následovně:

Nejprve budeme potřebovat několik proměnných:

Instanční

- `List<DrawInfo> drawInfo = new List<DrawInfo>();`

Lokální

- `State currentState = new State {Location = StartingPoint, CurrentAngle = 0};`
- `Stack<State> states = new Stack<State>();`

Nyní budeme iterovat znak po znaku vlastnoti `Sentence`, nechť právě zpracovávaný znak je `Sentence[i]`.

Pokud se jedná o nějaký znak z těchto čtyř: $+ - [\ ]$ tak v případě prvních dvou pouze přičtu / odečtu `RotationAngle` k `currentState.CurrentAngle`. V posledních dvou případech uložím aktuální stav na zásobník `states` / nastavím `currentState` na vrchol zásobníku a ten smažu.

Nyní zbývají znaky posouvající štětec $a - z$ a znaky kreslící úsečku $A - Z$, které však pozici štětce taktéž posunou. V obou případech tedy potřebujeme zjistit vektor posunu souřadnic štětce. Ten vypočítáme tak, že `DirectionPoint` orotujeme o `currentState.CurrentAngle` a vynásobíme `LineLength`.

Pokud `Sentence[i]` je znak $a - z$ tak k `currentState.Location` přičteme vektor posunu souřadnic a máme hotovo.

V případě znaku $A - Z$ si uložíme starou a novou pozici štětce do kolekce `drawInfo` spolu s kreslícím perem `Pen`, kterému nastavíme správnou barvu z asociativního pole `Colors` (pokud existuje záznam, jinak černou).

Po ukončení iterování stačí pouze překreslit `Panel`:

```
private void Panel_Paint(object sender, PaintEventArgs e)
{
        e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
        for (int i = 0; i < drawInfo.Count; ++i)
        {
            e.Graphics.DrawLine(drawInfo[i].Pen, drawInfo[i].Start.X,
                                drawInfo[i].Start.Y, drawInfo[i].End.X,
                                drawInfo[i].End.Y);
        }
    }
```

3.5.2 Fractal.cs

Jednoduchá třída sdružující kompletní informace o fraktálu.

```
class Fractal
{
        public Lsystem Lsys { get; set; }
        public Turtle Turtle { get; set; }
}
```

3.5.3 Form1.cs

Obsahuje třídu `LsystemForm : Form`. Její kód se spouští při startu aplikace. Zejména má na starosti:

- Nastavení vícenásobného bufferování panelu, na který se kreslí fraktály.
- Vykreslení (18)
- Veškeré reakce programu při interakci s uživatelským rozhraním

Její součástí je i kód 10 testovacích fraktálů, která se dají použít jako testovací data.

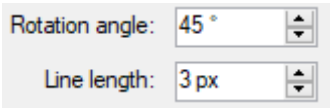
3.5.4 User Controly

RuleUC.cs Součástí je třída `RuleUC : UserControl`, přijímá znak (klíč) a k němu asociuje textový řetězec (hodnota), který uživatel zadá do textového pole. Viz (14).

ColorUC.cs Obsahuje třídu `ColorUC : UserControl`, která se stará o uživatelský vstup kreslícího znaku a asociuje k němu barvu, kterou uživatel vybere přes `ColorDialog`. Viz (17).

3.5.5 NumericUpDownUnit.cs

Vzhledem k tomu, že controla `NumericUpDown`, která je součástí WinForms neumožňuje zobrazit jednotku veličiny, využil jsem mírně upravenou verzi dostupnou z [3].



Obrázek 12: stupně a pixely

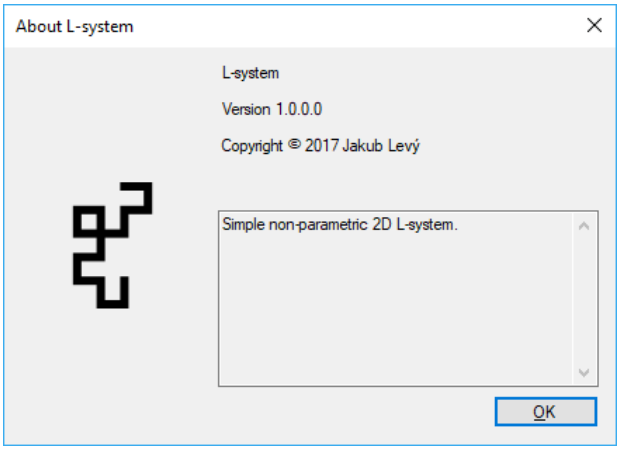
3.5.6 Utils.cs

Součástí je statická proměnná, regulární výraz, který se používá pro kontrolu vstupu axiomu a přepisovacích pravidel. Dále se zde nachází 5 statických metod:

1. zobrazuje `MessageBox` s chybovou hláškou
2. násobí vektor skalárem
3. sčítá dva vektory
4. na vstup dostane vektor a úhel, vrátí nový patřičně orotovaný vektor
5. převede vektor na nový vektor s nezápornými komponentami

3.5.7 AboutBox1.cs

Formulář obsahující informace o programu, zobrazí se při stisku tlačítka (3).

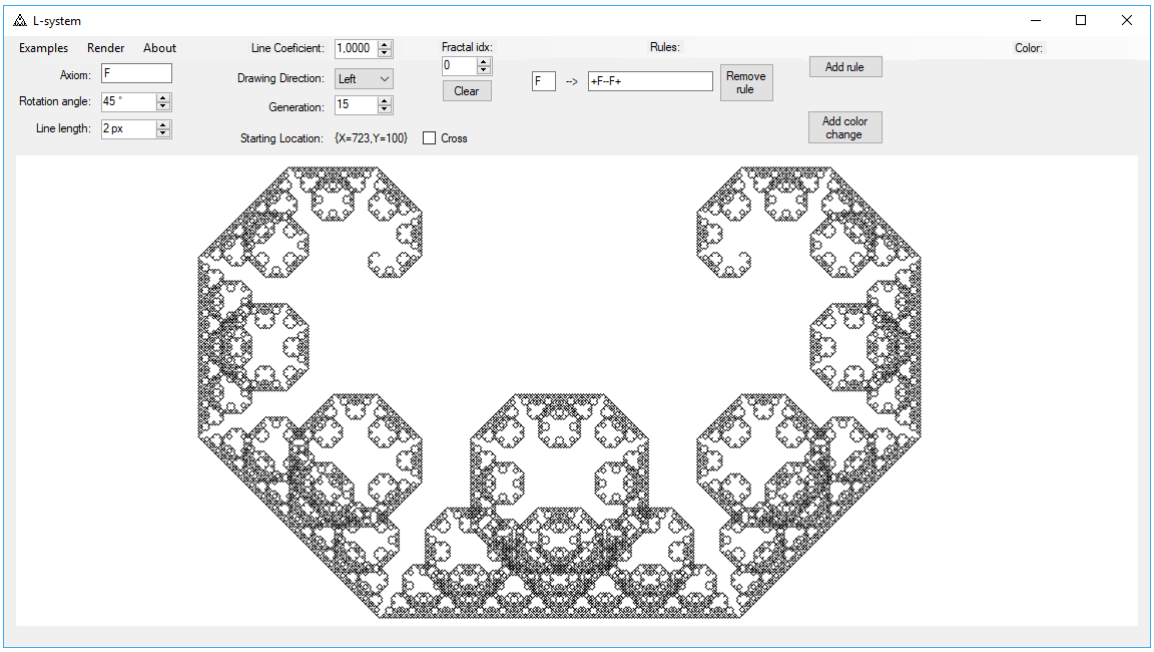


3.5.8 Další soubory

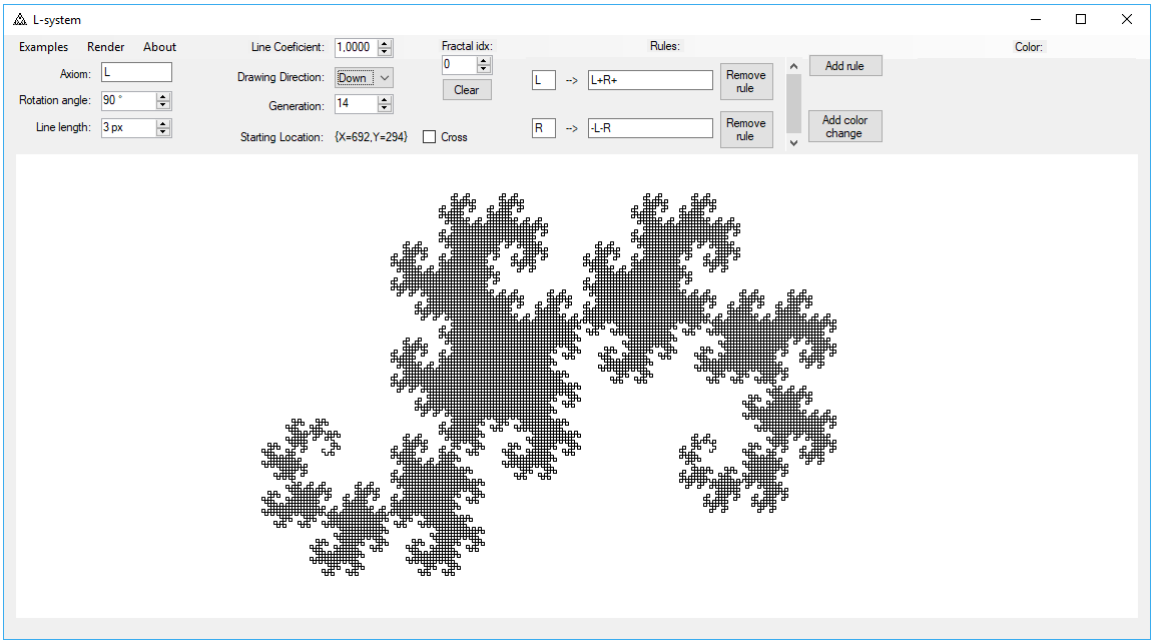
App.config XML soubor obsahující verzi prostředí .NET Framework nutnou ke spuštění programu.

L-system.csproj Soubor jehož součástí jsou informace o souborech přítomných v projektu Visual Studia

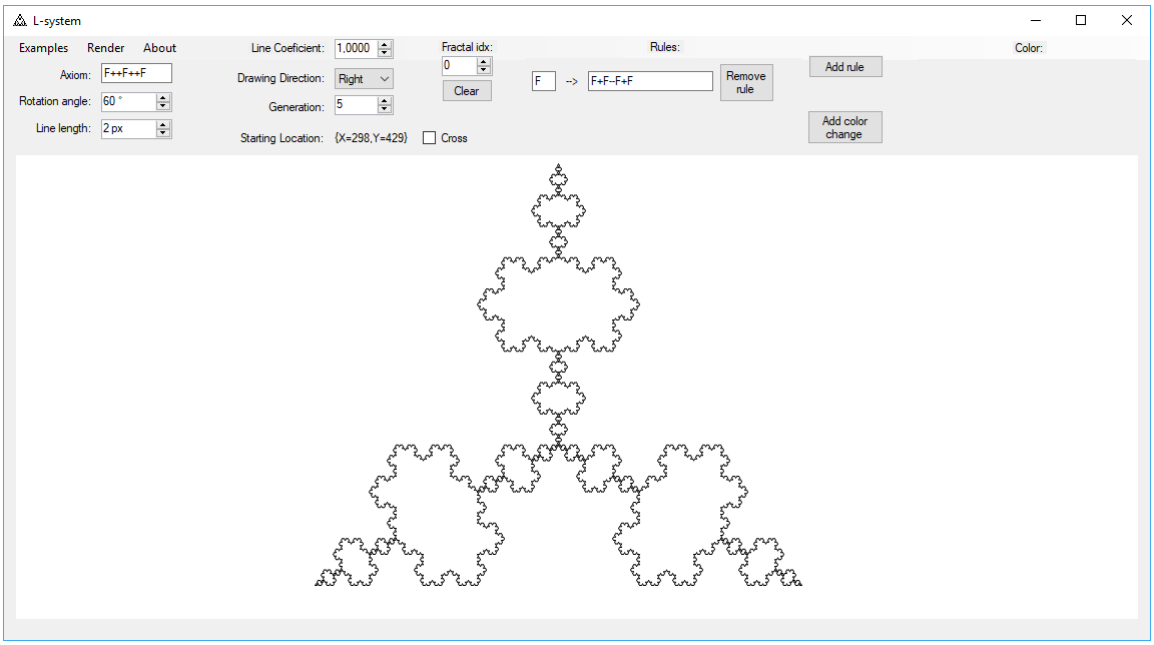
4 Další hezké fraktály



Obrázek 14: Lévyho C křivka



Obrázek 15: Dračí křivka



Obrázek 16: Kochova antivločka

5 Závěr

Dále by bylo možné a zajímavé rozšířit program na:

1. PL-system, který ke každému symbolu asociuje libovolný počet parametrů.
2. randomizující L-system, tedy L-system, který dostane nějakou pravděpodobnost na znáhodnění délky kroku a úhlu rotace
3. případně ještě L-system, který ke každé instrukci asociuje několik přepisovacích pravidel, která vybírá na základě nějakého parametru

Nicméně po implementaci těchto vylepšení by tento program byl již mnohem více složitý.

Seznam obrázků

Zdroje

[1] PRZEMYSLAW PRUSINKIEWICZ, Aristid Lindenmayer a WITH JAMES S. HANAN .. [ET AL.]. The algorithmic beauty of plants. New York: Springer-Verlag, 1996. ISBN 9780387946764.

[2] PRUSINKIEWICZ, Przemyslaw. Graphical applications of L-systems. Regina: Dept. of Computer Science, University of Regina, 1985. ISBN 9780773100350.

[3] `https://stackoverflow.com/questions/5921446/having-text-inside-numericupdown-control-after-the-number`