

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Generátor spirolaterálních objektů

BAKALÁŘSKÁ PRÁCE

Martin Bakeš

Brno, jaro 2008

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Jiří Chmelík

Poděkování

Děkuji prof. Ing. Ivo Serbovi CSc. a Mgr. Jiřímu Chmelíkovi ze jejich odborné vedení práce.

Shrnutí

Tato práce se věnuje spirolaterálním objektům, jejich tvorbě a využití, včetně ukázek tohoto využití. Dále je zde popsán vývoj a prostředí generátoru těchto objektů. Naznačuje také další možnosti, které spirolaterální objekty nabízejí a tím i možnosti, jak generátor rozšířit.

Klíčová slova

spirolaterální křivka, segment, úhel otočení, počet opakování, opačný krok, HPGL/PLT, SVG, Delphi

Obsah

1	Úvod	1
2	Spirolaterální křivky	2
3	Popis programu	6
3.1	Použité programové prostředky	6
3.1.1	Vývojové prostředí Delphi a jazyk ObjectPascal	6
3.1.2	Jazyk HPGL/PLT	6
3.1.3	Jazyk SVG	8
3.2	Uživatelské rozhraní aplikace	10
3.3	Implementace programu	12
3.3.1	Výpočet bodů spirolaterály	12
3.3.2	Vykreslení spirolaterály	14
3.3.3	Ukládání spirolaterály	19
3.3.4	Načítání spirolaterály	21
4	Návrhy k dalšímu vylepšení aplikace	22
5	Závěr	23
	Literatura	24
A	Ukázky upravených spirolaterál	25
B	Obsah přiloženého CD	28

Kapitola 1

Úvod

Téma týkající se spiroleterálních křivek jsem si vybral nejen proto, že to jsou matematicky zajímavé objekty, ale hlavně proto, že jsou zajímavé i po výtvarné stránce. Pro výtvarníky jsou tyto křivky velmi vhodné, neboť mohou sloužit jednak jako samostatné výtvarné dílo, ale i jako základ, který je možné použít při dalším zpracování. Výtvarník se při něm může nechat inspirovat různými druhy čáry, kterými je možné v aplikaci křivku vykreslit nebo může použít jen body, kterými křivka prochází. Další výhodou pro výtvarníka je to, že spirolaterální křivky jsou popsány jen několika málo parametry, jejichž změnou je možné jednoduše dostávat nové tvary.

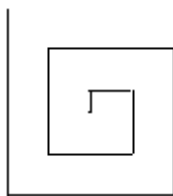
Programovou částí práce je aplikace sloužící ke generování spirolaterálních křivek. Ta je napsána v jazyce ObjectPascal ve vývojovém prostředí Borland Delphi 7 a je tedy spustitelná pouze na platformě MS Windows. Program umožňuje vykreslovat nejen „základní“ spirolaterálu tvořenou úsečkami, ale i několika dalšími způsoby jako například bézierovými křivkami nebo kruhovými oblouky. Dále je možné křivku ukládat do několika různých formátů. Na výběr jsou dva formáty rastrové grafiky, dva formáty vektorové grafiky a jeden formát sloužící k uchování konfigurace křivky.

Jelikož některé spirolaterální křivky svým tvarem připomínají ornamentální vzory, nemusí spirolaterály sloužit jen jak výtvarné hříčky pro potěchu oka. V praxi se mohou použít upravené spirolaterály například jako vzory tapet, ozdoby na šaty nebo jako ornamety na známky a bankovky.

Kapitola 2

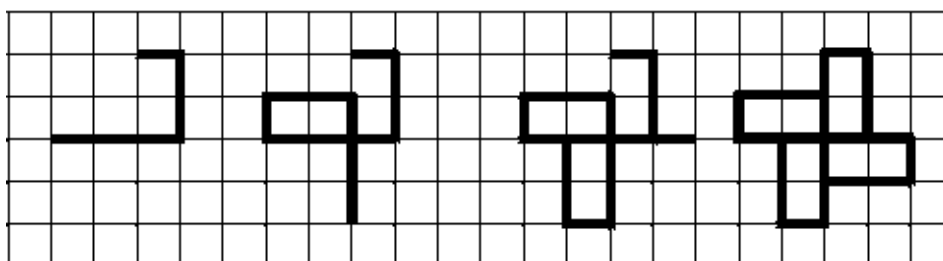
Spirolaterální křivky

Pojem spirolaterální křivka se poprvé objevil v práci Franka C. Oddse[8], který název spirolaterála odvodil ze dvou slov: *lateral*, znamenající rovný povrch a *spiro*, odkazující na spirálovitý tvar, protože původní sada spirolaterál byla vytvořena z „pravoúhlých spirál“. To je množina úseček (segmentů), z nichž první má délku jedné jednotky a každá následující úsečka je o jednu jednotku delší, než úsečka předcházející. Navíc je každý segment od předchozího pootočen o konstantní úhel, čímž vzniká spirálovitý tvar.



Obrázek 2.1: „Pravoúhlá spirála“

Spirolaterální křivka je určena třemi parametry. Prvním je počet segmentů, druhým je počet opakování a posledním je úhel, o který má být změněn směr jednotlivých segmentů. Pomocí počtu segmentů a úhlu otočení je možné vytvořit „pravoúhlou spirálu“ – první část spirolaterály. Z koncového bodu tohoto základu se začne stejným způsobem vykreslovat další část spirolaterály. Toto se opakuje tolikrát, kolik je zvolen počet opakování. Odds také stanovil formu pro jednoduchý zápis spirolaterály. Počet segmentů značí řád spirolaterály a uvádí se jako základ. Úhel otočení se zapisuje pomocí dolního indexu, proto potom 3_{90} značí křivku se třemi segmenty a úhlem otočení 90 stupňů.

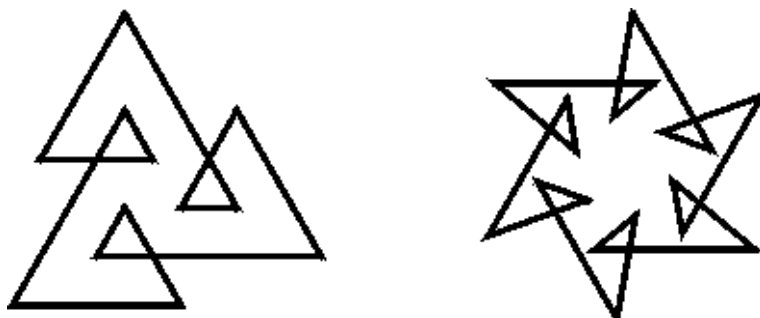


Obrázek 2.2: Vznik spirolaterály 3_{90} se 4 opakováními

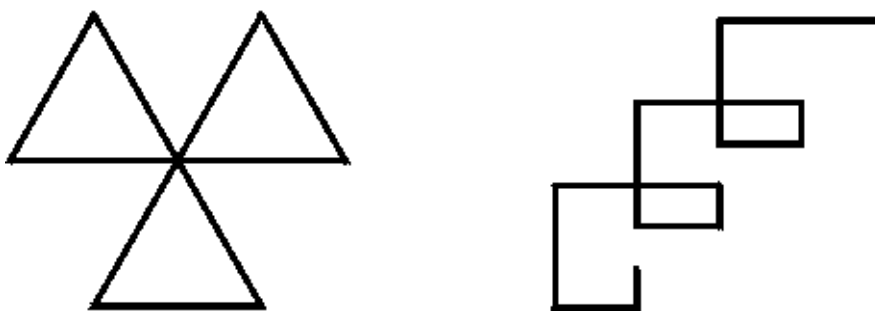
Matematicky zajímavým problémem při tvorbě spirolaterál je otázka uzavření křivky, která ještě ještě není zcela vyřešena. Uzavření křivky znamená, že po provedení všech kroků skončí křivka v bodě, ve kterém se začala vykreslovat. Odds tvrdil, že pomocí každého úhlu, který dělí 180 stupňů beze zbytku, je možné vygenerovat uzavřenou spirolaterálu. Toto tvrzení je pravdivé, ovšem nepokrývá všechny uzavřené křivky. Například křivka 3_{40} s šesti opakováními uzavřená je a přitom 40 nedělí 180 beze zbytku. Robert Krawczyk[4] při řešení problému uzavřenosti vycházel ze tří základních parametrů spirolaterál, a to z počtu segmentů, počtu opakování a úhlu otočení. Pro určení uzavřenosti vytvořil následující vztah. Pokud je celkový úhel otočení dělitelný 360, pak je křivka uzavřená.

celkový úhel otočení = úhel otočení \times počet segmentů \times počet opakování

Tento vztah platí pro libovolný úhel, ovšem také není úplně dokonalý. Spirolaterála nebude uzavřená, pokud je 360 stupňů dosaženo po jednom opakování.



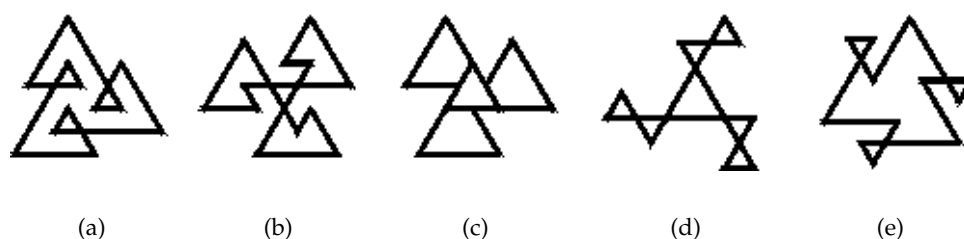
Obrázek 2.3: Ukázka uzavřené křivky založené na úhlu $180/n$ (4_{60}) a uzavřené křivky, založené na jiném úhlu (3_{40})



Obrázek 2.4: Křivka 2_{60} splňuje a křivka 4_{90} nesplňuje Krawczykovo pravidlo pro uzavřenost

Odds se dále zabýval spirolaterálami a možnostmi, jak je ozvláštnit. Přišel na to, že ne všechna otočení musí být ve stejném směru a některé úsečky se mohou otáčet o opačný úhel, než je zadáný. Tato úprava se nazývá opačné kroky. Navrhl také zápis takové spirolaterály. Ten pro křivku s šesti segmenty, úhlem otočení o devadesát stupňů a druhým a

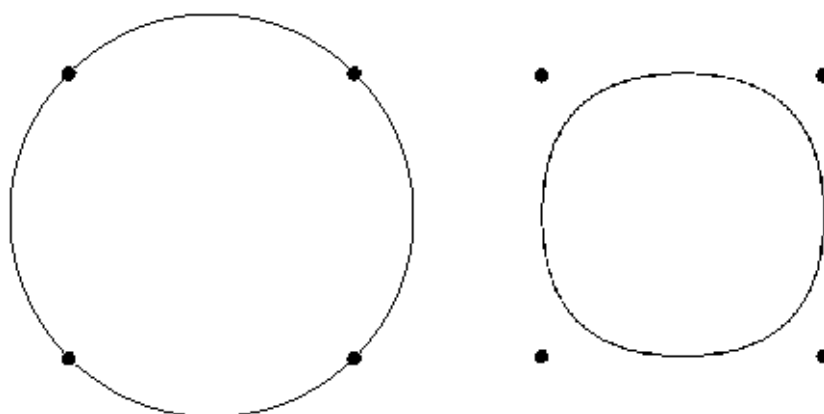
čtvrtým opačným krokem vypadá následovně: $6_{90}^{2,4}$. Díky této úpravě je možné vygenerovat mnohem více křivek, které nabývají nových, velmi zajímavých tvarů. Ovšem stejně jako u normálních spirolaterál, tak i u spirolaterál s opačnými kroky je stále nedořešená otázka uzavření křivky.



Obrázek 2.5: Křivky s opačnými kroky. (a) 4_{60} , (b) 4_{60}^1 , (c) 4_{60}^2 , (d) 4_{60}^3 , (e) $4_{60}^{1,2,3}$

Nyní by se mohlo zdát, že o spirolaterály budou mít zájem pouze matematici, kteří zkoumají jejich vlastnosti, jako je uzavření nebo různé parametry, pomocí kterých je možné vygenerovat nové, zajímavější křivky. Opak je ale pravdou. Jedním z lidí, kteří se významně zabývají spirolaterály je architekt Robert Krawczyk[5]. A protože architektura je řazena mezi umělecké obory, můžeme říci, že jeden z lidí, který se spirolaterály zabývá je umělec. Jak profesor Krawczyk[6] ukazuje, spirolaterály skrývají poměrně velký umělecký potenciál. K zajímavým výsledkům vede například obyčejná změna tloušťky čáry nebo vložení středové linky. Další možností, jak spirolaterály ozvláštnit, je nahrazení úseček různými obloučky, lomenými čarami či vlnovkami.

Pro informatiky-výtvarníky jsou spirolaterální křivky zajímavé z několika důvodů. Jedním z nich je malý počet parametrů, potřebný k vygenerování křivky. S tím souvisí poměrně snadné procházení množiny křivek a využití esteticky zajímavých křivek k dalšímu zpracování v nějakém grafickém programu. Další výhodou je, že spirolaterála je tvořená úsečkami. Z těchto úseček můžeme spočítat množinu koncových bodů, které křivku určují a ty následně použít jako řídicí body pro různé typy křivek, jako jsou například kruhové oblouky nebo kvadratická a kubická křivka.



Obrázek 2.6: Použití řídicích bodů k proložení kruhových oblouků a vytvoření kvadratické křivky

Kapitola 3

Popis programu

3.1 Použité programové prostředky

3.1.1 Vývojové prostředí Delphi a jazyk ObjectPascal

Program Generátor spirolaterál je vytvořen ve vývojovém prostředí Borland Delphi 7[1] v jazyce Object Pascal, který rozšiřuje jazyk Pascal o některé vlastnosti objektově orientovaného programování. Delphi je grafické vývojové prostředí určené pro vývoj aplikací pro platformu MS Windows. Toto prostředí umožňuje velmi jednoduchý návrh grafického uživatelského rozhraní aplikace, na jehož základě je vytvářena kostra zdrojového kódu, což poměrně hodně usnadňuje vývoj programu. Programování v Delphi je založeno na takzvaných komponentách. Jedná se o soubor funkcí, které vykonávají určitou činnost (zobrazení textu, obrázků...). Některé komponenty jsou v Delphi už integrované, jiné si lze už placeně nebo neplaceně stáhnout z Internetu, případně je možné si napsat vlastní komponentu. V tomto programu jsou používány převážně součásti komponenty VLC (Visual Component Library), která zapouzdřuje práci s ovládacími prvky MS Windows, jako jsou tlačítka, formuláře a popisky.

Jednou z mála nevýhod tohoto vývojového prostředí Delphi je ta, že vytvořené aplikace mohou běžet pouze v prostředí MS Windows. Pro kompilaci v operačním systému GNU/Linux sice vyvíjel Borland aplikaci Kylix, ta se ale příliš neujala, protože byla zaručena jen částečná přenositelnost kódu (mohly se používat jen některé komponenty) a její vývoj byl v roce 2002 ukončen.

Pro vývoj aplikace v prostředí Delphi jsem se rozhodl hlavně pro snadnost tvorby uživatelského rozhraní a také proto, že mám s vývojem v jazyce Object Pascal a v prostředí Delphi již určité zkušenosti.

3.1.2 Jazyk HPGL/PLT

Vektorový grafický formát HPGL (Hewlett-Packard Graphics Language)[3] byl vyvinut firmou Hewlett-Packard především jako komunikační formát pro své plottery. Díky své jednoduchosti jej ale začali používat i ostatní výrobci, a tak se tento formát rychle rozšířil a i když je už poměrně starý, stále se ještě používá. Tento formát jsem zvolil právě pro jednoduchost výsledného PLT souboru, do kterého se HPGL ukládá. Jedná se totiž o čistě textový formát ve kterém je každý řídicí nebo kreslicí příkaz zapsán pomocí dvojice ASCII znaků, za kterou mohou následovat číselné či textové parametry. Ty od sebe mohou být odděleny některým

3.1. POUŽITÉ PROGRAMOVÉ PROSTŘEDKY

ze znaků ",", (čárka), " " (mezera), "+" (znaménko plus) nebo "-" (znaménko minus). Jednotlivé příkazy jsou od sebe odděleny znakem ";" (středník) a slouží pro popis pohybu pera plotteru po papíru. Jazyk HPGL obsahuje celou řadu kreslicích příkazů, které umožňují vykreslovat úsečky, obdélníky, elipsy, oblouky, kruhové výseče a dokonce je možné vykreslovat i text. I když program generuje spirolaterály s různými typy oblouků, do PLT souboru je možné uložit křivku tvořenou pouze úsečkami, a to z prostého důvodu. Pro další zpracování spirolaterál v některém grafickém editoru jsou potřeba především koncové body úseček, ze kterých je možné dále vycházet a na to spirolaterála uložená v základní podobě dostačuje.

```
IN;  
VS32,1;  
PW1,1;  
LT;  
SP1;  
PU-457 383  
PD-457 383  
PD-457 550  
PD-124 550  
PD-124 50  
SP0;
```

Jak můžete vidět na ukázce, kde je uložena spirolaterála 3₉₀ s jedním opakováním, struktura PLT souboru je poměrně jednoduchá a skládá se ze dvou hlavních částí. První z nich je inicializace plotteru, která je v tomto případě neměnná, protože aplikace není určena ke komunikaci s plotterem a tato nastavení se dají v příslušných programech snadno změnit. Příkaz IN je první příkaz v souboru a je příkazem inicializace, který začíná jakoukoliv práci s plotterem. Za ním následuje několik příkazů, které nastavují vlastnosti plotteru. Jedním z nich je řídicí příkaz VS, který má dva parametry a nastavuje rychlost pohybu pera. Ten má význam pouze pokud chceme ovládat plotter a ne PLT soubor použít pouze jako formát pro uchování tvaru křivky. Proto jsem zvolil standardní rychlost pera, která je 32. Jako druhý parametr je číslo pera, pro které rychlost nastavujeme. Aplikaci používá pero jen jedno, proto stačí nastavit rychlost jen pro toto pero. Další důležitý příkaz je PW, který nastavuje šířku pera a má opět dva parametry. Prvním z nich je šířka pera, kterou jsem zvolil jednoduše 1, protože ve všech vektorových editorech je možné šířku pera následně změnit. Druhým parametrem je opět číslo pera, u kterého šířku nastavujeme, tedy 1. Další příkaz z inicializační části je LT. Ten nastavuje typ čáry, a protože ten je implicitně nastaven na normální čáru, není potřeba ho nijak nastavovat. Posledním z inicializačních příkazů je SP, který slouží k výběru pera, kterým se bude kreslit. A protože se doposud všechny parametry nastavovaly pro pero 1, tak se zvolí i zde.

V druhé části PLT souboru je popsán samotný objekt. K tomu slouží dva kreslicí příkazy. První z nich, příkaz PU, má dva parametry a slouží k přesunu zvednutého pera na zadané souřadnice. A protože je pero zvednuté, ještě stále se nic nevykresluje. K tomu slouží příkaz PD, který má také dva parametry a provede spuštění pera na zadaných souřadnicích. Následné opakované volání příkazu PD s různými souřadnicemi způsobí pohyb spuštěného

pera a tím vykreslení požadovaného tvaru. PLT soubor se ukončí příkazem SP0. Ten vybere pero číslo 0, které ovšem v souboru není definováno, tím dojde k uklizení pera 1 a k ukončení vykreslování.

3.1.3 Jazyk SVG

Možnost ukládat obrázek křivky do SVG jsem zvolil proto, že SVG je jeden z nejmodernějších standardů pro vektorovou grafiku a myslím si, že se tento formát bude používat i v budoucnosti, a to ze dvou důvodů. Za prvé proto, že se byl vyvinut především pro zobrazování vektorové grafiky na internetu, kde doposud podobný nástroj chyběl. Ne na všechno jsou totiž vhodné rastrové obrázky, které, na rozdíl od SVG, při zvětšení ztrácí kvalitu. Druhým důvodem je, že SVG nabízí široké možnosti a jeho zdrojový kód lze jednoduše upravovat podobně jako HTML.

SVG (Scalable Vector Graphics)[9] je jazyk, který popisuje dvourozměrnou grafiku pomocí XML, proto je svou strukturou podobný HTML, obsahuje ovšem specifické grafické elementy. SVG definuje tři základní typy grafických objektů, a to vektorové objekty (úsečka, lomená čára, obdélník, kružnice, elipsa, mnohoúhelník a křivka), rastrové obrazy a textové objekty. Objekty jsou vykreslovány ve stejném pořadí, ve kterém jsou zapsány ve zdrojovém souboru a na všechny lze aplikovat následující efekty: ořezávání objektů, alpha masking (masky průhlednosti) a filtrování obrazu (např. konvoluce). SVG také umožňuje vytvářet animace a upravovat grafiku pomocí skriptů.

SVG soubor se skládá ze dvou hlavních částí. První z nich je hlavička souboru, která je téměř neměnná a může vypadat například takto.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

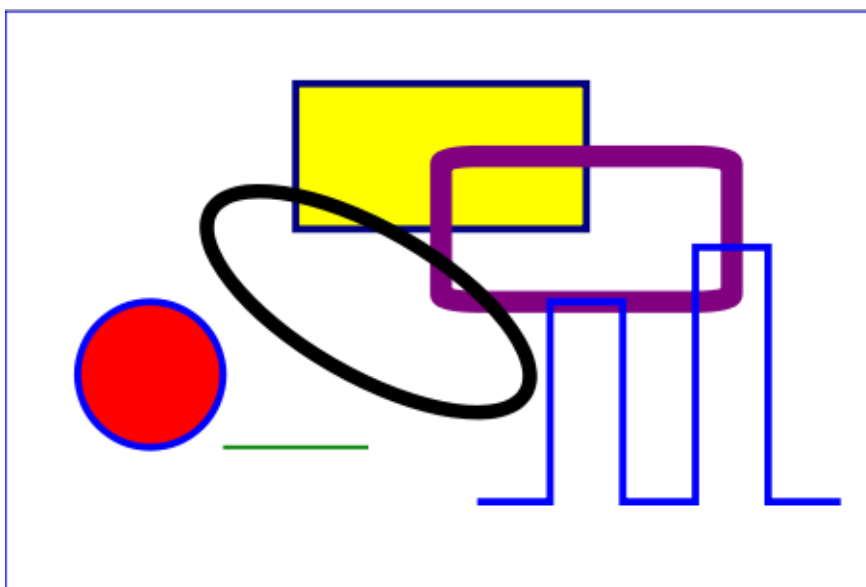
V každém XML souboru se značky zapisují do špičatých závorek a ne jinak tomu je i v SVG. Na prvním řádku je definována verze XML, ve které je dokument napsán. V tomto případě, stejně jako v aplikaci, je použito XML verze 1.0. Na druhém a třetím řádku je definováno DTD (Document Type Definition), na kterém je dokument založen. DTD říká, které elementy a atributy se mohou v dokumentu vyskytnout a v jakých mohou být vztazích. Přiřazením k určitému DTD tedy získáváme možnost kontrolovat správnou strukturu dokumentu.

V druhé části SVG souboru jsou již definovány jednotlivé objekty, které se mají vykreslit a může vypadat například takto:

```
<svg width="12cm" height="8cm" viewBox="0 0 1200 800"
xmlns="http://www.w3.org/2000/svg" version="1.1">
<title>Ukazka</title>
<rect x="400" y="100" width="400" height="200"
fill="yellow" stroke="navy" stroke-width="10" />
<rect x="600" y="200" width="400" height="200" rx="50" ry="10"
fill="none" stroke="purple" stroke-width="30" />
<circle cx="200" cy="500" r="100">
```

3.1. POUŽITÉ PROGRAMOVÉ PROSTŘEDKY

```
    fill="red" stroke="blue" stroke-width="10" />
<ellipse transform="translate(500 400) rotate(30)"
  rx="250" ry="100"
  fill="none" stroke="black" stroke-width="20"/>
<line x1="300" y1="600" x2="500" y2="600"
  stroke="green" stroke-width="5" />
<polyline fill="none" stroke="blue" stroke-width="10"
  points="650,675 750,675 750,400 850,400
    850,675 950,675 950,325 1050,325
    1050,675 1150,675" />
<rect x="1" y="1" width="1198" height="798"
  fill="none" stroke="blue" stroke-width="2"/>
</svg>
```



Obrázek 3.1: Takto vypadá SVG zapsané pomocí předcházejícího kódu.

Základním elementem SVG obrázku je element `<svg>`, který může obsahovat několik atributů. Já se budu věnovat pouze těm, které jsou použity v aplikaci. Atributy `width` a `height` definují šířku a výšku kreslicího plátna. Pokud nejsou definovány, použije prohlížeč hodnotu 100 %. V ukázkovém kódu jsou jako jednotky použity cm. V SVG je ale možné, stejně jako v CSS, použít px, mm, pt, em a další. Dále je v elementu `svg` definován implicitní jmenný prostor, který zajistí, že všechny další elementy budou náležet tomuto jmennému prostoru. Na obrázku výše jsou zobrazeny základní objekty bez jakýchkoliv transformací, které je možné vykreslit. V aplikaci jsou ovšem použity pouze elementy `<circle>`, `<rect>` a `<polyline>`, proto se budu věnovat jen těmto třem.

Element `<circle>` má několik atributů. Atributy `cx` a `cy` nastavují souřadnice středu kružnice na kreslicím plátně a atribut `r` nastavuje její poloměr. Pomocí atributu `fill` se nastavuje

barva výplně kružnice. Tu je možné zadat buď názvem barvy, nebo hexadecimálním kódem. Atribut *stroke* nastavuje barvu obrysové čáry a opět je možné ji zadat dvěma způsoby, stejně jako u atributu *fill*. Posledním atributem je *stroke-width*, který nastavuje šířku obrysové čáry.

V elementu `<rect>` se vyskytují atributy *x* a *y*, které určují souřadnice horního levého rohu obdélníku. Pomocí atributů *width* a *height* se nastavuje šířka a výška vykreslovaného obdélníku. Stejně jako v elementu `<circle>` se i zde vyskytují atributy *fill*, *stroke* a *stroke-width*, které mají opět stejné použití.

V elementu `<polyline>` se opět vyskytují atributy *fill*, *stroke* a *stroke-width*, jejichž použití zůstává i tady stejné. Posledním atributem je *points*. Ten určuje, kterými body má čára procházet. Tyto body se zadávají ve formě dvou čísel vzájemně oddělených čárkou (souřadnice *x* a *y* daného bodu). Tyto dvojice čísel jsou mezi sebou odděleny mezerou. SVG dokument končí uzavřením kořenového elementu pomocí značky `</svg>`.

3.2 Uživatelské rozhraní aplikace

Okno aplikace je rozděleno do dvou hlavních částí. V levé části okna se nachází prvky, které slouží k nastavení jak základních parametrů křivky (počet segmentů, počet opakování, úhel otočení a opačné kroky), tak prvky sloužící k nastavení efektů, jak křivku ozvláštnit (barva a síla vnější a vnitřní čáry, barva pozadí a styl čáry).

Pro popisky je použita standardní komponenta *Label*, pro zadávání číselných parametrů je použita komponenta *Edit*, která je v některých případech propojena s komponentou *UpDown*, která umožňuje kliknutím na příslušnou šipku zvětšovat nebo zmenšovat číselné hodnoty v poli pro zadávání parametrů. Tato možnost se vyskytuje jen u prvků, které umožňují nastavit pouze celočíselné hodnoty.

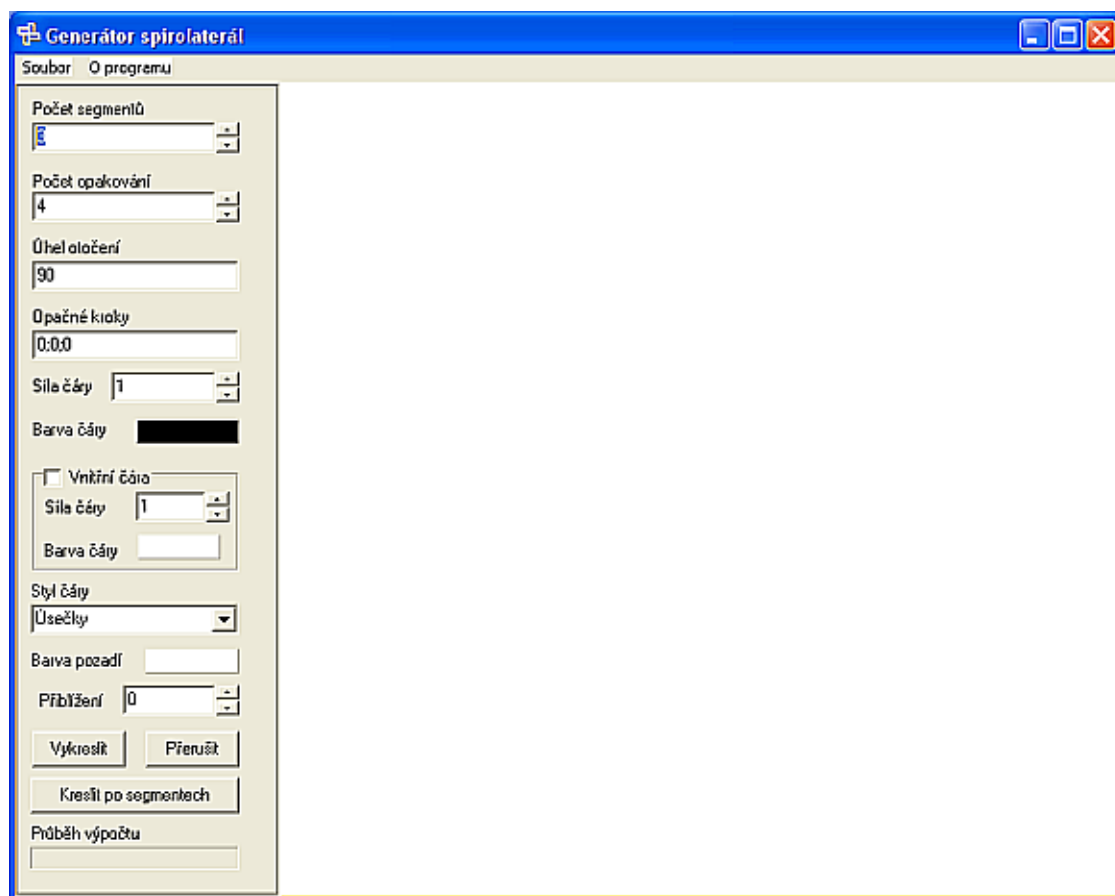
O výběr, zda použít vnitřní čáru, či nikoli se stará zaškrťovací políčko, v Delphi komponenta nazývaná *CheckBox*. Prvky nastavující vlastnosti vnitřní čáry jsou vloženy do komponenty *GroupBox*, která okolo nich vytváří rámeček a tím dává uživateli na vědomí, že se všechno, co je v této komponentě obsaženo, týká pouze vnitřní čáry.

Výběr barvy, ať už vnitřní, či vnější čáry nebo pozadí, umožňuje komponenta *Panel*. Po najetí myši na tento prvek se změní ukazatel myši a tím je uživateli oznámeno, že po kliknutí na panel nastane nějaká akce. Kliknutí na Panel vyvolá standardní dialogové okno pro volbu barvy, v Delphi realizované pomocí komponenty *ColorDialog*. Pokud uživatel zvolí novou barvu, tak se panel, ze kterého bylo okno vyvoláno obarví na nově vybranou barvu.

Pro výběr stylu čáry je použit rozbalovací seznam, který je v Delphi realizován pomocí komponenty *ComboBox*. Na výběr jsou prosté úsečky, body, obloukové úseky, vnitřní a vnější obloučky, vlnovka a kvadratická a kubická křivka.

Tlačítka jsou realizována pomocí komponenty *Button*. Tlačítko *Vykreslit* okamžitě vykreslí křivku se zadanými parametry. Kliknutím na tlačítko *Kreslit po segmentech* se vykreslí vždy jen jeden segment spoluterály. Uživatel tak má možnost postupným klikáním sledovat vznik celé křivky. Pokud uživatel zadá do počtu segmentů nebo opakování, pří-

3.2. UŽIVATELSKÉ ROZHRANÍ APLIKACE



Obrázek 3.2: Hlavní okno aplikace

padně do obojího, příliš velká čísla, výpočet bodů křivky se může nepříjemně prodloužit. Proto je zde k dispozici poslední tlačítko s názvem *Přerušit*. Po kliknutí na něj se výpočet bodů přeruší a uživatel může pokračovat v práci s programem. Posledním prvkem je ukazatel průběhu výpočtu bodů, který je vytvořen pomocí komponenty *ProgressBar*.

Všechny předchozí ovládací jsou umístěny na panel zvaný *ScrollBar*. Ten při zmenšení okna ve vertikálním směru zajistí to, že se na pravé straně tohoto panelu objeví rolovací lišta, která umožní uživateli posouvat se po panelu, a tím zpřístupnit ovládací prvky, které by v důsledku zmenšení okna nebyly vidět.

Po spuštění programu jsou všechny parametry nastaveny na implicitní hodnotu. Uživatel tak vidí, jaké hodnoty by měl do jednotlivých parametrů zadávat. To je důležité zejména u pole pro opačné kroky, protože ty se zadávají ve tvaru řady čísel oddělených středníky, což by mohlo bez vyplněné implicitní hodnoty činit uživateli problémy.

Pravá část okna slouží k vykreslování spirolaterálních křivek a je vytvořena komponentou *Image*. Ta umožňuje nejen vykreslovat, ale i ukládat to, co je na ní vykresleno do formátu BMP. O tom ale dále v popisu implementace aplikace.

Poslední částí uživatelského rozhraní aplikace je hlavní menu. To je tvořeno pomocí komponenty *MainMenu*. Menu má dvě hlavní položky. První z nich je *Soubor*, která nabízí dobře známé položky *Nový*, *Otevřít...*, *Uložit jako...* a *Konec*. Výběrem položky *Nový* se smaže vykreslená spirolaterála a nastaví se implicitní hodnoty parametrů. Pokud vykreslená křivka nebyla uložena, tak je uživatel dotázán, jestli si ji přeje uložit. Kliknutím na *Otevřít...* se vyvolá dialogové okno, v Delphi tvořené komponentou *OpenDialog*, pro výběr a otevření uložené křivky, která je uložena ve formátu SLC (*SpiroLateral Curve*). Jedná se pouze o textový soubor, ve kterém jsou uloženy všechny parametry spirolaterály, které je možné v aplikaci nastavit. Volba *Uložit...* otevře dialogové okno sloužící k ukládání vytvořené spirolaterály. To je vytvořeno pomocí komponenty *SaveDialog*. Křivku je možné ukládat do dvou rastrových formátů (BMP, JPG), do dvou vektorových (PLT, SVG) a v poslední řadě je možné uložit konfiguraci křivky do formátu SLC. To se může hodit, pokud uživatel vytvoří zajímavou spirolaterálu a rád by si ji uchoval pro další použití. Výběrem položky *Konec* se aplikace ukončí, pokud aktuální spirolaterála nebyla uložena, je uživatel opět dotázán na uložení.

Výběrem položky *O aplikaci* se otevře nové okno se stručnými informacemi o aplikaci a jejím tvůrci. Všechny položky v menu je možné vyvolat i pomocí standardních klávesových zkratk.

3.3 Implementace programu

3.3.1 Výpočet bodů spirolaterály

Jádro programu tvoří procedura sloužící k výpočtu bodů spirolaterály. Z nich se potom vychází při vykreslování křivek s libovolným typem čáry. Jelikož se výpočet bodů provádí pomocí vektorových transformací v rovině, bylo možné použít standardní postupy lineární algebry.

Hlavní procedury na výpočet bodů a vykreslení křivky jsou umístěny do komponenty zvané *ActionList*. Ta umožňuje takzvaným „akcím“ nastavovat různé vlastnosti, jako klávesové zkratky, název akce nebo to, jestli je akce povolena nebo zakázána. Tyto akce je potom možné navázat na ovládací prvky programu (tlačítka, hlavní menu) a ty od nich tyto vlastnosti přebírají. Takto je navázána například akce *ActKresli* na tlačítko *Vykreslit*.

Po kliknutí na toto tlačítko se spustí právě akce *ActVykresli*, která dále obsluhuje vykreslení spirolaterály. Nejprve je nutné načíst všechny potřebné parametry a vypočítat body, proto se zavolá akce *ActSpocitej*. V první části této procedury se načítají vstupní data. Pro každý parametr je použit kód velmi podobný tomuto:

```
val (EdtOpak.Text, opakovani, code);
if (code <> 0) or (opakovani < 0) or (opakovani > 10000) then
begin
  MessageDlg('Příslušné chybové hlášení', mtError, [mbOK], 0);
  EdtOpak.SetFocus;
  if code <> 0 then EdtOpak.SelStart := code-1
    else EdtOpak.SelStart := 0;
```

```

EdtOpak.SellLength := length(EdtOpak.text);
kresli := false;
exit;
end;

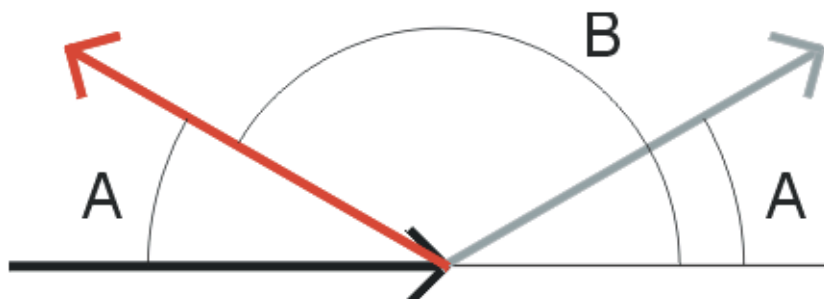
```

Funkce `val()` převádí řetězec na číslo a má tři parametry. Prvním z nich je řetězec, ze kterého se má číslo načíst, druhým je proměnná, do které se číslo uloží a třetím je proměnná, do které se uloží nula v případě, že převod proběhl bezchybně. V opačném případě se do ní uloží pozice prvního znaku, který nebylo možné na číslo převést. Pokud se převod nezdařil nebo pokud bylo zadané číslo mimo stanovený rozsah, vypíše se chybové hlášení, označí se chybný text v příslušném poli, aby uživatel viděl, kde nastala chyba a procedura pro zpracování hodnot a výpočet bodů se ukončí

Pokud načtení parametrů proběhne bez chyby, běh procedury pokračuje do její druhé části, kde probíhá samotný výpočet bodů. Pro reprezentaci bodu jsem mohl použít třídu *TPoint*, kterou Delphi nabízí. Ta je ovšem nevhodná, protože umožňuje ukládat pouze celočíselné souřadnice bodů. Proto jsem vytvořil vlastní třídu *TBod*, která má atributy *x* a *y* typu *real* (reálná čísla). Vypočítané body se ukládají do dynamického pole, protože dopředu nevíme, kolik segmentů a opakování uživatel zvolí. Toto pole je na začátku prázdné a s každým novým bodem se jeho délka o jedna zvětšuje.

Ještě před začátkem výpočtu je třeba určit počáteční bod křivky, protože výpočet nového bodu probíhá na základě bodu předchozího. Na jeho souřadnicích příliš nezáleží, protože před vykreslením je spirolaterála ještě posouvána a zvětšována nebo zmenšována tak, aby se vešla na vykreslovací plochu. Proto je počáteční bod nastaven jednoduše na bod $[0,0]$.

Na začátku výpočtu je potřeba změnit zadaný úhel, protože neodpovídá úhlu, který mezi sebou svírají jednotlivé segmenty. Nový segment se totiž vypočítává na základě koncového bodu segmentu předchozího (černý na obrázku 3.3). Pokud by se úhel nezměnil, nový segment (šedivý) by se byl od původního otočen o úhel *A*. To je ale špatně, protože je potřeba, aby úhel *A* segmenty svíraly, tak jako tomu je mezi červeným a černým segmentem na obrázku. Proto je potřeba určit nový úhel *B*, který se spočítá jako $180 - A$.



Obrázek 3.3: Výpočet úhlu mezi segmenty

Nyní už je možné spočítat souřadnice nového bodu. Nejprve je potřeba určit, zda se bude uplatňovat opačný krok a na základě toho buď přičíst, nebo odečíst úhel otočení. Pak je nutné převést úhel otočení na radiány, protože pro výpočet bodu jsou používány funkce *sin()* a *cos()*, které by s úhlem ve stupních dávaly chybné výsledky. Souřadnice nového bodu jsou určeny tak, že se přičte segment příslušné délky otočený o patřičný úhel k poslednímu vypočítanému bodu. Nakonec je bod uložen na konec pole. Na začátku cyklu je ještě uveden příkaz *Application.ProcessMessages*, který zajistí, že aplikace během náročného výpočtu bodu nepřestane reagovat na uživatelské akce, jako je například stisk tlačítka Přerušit, které zajistí přerušení výpočtu bodů. Následuje ukázka kódu, který počítá body a ukládá je do pole.

```
SetLength(body, 1);
body[Length(body)-1] := TBod.Create;
body[Length(body)-1].X := 0;
body[Length(body)-1].Y := 0;
uhelOtoc := 180 - uhelOtoc;
for i := 1 to opakovani do
  for j := 1 to segmenty do
    begin
      Application.ProcessMessages;
      if prerusit then break;
      provedenKrokZpet := false;
      for k:=0 to length(krokyZpet)-1 do
        if (j = krokyZpet[k]) then
          begin
            uhelDeg := uhelDeg - uhelOtoc;
            provedenKrokZpet := true;
            break;
          end;
      if (not provedenKrokZpet) then uhelDeg := uhelDeg + uhelOtoc;
      if uhelDeg > 360 then uhelDeg := uhelDeg - 360;
      if uhelDeg < -360 then uhelDeg := uhelDeg + 360;
      uhel := (PI/180) * uhelDeg;
      x := body[length(body)-1].X + j*(cos(uhel));
      y := body[length(body)-1].Y + j*(sin(uhel));
      SetLength(body, Length(body)+1);
      body[Length(body)-1] := TBod.Create;
      body[Length(body)-1].X := x;
      body[Length(body)-1].Y := y;
    end;
end;
```

3.3.2 Vykreslení spirolaterály

Nyní jsou spočítané body spirolaterály a zbývá ji jen vykreslit. Vykreslení všech stylů čáry probíhá podobně, přesto jsou mezi jednotlivými druhy jisté rozdíly. Proto zde uvedu stručný popis vykreslení pro každý styl čáry. Před samotným vykreslením je nejprve potřeba spočítat posun spirolaterály a koeficient zvětšení tak, aby byla vykreslená křivka ve středu kreslicí

plochy. Koeficient je počítán pro křivku tvořenou úsečkami, přičemž jsou ponechány určité okraje, aby nebyla vykreslená přes celou kreslicí plochu a také aby se do okna vešly křivky tvořené například kruhovými oblouky. Ty se tam sice při určitých konfiguracích křivky nevejdou, ale to se dá jednoduše napravit pomocí nastavení vhodného přiblížení. Dále je třeba vybarvit pozadí kreslicí plochy. K tomu je použita funkce *FloodFill()*, kterou nabízí komponenta *Image*. Poslední věcí, kterou je třeba udělat, je nastavit šířku a barvu pera, podle zadaných parametrů.

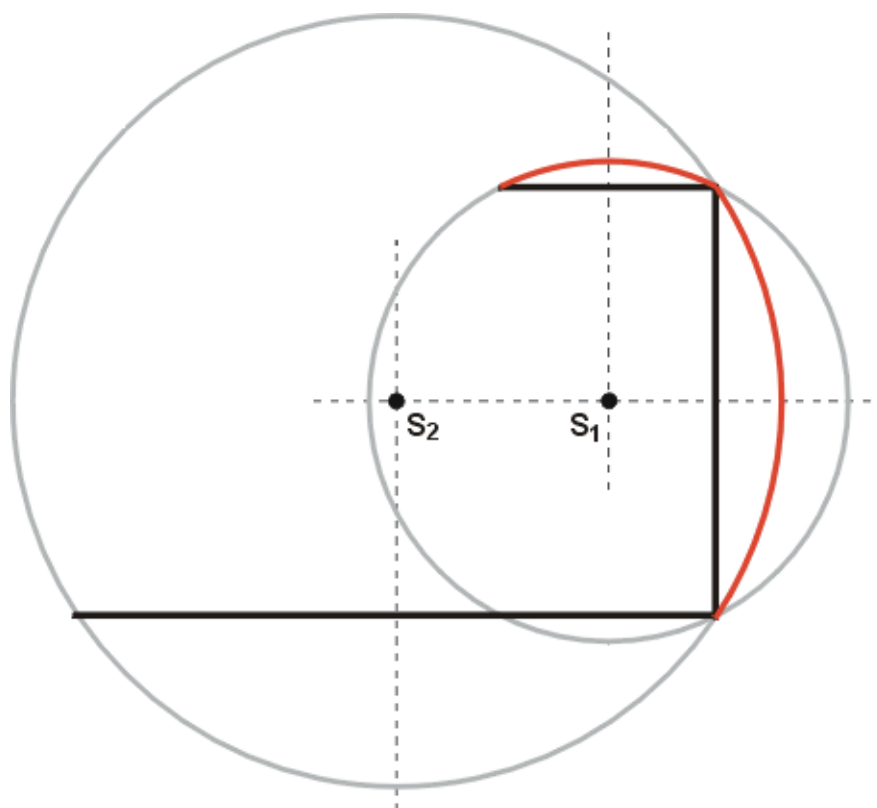
Pro vykreslení spirolaterály pomocí úseček je nejprve nutné nastavit pozici pera na počáteční bod křivky. O vykreslení se postará funkce *LineTo()*, kterou nabízí komponenta *Image*. Tato funkce vykreslí přímku z aktuální pozice pera do bodu, který je jí předán jako parametr. Proto stačí tuto funkci umístit do *for* cyklu a jako parametry jí předávat body křivky.

O vykreslení křivky pomocí bodů se stará funkce *Ellipse()*, která je opět dostupná v komponentě *Image*. Ta má čtyři parametry, souřadnice *x* a *y* horního levého a pravého dolního rohu obdélníku, který je elipse opsán. Opět tedy stačí umístit tuto funkci do *for* cyklu a za souřadnice horního levého rohu postupně dosazovat body spirolaterály, za souřadnice pravého dolního rohu tyto body zvětšené o dva.

Další možností, jak křivku vykreslit jsou bézierovy křivky. Pro ty je v Delphi připravena funkce *PolyBezier()*, které se jako jediný parametr předává pole řídících bodů, pro bézierovy křivky. Tyto body musí být typu *TPoint*, proto je potřeba vytvořit nové pole tohoto typu a body spirolaterály do něj pomocí posunu a koeficientu zvětšení převést. Navíc se křivka vykreslí pouze pokud je zadán přesný počet bodů, které bézierovy křivky určují. Proto je potřeba převést jen určitý počet bodů.

Vykreslování kruhových oblouků už je poněkud náročnější. K jejich vykreslování je použita funkce *Arc()*, která má osm parametrů. První čtyři určují obdélník (v tomto případě čtverec), který ohraničuje elipsu, ze které se bude oblouk vykreslovat. Další dva parametry určují bod od kterého se oblouk začít vykreslovat. Poslední dva parametry určují bod, ve kterém má vykreslování oblouku končit. Za počáteční a koncový bod lze jednoduše dosadit koncové body úsečky, nad kterou má být oblouk vykreslen. Pro výpočet krajních bodů ohraničujícího čtverce je nejprve potřeba spočítat střed kružnice a její poloměr. Střed kružnice je určen středem kružnice opsané trojúhelníku, který je tvořen třemi po sobě jdoucími body spirolaterály, přičemž kruhový oblouk je vykreslen vždy jen nad prvními dvěma. Střed je volen takto proto, aby na sebe kruhové oblouky co nejvíce navazovaly. Střed kružnice opsané trojúhelníku leží na průsečíku os jeho stran, proto je potřeba jej vypočítat. Obě osy jsou vyjádřeny parametrickou rovnicí přímky a jejich průsečík se vypočítá vyřešením soustavy dvou rovnic o dvou neznámých. Pro určení ohraničujícího čtverce je také potřeba znát poloměr kružnice. Ten je možné snadno spočítat Pythagorovou větou, protože známe obě kratší ramena trojúhelníku. Nyní už stačí dopočítat rohové body čtverce a předat funkci *Arc()* všechny parametry potřebné k vykreslení oblouku.

Dalšími třemi možnostmi, kterými je křivku možné ozvláštnit, jsou vykreslování pomocí vnějších nebo vnitřních obloučků, případně pomocí vlnovky, která je tvořena střídáním vnějšího a vnitřního obloučku. K vykreslení je použita znovu funkce *Arc()*, proto je potřeba spočítat poloměr a střed kružnice, ze které se bude oblouček vykreslovat, a z těchto hodnot

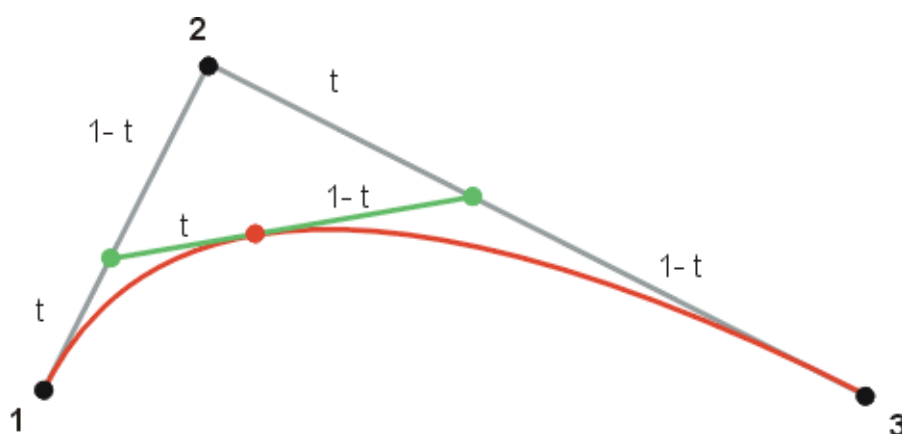


Obrázek 3.4: Určení středů kruhových oblouků

následně určit souřadnice horního levého a pravého dolního rohu čtverce, který kružnici ohraničuje. Pro všechny tři způsoby je výpočet kružnic stejný, proto je vykreslení realizováno pomocí jedné procedury, ve které se na konci rozhodne pouze o tom, která část kružnice se vykreslí. Vykreslování obloučku funguje tak, že nad segmentem délky jedna se vykreslí jeden oblouček, nad segmentem délky dva se vykreslí dva obloučky a tak dále. Proto je určení poloměru kružnice triviální, pomocí Pythagorovy věty je možné jej spočítat z délky prvního segmentu. Střed prvního oblouku který se nad segmentem vykreslí je určen tak, že se segment rozdělí na příslušný počet dílů. Protože se jedná o poloměr, je délka dílu ještě vydělena dvěma, a přičtena k výchozímu bodu segmentu. Tím vznikne první střed. Další středy už jsou odvozeny přičtením délky dílu, který vznikl při výpočtu prvního středu, k předcházejícímu středu. Z těchto vypočítaných hodnot se určí ohraničující čtverec, který se dosadí jako parametr funkce `Arc()`. V závislosti na typu čáry je třeba rozhodnout, zda za počáteční oblouku dosadit počáteční bod segmentu a za koncový bod oblouku koncový bod křivky, nebo naopak.

Program také nabízí vykreslovat spirolaterálu pomocí kvadratické bézierova křivky. Ta je určena třemi řídicími body. První a třetí bod jsou počáteční a koncový bod křivky druhý bod je bod řídicí a určuje tvar křivky. Křivka se vytváří tak, že se postupně od 0 do 1 mění

parametr t , který určuje pozici bodů na úsečce mezi body 1 a 2, a na úsečce mezi body 2 a 3. Tyto nově vzniklé body určují úsečku (na obrázku 3.5 zeleně), na které se opět pomocí parametru t určí jeden bod. Tento bod už je bodem křivky. Konstrukce kvadratické křivky je zobrazena na obrázku 3.5. V aplikaci je počáteční bod křivky volen jako střed jednoho segmentu, řídicí bod je volen jako koncový bod tohoto segmentu a koncový bod křivky volen jako střed segmentu následujícího. To proto, aby byla zaručena C^1 spojitost křivek. Ta je definována tak, že tečný vektor v koncovém bodě první křivky má stejný směr a je stejně velký, jako tečný vektor v počátečním bodě křivky navazující. Výběr bodů a konstrukce spirolaterály pomocí kvadratické křivky je vidět na obrázku 3.6. Body $S_1 - S_4$ určují počáteční a koncové body křivky, body 2 – 4 jsou řídicí body křivky.

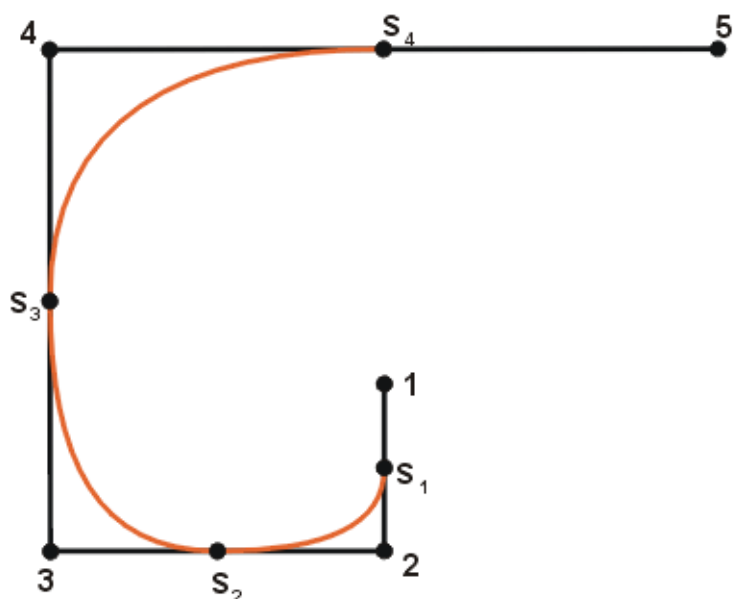


Obrázek 3.5: Konstrukce kvadratické křivky

V aplikaci je vykreslování křivky realizováno přesně podle tohoto algoritmu. Jedinou zajímavou věcí, která stojí za upřesnění je volba parametru t . Ten je zvolen tak, že se při jeho změně bod na úsečce posune vždy o jeden pixel. Toho je dosaženo tak, že se nejprve zjistí, zda je větší rozdíl mezi souřadnicemi x , nebo souřadnicemi y počátečního a koncového bodu křivky. Pro posun se vybere ta souřadnice, kde je větší rozdíl. To zaručuje větší přesnost při vykreslování. Následně se spočítá, kolik pixelů tento rozdíl tvoří a obrácením této hodnoty se určí velikost parametru pro posun o jeden pixel. Taková volba parametru je výhodná zejména proto, že nedochází ke „zbytečnému“ počítání, ke kterému by docházelo, kdyby byl parametr zvolen příliš malý, ani k nepřesnému počítání, ke kterému by docházelo, kdyby byl naopak zvolen příliš velký.

Poslední možností, pro vykreslení spirolaterály je kubická křivka. Ta je určena počátečním a koncovým bodem a dvěma body řídicími. Při určování jejího průběhu se postupuje podobně jako u kubické křivky, jen je nejprve potřeba pomocí parametru t určit dvě pomocné úsečky (na obrázku 3.7 modře) a z nich pomocí algoritmu popsaného u kvadratické křivky určit nový bod.

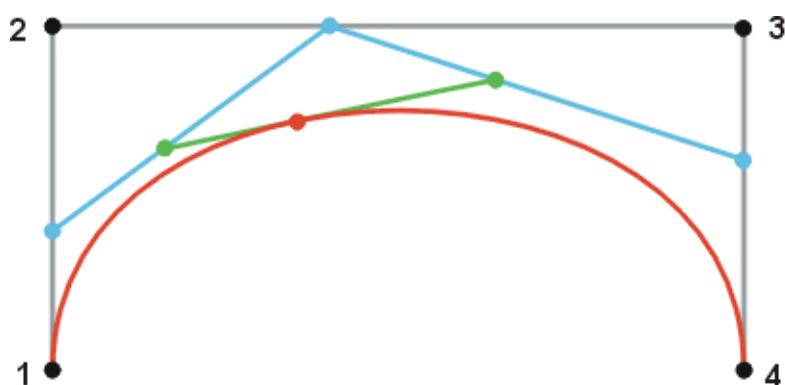
Jako výchozí bod křivky je zvolen vždy střed segmentu. Koncové body následujících segmentů jsou pak řídicími body křivky a jako koncový bod křivky je zvolen střed dalšího



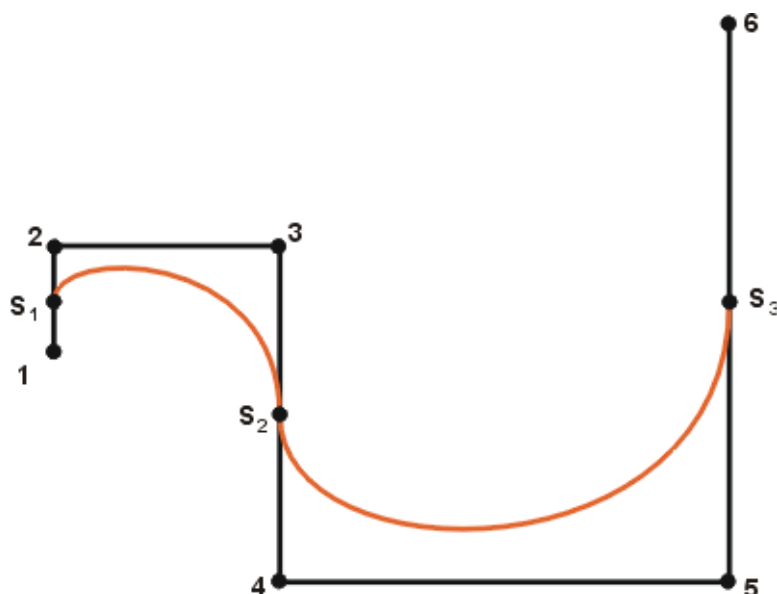
Obrázek 3.6: Určení počátečních, řídicích a koncových bodů a konstrukce křivky

segmentu. Takto zvolené body opět zaručují C^1 spojitost navazujících křivek. Tvorba spiro-laterály pomocí kubické křivky je znázorněna na obrázku 3.8. Body $S_1 - S_3$ určují počáteční a koncové body křivky, body 2 – 5 jsou řídicí body křivky. Implementace vykreslování kubické křivky je téměř stejná jako vykreslování kvadratické křivky, jen je třeba navíc spočítat pomocné body. I parametr t je volen stejným způsobem jako při vykreslování kvadratické křivky.

Pokud uživatel zvolil možnost vnitřní čáry, nastaví se šířka a barva pera na zvolené hodnoty a znovu se zopakuje celý postup vykreslení příslušného typu čáry. Tím je již vykreslená spirolaterála částečně překreslena novou a vznikne efekt vnitřní čáry.



Obrázek 3.7: Konstrukce kubické křivky



Obrázek 3.8: Určení počátečních, koncových a řídicích bodů kubické křivky

3.3.3 Ukládání spirolaterály

Další možnost, kterou program nabízí je ukládání. To je možné do pěti různých formátů. K dispozici jsou dva formáty rastrové grafiky (BMP a JPG), které nejsou už z principu rastrové grafiky příliš vhodné pro nějaké další zpracování, ale spíš pro zobrazení na monitoru. Dále je možné ukládat do dvou formátů vektorové grafiky (HPGL/PLT a SVG). HPGL/PLT soubory neumí zobrazit většina běžných prohlížečů, je možné je otevřít a editovat jen ve specializovaných grafických programech. V tomto formátu jsou uloženy pouze body spirolaterály, neukládá se ani informace o barvě, síle nebo typu čáry. Z nabízených formátů je tak nejvhodnější na následnou úpravu, protože nesvazuje grafickou fantazii tím, co všechno už je dáno. Formát SVG je také vektorový, ale spíš než na následnou úpravu se používá pro zobrazení vektorové grafiky na internetu. V aplikaci je ukládání do SVG implementováno zejména proto, aby bylo ukázáno, že existuje možnost do SVG ukládat, a to jak z důvodu, že SVG umožňuje snadno publikovat vektorovou grafiku na internetu, tak z důvodu, že je možné SVG načíst do grafických programů a dále je upravovat. A protože se jedná spíše o ukázkou této možnosti, je implementováno jen ukládání spirolaterál tvořených úsečkami nebo jednotlivými body. Další druhy jsem z důvodů časové náročnosti neimplementoval a jsou ponechány jako podněty k dalšímu vylepšení programu. Poslední formát, do kterého je možné ukládat je SLC (SpiroLateral Curve). Jedná se o textový soubor, ve kterém jsou na jednotlivých řádcích postupně uloženy všechny parametry, které je možné pro křivku zvolit, ne tedy jen základní parametry jako počet segmentů, kroků a úhel otočení, ale i ostatní volby jako barva, síla a styl čáry. To může být užitečné, pokud uživatel vytvoří křivku, která se mu líbí a chce si všechny volby někde uchovat pro příští použití. Uložení je

možné vyvolat buď vybraním položky *Uložit...* z hlavního menu nebo pomocí běžně užívané klávesové zkratky *Ctrl+S*. Po vybrání možnosti uložit se objeví standardní dialog pro ukládání, ve kterém je na výběr všech pět výše uvedených typů souborů.

Uložení do BMP je velmi jednoduché. Stačí jen zjistit, zda uživatel zadal k názvu souboru příponu *.bmp*. Pokud ne, je třeba ji doplnit a pak už jen stačí zavolat funkci *SaveToFile()*, kterou zpřístupňuje komponenta *Image* a předat jí jako parametr právě jméno souboru, do kterého se má obrázek uložit.

Při uložení do JPG je třeba nejprve vyvolat nové dialogové okno s dotazem na kompresi obrázku. Dále je potřeba do proměnné *j* typu *TJpegImage* přiřadit obrázek, který se má uložit a nastavit obrázku potřebnou kvalitu. Ta má rozsah od nuly do sta. Jelikož se v předcházejícím dialogu zadávala komprese, je třeba ji odečíst od sta. Tím získáme výslednou kvalitu obrazu. Opět je třeba zkontrolovat přítomnost přípony v názvu, případně ji k němu doplnit. Nyní již stačí zavolat funkci *SaveToFile()*, dostupnou z proměnné *j* a předat jí název souboru.

Při ukládání do PLT souboru je opět nejprve potřeba zajistit správné jméno souboru. Jelikož je PLT soubor jen textový soubor, stačí vytvořit proměnnou *f* typu *TextFile* a pomocí funkcí *AssignFile()* a *Rewrite()* jej připravit pro zápis. Poté se pomocí funkce *Writeln()* zapíše inicializační informace, které jsou vždy stejné. Následně se ve for cyklu zapisují jednotlivé body křivky. Na závěr se zapíše ukončovací příkaz a soubor se pomocí funkce *CloseFile()* uzavře.

SVG soubor je opět pouze textový, takže se pro zápis připravuje naprosto stejně jako při ukládání PLT souboru. Nejprve se zapíše hlavička SVG souboru, která se skládá z definice verze XML a definice DTD. Následně se zapíše značka *<svg>*, která má atributy *width* a *height*. Těm je předána šířka a výška vykreslovací plochy. Jelikož se do SVG ukládají i barvy a SVG používá barevné schéma RGB, kdežto v Delphi jsou barvy reprezentovány pomocí schématu BGR, je nejprve nutné barvy převést do modelu RGB. To je vyřešeno tak, že se barva, která je v Delphi reprezentována jako hexadecimální číslo, převede na string a patřičně se změní pořadí znaků v řetězci. Zpětný převod na hexadecimální číslo není potřeba, protože barva se předává ve formě textového řetězce. V SVG se jednotlivé grafické prvky vykreslují v pořadí, v jakém jsou zapsány ve zdrojovém souboru, proto objekty, které jsou popsány jako první budou překryty objekty popsány dále v souboru.

Nejprve je tedy nutné vykreslit pozadí. To se provádí pomocí elementu *<rect>*, kterému jsou jako atributy předány horní levý roh, šířka a výška obdélníku, který se má vykreslit. A jelikož má být obdélník přes celé plátno, za horní levý roh je dosazen bod [0,0] a za šířku a výšku je dosazena šířka a výška kreslicí plochy. Dále se ještě nastaví barva výplně a nulová šířka obrysu. Pokud se má vykreslit křivka pomocí úseček, je použit element *<polyline>*. U něj se nejprve nastaví barva obrysu a šířka čáry, kterou zadal uživatel. Následně se atributu *points* předají jednotlivé body, spirolaterály. Pokud je zvoleno vykreslení vnitřní čáry, znovu se použije element *<polyline>*, jen se mu předá jiná síla a barva čáry. Pokud se vykreslují jednotlivé body, je použit element *<circle>*, kterému se jako atributy předávají střed bodu a jeho poloměr. Střed je určen jako pozice bodu křivky a poloměr je polovina šířky čáry, kterou si zvolil uživatel. Dále je opět potřeba nastavit správnou barvu výplně

a sílu obrysu na nula. A stejně jako tomu bylo u vnitřní čáry u křivky vykreslené pomocí úseček, i u vnitřní čáry kreslené pomocí bodů se znovu použije element `<circle>` se změněnou barvou a poloměrem. SVG soubor se ukončí značkou `</svg>` a nyní je možné uzavřít soubor pomocí funkce `CloseFile()`.

Poslední způsob, jak spirolaterálu ukládat je zapsat ji do SLC souboru. Jedná se opět o čistě textový fomrát, takže se soubor pro zápis připraví pomocí funkcí `AssignFile()` a `Rewrite()`. Pak se na jednotlivé řádky zapíše všechny parametry křivky, které má uživatel možnost nastavit a soubor se uzavře.

3.3.4 Načítání spirolaterály

Poslední důležitou funkcí aplikace je načítání křivky z SLC souboru. Tu je opět možné vyvolat z hlavního menu programu zvolením možnosti *Otevřít...* nebo pomocí klávesové zkratky `Ctrl+O`. Po zvolení této možnosti se otevře dialogové okno pro otevření souboru. Zde je jako filtr pro soubory nastaven pouze typ SLC. To proto, aby se uživateli zabránilo načítat jakýkoliv soubor. Po vybrání souboru je nejprve soubor pomocí funkce `AssignFile()` a `Reset()` připraven ke čtení. To se provádí pomocí funkce `Readln()`, která vždy načte celý řádek do textového řetězce a přesune se na další řádek. Hodnota načtená do textového řetězce je následně předána do příslušného políčka pro zadávání parametrů. Po načtení všech hodnot je ještě křivka vykreslena.

Kapitola 4

Návrhy k dalšímu vylepšení aplikace

I když aplikace umí poměrně dost způsobů, jak křivku vykreslit (kruhové oblouky, vlnovka, bézierovy křivky...), stále zůstává dost možností, které by bylo možné ještě implementovat. Mezi nejjednodušší úpravy by patřilo přidání dalších způsobů vykreslování, které by nahrazovaly úsečku nějakým jiným typem čáry, například lomenou čarou, úsečkou tvořenou pomocí geometrických útvarů jako kružnice, čtverce a tak podobně. Nahrazení úsečky jiným typem čáry ovšem není tak graficky zajímavé jako další způsoby, které využívají řídicí body křivky k vykreslení výtvarných objektů. Jeden z nich popsali Robert Dixon[2] a Dennis Lawrance[7]. Jedná se o inverzi. Ta se provádí pomocí transformací, projekcí a mapování lineárních útvarů na křivkové. Inverze dle definice rekonstruuje základní tvar spirolaterál do křivek. Bylo zjištěno, že pomocí těchto transformací vznikají vizuálně nejzajímavější útvary při použití spirolaterál, které jsou uzavřené okolo středu, to znamená, že úsečky neprocházejí středem.

S dalším způsobem, jak spirolaterály ozvláštnit, přišel opět Dixon a tuto metodu nazval antiMercator. Při této transformaci jsou jedním způsobem zpracovány vodorovné úsečky, jiným způsobem svislé úsečky a dalším způsobem úhlopříčné úsečky.

Spirolaterály jsou zatím pouze rovinné útvary a jistě by bylo zajímavé pokusit se je zobrazit v prostoru. Toto byla by další z možností, jak program výrazně obohatit.

Dalším způsobem kterým by bylo možné aplikaci vylepšit by bylo dokončení ukládání do SVG. Nyní je možné ukládat pouze spirolaterály tvořené úsečkami nebo jednotlivými body. SVG ale nabízí možnosti, kterými je možné popisovat i oblouky a křivky, proto by bylo možné realizovat i ukládání křivky s ostatními druhy čar.

Jako poslední způsob, jak aplikaci vylepšit by bylo možné umožnit nastavit velikost vykresleného obrázku před uložením spirolaterály do některého z rastrových formátů, případně přidat ukládání do dalších formátů.

Kapitola 5

Závěr

V úvodní části této práce je popsáno, co to spirolaterální křivka je, z čeho se vyvinula a jaké má základní parametry. Těmi jsou počet segmentů, počet opakování a úhel otočení. Dále je popisován problém uzavření spirolaterálních křivek, který doposud není spolehlivě vyřešen. Jsou také popsány možnosti, jak křivky ozvláštnit. Mezi ně patří použití opačných kroků nebo použití obloučků, vlnovek, lomených čar, kruhových oblouků nebo křivek místo běžných úseček.

V další části práce jsou popsány použité programové prostředky. Čtenář je stručně seznámen s jazykem ObjectPascal a s vývojovým prostředím Borland Delphi 7 a jeho výhodami i nevýhodami. Dále je popsán jazyk HPGL/PLT do kterého je možné spirolaterálu uložit a následně ji načíst v některém grafickém editoru a použít pro další zpracování. Také je představen a popsán jazyk SVG, který se používá především pro zobrazení vektorové grafiky na internetu.

Dále je popsáno uživatelské rozhraní aplikace, které je navrženo s ohledem na přehlednost a přístupnost. Také jsou popsány jednotlivé komponenty, které jsou v okně programu použity.

Další část práce se věnuje popisu implementace jednotlivých funkcí programu. Jedná se hlavně o načítání parametrů křivky a následný výpočet jejích bodů. Následně je popsáno vykreslování křivky se všemi druhy čáry. Podle potřeby jsou zde uváděny ukázky zdrojového kódu nebo obrázky, které názorně ukazují danou situaci.

Literatura

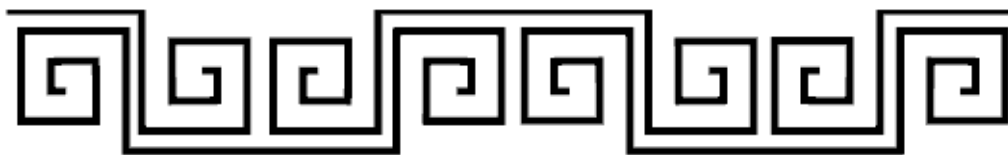
- [1] Wikipedia contributors: *Borland Delphi*, Wikipedia, The Free Encyclopedia, 18. 5. 2008, <http://en.wikipedia.org/wiki/Borland_Delphi>, citováno: 20. 5. 2008 . 3.1.1
- [2] Dixon, R.: *Mathographics*, Dover Publications, Inc., 1987. 4
- [3] Tišnovský, P.: *Vektorový grafický formát HPGL*, Root.cz, 8. 3. 2007, <<http://www.root.cz/clanky/vektorovy-graficky-format-hppl/>>, citováno: 20. 5. 2008 . 3.1.2
- [4] Krawczyk, R.: *Spirolaterals, Complexity from Simplicity*, International Society of Arts, Mathematics and Architecture 99, 1999. 2
- [5] Krawczyk, R.: , <<http://www.iit.edu/~krawczyk/>>, citováno: 20. 5. 2008 . 2
- [6] Krawczyk, R.: *The art of spirolaterals*, The Millenial Open Symposium on the Arts and Interdisciplinary Computing 99, 1999. 2
- [7] Lawrence, D.: *A Catalog of Special Curves*, Dover Publications, Inc., 1972. 4
- [8] Odds, F.: *Mathematics Teacher*, 1973. 2
- [9] Hejral, M.: *Průvodce SVG – Grafika – Webdesign – Interval.cz*, Interval.cz, 15. 4. 2003, <<http://interval.cz/clanky/pruvodce-svg/>>, citováno: 20. 5. 2008 . 3.1.3

Příloha A

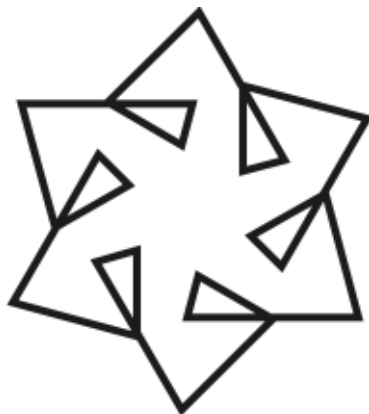
Ukázky upravených spirolaterál



Obrázek A.1: Původní spirolaterála 9_{90}^9



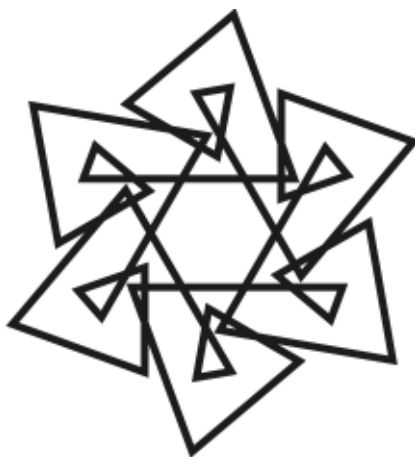
Obrázek A.2: Ukázka neuzavřené spirolaterály po zpracování v programu CorelDraw



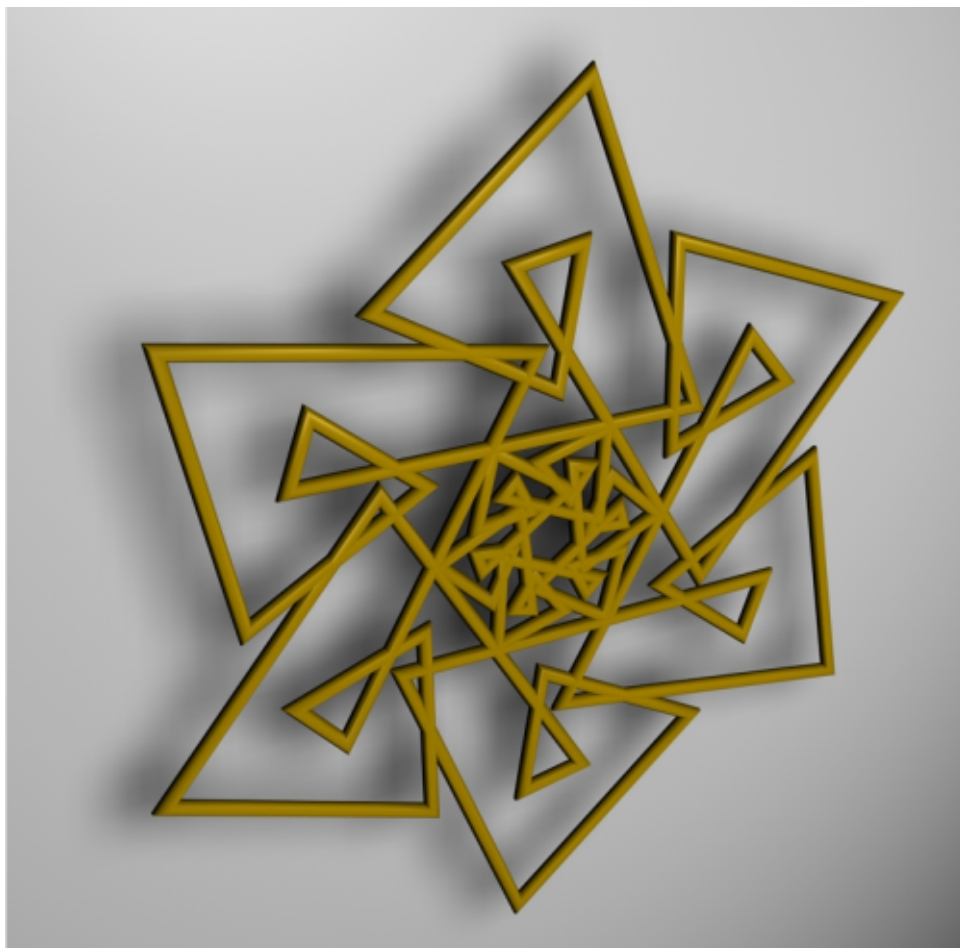
Obrázek A.3: Původní spirolaterála 4_{75} s šesti opakováními



Obrázek A.4: Ukázka uzavřené spirolaterály po zpracování v programu CorelDraw



Obrázek A.5: Původní spirolaterála 6_{70} s šesti opakováními



Obrázek A.6: Ukázka zpracování spirolaterály ve 3D pomocí programu Cinema4D

Příloha B

Obsah přiloženého CD

Součástí této práce je také přiložené CD, obsahující

- zdrojový dokument této práce ve formátu XML (podle DTD DocBook),
- tuto práci ve formátu PDF,
- zdrojové kódy aplikace Generátor spirolaterál,
- přeloženou verzi aplikace Generátor spirolaterál.