Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Grafický nástroj pro podporu sazby jednoduchých obrázků

2020

Karolína Zemenová

Vedoucí práce: doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Karolína Zemenová
Název práce:	Grafický nástroj pro podporu sazby jednoduchých obrázků
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	30
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Karolína Zemenová
Title:	Graphical tool supporting typesetting of simple pictures
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, full-time form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	30
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Práce popisuje grafický editor pro tvorbu obrázků, skládajících se z jednoduchých geometrických tvarů. Je vhodný pro kreslení svazů a konečných automatů. Výsledné obrázky lze exportovat do formátu png, svg, pdf a také do jazyka TikZ

Synopsis

Thesis describes graphic editor for typesetting of pictures composed from simple geometric shapes. It is useful for drawing lattices and finite automata. Created pictures can be exported to png format, svg, pdf and TikZ language

Klíčová slova: grafický editor; Java; JavaFX; TikZ; LaTeX

Keywords: graphic editor; Java; JavaFX; TikZ; LaTeX

Děkuji vedoucímu doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za jeho vedení a rady, které mi pomohly při vývoji aplikace.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracovala samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Vytvoření jednoduchého obrázku pouze s pomocí textových příkazů může být zbytečně náročné. Cílem grafického editoru, který je výsledkem této práce, je uživateli usnadnit vytváření obrázků. Uživatel má možnost kreslit základní geometrické tvary přímo na kreslící plátno. Nakreslené objekty je možné dále upravovat, přemisťovat a rotovat. Rozpracovaný obrázek je možné uložit a poté znovu načíst. Výsledek může být exportován do několika grafických formátů. Při vývoji jsem si kladla za cíl, aby aplikace byla co nejjednodušší na použití.

V této práci se nejprve podíváme na použité technologie, dále se budeme zabývat nejdůležitějšími částmi aplikace z pohledu programátora, na strukturu aplikace a nejdůležitější třídy a rozhraní a na problémy, na které jsem při vývoji aplikace narazila. Poté se podíváme na jednoduchou uživatelskou dokumentaci, kde popíšeme uživatelské rozhraní a vysvětlíme jakým způsobem se aplikace ovládá.

1.1 Požadavky

Požadováno bylo, aby editor umožňoval:

- Kreslení geometrických tvarů (elipsa, obdélník, trojúhelník, úsečka, šipka, bézierova křivka) a textových popisků;
- Export do bitmapy, pdf a do prostředí picture.

Původní zadání vyžadovalo export do prostředí picture, ale vzhledem k jeho zastaralosti, jsme se s vedoucím rozhodli raději použít prostředí TikZ. Možnost exportu do prostředí TikZ je také vhodná k další úpravě obrázku s využitím pokročilých možností TikZu.

1.2 Použité technologie

1.2.1 Java

Java je objektově orientovaný programovací jazyk vytvořený společností Sun Microsystems v roce 1995. Jeho cílem je umožnit vývojářům aplikací program napsat jednou a spustit kdekoliv (write once, run anywhere), což znamená, že zkompilovaný Java kód může běžet na všech platformách podporujících Javu, bez nutnosti rekompilace. Java aplikace se typicky překládají do bytekódu, který běží na Java virtual machine (JVM), Java tak může bežet na všech zařízeních, kde je JVM k dispozici. [1]

1.2.2 JavaFX

JavaFX je softwarová technologie, která v kombinaci s Javou, umožňuje vytváření moderně vypadajících aplikací s bohatým obsahem, audiem a videem. [2]

1.2.3 TikZ

PGF/TIKZ je dvojice jazyků pro vytváření vektorové grafiky. Umožňují kreslení základních tvarů jako jsou: body, přímky, cesty, elipsy apod. PGF (Portable Graphics Format) je nízkoúrovňový jazyk, zatímco TikZ (TikZ ist kein Zeichenprogramm – německy TikZ není kreslící program) je množina víceúrovňových maker, která používají PGF.

TikZ obsahuje knihovny pro kreslení různých druhů diagramů. Příkladem mohou být konečné automaty, Turingovy stroje, myšlenkové mapy (mind maps), matice a mnoho dalších. [3]

1.2.4 LaTeX

LaTeX je systém na přípravu dokumentů. Je nejčastěji používán pro vědecké dokumenty, ale může být použit pro téměř jakoukoliv formu publikace. LaTeX umožňuje sazbu článků, technických zpráv, knih a prezentací. Poskytuje kontrolu nad velkými dokumenty, referencemi a tabulkami. Také se často používá pro sazbu matematických vzorců. [4]

1.2.5 XML

XML (eXtensible Markup Language) je značkovací jazyk, který je dobře čitelný, jak pro člověka, tak i pro stroj. Byl vytvořen k ukládání a přenášení dat.[5]

1.2.6 SVG

SVG (Scalable Vector Graphics) je formát založený na XML. Slouží k definování vektorové grafiky pro web, je tedy možné ho neomezeně škálovat, přibližovat a neztrácí na kvalitě. [6]

1.2.7 Apache Batik

Knihovna, která slouží manipulaci, generaci a zobrazování obrázků ve formátu svg. Umožnuje konvertování svg do různých formátů, jak rastrových, tak vektorových. V práci jsem použila právě převod z svg do pdf formátu. [7]

2 Programátorská dokumentace

V této části bude popsána celková struktura programu a jeho nejzásadnější části. Popíšeme práci s JavaFX, základ uživatelského rozhraní, tvary a vybrané třídy a rozhraní. Dále se podíváme na to, jakým způsobem se provádí výstupy a na jaké problémy jsem při tvorbě aplikace narazila.
Aplikace je rozdělena do sedmi balíčků, nyní krátce popíšeme jejich obsah a úlohu.

- `core` – obsahuje většinu tříd řídících hlavní logiku aplikace. V tomto balíčku se nacházejí třídy `DrawingPaneController`, `ShapeEditor`, `TextNodeEditor`, `ShapeCopyGenerator` a rozhraní `Editable`, které jsou podrobněji popsány v sekci 2.4. Dále obsahuje rozhraní `Container`, které implementují všechny uzly (nodes), které jsou podrobněji popsány sekci 2.3. A také rozhraní `EditableText`, které implementují všechny objekty obsahující text. Společně tyto třídy a rozhraní tvoří základ aplikace. Poskytují hlavní funkcionalitu pro vytvoření objektů a jejich úpravu.

- `shapes` – obsahuje třídy reprezentující všechny tvary, které je možno v editoru kreslit. A také obsahuje abstraktní třídy, které využívají šipky a uzly, podrobněji popíšeme v sekci 2.3.

- `actions` – obsahuje třídy reprezentující uživatelské akce a třídu History, která poskytuje možnost vykonat operace zpět a znovu, podrobnější popis fungování najdeme v sekci 2.7.

- `io` – obsahuje třídy, generující výstupy, které jsou podrobněji popsány v sekcích 2.5.1 až 2.5.4. A také třídy zajišťující ukládání a načítaní rozpracovaného obrázku.

- `ui` – obsahuje třídy, které generují uživatelské rozhraní a reagují na uživatelské vstupy, soubory typu fxml a css, které využívají, jsou ve složce resources.

- `utils` – obsahuje pomocné třídy s jednoduchou funkcionalitou.

- `enums` – obsahuje výčtové typy.

2.1 Práce s JavaFX

Na začátek se podívejme na krátky popis toho, jakým způsobem se skládají prvky v JavaFX, aby vytvořili uživatelské rozhraní.

Základem je stromová struktura, každý uzel může mít, buď jednoho, nebo žádného rodiče. Uzel, který rodiče nemá, je kořenem. Další dva typy uzlů jsou větve a listy. První mohou mít další děti, příkladem uveďme `AnchorPane`, obecně třída `Parent` a její podtřídy. Druhé už žádné další potomky mít nemohou, například `Rectangle` nebo `Circle`. Jeden konkrétní uzel se může vyskytovat

jako kořen, nebo potomek pouze jednou a cykly jsou zakázané. Stromy, které jsou součástí scény, jsou vykresleny. Scéna je kontejnerem pro kořenový uzel. [2]

JavaFX poskytuje základní tvary v balíčku javafx.scene.shapes, které v aplikaci používám. Stejně tak obsahuje mnoho možností pro rozvržení obsahu, jako jsou různé panely a boxy. V aplikaci používám `AnchorPane`, který obsahuje potomky, mohou to být například kreslené tvary. Panel může reagovat na události kliknutí myší, stejně tak jeho potomci. V programu každému uživatelem vytvořenému tvaru přidávám reakci na událost kliknutí myší. V tomto případě se nakliknutý tvar označí.

2.2 Uživatelské rozhraní

Základ uživatelského rozhraní je vytvořen pomocí fxml, další proměnlivé části jsou vytvářeny za pomoci třídy `UICreator`, jedná se primárně o grafiku tlačítek. Pro celkové rozložení hlavního okna aplikace byl použit `BorderPane`. O reakci na změny v uživatelském rozhraní aplikace se stará `UIController`. Kontroluje, která tlačítka jsou přístupná, uživatelské vstupy a podobně. Existují tři možnosti velikosti ovládacích prvků (malé, střední, velké). Uživatel si mezi nimi může libovolně přepínat (viz obrázek 2).

Kreslící plátno, které můžeme vidět v aplikaci, se skládá ze tří instancí třídy `AnchorPane`, které jsou umístěny v instanci třídy `StackPane`, která slouží ke skládání svého obsahu přímo nad sebe. Nejspodnější vrstvou je vlastní kreslící plátno, které obsahuje jen nakreslené objekty. Nad ním je editovací panel, který obsahuje prostředky pro editaci tvarů a pomocné objekty, jako editační body a podobně. Nad nimi se nachází panel s mřížkou.

Mřížku jsem oddělila od editačního panelu kvůli možnosti přibližování. Přibližování se aplikuje pouze na kreslící plátno a editační panel, protože mřížka je realizována jako malý obrázek, který se neomezeně opakuje na pozadí vrchního panelu. Jeho zvětšování tedy není žádoucí, protože by se tím zvětšovala i tloušťka čar reprezentujících mřížku. Nakreslená mřížka tedy neslouží pro vypočítání, kam se mají objekty přichytávat, ale jenom jako pomocník pro uživatele a pro výpočet stačí znát jen výšku a šířku jednoho pole mřížky.

2.3 Tvary

V této sekci se podíváme na tvary které editor umí kreslit, a jakým způsobem s nimi pracuji.

Všechny útvary, které mohou být kresleny, implementují rozhraní `Editable`, které je obohacuje o nové funkce, které editoru pomáhají při práci s nimi. Dále tvary rozděluji na dva typy a to jednoduché a složené.

V aplikaci také používám obalující třídu `ShapeRepresentation`, která obsahuje vždy jeden tvar.

Obecně má každý tvar nějaké specifické funkce pro nastavení vlastností jako je u kruhu například poloměr. Potom existují vlastnosti společné různým tvarům,

jako je například posunutí, rotace, barva čáry, tloušťka čáry, typ čáry atd.

Vytváření tvarů řeším u většiny z nich tak, že uživatel tažením myši vytvoří obdélník, do něhož bude tvar vepsán. U polygonů to tak nejde, v tomto případě je třeba, aby uživatel kliknutím vybral, kde se budou nacházet body polygonu.

2.3.1 Jednoduché tvary

Jednoduché tvary dědí odpovídající tvary z balíčku `javafx.scene.shape` a jsou doplněny o dodatečné funkce například `copy()` nebo `getLeft()` . Těmito tvary jsou: úsečka, obdélník, elipsa, trojúhelník, polygon, křivka, text.

2.3.2 Složené tvary

Složené tvary mají jako rodiče třídu `Group`, jedná se tedy o skupinu tvarů. Ve třídě `Group` mohou být další složené tvary. Mezi ně patří šipky, uzly, ale také kruh, protože kvůli požadavku, aby se v editoru dobře kreslily konečné automaty mi přišlo vhodné, pro reprezentaci koncového stavu použít kolečko s dvojitým ohraničením a jelikož jsem nenašla nějakou možnost úpravy obyčejného kolečka, vyřešila jsem tento požadavek složeným tvarem. Takový kruh se skládá ze dvou kruhů, kde vnitřní kruh má menší poloměr než vnější. V okamžiku, kdy uživatel vypne možnost dvojitého ohraničení, vnitřní kruh nebude viditelný.

Dále sem patří uzly (nodes), útvary obsahující kontejner a text. I tyto objekty jsem se rozhodla použít právě kvůli zjednodušenému kreslení konečných automatů, či svazů. Ale nezůstala jsem pouze u kruhů, díky abstraktní třídě `ContainedText`, která je rodičem všech uzlů, by bylo jednoduché implementovat tuto funkcionalitu i pro ostatní tvary, nicméně mi přišlo vhodné použít právě kruh, obdélník a elipsu, které jsou vhodné i pro jiné účely.

Dále má uživatel možnost nastavení pozice textového popisku do devíti různých pozic. Problémem třídy `Text` je, že bod určující jeho pozici je levý dolní roh textu, je tedy nutné pozici textu při jakékoliv jeho změně přepočítávat.

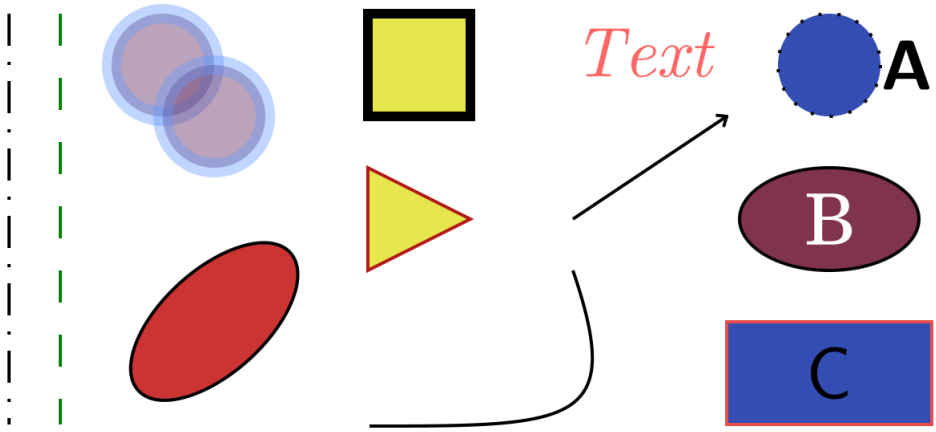
Šipky se od ostatních objektů liší hlavně tím, že mají definovaný kontrolní bod, který určuje směr konce šipky. Jinak se jedná pouze o úsečku, či křivku doplněnou o dvě úsečky.

Na obrázku 1 můžeme vidět příklady většiny tvarů, které je možné kreslit.

2.4 Třídy a rozhraní

Vybrala jsem k popisu nejdůležitější třídy v aplikaci.

Třída `DrawingPaneController` – spravuje kreslící plátno a obsahuje seznam nakreslených objektů. Pamatuje si, které objekty jsou vybrány a má též na starosti přemisťování objektů. V případě potřeby používá třídy `ShapeCreator`, `ShapeEditor` a `ShapeRotator` pro vytvoření nových tvarů, jejich úpravu a rotaci. Dále také nastavuje vzhled vybraným objektům. Primárně reaguje na pod-



Obrázek 1: Ukázka možností kreslení

něty od třídy `UIController`, nebo na události myši nad kreslícím plátnem.

Třída `ShapeEditor` – způsobí vytvoření editovacích bodů, kterými je možné posunovat a tím tvar upravovat. V některých případech jsou přímo navázané na pozice bodů definujících daný tvar. Příkladem může být úsečka, kde upravujeme přímo její počáteční nebo koncový bod, ale někdy je potřeba celkově přepočítat celý tvar, například obdélník, který se v JavaFX definuje pomocí levého horního rohu a šířky a výšky.

Třída `TextNodeEditor` – speciální třída pouze pro editování textových popisků. Upravuje objekty implementující rozhraní `EditableText`. Proces úpravy textů se výrazně liší od úpravy ostatních objektů. Tato třída může upravovat i složené objekty, které obsahují text. Nejprve dojde k přepočtu, na odpovídajícím místě se vytvoří instance třídy `TextField` se stejným textovým obsahem, jako měl původní text. Původní text se nastaví na neviditelný. Poté může uživatel zapisovat a upravovat, nemůže ovšem text posouvat ani rotovat. Původně to bylo možné, ale nastával problém, co udělat, když uživatel použije například zkratku ctrl+C. Chce okopírovat textový obsah, nebo celý objekt? Právě tyto problémy mě vedly k oddělení těchto dvou typů úprav. Celý objekt bude zkopírován v obvyklém módu úprav, tedy když je tvar označen a text v textovém módu.

Třída `UIController` – Spravuje uživatelské rozhraní, kromě kreslícího plátna a klávesových zkratek. Získané informace posílá ostatním třídám, popř. se jí třídy na ně mohou dotazovat.

Rozhraní `Editable` – každý objekt, který má být v editoru vykreslen, musí implementovat toto rozhraní. Rozhraní obsahuje mnoho metod, které jsou nezbytné pro fungovaní aplikace. Uvedeme zde příklad některých z nich:

- `getEditPoints()` – vrátí body potřebné pro úpravu tvaru.
- `toTikz()` – vrátí textový zápis útvaru v prostředí TikZ.
- `copy()` - vrátí novou instanci třídy, která má stejné vlastnosti, kromě některých vzhledových vlastností, které jsou společné více objektům.
- `getTop()` – vrátí nejmenší y souřadnici, kterou tvar na kreslícím panelu zabírá bez započtení hodnoty posunutí nebo současné hodnoty rotace. Podobně pro ostatní směry.
- `getGridAnchorX()` – vrátí x souřadnici, kterou se má tvar zachytávat k mřížce, obdobně pro y souřadnici.

Třída `ShapeCopyGenerator` – má na starosti kopírování. Ukládá si seznam objektů, které byly uživatelem zkopírovány a když uživatel vykoná akci vložení, vytvoří nové kopie uložených objektů, které budou vloženy na kreslící plátno. Objekty mají pro tento účel metodu `copy()`.

Třídy `ProjectLoader a Project Saver` – mají za úkol ukládání a načítání projektu ve formátu xml. Výsledný xml soubor není nijak ošetřen nebo šifrován. Uživatel může tento soubor přepisovat, ale nedoporučuje se to. Proto je možné, že při načítání se mohou do aplikace dostat nesmyslná data. Některé ze závažných problémů jsou při načítání ošetřeny, ne však všechny, které mohou nastat.

2.5 Exporty

V této kapitole popíšeme jakým způsobem editor přistupuje k exportu nakresleného obrázku do formátu png, svg, pdf a do prostředí TikZ.

2.5.1 Export do PNG

PNG (Portable Network Grahpics) je rastrový grafický formát s bezztrátovou kompresí. Vytvoření takového obrázku je velmi snadné. Stačí nad `Node` zavolat funkci `snapshot(SnapshotParameters params, WritableImage image)`. Funkce zapíše do image obraz vzledu daného uzlu. V našem případě je uzlem kreslící plátno. Pomocí volitelných parametrů můžeme nastavovat například viewport.

2.5.2 Export do PDF

K vytvoření pdf souboru z nakresleného obrázku jsem měla na výběr ze dvou možností.

První z nich byla použít program pdfLaTeX, aby ze zdrojového kódu výstupu v prostředí TikZ vygeneroval soubor formátu pdf.

Jelikož jsem se snažila, aby aplikace byla pro uživatele co nejvíce příjemná a jednoduchá na použití, rozhodla jsem se pro druhou možnost a to je použití knihovny. Vybrala jsem knihovnu Apache Batik, která mi umožnila převést obrázek ve formátu svg do formátu pdf.

Jedna z výhod, kterou tento přístup přináší je, že uživatel programu nemusí mít nainstalovaný program pdfLaTeX tím zpřístupňuje program k použití většímu množství uživatelů, kteří mohou chtít využívat ostatní funkce programu.

Problém, na který jsem při používání této knihovny narazila byl, že Batik při převodu z svg do pdf nepoužíval moje dodané fonty. Přestože uvedená cesta k nim byla správná a v různých webových prohlížečích se zobrazovaly správně. Tento problém lze vyřešit zaregistrováním vlasních fontů do grafického prostředí pomocí funkce `registerFont(java.awt.Font font)`. Načítat je pomocí `loadFont(java.io.InputStream in, double size)` nestačí, knihovna Batik je při převodu nepoužije.

2.5.3 Export do SVG

Samostatný export do SVG jsem se rozhodla zpřístupnit, protože původně jsem ho realizovala jako mezikrok pro výstup do PDF s pomocí knihovny Batik. Ale jelikož se i výstup do tohoto formátu může hodit, zařadila jsem ho jako možnost pro uživatele. Podívejme se na zápis obrázku v SVG. Zajímavé je, že vlastnosti objektů mají shodná jména s vlastnostmi nastavovanými objektům v JavaFX. Podívejme se jak vypadá zápis některých podčástí obrázku 1 v SVG.

Na řádcích 3 až 6 přidáváme font, aby byl v dokumentu dostupný. Na řádku 4 definujeme název fontu a na řádku 5 určíme cestu k souboru s tímto fontem.

Dále pokračuje výčet tvarů a jejich specifických vlastností. Úsečku definujeme podle počátečního a koncového bodu, kruh pomocí středu a poloměru apod. Křivky jsou definovány pomocí cesty (path).

Na řádku 13 až 16 se nachází skupina `<g>`, která sdružuje text a elipsu, reprezentuje tedy Ellipse Node v editoru. Tímto způsobem mohou mít oba tvary společné posunutí a rotaci. Tyto transformace se provádí pomocí atributu transform, v příkladu můžeme vidět posunutí doleva o 12. Všimněme si, že text odkazuje na výše definovaný font.

2.5.4 Export do prostředí TikZ

Každý tvar musí mít metodu `toTikZ()`. Tato metoda vrací text reprezentující daný tvar v jazyce TikZ, ve většině případů tento výstup vypadá jako v kódu 2.

Za zmínku stojí, že v editoru y souřadnice roste směrem dolů zatímco v TikZu je tomu naopak, je tedy potřeba znát celkovou výšku obrázku.

Při převodu obrázku do prostředí TikZ nejprve projdeme všechny objekty a zjistíme jaké používají fonty a barvy. Pokud je barva použita více než jednou, vytvoří se pojmenovaná barva na kterou se potom budeme dále odkazovat. To hlavně zvyšuje čitelnost zápisu. Stejná barva s jinou průhledností se počítá jako

```
<svg version="1.1" baseProfile="full" width="1000.00" height
   ="500.00" xmlns="http://www.w3.org/2000/svg">
<style>
 @font-face {
      font-family: 'TeXGyreSchola-Regular';
      src: url('fonts/texgyreschola-regular.otf') format('truetype')
      }
</style>
 <line x1="32.00" y1="16.00" x2="32.00" y2="144.00" stroke="rgb
    (9,130,9)" stroke-width="1.00" stroke-dasharray="10.0 10.0 "/>
 <circle cx="64.00" cy="32.00" r="16.00" stroke="rgb(102,154,255)"
    stroke-opacity="0.40" stroke-width="6.00" fill="rgb(102,51,102)
    " fill-opacity="0.44"/>
 <rect x="128.00" y="16.00" width="32.00" height="32.00" stroke="
    black" stroke-width="3.00" fill="rgb(230,230,77)"/>
 <polygon points="128.0, 96.0 128.0, 64.0 160.0, 80.0" stroke="rgb
    (179,26,26)" stroke-width="1.00" fill="rgb(230,230,77)"/>
 <path d="M 128.5, 144.5 C 192.5, 144.5 208.5, 144.5 192.0, 96.0"
    stroke="black" stroke-width="1.00" fill="white"/>
 <g transform=" translate(-12.00 0.00)">
  <ellipse cx="284.00" cy="80.00" rx="28.00" ry="16.00" stroke="
     black" stroke-width="1.00" fill="rgb(128,51,77)"/>
  <text x="276.30" y="87.94" font-family="TeXGyreSchola-Regular"
     font-size="22.00" textAnchor="center" fill="white">B</text>
 </g>
</svg>
```

Zdrojový kód 1: Popis vybraných tvarů z obrázku 1 v SVG

```
\draw[<styl>] (<počátek>) <název_objektu> <specifické_údaje_pro_daný
   _objekt>;
```

Zdrojový kód 2: Obecný zápis tvaru

jedna barva, protože v TikZu průhlednost definujeme zvlášť. Potom nadefinujeme příkaz na nastavení fontu. Používám velikost 10pt a text potom zvětšuji pomocí škálování, nastavovat velikost textu přímo v některých případech nefungovalo správně.

Potom už se začnou zapisovat jednotlivé objekty. Podobně jako v svg objekty typu uzel (node) v editoru se zapisují společně, nejprve tvar a potom text, ten má relativní souřadnici vzhledem k tvaru.

V tabulce 1 můžeme vidět názvy vlastností objektů v JavaFX a vedle jejich ekvivalenty v prostředí TikZ. Berme v úvahu, že tyto údaje se týkají pouze objektů, které je možné v editoru vykreslovat a vlastností, které jim lze nastavovat. A samozřejmě všechny tyto vlastnosti nejsou přístupné všem objektům např. úsečka nemá výplň apod.

```
\usepackage{color}
\usepackage{tikz}
\usepackage{xcolor}
\begin{tikzpicture}
\definecolor{color1}{RGB}{102,154,255};
\definecolor{color2}{RGB}{102,51,102};
\definecolor{color3}{RGB}{51,77,179};
\definecolor{color4}{RGB}{255,255,255};
\definecolor{color5}{RGB}{230,230,77};
\newcommand{\customqcs}{\fontfamily{qcs}\fontsize{10pt}{12
    pt}\selectfont}
\newcommand{\customcmr}{\fontfamily{cmr}\fontsize{10pt}{12
    pt}\selectfont}
\draw[ draw = black,line width=0.48pt,fill={rgb,255:red,204;green
    ,51;blue,51}, rotate around={42.10 :(38.25pt, 185.22pt)}](38.29
    pt,185.22pt) ellipse (15.32pt and 7.70pt);
\draw[draw = black,line width=1.44pt ,fill = color5] (61.26pt,231.17
    pt) rectangle ++(15.32pt,-15.32pt);
\draw[draw = {rgb,255:red,179;green,26;blue,26},line width=0.48pt ,
    fill = color5] (61.26pt,192.88pt) -- (61.26pt,208.20pt) --
    (76.58pt,200.54pt) -- cycle;
\draw[draw = black,line width=0.48pt ,fill = color4]( 61.50pt,169.43
    pt ) .. controls (92.13pt,169.43pt) and (99.79pt,169.43pt) ..
    (91.89pt,192.88pt);
\node [text={rgb,255:red,255;green,102;blue,102}, font=\customcmr,
    scale=1.05pt, ] at (103.40pt,224.95pt) {\itshape{Text}};
\draw[ draw = black,line width=0.48pt,fill={rgb,255:red,128;green
    ,51;blue,77}](130.18pt,200.54pt) ellipse (13.40pt and 7.66pt)
  node [text=color4 , font=\customqcs, scale=1.05pt, ] at ++(0.00pt
     ,-0.00pt) {B};
\draw[ draw = color1,line width=2.87pt,fill=color2,draw opacity
    =0.40,fill opacity=0.44](38.29pt,215.85pt) circle (7.66pt);
\end{tikzpicture}
```

Zdrojový kód 3: Ukázka zápisu vybráných tvarů z obrázku 1 v prostředí TikZ

2.6 Ukládání a načítání projektu

Podobně jako při exportu do jazyka TikZ, kde mají objekty metodu `toTikZ()`, má i každý objekt metodu `serialize()`, která vrací popis objektu v XML. XML text je uložen do souboru, který může být poté načten. Při načtení aplikace objekty zpět deserializuje a tím vytvoří se původní objekty. Pokud je v zápisu chyba, soubor není možné načíst a objeví se okno oznamující, že došlo k chybě.

2.7 Balíček actions

Tento balíček obsahuje výhradně třídy pro undo/redo neboli zpět a znovu funkcionalitu. Rozhraní `Action` obsahuje metody `execute()` pro znovuvykonání dané akce a `executeInverse()` pro vykonání opačné akce, tedy k odstranění

Tabulka 1: Vybrané pojmenování vlastností objektů v JavaFX a jejich ekvivalenty v prostředí TikZ

	Popis	**JavaFX**	**TikZ**
1	Tloušťka čáry ohraničující tvar	StrokeWidth	line width
2	Barva čáry ohraničující tvar	Stroke	draw
3	Průhlednost čáry	– (součástí barvy)	draw opacity
4	Barva výplně objektu	Fill	fill
5	Průhlednost výplně	– (součástí barvy)	fill opacity
6	Typ čáry, přerušovanost	StrokeDashArray	dash pattern
7	Typ fontu	Font	font
8	Rotace	Rotate	rotate
9	Barva mezi dvojitými čarami	– (neposkytuje)	double
10	Vzdálenosti mezi čarami	– (neposkytuje)	double distance

dané změny. Implementována je pomocí dvou zásobníků undo a redo. Vykonaná akce se spolu s tím nad jakými objekty byla vykonána a hodnotou před a po vykonání, uloží na zásobník undo. Poté, pokud je stisknuto tlačítko undo, se daná akce ze zásobníku odstraní a vloží se na zásobník redo. Po vykonání nové akce uživatelem se redo zásobník promaže.

2.8 Rotace

Třída `ShapeRotator` zodpovídá za rotování objektů. V programu je možné rotovat pouze jeden objekt na jednou a je rotován podle svého středu. K tomuto účelu používám třídu `javafx.scene.transform.Rotate`. Každý tvar má seznam, do kterého můžeme ukládat jeho transformace. Třída `Rotate` má jednoduchý konstruktor, kde stačí zapsat úhel a souřadnice x a y, což je bod, kolem kterého se bude rotovat. Když ji přidáme do pole transformací objektu, bude se vykreslovat právě takto rotovaný.

Kvůli čitelnosti výsledného produktu v prostředí TikZ či v svg, jsem se rozhodla, že by bylo vhodné udržovat vždy jen jednu transformaci. Pro tento účel používám `Transform.createConcatenation(Transform t)`, která vrací novou transformaci tak, že je ekvivalentní s provedením první a potom druhé transformace v tomto pořadí.

S rotací se vyskytl i jeden problém. Může se stát, že výsledná transformace může být rotací kolem velmi vzdáleného bodu o velmi malý úhel. V tomto pokud se snažíme přeložit výstup aplikace pomocí pdfLaTeXu, nastane chyba „Dimension too large“. Řešením je daný objekt smazat a vytvořit ho znovu. V ostatních formátech, do kterých jsou obrázky exportovány tato skutečnost nezpůsobuje problém.

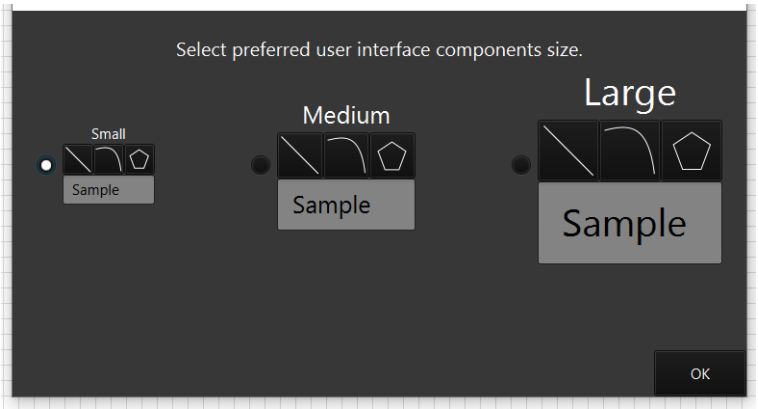
2.9 Fonty

Jelikož hlavním výstupem tohoto editoru má být právě obrázek v TikZu, zvolila jsem takový přístup, že do aplikace dodám fonty používané v LaTeXu. Tyto fonty jsou potom dále použity v ostatních exportech.

3 Uživatelská příručka

3.1 Uživatelské rozhraní

Při prvním spuštění aplikace se objeví okno, kde můžeme vybrat preferovanou velikost ovládacích prvků. Na výběr jsou small, medium a large spolu s ukázkami velikostí. Toto okno je možné kdykoliv otevřít znovu v menu v Settings/User Interface.



Obrázek 2: Výběr velikosti ovládacích prvků

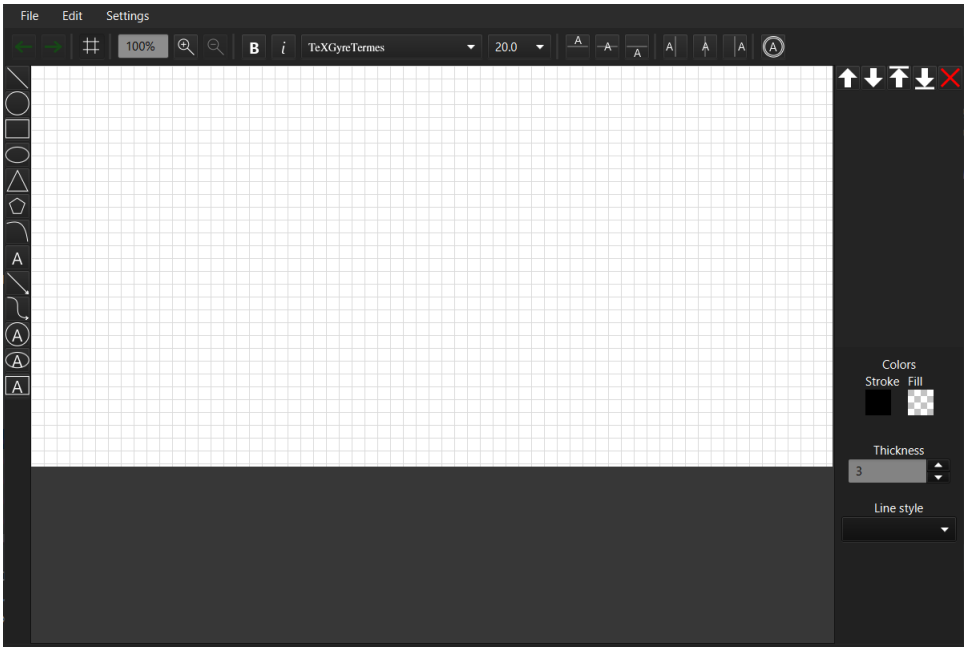
Hlavní okno aplikace se skládá z pěti částí, můžeme ho vidět na obrázku 3.

1. Aplikační menu.
2. Panel nástrojů.
3. Panel pro výběr tvarů.
4. Plátno pro kreslení.
5. Panel pro práci s objekty.

3.1.1 Aplikační menu

Nachází se v horní části aplikace.
File obsahuje položky:

- *New* slouží k vyprázdnění plátna.
- *Save* a *Load* slouží k ukládání a načítání rozpracovaného projektu. Ukládáno je do formátu xml.
- *Export* obsahuje 4 položky:
  - *Export To PNG* (exportovat do PNG) – vytvoří soubor typu png na uživatelem vybrané cestě.



Obrázek 3: Hlavní okno aplikace

  - *Export To PDF* (exportovat do PDF) – vytvoří pdf soubor na uživatelem vybrané cestě.
  - *Export To TikZ* – aplikace vytvoří v novém okně text, který popisuje obrázek v LaTeXovém prostředí TikZ.
  - *Export To SVG* – vytvoří zápis nakreslených objektů v svg. Použité fonty jsou vyexportovány spolu s tímto souborem a musí zůstat na správné cestě, tedy ve složce fonts na úrovni svg souboru, jinak nebudou ve výsledném obrázku použity.

- *Exit* ukončí aplikaci.

Settings obsahuje položky:

- Grid slouží k nastavení mřížky. Nastavuje se velikost na ose x a na ose y. Objekty se k mřížce přichytávají.
- Canvas slouží ke zmeně velikosti kreslícího plátna. Maximální velikost plátna je 2000×2000. Pokud dojde ke zmenšení plátna a některé objekty se tím ocitnou mimo plátno, aplikace se zeptá, jestli chceme objekty mimo plátno smazat, pokud ne, nedovolí plátno zmenšit.
- User Interface slouží k nastavení velikosti prvků uživatelského rozhraní.

Edit obsahuje položky:

- Undo pro vrácení poslední akce.

- Redo pro znovuprovedení vrácené akce.
- Cut pro odstranění vybraných objektů z plátna s tím, že jsou uloženy pro možné vložení.
- Copy pro zkopírování vybraných objektů.
- Paste pro vložení zkopírovaných objektů.
- Delete pro odstranění objektů.
- Select All pro vybrání všech objektů.
- Grid Toggle pro zapnutí/vypnutí mřížky.
- Bold Text pro ztučnění vybraných textových položek.
- Italic Text pro použití kurzívy na vybrané textové položky.

3.1.2 Panel nástrojů

Panel nástrojů obsahuje tlačítka pro práci s akcemi, mřížkou, textem, uzly a přibližováním/oddalováním. V tomto panelu se nachází:

- Tlačítko pro vrácení akce.
- Tlačítko pro znovuvykonání akce.
- Tlačítko pro zapnutí/vypnutí mřížky (klávesová zkratka ctrl + G).
- Textové pole zobrazující aktuální procento přiblížení kreslícího plátna.
- Tlačítko pro přiblížení (klávesová zkratka +).
- Tlačítko pro oddálení (klávesová zkratka −).
- Tlačítko pro tučný text.
- Tlačítko pro kurzivu.
- ComboBox (kombinované pole) pro výběr typu písma.
- Textové pole pro zadání velikosti písma.
- Tři tlačítka pro nastavení vertikálního zarovnání textu v uzlu (vlevo, na střed, vpravo).
- Tři tlačítka pro nastavení horizontálního zarovnání textu v uzlu (nahoru, na střed, napravo).
- Tlačítko pro nastavení/zrušení dvojité čáry u kruhového uzlu.

3.1.3 Panel pro výběr tvarů

Přes tento panel můžeme zvolit jaký tvar chceme kreslit.

Výběr tvarů – v tomto pořadí: úsečka, kruh, obdélník, elipsa, trojúhelník, polygon, kubická křivka, text, šipka, křivka zakončená šipkou, kruhový uzel, eliptický uzel a obdélníkový uzel.

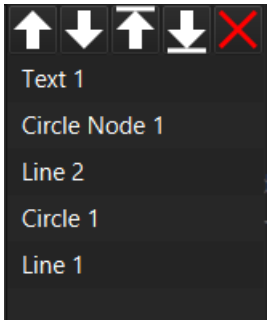
3.1.4 Kreslící plátno

Na kreslící plátno probíhá kreslení, můžete si nastavit jeho velikost. Je na něm vykreslena mřížka, kterou doporučuji používat, protože výrazně zpřesňuje kreslení, ale dá se v případě potřeby vypnout v panelu nástrojů. Kreslící plátno je možné přibližovat a to až do přiblížení 2000 %.

3.1.5 Panel pro práci s objekty

Seznam objektů – obsahuje názvy všech objektů na plátně. Kliknutím na objekt v tomto seznamu dojde k označení odpovídajícího objektu na kreslícím plátně. Při označování objektů je možné používat modifikátor shift nebo ctrl.

Použití tohoto seznamu nám umožňuje označit objekt, který je na kreslícím plátně skrytý za jiným objektem.



Obrázek 4: Seznam objektů

Šipky – slouží k přesouvání tvarů do popředí nebo do pozadí na kreslícím plátně. Platí, že prvek, který je v seznamu nejvýše, je nejvíce vpředu. Tlačítko šipky přesune objekt nebo skupinu objektů o jednu úroveň výše/níže, tlačítko šipky s úsečkou přesune objekty úplně do pozadí/popředí.

Tlačítko delete – odstraní vybraný objekt, jak ze seznamu, tak z kreslícího plátna.

Výběr barev – můžeme zde vidět dva čtverce nadepsané Stroke a Fill. Stroke nastavuje barvu čáry, Fill nastavuje barvu výplně. Po zapnutí aplikace je Stroke nastaven na černou barvu a Fill na průhlednou. Kliknutím na jeden ze čtverců se otevře nástroj pro výběr barvy, který umožnuje vybranou barvu změnit. Všimněme si volby „custom color“, která poskytuje rozmanitější nastavení barev a také umožňuje nastavovat průhlednost. Nové tvary se budou kreslit vybranými

barvami. Barvy již nakreslených jiných tvarů změníme vybráním jiné barvy přitom když je označen. Pokud je zvolená nová barva na objekt, na který se nedá aplikovat (např. šipka nemá výplň) bude volba ignorována.

Tloušťka čáry (Thickness) – hodnoty jsou omezeny od 0 do 255, můžeme je nastavovat buď šipkami, nebo napsat konkrétní hodnotu a stisknout enter.

Typ čáry můžeme nastavit čtyři druhy čar. K dispozici jsou možnosti: plná, čárkovaná, čerchovaná, tečkovaná.

3.2 Ovládání

V této sekci si popíšeme základy ovládání aplikace.

3.2.1 Kreslení objektů

K dispozici máme kreslící plátno. Aplikace se zapíná s nastavenou mřížkou 16×16. Když je mřížka zapnutá objekty se k ní přichytávájí a to levým horním rohem.Úsečka, šipka, křivka a zakřivená šipka jsou přichytávány počátečním bodem.

Po kliknutí na tlačítko s ikonou daného tvaru stačí jen na kreslícím plátně stisknutím a tažením levého tlačítka myši objekt nakreslit. Po uvolnění tlačítka myši je tvar dokončen. Toto chování se liší u polygonů a trojúhelníků, u kterých stačí jenom klikat na vybraná místa, kde se mají vytvořit body. Kreslení polygonu ukončíme v případě, že buď nakreslený bod bude velmi blízko počátečního, nebo stiskneme pravé tlačítko myši. Trojúhelník se ukončuje sám, pokud má tři body.

3.2.2 Označování objektů

Objekt je označen, pokud je jeho kreslení dokončeno. Jinak je možné označit jiný objekt a to kliknutím na něj myší. Stejně tak jako v seznamu objektů můžeme použít shift a ctrl. Pokud máme označený nějaký objekt a chceme označení zrušit, můžeme kliknout na kreslící plátno mimo jakýkoliv objekt, nebo použít klávesu Esc.

Kolem vybraného objektu se objeví ohraničující obdélník, modré kolečko se šedým okrajem a množina šedých koleček s modrým okrajem. Tažením za modré kolečko objekty posouváme. Toto se hodí hlavně když je posouvaný objekt překrytý jiným objektem. Kolečka s modrým okrajem jsou body pro úpravu.

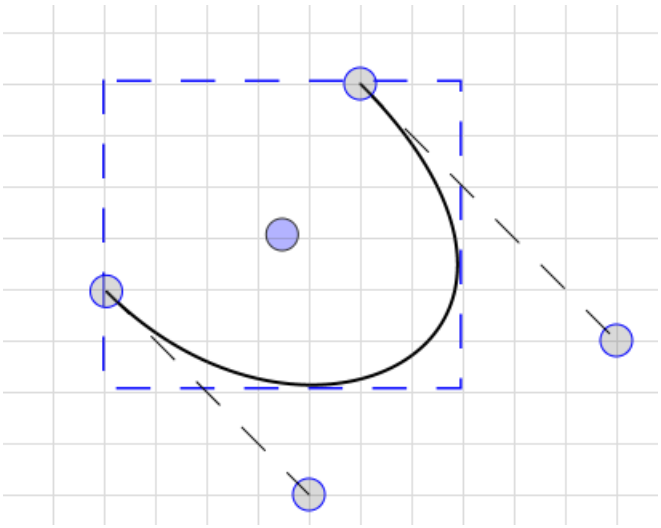
Pokud je objekt, nebo množina objektů vybrán, je možné aplikovat jakékoliv změny na všechny označené objekty, které to umožňují, stejně tak je možné posouvat všechny vybrané objekty najednou. Můžeme například použít nastavení barvy výplně na všechny objekty najednou.

3.2.3 Upravování objektů

Pokud je označen pouze jeden objekt, zobrazí se body pro jeho úpravu, můžeme je vidět na obrázku 5. Tažením těchto bodů daný tvar upravujeme. Pokud je

mřížka zapnutá body se k ní přichytávají. U několika typů objektů tažením bodů upravujeme obdélník, do kterého je objekt vepsán, jako například u elipsy. U ostatních upravujeme tažením bodů přímo i body daného tvaru, například u úsečky upravujeme její počáteční a koncový bod.

Křivky a objekty obsahující text by měly být ihned po nakreslení upraveny. U křivky potřebujeme nastavit její kontrolní body. U textu potřebujeme vyplnit textový obsah, toho docílíme tak, že na označený objekt klikneme dvojklikem anebo stiskneme klávesu enter, potom můžeme textový obsah upravovat. Ve chvíli, kdy je takto označem text, je možné upravovat jeho barvu. Ukončíme buď znovu klávesou enter, nebo kliknutím mimo upravovaný objekt, nebo použitím klávesy Escape.



Obrázek 5: Úprava křivky

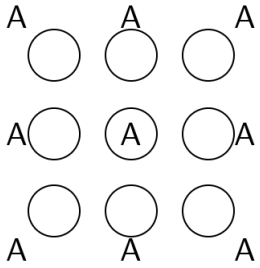
Na obrázku 5 můžeme vidět právě upravovanou křivku a její kontrolní body.

Pokud máme vybraný jeden objekt, můžeme ho rotovat podle jeho středu. Pokud najedeme kurzorem myši k rohům jeho ohraničujícího obdélníku uvidíme, že kurzor změní vzhled. Potom je možné tažením myši objekt rotovat.

Speciálním typem objektů jsou tzv. uzly. Jsou to objekty, které obsahují nějaký tvar a k tomu ještě textový popisek. Kreslí se stejně jako jakýkoliv jiný objekt a přístup k editaci textu je popsán výše. Mají ale speciální vlastnost a to je zarovnání textu. Posledních sedm tlačítek v panelu nástrojů ovlivňuje právě tento typ objektů. Můžeme si vybrat, kde má být textový popisek vzhledem k tvaru. Na výběr je devět kombinací (viz obrázek 6). Poslední z těchto tlačítek se dá aplikovat pouze na kruhový uzel a slouží ke zjednodušení kreslení koncových stavů konečných automatů.

3.2.4 Ukládání a exportování

Rozdělaný či hotový obrázek můžeme uložit. Buď pomocí položky v menu nebo pomocí klávesové zkratky ctrl + s. Uložený objekt je ve formátu .xml. Pobdob-



Obrázek 6: Možné nastavení pozice textu vzhledem ke kontejneru

ným způsobem lze projekt do programu načíst. Pokud by byl soubor s uloženými údaji neplatný nebo poškozený, aplikace to oznámí chybovou hláškou a soubor nebude možné načíst.

K exportu máme k dispozici čtyři možnosti. Export do pdf a png jednoduše vytvoří nový soubor typu .pdf či .png na uživatelem vybranou cestu. Export do TikZu otevře nové okno, které obsahuje zápis v TikZu. Uživatel si může tento text zkopírovat a použít ho ve svém LaTeXovém dokumentu, případně může použít pdfLaTeX pro vytvoření pdf obrázku z tohoto kódu (některé detaily se mohou lišit od pdf vytvořeného aplikací například šipky).

Při exportu do svg navíc kromě souboru typu .svg program vytvoří složku fonts, která bude obsahovat všechny fonty použité tímto svg. V svg souboru je uvedená cesta k těmto fontům.

Závěr

Výsledkem práce je grafický editor, napsaný v jazyce Java, který umožňuje kreslení textu a tvarů: úsečka, kruh, obdélník, trojúhelník, polygon, elipsa, šipka, křivka, křivka zakončená šipkou, kruh obsahující text, elipsa obsahující text, obdélník obsahující text. Nakreslené tvary lze upravovat, přemisťovat, rotovat a mazat. Rozdělané projekty je možné ukládat a načítat. Výsledné obrázky lze exportovat do png, svg, pdf a do jazyka TikZ.

Conclusions

The result of this thesis is graphic editor written in Java language which supports drawing of text and shapes: line, circle, rectangle, triangle, polygon, ellipse, arrow, curved arrow, circle containing text, ellipse containing text, rectangle containing text. Drawn shapes can be edited, translated, rotated and removed. Unfinished project can be saved and loaded. It is possible to export created pictures to png, svg, pdf and TikZ language.

A Obsah přiloženého DVD

`bin/`
: Soubory potřebné ke spuštění programu.

`doc/`
: Text práce ve formátu pdf a všechny soubory potřebné k bezproblémovému vygenerování pdf.

`src/`
: Kompletní zdrojové texty programu, potřebné pro pro bezproblémové vytvoření spustitelných verzí programu.

`readme.txt`
: Instrukce ke spuštění programu a další informace.

Literatura

[1] WIKIPEDIA (ed.). *Java programming language* [online] [cit. 2020-7-28] Dostupné z: https://en.wikipedia.org/wiki/Java_(programming_language)

[2] JAVA (ed.). *General information on JavaFX* [online] [cit. 2019-8-22]. Dostupné z: https://www.java.com/en/download/faq/javafx.xml

[3] WIKIPEDIA (ed.). *PGF/TikZ* [online] [cit. 2020-8-7]. Dostupné z: https://en.wikipedia.org/wiki/PGF/TikZ

[4] THE LaTeX-PROJECT (ed.). *An Introduction to LaTeX* [online] [cit. 2019-8-22]. Dostupné z: https://www.latex-project.org/about/

[5] W3SCHOOLS (ed.). *XML Tutorial* [online] [cit. 2020-8-13]. Dostupné z: https://www.w3schools.com/xml/

[6] W3SCHOOLS (ed.). *SVG Tutorial* [online] [cit. 2019-8-22]. Dostupné z: https://www.w3schools.com/graphics/svg_intro.asp

[7] THE APACHE™ BATIK PROJECT (ed.). *Apache™ Batik SVG Toolkit* [online] [cit. 2020-7-28]. Dostupné z: https://xmlgraphics.apache.org/batik/

[8] ORACLE (ed.). *Class Node* [online] [cit. 2020-8-5]. Dostupné z: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html