Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Software pro konstrukci střihů

2024

Bc. Karolína Zemenová

Vedoucí práce:
Mgr. Markéta Trnečková, Ph.D.

Studijní program: Aplikovaná informatika,
Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Karolína Zemenová
Název práce:	Software pro konstrukci střihů
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Aplikovaná informatika, Specializace: Vývoj software
Vedoucí práce:	Mgr. Markéta Trnečková, Ph.D.
Počet stran:	45
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Bc. Karolína Zemenová
Title:	Software for creating sewing patterns
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	Mgr. Markéta Trnečková, Ph.D.
Page count:	45
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Tato práce se zabývá popisem a dokumentací softwaru pro konstrukci šicích střihů, který byl vytvořen v rámci praktické části. Obsahuje přehled několika interpolačních metod a Bézierových křivek. Součástí je i srovnání vytvořeného software s již existujícími řešeními.

Synopsis

This thesis describes and documents software for creating sewing patterns, which was created in the practical part. It contains a summary of several interpolation methods and Bézier curves. It also compares the created software with existing software solutions.

Klíčová slova: Šicí střih; React; ANTLR; PDF

Keywords: Sewing pattern; React; ANTLR; PDF

Chtěla bych poděkovat svému příteli za podporu při tvorbě této práce.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod **8**
- 1.1 Existující řešení . . . 8
  - 1.1.1 CLO . . . 8
  - 1.1.2 Seamly2D . . . 9
  - 1.1.3 Patternlab . . . 9

2 Interpolační metody **12**
- 2.1 Explicitní, implicitní a parametrický popis křivky . . . 12
- 2.2 Interpolace algebraickým polynomem . . . 12
- 2.3 Lagrangeův tvar polynomu . . . 13
- 2.4 Newtonův polynom . . . 14
- 2.5 Hermitův interpolační polynom . . . 14
- 2.6 Interpolace po částech . . . 15
  - 2.6.1 Geometrická a parametrická spojitost . . . 15
- 2.7 Spline interpolace . . . 15
- 2.8 Kubický spline . . . 16
- 2.9 Bézierovy křivky . . . 17

3 Návrh software a použité technologie **19**
- 3.1 React . . . 19
- 3.2 ANTLR . . . 19
- 3.3 TypeScript . . . 20
- 3.4 Node.js . . . 20

4 Programátorská dokumentace **21**
- 4.1 Základní struktura . . . 21
- 4.2 Interpret . . . 21
- 4.3 Uživatelské rozhraní . . . 21
- 4.4 Úprava kódu . . . 22
- 4.5 Generování PDF . . . 23

5 Uživatelská příručka **25**
- 5.1 Ovládací panel . . . 26
- 5.2 Editor kódu . . . 27
- 5.3 Obrazový editor . . . 27
- 5.4 Generování PDF a tisk . . . 27

6 Dokumentace jazyka **30**
- 6.1 Proměnné a základní operace . . . 30
- 6.2 Kreslení objektů . . . 30
  - 6.2.1 Bod . . . 31
  - 6.2.2 Úsečka . . . 31
  - 6.2.3 Kružnice . . . 32

6.2.4 Křivka 32
6.2.5 Oblouk 33
6.3 Další funkce 35
6.3.1 Dělení úsečky v daném poměru 35
6.3.2 Průnik dvou objektů 35
6.3.3 Používání vlastností objektů 36
6.3.4 Komentáře 37
6.4 Styl čar 37

7 Zhodnocení a srovnání s existujícím software **39**
7.1 Ukázkové střihy 39
7.2 Zhodnocení 39
7.3 Srovnání 40

Závěr **42**

Conclusions **43**

A Obsah elektronických dat **44**

Literatura **45**

Seznam obrázků

1 CLO . . . 9
2 Seamly2D . . . 10
3 PatternLab . . . 11
4 Bernsteinovy polynomy . . . 18
5 Část syntaktického stromu . . . 23
6 Aplikace . . . 25
7 Ovládací panel (levá část) . . . 26
8 Ovládací panel (pravá část) . . . 27
9 Stránka se shrnutím . . . 28
10 Ovlivnění tvaru křivky kontrolními body . . . 33
11 Záměna počátečního a koncového úhlu oblouku . . . 34
12 Body průniku . . . 36
13 Kruhová sukně . . . 40
14 Halenka . . . 41

Seznam zdrojových kódů

1 React JSX . . . 20
2 Vykonání vlastního kódu při průchodu syntaktickým stromem . . . 20
3 Použití ANTLR . . . 22
4 Vytvoření bodu . . . 23
5 Použití svg2pdf.js . . . 23
6 Nastavení vlastností kreslení . . . 24
7 Přiřazení hodnoty do proměnné . . . 30
8 Vytvoření bodu . . . 31
9 Úsečka zadaná pomocí dvou bodů . . . 32
10 Úsečka zadaná pomocí bodu, úhlu a délky . . . 32
11 Kružnice . . . 32
12 Křivka s kontrolními body . . . 33
13 Křivka bez kontrolních bodů . . . 33
14 Křivka s vektory místo kontrolních bodů . . . 34
15 Spojení dvou bodů obloukem . . . 34
16 Část kružnice . . . 34
17 Dělení úsečky v daném poměru . . . 35
18 Získání bodů průniku . . . 36
19 Přístup k souřadnicím bodu . . . 36
20 Komentáře . . . 37
21 Přepínání stylu kreslení . . . 37
22 Kruhová sukně . . . 39

1 Úvod

Pro ušití dobře padnoucího oděvu, je základem šicí střih. Mým cílem bylo navrhnout aplikaci, která slouží k usnadnění návrhu šicích střihů. Konstruovat střih na papír je časově náročné a případná úprava znamená změnu celé konstrukce. S použitím software se změny dělají snadno.

V aplikaci je střih zapsán pomocí geometrických útvarů. Výhodou je, že je možné si takto vytvořit střih přímo na míru a v případě potřeby ho kdykoliv upravit a ušetřit si tak práci se zdlouhavým ručním překreslováním.

Na trhu je k dispozici bohatá nabídka šicích střihů ať už v papírové formě, nebo v elektronické podobě. Tyto střihy často obsahují několik univerzálních velikostí, což ale nemusí odpovídat proporcím konkrétního člověka, a proto může být potřeba dělat před šitím úpravy. Možností je navržení a vytvoření střihu přesně podle vlastních představ.

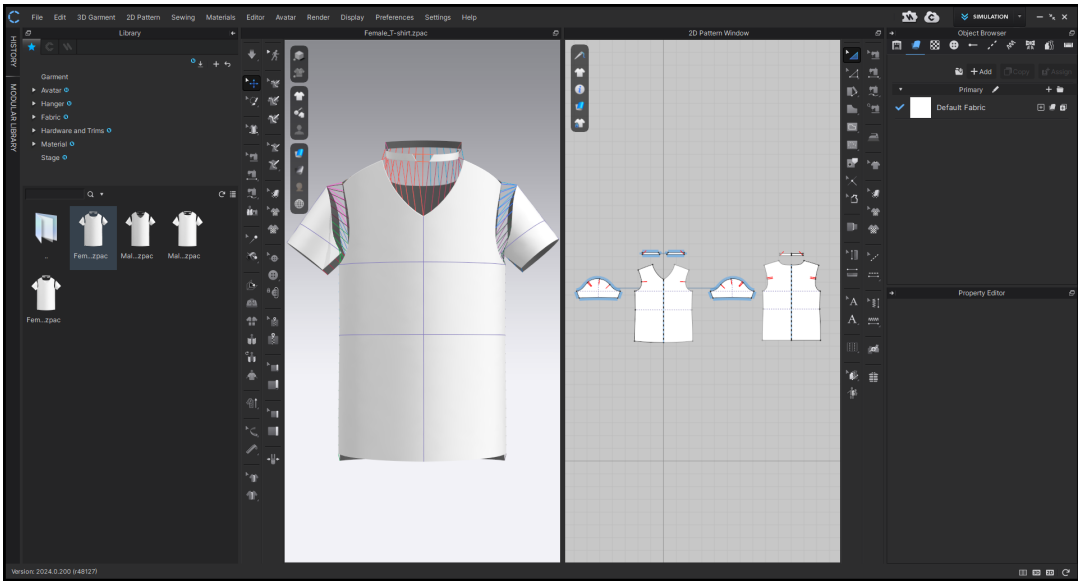
V této kapitole představím několik existujících řešení, programů, které se zabývají tvořením šicích střihů. V druhé kapitole se věnuji interpolačním křivkám a Beziérovým křivkám. Ve třetí kapitole se zaměřuji na návrh software a použité technologie. Čtvrtá kapitola se zabývá programátorskou dokumentací. Pátá kapitola obsahuje uživatelskou příručku. Šestá kapitola se zabývá dokumentací vytvořeného jazyka. Sedmá kapitola rozebírá střihy vytvořené pomocí aplikace, její zhodnocení a srovnání s existujícím software.

1.1 Existující řešení

Pro tvoření šicích střihů existují různé typy aplikací. K dispozici jsou řešení, která se hodí pro profesionální návrháře a velké podniky, ale najdou se i řešení pro jednotlivce a šicí nadšence. Setkala jsem se i aplikacemi, které slouží jako konfigurátory střihu, kde je možné si nastavit míry a zvolit si různé parametry vybraného oděvu. Často bývá nutné platit za výsledný střih jednorázově, nebo nabízejí měsíční předplatné. Ve zkratce představím jednoho zástupce z každé skupiny.

1.1.1 CLO

CLO je software zaměřený na profesionální návrháře. Soustředí se na rychlý návrh oděvu a jeho vyzkoušení na postavách ve 3D, kde může návrhář rychle vidět, jak bude oděv na postavě vypadat a jak se bude chovat při pohybu. Po zapnutí má aplikace připravený ukázkový střih trika, které je možné si rovnou vizualizovat na jednom z avatarů, které jsou k dispozici. Je možné střih upravovat a sledovat, jak se změny projevují ve 3D. Obsahuje velké množství funkcionality a pro správné používání je vhodné absolvovat trénink. Náhled aplikace je vidět na obrázku 1. Celkově se jedná o velmi složitou a propracovanou aplikaci, která nabízí i plán pro jednotlivce s cenovkou 50$ měsíčně.



Obrázek 1: CLO

1.1.2 Seamly2D

Seamly2D je open-source program pro design šicích střihů. Je k dispozici pro operační systémy Windows, MacOS, a Linux [1].

Seamly kromě samotné aplikace obsahuje ještě program, který slouží pro zadávání tělesných rozměrů SeamlyMe. Má předdefinované desítky různých měr, které se dají po importu souboru do Seamly používat. Na obrázku 2 je vidět okno aplikace s rozpracovaným střihem halenky.

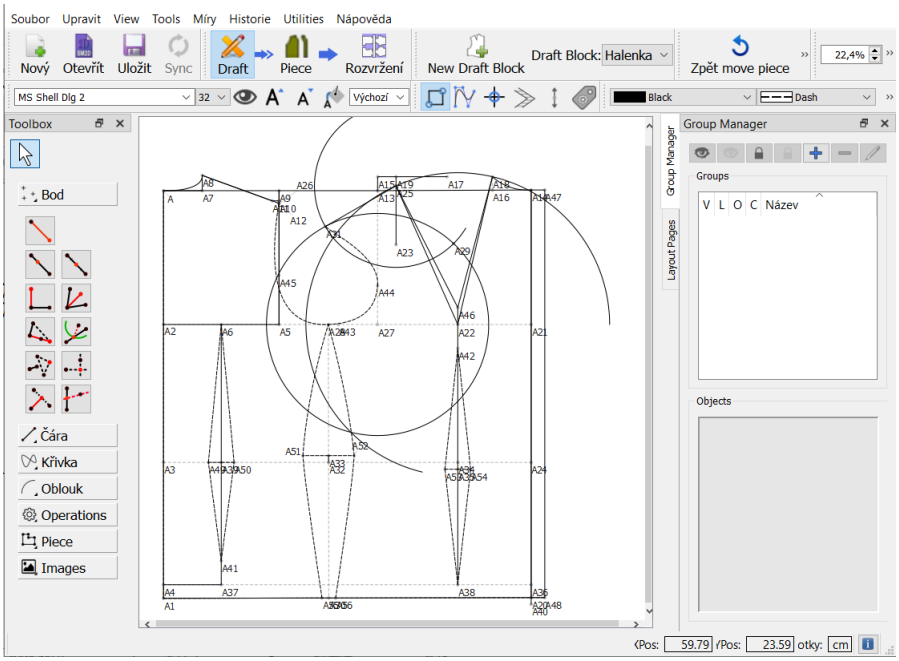
Samotný program Seamly2D umí kreslit střih od základu pomocí bodů, úseček a dalších útvarů. K dispozici má mnoho nástrojů pro kreslení podle různých pravidel. Umožňuje využívat nadefinované míry, proměnné, nebo vlastnosti již nakreslených útvarů, jako je například délka již nakreslené úsečky a používat tyto hodnoty ve vzorcích. Dle mého názoru bylo kreslení poměrně snadné, jakmile jsem si zvykla na ovládání a pochopila, jak fungují různé nástroje.

Jako negativum bych zmínila, že je hodně složitý pro nezkušeného uživatele. Osobně jsem měla problémy s použitím některých nástrojů, například největší problémy mi způsobovalo použití nástroje Add New Pattern Piece.

1.1.3 Patternlab

Webová aplikace, která neslouží přímo k samotnému kreslení střihů, ale spíš ke konfiguraci střihu. Dovoluje vybrat obecnou velikost, nebo vytvořit profil a naměřit si vlastní velikosti. Umožňuje zadat různé rozměry například výšku, délku rukávů apod.

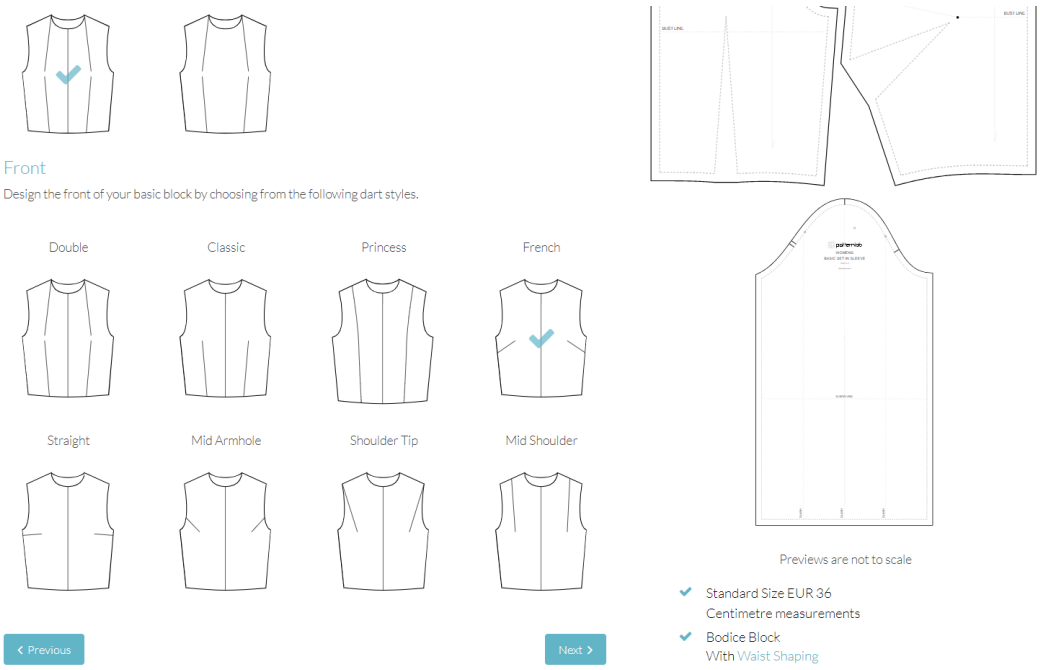
Po zadání rozměrů dává na výběr z různých částí oděvu, na které umí vytvořit střih, například sukně, šaty, kalhoty atd. Dále umožňuje vybraný oděv konfigurovat, na výběr je celá řada možností například styly záševků, tvarování



Obrázek 2: Seamly2D

v pase, kde budou švy a další. Ukázka možností výběru je na obrázku 3.

Nakonec umožňuje za poplatek stáhnout výsledný střih, a to ve formátu PDF, ale nabízí i formát SVG, kde přímo doporučuje, že si mohu střih upravit. Platit je potřeba za každý střih, například střih na kalhoty v době psaní této práce vyšel na 14.99 liber.



Obrázek 3: PatternLab

2 Interpolační metody

V následující kapitole představím různé metody interpolace křivek. Nejprve vysvětlím, jaký je rozdíl mezi interpolací a aproximací. Poté se budu věnovat několika interpolačním metodám a nakonec se budu zabývat aproximačními Bézierovými křivkami, které jsem v aplikaci použila.

Interpolace je prováděna v případě, že je dána množina bodů a hledá se křivka, která těmito body prochází. Takovým bodům se říká opěrné body (data points) a křivka se nazývá interpolační křivka.

Oproti tomu při aproximaci je snaha o nalezení křivky, která se blíží k daným bodům, ale nemusí jimi přímo procházet. Tyto body se nazývají řídící. Můžete si představit, že řídící body přitahují křivku blíže k sobě, ale křivka samotná jimi neprochází. [2]

2.1 Explicitní, implicitní a parametrický popis křivky

Křivky je možné zadávat různými způsoby.

- Křivka zadaná ve tvaru $y = f(x)$ je zadána *explicitně*. Z hodnot x se získávají hodnoty y. Je možné ji vykreslit jako křivku, ale takový zápis má nevýhodu. Není možné mít křivku procházející dvěma body se stejnou souřadnicí x.

- Křivka zadaná ve tvaru $F(x, y) = 0$ je zadána *implicitně*. Takto se dají reprezentovat i křivky, které mají dvě různé hodnoty y pro stejnou hodnotu x. Jako příklad uvedu kružnici, jejíž implicitní zadání je $x^2 + y^2 - R^2 = 0$.

- Křivka ve tvaru $P(t) = (f(t), g(t))$ je zadána parametricky. Dosazením hodnoty parametru t se získají body (x, y). Parametr t bývá obvykle určen intervalem $[0, 1]$, ale v některých případech se může hodit použít i jiný rozsah intervalu.

Křivka je *regulární*, pokud pro každé $t \in I$ platí $P'(t) \neq 0$.

Z implicitního zadání křivky je možné přejít k parametrickému, pokud je křivka *regulární*. Je-li křivka ve tvaru $y = f(x)$, pak její parametrické zadání by bylo ve tvaru $P(t) = (x, f(x))$. Přechod opačným směrem, ale obecně možný není. [3]

2.2 Interpolace algebraickým polynomem

Cílem interpolace algebraickým polynomem je nalézt předpis polynomu, který prochází všemi zadanými body. Pokud jsou zadány dva body, potom je možné je interpolovat přímkou (polynomem stupně 1). V případě, že jsou body tři, přímka už obecně stačit nebude a bude potřeba polynom stupně 2, tedy parabola. Je-li zadáno $n + 1$ bodů, pak výsledkem bude polynom stupně n. [3]

Zadány jsou 3 body $P_0 = (x_0, y_0), P_1 = (x_1, y_1)$ a $P_2 = (x_2, y_2)$.

To znamená, že polynom bude stupně 2, bude tedy ve tvaru

$$P(x) = a_0x^2 + a_1x^1 + a_2.$$

Je nutné najít koeficienty a_0, a_1 a a_2. Pro takto zadaných $n+1$ bodů se vytvoří $n+1$ rovnic. Pro každý bod $P_i = (x_i, y_i)$ se vytvoří rovnice tak, že se dosadí za x do předchozího předpisu souřadnice x_i a výsledek bude roven y_i. Tímto způsobem vzniknou 3 rovnice

$$\begin{aligned} a_0x_0^2 + a_1x_0^1 + a_2 &= y_0 \\ a_0x_1^2 + a_1x_1^1 + a_2 &= y_1 \\ a_0x_2^2 + a_1x_2^1 + a_2 &= y_2 \end{aligned}$$

se třemi neznámými. Vypočtením této soustavy rovnic se získají hodnoty koeficientů a_0, a_1 a a_2. Dosazením zpět do předpisu polynomu vznikne interpolační polynom, který prochází body P_0, P_1 a P_2. Správnost výpočtu je možné ověřit dosazením souřadnic x_i každého bodu $P_i = (x_i, y_i)$ do výsledného polynomu, výsledkem musí být jeho souřadnice y_i.

Může se stát, že i pro $n+1$ bodů bude výsledkem interpolační polynom nižšího stupně než n. Příkladem mohou být tři body, které leží na přímce. [3]

2.3 Lagrangeův tvar polynomu

Je obecný způsob získaní polynomu, který prochází danými $n+1$ body. Předpis polynomu se získá dosazením do následujícího vzorce.

$$P_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Žádné dva body nemohou mít stejnou souřadnici x. Výsledek produktu se násobí hodnotou y_i, je tedy nutné, aby hodnota produktu byla 1 v případě, že $x = x_i$ a 0 v ostatních případech.

V případě že jsou zadány tři body $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ a $P_2 = (x_2, y_2)$, bude vzorec před dosazením konkrétních hodnot vypadat takto

$$P_2 = y_0\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + y_2\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

Použitím Lagrangeova polynomu, lze získat pouze jeden předpis křivky, která prochází danými body, ale těchto křivek je nekonečně mnoho. Například přidáním jednoho dalšího bodu vznikne polynom stupně o jedna vyšší, který stále prochází původními dvěma body. Interpolační polynom, který vznikne dosazením do vzorce nemusí být předpisem právě hledané křivky. Navíc existuje pouze jeden polynom stupně n, který prochází danými body. [2]

Polynomický předpis křivky má několik nevýhod. Jednou z nich je změna celého polynomu při změně jednoho bodu. Další nevýhodou je, že křivka se vlní neboli osciluje, a to již při poměrně nízkém stupni polynomu. V praxi se obvykle používají polynomy stupně tři, nebo maximálně do stupně pět.[3]

2.4 Newtonův polynom

Newtonův polynom se liší postupem výpočtu, ale výsledek bude stejný, jako při použití Lagrangeova polynomu. Jeho výhodou je interaktivita, jelikož umožňuje přidávat další body, přičemž není potřeba počítat celý postup znovu, pouze určitou část.[2]

Je zadaných $n+1$ hodnot $P_0, P_1, \ldots, P_n$. K nim jsou přiřazeny hodnoty parametru t a seřazeny tak, že

$$t_0 = 0 < t_1 < t_2 < \cdots < t_{n-1} < t_n = 1.$$

Hledaný polynom stupně n bude ve tvaru

$$P(t) = \sum_{i=0}^{n} N_i(t) A_i,$$

kde N_i závisí pouze na hodnotách parametru t, nikoliv na hodnotách P_i.

$$N_0(t) = 1 \text{ a } N_i(t) = (t - t_0)(t - t_1)\ldots(t - t_{i-1}), \text{ pro } i = 1, \ldots, n$$

Dále je potřeba zjistit koeficienty A_i, které závisí na předchozích hodnotách A a na parametru t. Pro kratší zápis rovnic se používá metoda poměrných diferencí.

$$[t_i, t_k] = \frac{P_i - P_k}{t_i - t_k}$$

Rovnice potom vypadají takto:

$$\begin{aligned} A_0 &= P_0 \\ A_1 &= [t_1, t_0] = \frac{P_1 - P_0}{t_1 - t_0} \\ A_2 &= [t_2, t_1, t_0] = \frac{[t_2, t_1] - [t_1, t_0]}{t_2 - t_0} \end{aligned}$$

Koeficienty je možné vypočítat postupně. V případě přidání dalšího bodu se bude navíc počítat jedna hodnota N_{i+1} a jeden koeficient A_{n+1}

2.5 Hermitův interpolační polynom

Pro výpočet Hermitova interpolačního polynomu stupně 3 je potřeba počáteční bod P_1, koncový bod P_2 a hodnoty derivací v těchto bodech y_1' a y_2'. Tvar křivky je tak navíc ovlivněn hodnotami derivací v těchto bodech.[3]

Pro $n+1$ zadaných bodů a $n+1$ hodnot derivace v těchto bodech, bude výsledkem polynom stupně $2n+1$.

Představím postup pro dva body $P_0 = (x_0, y_0), P_1 = (x_1, y_1)$ a dvě hodnoty derivací v těchto bodech y_0 a y_1, hledaný polynom bude stupně 3. Ten bude ve tvaru:

$$P(x) = ax^3 + bx^2 + cx + d$$

Vytvoří se soustava rovnic o čtyřech neznámých. Dvě rovnice se získají dosazením souřadnic bodů do vzorce stejným způsobem jako v 2.2.

$$y_0 = ax_0^3 + bx_0^2 + cx_0 + d$$
$$y_1 = ax_1^3 + bx_1^2 + cx_1 + d$$

Vypočítá se $P'(x)$, což v tomto případě bude

$$P(x) = 3ax^2 + 2bx + c.$$

Protože máme zadané hodnoty derivací v bodech, je možné je dosadit do vzorce a tím získáme chybějící dvě rovnice

$$y'_0 = 3ax_0^2 + 2bx_0 + c$$
$$y'_1 = 3ax_1^2 + 2bx_1 + c$$

Nakonec už jen zbývá vyřešit soustavu rovnic, a tím získat koeficienty a, b, c a d, které stačí dosadit to předpisu polynomu.

2.6 Interpolace po částech

Pokud je zadáno více bodů tak, že by interpolací polynomem vznikl polynom vyššího stupně, je žádoucí použít interpolaci po částech. Body seřazené podle souřadnice x vzestupně se rozdělí do skupin např. po čtyřech a najdou se interpolační polynomy třetího stupně. Potom je nutné se zabývat návazností křivek v krajních bodech, protože je žádoucí zaručit určitou úroveň spojitosti.

2.6.1 Geometrická a parametrická spojitost

Pokud na sebe dvě křivky navazují v krajním bodě, mají geometrickou spojitost G^0. Pokud mají v tomto bodě i stejný směr tečných vektorů, potom mají geometrickou spojitost G^1. Má-li ve sdíleném bodě každá derivace až do stupně n stejný směr, potom má křivka geometrickou spojitost stupně G^n. Pokud mají ve sdíleném bodě i stejnou velikost první derivace, potom mají parametrickou spojitost C^1. [2] Křivka má parametrickou spojitost C^n, pokud pro každý její bod existují spojité derivace až do řádu n.[3]

2.7 Spline interpolace

Spline je množina polynomů stupně k, které jsou hladce spojeny v určitých bodech. Ve všech bodech, kde se polynomy spojují mají stejnou hodnotu derivace. Také všechny derivace v těchto bodech musí být stejné až do stupně $k - 1$. [2]

2.8 Kubický spline

Zadaných je n bodů, jsou seřazeny vzestupně podle jejich souřadnice x. Existuje nekonečně mnoho křivek, které těmito body procházejí. Obvykle je ale hledaná nějaká konkrétní křivka, a proto je potřeba, aby metoda interpolace byla interaktivní, aby umožnila najít přáve tu jednu konkrétní křivku. Tyto požadavky splňuje kubický spline. [2]

Je zadaných $n+1$ opěrných bodů, pro propojení se využije n Hermitových segmentů. Tyto segmenty budou na sdílených bodech mít shodnou hodnotu první, ale i druhé derivace. Díky tomu nebude ve výsledku poznat, kde se nacházely jednotlivé opěrné body.

Z bodů se utvoří n dvojic $(P_0, P_1), (P_1, P_2) \dots (P_{n-1}, P_n)$. Z každé dvojice bodů vznikne jeden Hermitův segment, je třeba mu dodat hodnoty derivace v těchto bodech. Hodnoty derivací jsou shodné na sdílených bodech, kterých je $n-1$. Pro každou vnitřní Hermitovu kubiku vzniknou 4 neznámé, tím tedy bude pro získání výsledku potřeba zjistit $4n$ neznámých. Ze zadaných opěrných bodů se získá $n+1$ podmínek, z napojení segmentů ve vnitřních bodech se získá $n-1$ podmínek. Z rovnosti prvních a druhých derivací ve vnitřních bodech vznikne dalších $2n-2$ podmínek. Dohromady tedy $4n-2$ podmínek, chybějící dvě hodnoty derivací v prvním a posledním bodě spline křivky zadává uživatel programu.[3]

Nastíním postup pro čtyři body $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ a $P_3 = (x_3, y_3)$. Hledají se tři polynomy třetího stupně ve tvaru

$$\begin{aligned} P_0(x) &= a_0x^3 + b_0x^2 + c_0x + d_0, \\ P_1(x) &= a_1x^3 + b_1x^2 + c_1x + d_1, \\ P_2(x) &= a_2x^3 + b_2x^2 + c_2x + d_2. \end{aligned}$$

Je třeba najít dvanáct koeficientů. Čtyři podmínky vycházejí ze zadaných opěrných bodů

$$\begin{aligned} y_0 &= a_0x_0^3 + b_0x_0^2 + c_0x_0 + d_0 \\ y_1 &= a_0x_1^3 + b_0x_1^2 + c_0x_1 + d_0 \\ y_2 &= a_1x_2^3 + b_1x_2^2 + c_1x_2 + d_1 \\ y_3 &= a_2x_3^3 + b_2x_3^2 + c_2x_3 + d_2. \end{aligned}$$

Dvě podmínky se získají ze spojitosti ve vnitřních bodech

$$\begin{aligned} a_0x_1^3 + b_0x_1^2 + c_0x_1 + d_0 &= a_1x_1^3 + b_1x_1^2 + c_1x_1 + d_1 \\ a_1x_2^3 + b_1x_2^2 + c_1x_2 + d_1 &= a_2x_2^3 + b_2x_2^2 + c_2x_2 + d_2. \end{aligned}$$

Čtyři podmínky ze spojitosti první a druhé derivace ve společných bodech

$$\begin{aligned}
3a_0x_1 + 2b_0x_1 + c_0 &= 3a_1x_1 + 2b_1x_1 + c_1 \\
3a_1x_2 + 2b_1x_2 + c_1 &= 3a_2x_2 + 2b_2x_2 + c_2 \\
6a_0x_1 + 2b_0 &= 6a_1x_1 + 2b_1 \\
6a_1x_2 + 2b_1 &= 6a_2x_2 + 2b_2.
\end{aligned}$$

Přidáním informace o hodnotách derivací v bodech P_0 a P_3 je možné vyřešit soustavu rovnic a získat tak všech dvanáct koeficientů a tím získat předpisy kubických polynomů.

2.9 Bézierovy křivky

Tyto křivky jsem použila v aplikaci, jelikož SVG využívá pro kreslení křivek Beziérovy křivky. Jedná se o aproximační metodu, kde daná křivka obecně neprochází všemi danými body, ale pouze počátečním a koncovým bodem. Křivka je k řídícím bodům přitahována a jejich vliv na ni je největší v jejich blízkosti. Výhodou je snadná úprava. [2]

Bézierova křivka je zadaná parametricky pro $n + 1$ řídících bodů jako

$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t), \text{ kde } t \in [0, 1].$$

Každému bodu je přiřazena váha, která je zde dána pomocí Bernsteinových polynomů, které jsou definovány jako

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}.$$

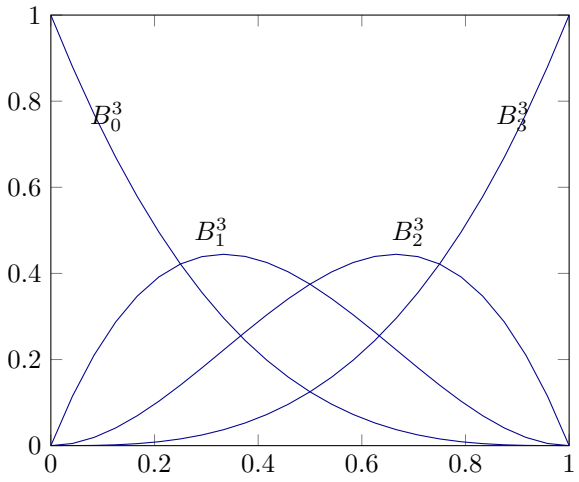
Jejich vlastností je, že součet všech Bernsteinových polynomů pro libovolné $t \in [0, 1]$ je roven 1. [3] Na obrázku 4 vidíme Bernsteinovy polynomy pro $n = 3$. Vidíme, že pro $t = 0$ je pouze hodnota prvního polynomu rovna jedné a všech ostatních rovna nule, to nám zaručuje, že křivka bude procházet svým počátečním bodem. Podobně hodnota posledního polynomu pro $t = 1$ je rovna jedné a ostatních je rovna nule a to způsobí, že křivka prochází svým koncovým bodem.

V praxi se nejčastěji používají Bézierovy kubiky, které mají tvar

$$P(t) = \sum_{i=0}^{3} P_i B_i^3(t), \ kde \ t \in [0, 1],$$

a jednotlivé Bernsteinovy polynomy budou ve tvaru

$$\begin{aligned}
B_0^3 &= (1-t)^3, \\
B_1^3 &= 3t(1-t)^2, \\
B_2^3 &= 3t^2(1-t), \\
B_3^3 &= t^3.
\end{aligned}$$



Obrázek 4: Bernsteinovy polynomy

3 Návrh software a použité technologie

Mým cílem bylo vytvořit aplikaci, která z popisu střihu vytvoří obrázek střihu. Pro popis jsem zvolila vlastní jednoduchý jazyk, který by měl být snadno pochopitelný. Střih se konstruuje pomocí geometrických útvarů, jako jsou body, úsečky, křivky a kružnice, proto je hlavní funkcí jazyka vytváření těchto útvarů.

Chtěla jsem, aby konstrukce závisela na měřených mírách tak, aby se jeden návrh střihu mohl používat opakovaně pro více osob. To byl důvod, proč jsem do jazyka zavedla proměnné. Jejich hodnotu není po přiřazení možno měnit, ačkoliv by to nebylo těžké implementovat, nemyslím si, že by z toho plynul nějaký užitek, a proto jazyk tuto funkcionalitu nemá.

Další požadovanou vlastností aplikace byla určitá interaktivita, například možnost dělat drobné úpravy pomocí obrazového editoru, protože takovým způsobem je mnohem snadnější dostat například žádaný tvar křivky. Zavedla jsem tedy zpětnou úpravu kódu, podle změn provedených v editoru obrazu tak, aby kód byl vždy aktuální.

Rozhodla jsem se pro webovou aplikaci, kvůli výhodám z toho plynoucím. Uživatel nemusí nic instalovat a aplikace je přístupná odkudkoliv, kde je k dispozici internet. Nezaměřovala jsem se na podporu malých zařízení, jako jsou mobilní telefony, protože aplikace se zaměřuje hlavně na psaní kódu a následnou úpravu obrázku, ani jeden z těchto úkonů by se na malém zařízení nedělal dostatečně pohodlně.

Pro uživatelské rozhraní jsem zvolila knihovnu React.js. Dále jsem se rozhodla použít jazyk Typescript, kvůli statické kontrole typů a snadnějšímu hledání chyb při vývoji. Pro vytvoření vlastního jazyka jsem si vybrala ANLTR.

3.1 React

React je javascriptová knihovna pro tvoření uživatelských rozhraní pomocí komponent. Složením idviduálních komponent vzniká komplexní uživatelské rozhraní. [4] S Reactem je možné používat JSX, jedná se o rozšíření syntaxe JavaScriptu, který umožňuje psát kód podobný HTML v javascriptovém souboru.[5]

V kódu 1 je vidět ukázka použití JSX v aplikaci. Ukazuje využití JSX pro zobrazení reprezentace stránky formátu A4 ve shrnutí, jak bude výsledný střih rozložen na stránkách při tisku.

3.2 ANTLR

ANTLR (ANother Tool for Language Recognition) je generátor parserů. Z formálního popisu jazyka nazývaného gramatika, umí vytvořit parser pro daný jazyk, který dokáže generovat syntaktické stromy. ANTLR také vygeneruje rozhraní k procházení syntaktického stromu, kde už je možné použít svou vlastní implementaci a vykonávat vlastní kód na základě získaného syntaktického stromu [6]. V ukázce 2 je zobrazen příklad vykonání vlastního kódu při návštěvě odkazu na proměnnou.

```
1        <React.Fragment>
2           <rect fill="none" strokeWidth={0.1 + "mm"} stroke="#888888"
                 x={a4.x} y={a4.y} width={a4.w} height={a4.h}/>
3           <text textAnchor="middle" alignmentBaseline="middle"
                fillOpacity={0.5} x={a4.x + (a4.w / 2)} y={a4.y + (a4.h
                 / 2)}
4                fontSize={a4.h / 5}>{props.index}</text>
5        </React.Fragment>
```

Zdrojový kód 1: React JSX

```
1  visitVariable = (ctx: VariableContext) => {
2     const variable = this.objectsManager.getVariable(ctx._name.text)
3     if (variable === undefined) {
4           throw new VisitorError("Variable " + ctx._name.text + " is
               not defined", ctx)
5     }
6     return variable.content
7  }
```

Zdrojový kód 2: Vykonání vlastního kódu při průchodu syntaktickým stromem

3.3 TypeScript

Typescript je silně typovaný programovací jazyk, který staví na Javascriptu. Přidává k Javascriptu další syntax pro používání typů. Kód v Typescriptu se konvertuje do Javascriptu a tak beží kdekoliv, kde beží Javascript. [7]

Díky obohacení o typy, pomáhá hledat chyby v kódu, nebo jim předcházet a při psaní může napovídat atributy, ke kterým můžeme přistoupit. Používá primitivní typy `string`, `number` a `boolean`, pro pole používá `[]` (například `string[]`) a má speciální typ `any`, který nepodléhá kontrole.

3.4 Node.js

Node.js je open-source, multiplatformní prostředí pro běh Javascriptu. Umožňuje vytvářet servery, webové aplikace, nástroje příkazového řádku a scripty [8].

4 Programátorská dokumentace

Tato kapitola se zabývá aplikací z pohledu programátora. Popisuje základní strukturu programu a představuje ukázky z některých částí například interpret, jak funguje zobrazení, jak se kód přizpůsobuje změnám v obrázku, nebo generování PDF.

4.1 Základní struktura

Aplikace je napsaná v jazyce Typescript, využívá knihovnu React.js pro uživatelské rozhraní a interpretuje uživatelský kód s pomocí nástroje ANTLR.

Uživatel píše kód ve webovém prostředí, ten je interpretován, výsledkem jsou objekty, které se zobrazí v editoru obrazu. Uživatel může dál v omezené míře interagovat s vykreslenými objekty a případné změny se projevují v původním napsaném kódu tak, aby jeho vyhodnocením vždy vznikl stejný obraz. Z obrazu je potom možné vygenerovat dokument typu PDF, kde je celý obraz rozdělen na jednotlivé stránky, připraven k tisku.

Hlavní části aplikace jsou interpret, který zpracovává jazykový vstup, uživatelské rozhraní, generátor kódu a generátor PDF.

4.2 Interpret

Interpret byl vytvořen s pomocí nástroje ANTLR. Nejprve bylo zapotřebí vytvořit gramatiku, ze které potom nástroj ANTLR vygeneruje parser.

Gramatika je uložena v souboru typu g4. Já jsem využila možnosti mít v jednom souboru pravidla lexeru i parseru společně, je ale možné je mít v rozdílných souborech. ANTLR dokáže generovat parsery v různých jazycích, v tomto případě v jazyce Typescript. Voláním parseru nad konkrétním kódem vznikne syntaktický strom. V ukázce 3 je zobrazeno základní volání parseru s navázáním vlastního `error listeneru`.

Z průchodu stromem vzniknou proměnné a objekty a provedou se potřebné výpočty. Potom se provede konverze objektů na objekty modelové, které budou zobrazeny Reactem.

Příkazy pro tvorbu objektů mají různé parametry, parametry se dají používat pojmenované, nebo nepojmenované. Parametry se jmény, jsou podle mě snadno pochopitelné, ale je zdlouhavé je psát. Proto jsem přidala možnost používat i nepojmenované parametry pro zkušenější uživatele, kteří už vědí, které parametry mají jednotlivé objekty k dispozici a jejich správné pořadí.

4.3 Uživatelské rozhraní

Pro tvorbu uživatelského rozhraní jsem využila knihovnu React.js.

Pro zobrazení střihu používám SVG. Objekty, které vygeneruje interpret, se nejprve konvertují na modelové objekty z nich se později s použitím JSX vytvoří

```
export function parseTpm(code : string) : VisitorResult {

    const inputStream = CharStreams.fromString(code);
    const lexer = new PatternMakingLexer(inputStream);
    const tokenStream = new CommonTokenStream(lexer);
    const parser = new PatternMakingParser(tokenStream);
    const errorListener = new ErrorListener()
    parser.addErrorListener(errorListener)

    const tree = parser.init()
    if(errorListener.hasErrors()){
        const error = errorListener.getFirstError()
        throw Error(`Error on line ${error.line}: ${error.msg}`)
    }

    const visitor = new TPMVisitor()
    return visitor.visit(tree)
}
```

Zdrojový kód 3: Použití ANTLR

SVG elementy, které se vykreslí do SVG plátna. Toto plátno spravuje komponenta `Canvas`. SVG jsem zvolila, protože nabízí jednoduchou manipulovatelnost s objekty, kde je možné na jednotlivé objekty navázat `onClickEvent` a tím zjistit, na který z nich uživatel klikl.

4.4 Úprava kódu

Protože umožňuji, aby uživatel měnil polohu bodů v obrázku, musela jsem najít způsob, jak upravit kód odpovídajícím způsobem, aby reflektoval uživatelské změny a vyhodnocením nového kódu vznikl stejný obraz. Tuto funkcionalitu obsluhuje třída `CodeEditor`.

V případě posunutí bodu dojde k přepočtu, je zjištěn kontext posunutého bodu a na základě tohoto kontextu se vykoná odpovídající přepočet. Například pro bod, který vznikl dělením úsečky se vypočítá nový poměr dělení, který bude odpovídat místu, kam byl bod přesunut.

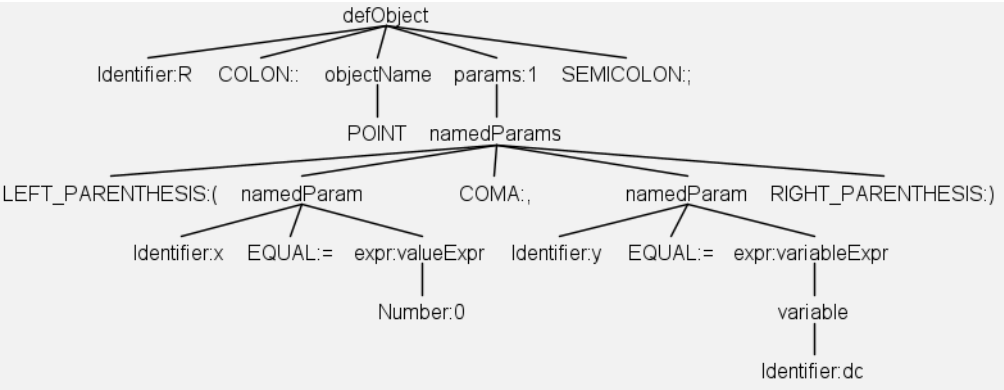
Při posunutí bodu, který vznikl zadáním souřadnic pomocí nějakého výpočtu, se `CodeEditor` snaží tento výpočet upravit co nejméně, aby zůstala zachována původní závislost. Například z `p : Point(point0.x / 5, point1.y - 5);` posunutím bodu p může vzniknout `p : Point(point0.x / 5+0.9, point1.y +15.2);`, zůstane tak zachována původní závislost na souřadnicích jiných bodů.

Aby toto bylo možné objekty si uchovávají kontext, ze kterého vznikly. Tento kontext je část syntaktického stromu, který vznikl voláním parseru. `CodeEditor` tak najde potřebnou část kódu, kterou je potřeba změnit a nahradí ji za novou, aktuální.

Můžeme si to ukázat na jednoduchém příkladu, kdy se změní souřadnice bodu, který je zapsán v kódu 4. Na obrázku 5 je vidět syntaktický strom, který vznikl na základě tohoto kódu. `CodeEditor` může ve stromu najít uzel odpovídající hodnotě parametru `y`, což je v tomto případě proměnná `dc` a upravit hodnotu tohoto parametru odpovídajícím způsobem.

```
R : Point(x = 0, y = dc);
```

Zdrojový kód 4: Vytvoření bodu



Obrázek 5: Část syntaktického stromu

4.5 Generování PDF

Pro generovaní PDF jsem použila knihovnu svg2pdf.js. Sama tato knihovna využívá ještě knihovnu jsPDF. Je jednoduchá na použití jak je vidět v kódu 5.

```
const doc = new jsPDF('p', 'mm', [A4.A4_HEIGHT_IN_MM, A4.
    A4_WIDTH_IN_MM]);

const result = await doc.svg(element, {
    x: -page.x + A4.MARGIN_LEFT,
    y: -page.y + A4.MARGIN_TOP,
    width: A4.A4_WIDTH_IN_MM,
    height: A4.A4_HEIGHT_IN_MM,
    loadExternalStyleSheets: true
})
```

Zdrojový kód 5: Použití svg2pdf.js

Jeden z problémů, na který jsem narazila, je, že tato knihovna umí tvořit PDF pouze z existujícího elementu na stránce. Pro potřeby v programu by bylo

lepší, kdyby jí stačilo poskytnout svg ve formě textu. Protože PDF má několik složek, obsahuje stránku se shrnutím a potom jednotlivé kusy střihu rozdělené na více stránek. Toto omezení se mi podařilo obejít tím, že jsem do SVG, které vidí uživatel přidala dvě další části, které mají jiné vlastnosti (např. tloušťku čar), tak, aby se v PDF zobrazily správně, a mají nastaveno `visibility: none`.

Dohromady se z těchto částí složí výsledné PDF a přidají se popisky, čísla stránek a ohraničení. Knihovna jsPDF umožňuje kreslení na stránku PDF přímočarým způsobem, jak je vidět v kódu 6.

```
1 doc.setFillColor(255, 255, 255)
2 doc.setDrawColor(0, 0, 0)
3 doc.setLineWidth(0.1)
4 doc.setLineDashPattern([2, 2], 0)
5 doc.rect(0, 0, A4.A4_WIDTH_IN_MM, A4.MARGIN_TOP, "F")
```

Zdrojový kód 6: Nastavení vlastností kreslení

Před vytvořením PDF dokumentu dojde k přepočtu jednotlivých stránek a dojde k pokusu o odstranění zbytečných prázdných stránek, to jsou stránky, které sousedí s okrajem, nebo jinou prázdnou stránkou a jsou prázdné. Tato funkcionalita neodstraní všechny přebytečné prázdné stránky, protože rozměry objektů počítá pomocí obdélníků a takový výpočet není dostatečně přesný, aby odhalil všechny takové případy.

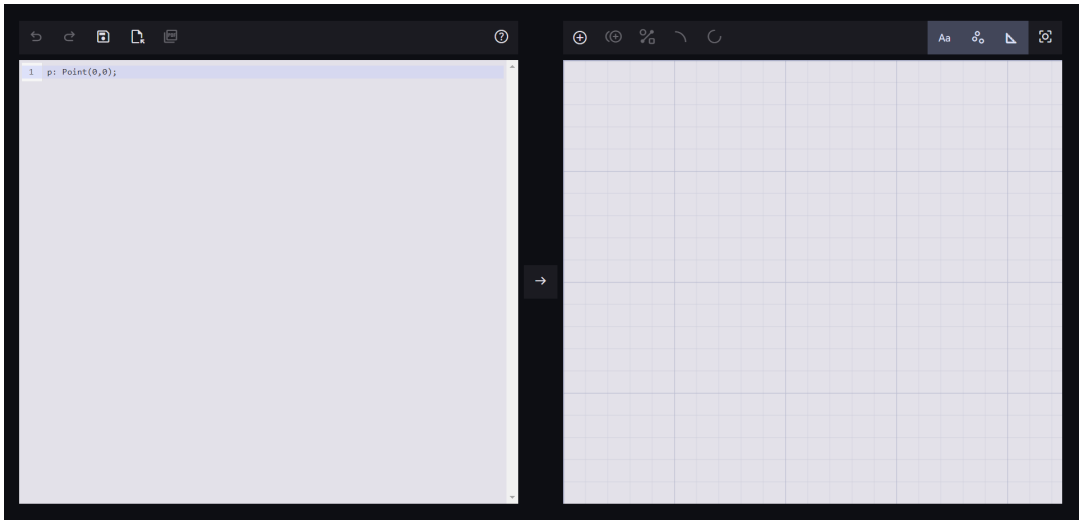
5 Uživatelská příručka

V této kapitole vysvětlím, jak aplikaci používat. Popíšu ovládání, jak si vytvořit střih pomocí jazyka a jak si navržený střih vytisknout. Pro účely testování aplikace běží na http://158.194.92.115:3000/.

Na obrázku 6 je vyobrazena aplikace, takto vypadá po zapnutí. Skládá se ze dvou hlavních částí, textového editoru a obrazového editoru. Nad oběma částmi se nachází několik tlačítek.

Zamýšlený způsob použití je takový, že v levém editoru kódu si uživatel zapíše předpis pro střih pomocí speciálního jazyka, aplikace tento předpis vyhodnotí a zobrazí podle něj výsledný střih, který si může ještě upravit. Dále je zde možnost si ho vytisknout na papíry velikosti A4, které se potom slepí k sobě.

Levá část, editor kódu, slouží pro psaní kódu, jak psát kód je vysvětleno v kapitole 6. Pomocí kódu je možné postupně tvořit střih použitím bodů, úseček, kružnic, křivek a oblouků. Stisknutím tlačítka uprostřed se symbolem šipky se provede vyhodnocení kódu a vše co bylo zapsáno kódem se zobrazí v pravém okně, kde je možné také provádět úpravy, které se ještě zpětně projevují kódu tak, aby byl vždy aktuální.



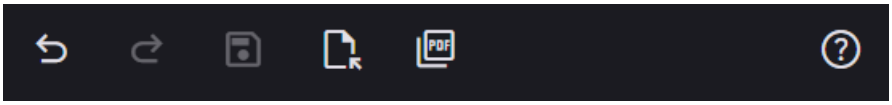
Obrázek 6: Aplikace

Vpravo, v editoru obrázku, je vidět výsledek, který vznikl ze zapsaného kódu. Body, které jsou reprezentovány šedými kruhy a je možné je posouvat pomocí stisknutí levého tlačítka myši a tažení myší. Kliknutím na bod jej označíme, to se projevuje změnou jeho barvy. V obrazovém editoru se můžeme pohybovat pohybem myši při stisknutí kolečka. Otáčením kolečka myši si obraz přibližujeme a oddalujeme.

5.1 Ovládací panel

Levá část ovládacího panelu je zobrazena na obrázku 7. Na obrázku vidíme šest tlačítek.

- Undo – vrátí poslední vykonanou akci zpět
- Redo – zopakuje poslední vrácenou akci pomocí tlačítka undo
- Uložit – stáhne soubor se současným zdrojovým kódem
- Nahrát soubor – umožní nahrát soubor se zdrojovým kódem typu tpm
- Přepnutí do pohledu pro pdf – aplikace se přepne se do náhledu před vygenerováním pdf souboru
- Nápověda – zobrazí nápovědu k jazyku v novém okně



Obrázek 7: Ovládací panel (levá část)

Na obrázku 8 vidíme pravý ovládací panel. Tlačítka v tomto panelu slouží pro interakci s editorem obrazu.

- Přidávání bodů – tlačítko zapne, nebo vypne režim přidávání bodů kliknutím na místo v editoru pomocí levého tlačítka myši
- Přidání relativního bodu – vyžaduje jeden označený bod. Vytvoří bod, který má nastavené souřadnice stejné jako označený
- Přidání úsečky – vyžaduje dva označené body. Vytvoří úsečku, která spojuje tyto dva body, v pořadí v jakém byly vybrány
- Přidání křivky – vyžaduje dva nebo čtyři označené body. V případě dvou označených bodů budou tyto body začátek a konec křivky a kontrolní body se vytvoří uprostřed. V případě čtyř bodů bude křivka vytvořena podle pořadí označených bodu a to počáteční bod, první kontrolní bod, druhý kontrolní bod a koncový bod
- Viditelnost popisků – zapne nebo vypne zobrazení popisků objektů.
- Viditelnost bodů – zapne nebo vypne zobrazení bodů
- Viditelnost měření – zapne nebo vypne zobrazení hodnot měření délek
- Zobrazit celý střih – nastaví editor tak, aby byl zobrazen celý střih

Obrázek 8: Ovládací panel (pravá část)

Tlačítko se symbolem šipky mezi textovým a obrazovým editorem je tlačítko pro vyhodnocení kódu, pokud na něj klikneme aplikace přečte kód a najde případné chyby, nebo varování a pokud k žádné chybě nedojde, zobrazí obrázek v obrazovém editoru. Stejného efektu dosáhneme i pokud po napsání textu klikneme mimo editor kódu.

5.2 Editor kódu

Do textového editoru zapisujeme kód. Tento editor má na své levé straně panel s čísly řádků, to nám může pomoci při hledání chyby. Pokud v kódu nastane chyba, objeví se červené oznámení o chybě na spodní straně textového editoru, toto oznámení obsahuje spolu s informací o chybě také číslo řádku. Kromě chyby se také může objevit i žluté varování, které upozorňuje na možný problém. Obecně je doporučeno psát každý příkaz na nový řádek, je tak snadnější nalézt chybu. Chyba se obecně může nacházet i na jiném řádku, než je napsáno v oznámení o chybě, například o jeden řádek výš.

5.3 Obrazový editor

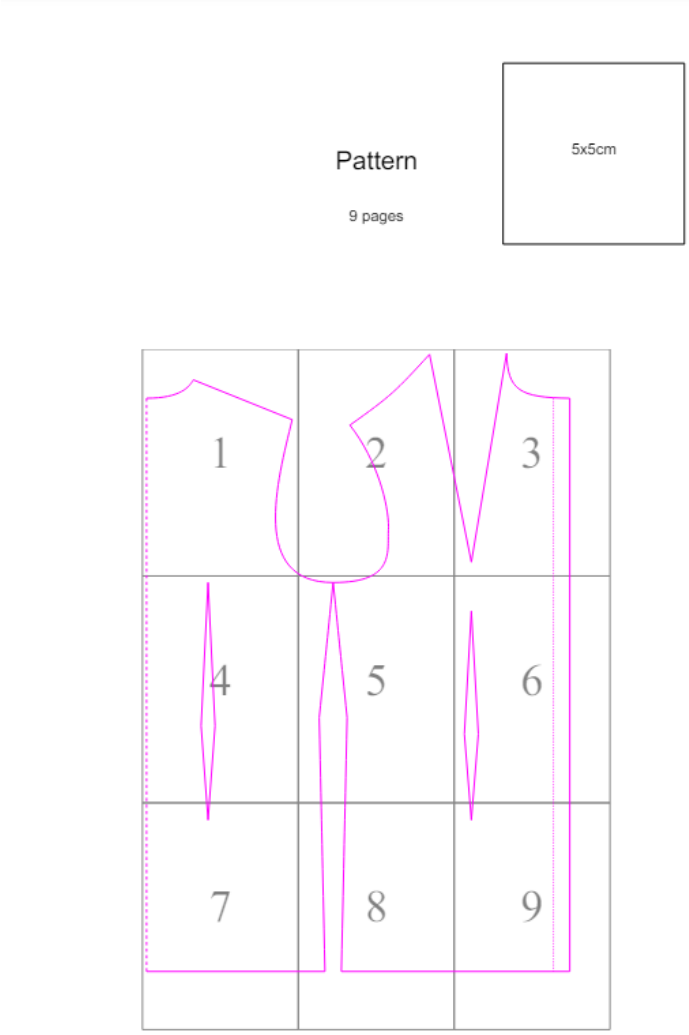
V obrazovém editoru vidíme výsledný obrázek, který vznikl vyhodnocením kódu. Vidíme objekty, jejich jména, u úseček i jejich délku. Body můžeme posouvat, pokud tak učiníme, dojde k přepsání kódu. Změny provedené v obrazovém editoru, se automaticky projeví v kódu, ať už jde o přidání objektů, nebo o posunutí existujících bodů.

Pokud si v kódu zavedeme nějaká omezení, například bod ležící na úsečce, bude toto omezení platit i v editoru, nebudeme schopni bod posunout mimo úsečku. Body, které byly definovány jako průniky, také není možné posouvat.

5.4 Generování PDF a tisk

Program umožňuje vygenerovat dokument typu PDF, který je přizpůsoben pro tisk střihu na tiskárně. K získání PDF souboru je třeba nejprve mít funkční kód, který po stisknutí tlačítka se symbolem šipky vytvoří střih. Potom bude možné stisknout tlačítko PDF, kterým se aplikace přepne do PDF pohledu. V tomto pohledu už nejsou vidět některé objekty v obrázku, například body, popisky, také nejsou vidět objekty nakreslené štětcem `noprint`. Je zde nepovinné pole pro pojmenování střihu. Po kliknutí na tlačítko Download PDF, by se měl stáhnout soubor PDF, případně je nutné stažení souboru potvrdit.

Na obrázku 9 je zobrazena první stránka vygenerovaného PDF, která obsahuje shrnutí střihu, náhled střihu, jak je rozdělený na jednotlivých očíslovaných stranách A4. Na této stránce se nachází i kalibrační čtverec.



Obrázek 9: Stránka se shrnutím

Před tiskem je nutné si ověřit, že tisk proběhne v měřítku 100 % a nebude docházet ke zmenšení, nebo naopak zvětšení vytisknutého obrázku. V takovém případě rozměry střihu nebudou odpovídat nastaveným rozměrům v programu. Zprvu je vhodné vytisknout si pouze první stránku a ověřit, že kalibrační čtverec má strany dlouhé přesně 5 cm. Pokud tomu tak není, střih se nevytiskne správně.

Po vytištětní všech stran ve správném měřítku, už se mohou slepit k sobě. Na první stránce se shrnutím je zobrazeno, jak mají být stránky seřazeny, aby vytvořily daný střih, číslo ve shrnutí je vytištěno na spodním pravém rohu stránky. Každá stránka má okraje, vyobrazené přerušovanou čárou, tyto okraje můžeme odstřihnout.

Někdy se může stát, že v dokumentu PDF budou stránky bez obsahu (kromě okrajů a čísla stránky). Pokud je se jedná o stránku na kraji celého střihu, která není ze všech stran obklopena stránkami s obsahem, doporučuji tuto stránku z tisku vyřadit.

6 Dokumentace jazyka

V této kapitole vysvětlím, jak psát kód. Použití vysvětlím na příkladech. Všechny hodnoty, které v kódu zadávám jsou v milimetrech. Pro zadávání střihů je k dispozici jednoduchý jazyk, který umí kreslit několik druhů geometrických útvarů, také umožňuje uložit si číselnou hodnotu do pojmenované proměnné, provádět jednoduché výpočty a dokáže přistupovat k vlastnostem vytvořených objektů.

6.1 Proměnné a základní operace

V jazyce jsou k dispozici proměnné, do kterých je možné uložit číslo. Proměnná má své jméno a hodnotu. Pro definici je potřeba použít klíčové slovo `var`. Hodnoty jsou v milimetrech.

```
1 var obvod_pasu = 600;
2 var pas_predni = obvod_pasu / 2 + 10;
```

Zdrojový kód 7: Přiřazení hodnoty do proměnné

V kódu 7 se vytvořila proměnná `obvod_pasu`, do které se uložila hodnota 600, hodnota je v milimetrech a odpovídá 60 cm. S použitím této hodnoty se potom vypočítala nová hodnota a uložila se do proměnné `pas_predni`.

V aplikaci je možné používat operace sčítání (`+`), odčítání (`-`), násobení (`*`) a dělení (`/`) a závorky `(` a `)` pro přednost operací. Tyto operace se mohou používat nejen při vytváření proměnných, ale také při zadávání parametrů jednotlivých objektů.

Proměnné by měli sloužit hlavně k tomu, aby se do nich uložili míry a také výsledky výpočtů z nich odvozených. Pokud se potom při práci s těmito hodnotami dobře nastaví vzájemné závislosti mezi objekty, dá se jednoduše změnit hodnota proměnné podle míry daného člověka, pro kterého bude střih vytvořen, a celý střih se vykreslí s novými hodnotami. Potom možná bude potřeba udělat nějaké drobné úpravy v závislosti na konkrétním střihu.

6.2 Kreslení objektů

Pomocí jazyka je možné kreslit pět základních geometrických útvarů:

- bod,
- úsečka,
- kružnice,
- křivka,
- oblouk.

Každý útvar musí mít své jméno, pomocí tohoto jména je možné se na něj později odkazovat. Jeho jméno je také zobrazeno v obrázku.

Definice objektů mají stejný základ, objektu je nutné přiřadit jméno, zadat jeho typ a dodat parametry, které se liší u jednotlivých typů objektů (Definice bodu a křivky budou mít různé parametry). Parametry jsou umístěné v závorkách „(“ a „)“. Každý pojmenovaný parametr je ve tvaru: `<jméno> = <hodnota>`. Pokud je pojmenovaných parametrů více jsou odděleny čárkami a na jejich pořadí nezáleží. Parametry je možné zadávat i nepojmenované, oddělené čárkami, v tomto případě záleží na jejich pořadí. Ukázky kódu v následujících kapitolách, které používají pojmenované parametry, je používají v pořadí, v jakém by se používali v nepojmenovaném tvaru. V definici jednoho objektu se nesmí střídat pojmenované a nepojmenované parametry.

6.2.1 Bod

Bod je možné zadat několika způsoby, nejjednodušší způsob je pomocí souřadnic x a y. Kód pro jednoduché vytvoření bodu je v ukázce 8. Pojmenování parametrů se může vynechat, v takovém případě budeme muset vždy zadat jako první hodnotu souřadnice x a na druhém místě vždy hodnotu souřadnice y.

```
1 p : Point(x = 10, y = 10);
2 q : Point(15, 15);
```

Zdrojový kód 8: Vytvoření bodu

Tímto zápisem se vytvoří bod se jménem p, který bude ležet na souřadnicích [10,10], a bod q, který bude ležet na souřadnicích [15,15]. Po vložení tohoto kódu do okna editoru a stisknutí tlačítka pro vykonání kódu se v obrazovém editoru zobrazí body p a q. Tyto body se dají libovolně posouvat. Když se bod posune tímto způsobem, změní se i hodnoty parametrů bodu v kódu.

Všechny ostatní objekty body nějakým způsobem využívají, body tvoří základ celého střihu, a pouze body je možné posouvat v editoru obrazu. Každý objekt je pojmenovaný, jméno vidíme v editoru. Pokud budeme chtít nějaký bod využít při vytváření dalšího objektu, odkážeme se na něj jeho jménem, stačí do hodnoty parametru napsat jméno bodu, na který se chceme odkázat.

6.2.2 Úsečka

Jeden způsob, jak vytvořit úsečku, je pomocí dvou bodů, je vidět v ukázce 9. Všimněte si, že v příkladu se odkazuje na dříve definované body `p1` a `p2`. Opět zde jsou dvě možnosti vytvoření (s pojmenovanými i nepojmenovanými parametry).

Úsečku je možné definovat také pomocí jednoho bodu, úhlu a délky, jak je vidět v kódu 10. Start bude určovat začátek úsečky, orientation určuje úhel úsečky,

```
p1 : Point(x = 100, y = 0);
p2 : Point(x = 200, y = 50);
l : Line(start = p1 , end = p2);
k : Line(p1, p2);
```

Zdrojový kód 9: Úsečka zadaná pomocí dvou bodů

tak že 0 je vodorovná úsečka, 90 je svislá. Pro tento případ je možné použít i slovo „horizontal“ resp. „vertical“. Length určuje délku úsečky.

```
p : Point(x = 0, y = 0);
l : Line(start = p, orientation = 45, length = 20);
```

Zdrojový kód 10: Úsečka zadaná pomocí bodu, úhlu a délky

Úsečka bude začínat v bodě p a bude pokračovat pod úhlem 45° až do délky 2 cm. Ačkoliv v zadaní úsečky není koncový bod, v obrázku se nachází bod se jménem l.end. I tento bod se dá posouvat, ale omezuje ho nastavená orientace úsečky, posunutí mění délku úsečky, ale nemění její směr. I na takto vytvořený bod je možné se odkázat použitím jeho jména.

6.2.3 Kružnice

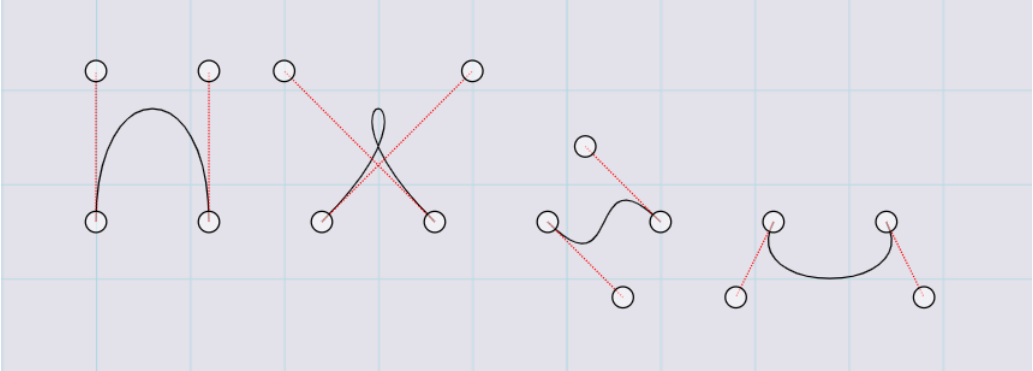
Nakreslení kružnice je ukázáno v kódu 11, stačí pouze jeden bod, který bude středem kružnice a poloměr. Pokud není žádoucí celá kružnice, ale pouze její část, nabízí jazyk také kruhový oblouk, který je vysvětlen v sekci 6.2.5.

```
p : Point(x = 0, y = 0);
c : Circle(center = p, radius = 20);
```

Zdrojový kód 11: Kružnice

6.2.4 Křivka

Křivka se zadává pomocí dvou nebo čtyř bodů. Křivka vždy musí mít počáteční a koncový bod, které určují její začátek a konec, má také dva kontrolní body, které však není nutné přímo zadávat. Vytvoření křivky se zadanými kontrolními body je v kódu 12. V případě, že kontrolní body nejsou zadány, jako v ukázce 13, aplikace si je sama vytvoří. Posunutím kontrolních bodů se mění tvar křivky, nejjednodušší způsob je posouvat body přímo v editoru obrazu a sledovat, jak jejich poloha ovlivňuje tvar křivky. Toto ovlivňování je vidět i na obrázku 10.



Obrázek 10: Ovlivnění tvaru křivky kontrolními body

```
1 s~: Point(x = 0, y = 0);
2 e : Point(x = 100, y = 0);
3 c1 : Point(x = 40, y = 50);
4 c2 : Point(x = 60, y = 50);
5 c : Curve(start = s, end = e, control1 = c1, control2 = c2);
```

Zdrojový kód 12: Křivka s kontrolními body

Pokud některý z kontrolních bodů není zadán, bod se automaticky vytvoří ve středu mezi počátečním a koncovým bodem a je k dispozici v editoru obrazu k posunutí.

```
1 s : Point(x = 0, y = 0);
2 e : Point(x = 100, y = 0);
3 c : Curve(start = s, end = e);
```

Zdrojový kód 13: Křivka bez kontrolních bodů

Namísto kontrolního bodu křivka přijímá i vektor. Vektor zadáváme ve tvaru `(x,y)`. Výhodou takového zadání je, že vektor určuje polohu kontrolních bodů relativně vzhledem k počátečnímu resp. koncovému bodu. Kdyby došlo k posunutí jednoho z bodů křivky, nezmění se poloha kontrolního bodu vůči změněnému. Použití vektorů je ukázáno v kódu 14.

6.2.5 Oblouk

Kruhový oblouk se dá vykreslit dvěma způsoby. První způsob je užitečný v případě, kdy je dispozici počáteční a koncový bod oblouku a poloměr. V takovém případě dojde k propojení těchto dvou bodů obloukem, částí kružnice s daným poloměrem. Toto použití je zobrazeno v kódu 15. Pokud není možné body spojit, protože poloměr je příliš malý, automaticky se zvětší tak, aby to bylo možné.

```
s : Point(x = 0, y = 0);
e : Point(x = 100, y = 0);
curve0: Curve (start = s, end = e, control2 = (-22.5,40.5), control1
    = (23.5,38.5));
```

Zdrojový kód 14: Křivka s vektory místo kontrolních bodů

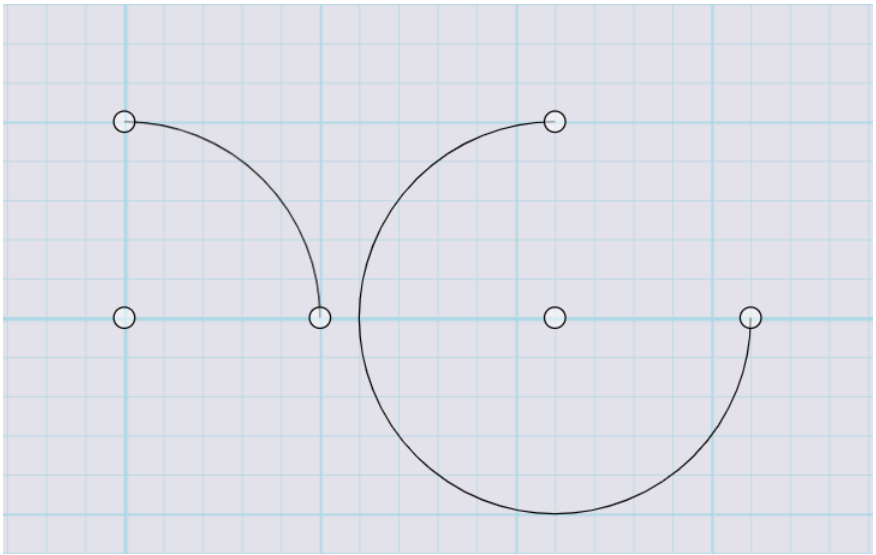
```
point0: Point(x = 0, y = 0);
point1: Point(x = 75, y = 0);
arc0 : Arc(start = point0, end = point1, radius = 60);
```

Zdrojový kód 15: Spojení dvou bodů obloukem

Druhý způsob kreslení oblouku se hodí v případě, že je znám střed a je potřeba nakreslit pouze část kružnice. V takovém případě se pro zadání oblouku použijí střed, poloměr, počáteční úhel a koncový úhel, jako v příkladu 16. Kružnice se vykreslí od počátečního úhlu v proti směru hodinových ručiček ke koncovému úhlu. Záměnou těchto dvou úhlů dojde k vykreslení opačné části kružnice, jak je vidět na obrázku 11.

```
stred: Point(x = 0, y = 0);
arc : Arc(center = stred, radius = 35, angle1 = 50, angle2 = 90);
```

Zdrojový kód 16: Část kružnice



Obrázek 11: Záměna počátečního a koncového úhlu oblouku

6.3 Další funkce

Body není nutné vytvářet pouze na určitých souřadnicích, ale mohou být závislé na některých už existujících objektech. Například jeden bod může ležet uprostřed úsečky a druhý se může nacházet v místě průniku dvou úseček. Využívání těchto funkcí je důležité, pokud má střih být obecný a má se používat opakovaně s jinými mírami.

6.3.1 Dělení úsečky v daném poměru

Jako první ukážu na příkladu, jak vytvořit bod na úsečce tak, že ji bude dělit v nějakém poměru. V kódu 17 stačí použít příkaz `divide`, jehož argumenty jsou úsečka a poměr dělení. Výsledkem je bod, který dělí úsečku v daném poměru, v příkladu je pojmenován `a`. Bod `a` tedy dělí úsečku v poměru jedna ku dvěma to znamená, že bude ležet ve vzdálenosti $\frac{1}{3}$ délky úsečky od počátečního bodu `point0`.

```
1 point0: Point(x = 0, y = 0);
2 point1: Point(x = 20, y = 0);
3 line0: Line(start = point0,end = point1);
4 a : divide(line0, 1:2);
```

Zdrojový kód 17: Dělení úsečky v daném poměru

Namísto zadání poměru, podporuje příkaz `divide` i použití zápisu v procentech. Předchozí příklad by se v procentech zapsal přibližně jako `a : divide(line0, 33%);`. Povolené hodnoty jsou od 0 % do 100 %.

6.3.2 Průnik dvou objektů

Další možností jak získat body, je využití průniků. Aplikace dokáže najít body průniku mezi úsečkami a kružnicemi. Nepodporuje výpočty průniků s body, křivkami ani oblouky.

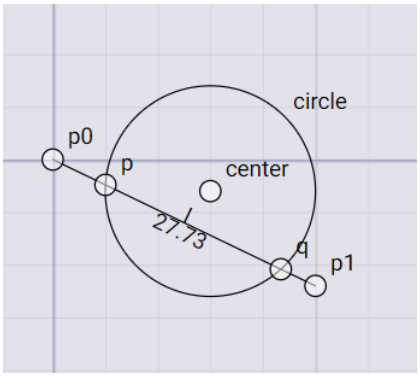
K nalezení průniku dvou objektů stačí použít příkaz `intersect`, kde jako argumenty zadáváme jména objektů, mezi kterými hledáme průnik. Použití je podobné jako u příkazu `divide`, ale jeho výsledkem může být více bodů, proto je možné zadat více názvů bodů oddělených čárkami.

V kódu 18 došlo k vytvoření úsečky a kružnice. Na řádku 6 se použil příkaz `intersect` pro nalezení bodů průniku. Tyto body se uložily do proměnných p a q. Výsledek je zobrazen na obrázku 12.

Bodů průniku ale může být proměnlivý počet v závislosti na objektech mezi kterými se průnik počítá. Mohlo by se stát, že se očekávají dva body průniku, ale při dané poloze objektů bude existovat jen jeden. V takovém případě dojde k upozornění pomocí varování, která se zobrazují v levé dolní části. Dokud nedojde k odkázání na neexistující bod v dalším příkazu nenastane chyba. V opačném

```
p0: Point(x = 0, y = 0);
p1: Point(x = 25, y = 12);
l : Line(start = p0, end = p1);
center : Point(x = 15, y = 3);
circle : Circle(center = center, radius = 10);
p,q : intersect(l, circle);
```

Zdrojový kód 18: Získání bodů průniku



Obrázek 12: Body průniku

případě, kdy je potřeba pouze jeden bod průniku, ale je jich k dispozici více, aplikace také zobrazí varování.

6.3.3 Používání vlastností objektů

Další funkcí jazyka, je přistupování k hodnotám parametrů už zadaných objektů pomocí tečky. V příkladu 19 je zadán bod `point0` na souřadnicích [25,25], Druhý bod byl zadefinován tak, že jeho souřadnice x bude vždy stejná, jako souřadnice x prvního bodu, jeho souřadnice y bude vždy o 10 mm zmenšená, takže bod `point1` se bude nacházet nad bodem `point0`. Změnou souřadnice prvního bodu, se změní i souřadnice druhého bodu.

```
point0: Point(x = 25, y = 25);
point1: Point(x = point0.x, y = point0.y - 10);
```

Zdrojový kód 19: Přístup k souřadnicím bodu

Hodnoty parametrů jsou přístupné i u ostatních objektů, například u úsečky, se dá přistoupit i k její délce, což se hodí k měření vzdáleností v nakresleném střihu. I když existují výjimky, například oblouk nemusí vždy mít vytvořený středový bod.

6.3.4 Komentáře

Komentáře slouží k přidávání textových popisků přímo do kódu. Jednořádkový komentář se zapíše pomocí `//` a komentářový blok se zapíše pomocí `/*` pro začátek a `*/` pro ukončení. Příklad použití komentářů je zobrazen v kódu 20.

```
1 //Komentar na jeden radek
2 /*
3  Komentarovy blok
4  na vice radku
5 */
```

Zdrojový kód 20: Komentáře

6.4 Styl čar

Příkaz `Brush` slouží k přepínání aktuálního stylu kreslení, mění typ a barvu čar kreslených objektů. Má tvar: `Brush <styl> <barva>;`. Za `<styl>` je možno dosadit jedno z následujících slov

- solid – plná čára,
- dashed – přerušovaná čára,
- dotted – tečkovaná čára,
- dashdotted – střídavě přerušovaná a tečkovaná čára,
- noprint – speciální typ čáry, která je vidět pouze při kreslení a nezobrazí se ve výsledném obrázku PDF, slouží hlavně pro kreslení pomocných objektů, které ve výsledném střihu nebudou zobrazeny.

Za `<color>` je možno dosadit jednu z následujících barev: black, white, red, green, blue, yellow, purple, orange, pink, brown, gray, cyan, magenta, lime, silver, teal, maroon, navy, nebo olive. Pro jinou barvu se dá použít hexadecimální zápis barvy, kdy barva začíná znakem # a následuje šest hexadecimálních číslic.

```
1 ...
2 Brush solid blue;
3 ...
4 Brush dashed #00FF00;
5 ...
```

Zdrojový kód 21: Přepínání stylu kreslení

V kódu 21 je vidět, že barvu je možné přepínat. Jakmile je barva nastavená pomocí příkazu Brush, budou všechny následující příkazy používat nový styl a barvu čar, dokud se znovu nepřepne. Pokud příkaz Brush není použit, budou objekty vykresleny předvoleným štětcem, který odpovídá příkazu `Brush solid black;`.

7 Zhodnocení a srovnání s existujícím software

Tato kapitola se zabývá ukázkami střihů vytvořených s pomocí aplikace. Následuje srovnání s existujícími řešeními a zhodnocení aplikace.

7.1 Ukázkové střihy

Postup konstrukce střihů jsem čerpala z knihy Střihněte si na šaty. Nejprve představím velice jednoduchý střih na kruhovou sukni a potom složitější střih halenky.

Pomocí software jsem vytvořila střih na kruhovou sukni. Na začátku jsem si zapsala rozměry `op` pro obvod pasu a `ds` pro délku sukně. Pro změnu délky sukně stačí přepsat hodnotu `ds` a nechat kód znovu vyhodnotit. Stejně jednoduše lze změnit obvod pasu.

```
1  //Míry
2  var op = 800;
3  var ds = 500;
4  //Výpočty
5  var r_vnitrni = op / (3.14 * 2);
6  var r_vnejsi = ds + r_vnitrni;
7  //Střih
8  stred : Point(0, 0);
9  kruh_vnejsi : Arc(stred, r_vnejsi, 270, 90);
10 kruh_vnitrni : Arc(stred, r_vnitrni, 270, 90);
11 line0: Line(start = kruh_vnejsi.start,end = kruh_vnitrni.start);
12 line1: Line(start = kruh_vnitrni.end,end = kruh_vnejsi.end);
```

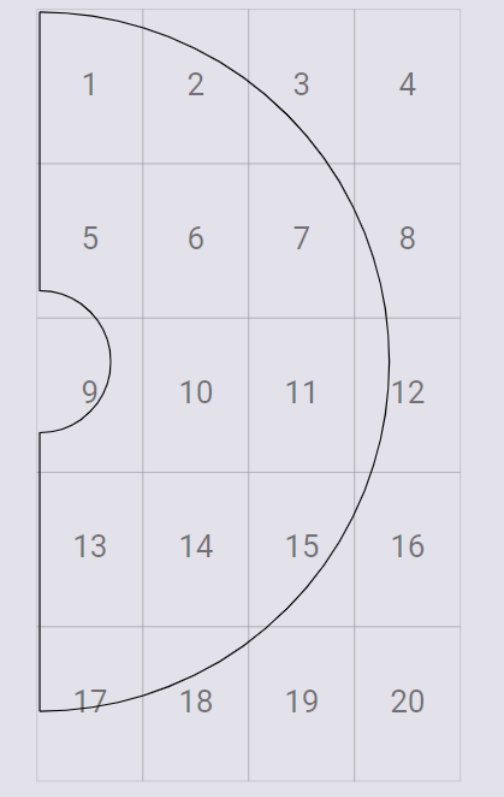
Zdrojový kód 22: Kruhová sukně

Na obrázku 13 je výsledek interpretace kódu 22 v náhledu před vytvořením PDF souboru pro tisk. Ukazuje, jak bude střih rozložený na jednotlivých stránkách.

Následuje složitější příklad, je to střih na halenku. Výsledek můžete vidět na obrázku 14. Kód v tomto případě je už poměrně dlouhý, a proto jej přikládám, jako součást elektronických dat práce. Návod ke konstrukci jsem opět čerpala z knihy Střihněte si na šaty. Na střih halenky je potřeba naměřit více rozměrů, jako jsou například obvod pasu, obvod hrudi, obvod krku, hloubka sedu a další. Do ukázkového střihu jsem nezahrnula rukávy. S použitím závislostí v kódu se střih dobře přizpůsobuje změnám zadaných rozměrů. Problémová část v tomto příkladu je průramek, který často při přepočítání vyžaduje úpravu. I přesto je to mnohonásobně rychlejší, než kreslit celý střih znovu.

7.2 Zhodnocení

Myslím, že se mi podařilo vytvořit aplikaci, která zajímavým způsobem přistupuje ke konstrukci střihu pomocí vlastního jazyka. S pomocí aplikace uživatel



Obrázek 13: Kruhová sukně

dokáže kreslit objekty používané ke konstrukci šicího střihu, zároveň aplikace poskytuje uživateli nástroje k tomu, aby střihy mohly být znovupoužitelné, ale není to podmínkou.

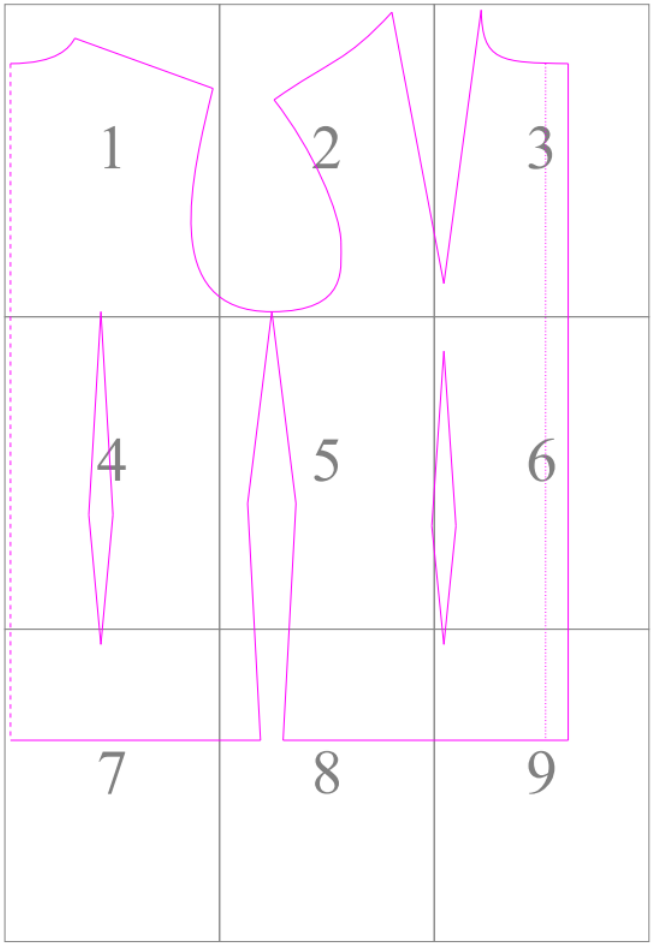
Také si myslím, že aplikace na první pohled působí jednoduše a nezastrašuje nového uživatele množstvím různých tlačítek. Z nakresleného obrázku je možné získat střih prakticky ihned.

Vytvořený jazyk by mohl na první pohled působit složitě, ale snaží se být konzistentní a princip vytváření objektů je stále stejný. Navíc některé operace se dají dělat přes obrazový editor. Zapisování výpočtu v kódu má výhodu, že je ho možné snadno upravit, bez nutnosti otevíraní různých oken a hledání proměnných v tabulkách.

7.3 Srovnání

Aplikace vytvořená v rámci této práce se z vybraných existujících řešení v sekci 1.1, dle mého názoru, v přístupu ke konstrukci střihu nejvíce podobá aplikaci Seamly2D, proto budu srovnávat s touto aplikací.

Obě aplikace používají podobný přístup ke kreslení střihu pomocí bodů a pomocí různých pravidel, která udávají polohu dalších bodů. Stejně tak obě aplikace umožňují používat vzorce. A obě aplikace umí vygenerovat PDF z nakresleného střihu, kde se střih rozdělí na strany. V obou aplikacích jsem vytvořila střih na



Obrázek 14: Halenka

halenku, ačkoliv v Seamly2D se mi nepovedl střih vytvořit přesně tak, jak jsem potřebovala, protože nemám dostatek zkušeností s tímto programem. Seamly2D má větší množství různých nástrojů pro kreslení a celkově více funkcionality, což může být pro nového uživatele matoucí.

Myslím si, že výhodou vytvořené aplikace je jednoduchost, uživatelské rozhraní neobsahuje mnoho prvků a je tak jednodušší správně používat nástroje, které jsou k dispozici. Jelikož se od nakresleného střihu přechází rovnou k vytvoření PDF pro tisk, není mezi těmito kroky žádný složitý proces, který by mohl uživatel snadno zkazit. Nevýhodou tohoto přístupu je, že aplikace nepodporuje například přidání přídavku na švy.

Aplikace se také liší tím, že používá vlastní jazyk, ze kterého se dá lépe poznat, jak probíhá postup konstrukce než při používání pouze grafického editoru. A také se tímto způsobem snadno upravují hodnoty proměnných.

Závěr

Výsledkem této práce je jednoduchá webová aplikace, která slouží ke kreslení šicích střihů, pomocí jednoduchého jazyka. Střih je možné do jisté míry upravovat i pomocí grafického editoru. Z vytvořeného střihu je možné vygenerovat soubor PDF připravený pro tisk střihu. Textová část obsahuje uživatelskou příručku k aplikaci a vytvořenému jazyku.

Conclusions

The result of this thesis is a simple web application which is designed to draw sewing patterns using a simple language. The pattern can be edited using the graphic editor to some extent. From the created sewing pattern it is possible to generate a PDF file designed for printing. This thesis contains user guide to the application and the created language.

A Obsah elektronických dat

`text/`
: Adresář s textem práce ve formátu PDF a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu.

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

`patternmaker/`
: Adresář s kompletními zdrojovými kódy programu, potřebné pro pro bezproblémové vytvoření spustitelných verzí programu.

`strihy/`
: Adresář s kódy pro zobrazení střihu halenky a kruhové sukně a vygenerovaná PDF.

Literatura

[1] *Seamly2D*. [online]. [cit. 2024-4-22]. Dostupný z: ⟨https://github.com/FashionFreedom/Seamly2D⟩.

[2] SALOMON, D. *Curves and Surfaces for Computer Graphics*. 2006.

[3] SOJKA, E.; NĚMEC, M.; FABIÁN, T. *Matematické základy počítačové grafiky*. 2012.

[4] *React*. [online]. [cit. 2024-4-21]. Dostupný z: ⟨https://react.dev/⟩.

[5] *Writing Markup with JSX*. [online]. [cit. 2024-4-21]. Dostupný z: ⟨https://react.dev/learn/writing-markup-with-jsx⟩.

[6] *About The ANTLR Parser Generator*. [online]. [cit. 2024-4-21]. Dostupný z: ⟨https://www.antlr.org/about.html⟩.

[7] *TypeScript*. [online]. [cit. 2024-4-21]. Dostupný z: ⟨https://www.typescriptlang.org/⟩.

[8] *Node.js*. [online]. [cit. 2024-4-22]. Dostupný z: ⟨https://nodejs.org/en⟩.

[9] VELEBOVÁ, L. *Střihněte si na šaty*. 2016. ISBN 978-80-271-0226-6.