Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Počítačová podpora výuky analytické geometrie

2017

Bc. Lenka Kalivodová

Vedoucí práce: Mgr. Tomáš Kühr, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Lenka Kalivodová
Název práce:	Počítačová podpora výuky analytické geometrie
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	37
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Lenka Kalivodová
Title:	Software for Teaching of Analytic Geometry
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	37
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Tato práce obsahuje uživatelskou a programátorskou příručku k aplikaci AnaGeom2D a zároveň popis použitých algoritmů. Aplikace demonstruje základní pojmy a algoritmy analytické geometrie a je určena pro výukové účely.

Synopsis

The thesis contains user and programming manual for AnaGeom2D application and also descripiton of used algorithms. The application demonstrates the basic terms and algorithms of analytic geometry and is intended for educational purposes.

Klíčová slova: analytická geometrie; výukový program; geometrické objekty

Keywords: Analytic geometry; Software for Teaching; Geometric objects

Děkuji Mgr. Tomáši Kührovi, Ph.D. za jeho konzultace, inspirativní doporučení a celkové vedení mé bakalářské práce.

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 zdrojových kódů

1 Úvod

Tato bakalářská práce je zaměřena na zpracování základních teoretických údajů z oblasti analytické geometrie a jejich následné využití v praktické části práce, kterou je samotná aplikace.

Práce je rozdělena do tří částí. První část je zaměřena na vysvětlení základních pojmů a algoritmů analytické geometrie. Druhou část tvoří seznámení s uživatelským rozhraním aplikace. Třetí část pak tvoří programátorská dokumentace, ve které jsou popsány hlavní třídy a metody.

1.1 Cíle práce

Hlavním cílem této práce je vytvoření softwarové aplikace umožňující demonstraci základních algoritmů analytické geometrie. Software bude podporovat zejména vizualizace objektů v rovině (body, úsečky, přímky, polopřímky, vektory, kružnice a elipsy) a jejich základní afinní transformace (posun, změna měřítka, rotace). Aplikace bude vytvořena takovým způsobem, aby se mohla stát nástrojem pro podporu výuky analytické geometrie.

1.2 Existující software s příbuznou tématikou

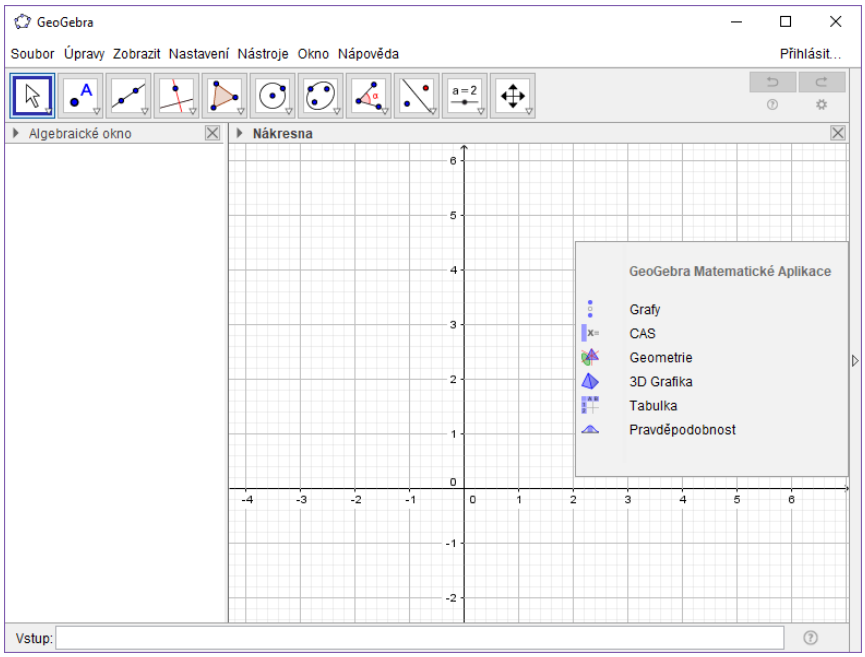
V současnosti existuje celá řada softwarových nástrojů určených pro výuku geometrie. Mezi nejznámější patří například:

1. *GeoGebra*[4] je multiplatformní open source software určený primárně pro studenty základních a středních škol, který nabízí řadu nástrojů pro konstrukci různých geometrických útvarů a výpočty matematických rovnic. Software dále mimo jiné umožňuje znázornění grafů, statistiku a infinitezimální počet.

   Vzhledem k tomu, že je *GeoGebra* vyvíjena více než deset let týmem zkušených vývojářů, obsahuje v porovnání s *AnaGeom2D* daleko více funkcionalit, např. je možná interakce s objekty na vykreslovací ploše pomocí myši. Samozřejmostí je také větší množství geometrických objektů a s nimi spojených výpočtů.

2. *Cabri Geometrie II Plus*[5] je softwarový nástroj pro výuku geometrie na základních a středních školách, který umožňuje vytváření rovinných geometrických útvarů. Nevýhodou programu je nutnost jeho zakoupení po 30 denní lhůtě pro vyzkoušení.

   Tento program nabízí podobné geometrické objekty jako *AnaGeom2D*, navíc však umožňuje konstrukci dalších mnohoúhelníků a následné operace s nimi. Kromě posunu a rotace objektu navíc obsahuje nástroj pro zobrazení osové a středové souměrnosti nebo stejnolehlosti.



Obrázek 1: Program GeoGebra

1.3 Volba konvencí

V celém textu budeme předpokládat, že se pohybujeme v rovině. Tedy i popisované geometrické objekty, jejich souřadnice a rovnice budou zadány v rovině. Dále v textu budeme předpokládat základní znalosti vektorových a afinních prostorů. Pokud nebude uvedeno jinak, budeme pod označením V chápat vektorový prostor nad množinou uspořádaných dvojic reálných čísel. Vektory z V budeme zapisovat do kulatých závorek a body do hranatých závorek.

2 Použité technologie

2.1 Java

Java[6] je vysokoúrovňový jazyk vyvíjený společností Oracle. Mezi jeho základní vlastnosti patří, že je objektově orientovaný a silně typovaný. Jazyk Java je široce rozšířený, zejména i díky jeho platformové nezávislosti. Aktuální verzí je Java 8 zveřejněná v roce 2014.

2.2 JavaFX

Platforma JavaFX je v současné době oficiálně doporučenou grafickou knihovnou pro aplikace vyvíjené na platformě Java 8.[7] JavaFX 8 obsahuje všechny možnosti předchozích verzí a podporu platforem Windows, Linux, a macOS. Platforma JavaFX 8 umožňuje vývoj aplikací dvěma základními způsoby:

1. Aplikace je možné vytvářet stejně jako klasické Java aplikace s uživatelským rozhraním ve Swing. Uživatelské rozhraní je zapsáno v jedné nebo více samostatných třídách v jazyce Java.

2. Druhým způsobem je využití FXML. FXML je jazyk založený na XML, který poskytuje strukturu pro vytváření uživatelského rozhraní odděleně od aplikační logiky kódu programu.[8] Dále aplikace musí obsahovat controllery k FXML dokumentům, což jsou třídy implementující rozhraní *Initializable* s jedinou metodu *initialize*.[9]

3 Úvod do analytické geometrie

Abychom se mohli blíže zabývat analytickou geometrií v rovině a základními geometrickými objekty, musíme nejprve definovat kartézskou soustavu souřadnic, kterou budeme pro jejich zobrazení používat.

Kartézskou soustavou souřadnic v rovině rozumíme dvojici číselných os x a y, pro které platí:

1. obě osy jsou navzájem kolmé,
2. jejich průsečíku odpovídá na obou stranách číslo 0.

Pokud máme zadanou soustavu souřadnic, můžeme libovolnému bodu X v rovině jednoznačně přiřadit uspořádanou dvojici čísel x_1, x_2. Čísla x_1, x_2 pak nazveme *souřadnice bodu* X v kartézské soustavě souřadnic.

3.1 Bod

Bod je nejzákladnější geometrický objekt, pomocí něhož můžeme definovat další geometrické útvary. Je to uspořádaná dvojice reálných čísel, která reprezentuje souřadnice daného bodu. Body budeme zapisovat velkým písmem a jeho souřadnice malým písmem v hranatých závorkách.

$$B[b_1; b_2]. \tag{1}$$

Bod nemá žádný rozměr a v souřadnicovém sytému bývá nejčastěji znázorňován vybarveným kolečkem či křížkem.

3.2 Vektor

Vektor je veličina, která má daný směr a velikost. Předpokládejme, že máme dva body $A[a_1; a_2], B[b_1; b_2]$. Pak vektor můžeme zapsat

$$\vec{u} = (u_1; u_2), \tag{2}$$

kde $u_1 = b_1 - a_1, u_2 = b_2 - a_2$.

Každý vektor je tedy jednoznačně určen dvěma body. Je-li vektor $\vec{u} = B - A$, pak nazýváme vektor $A - B$ *opačný vektor* k vektoru $\vec{u}$ a značíme ho $-\vec{u}$.

3.3 Přímka

Přímka je jednorozměrný geometrický objekt. Obvykle bývá zadávána dvěma body, protože pro každé dva libovolné body platí, že jimi lze vést právě jednu přímka. Nejkratší spojnici mezi takovými body nazýváme *úsečkou*. Dále existuje přímce podobná *polopřímka*, která ale na rozdíl od přímky má pevně daný počátek, ze kterého vede do nekonečna. V rovině je každá přímka popsána lineární rovnicí, která může být zadána v různých tvarech.

3.3.1 Obecná rovnice

Každá přímka v rovině lze vyjádřit obecnou rovnicí, můžeme ji úpravami odvodit z parametrické rovnice a obráceně.

Obecná rovnice přímky p má tvar

$$p : ax + by + c = 0, \tag{3}$$

kde jsou $a, b, c \in \mathbb{R}$ a alespoň jedna z konstant $a, b \neq 0$.

Pokud platí $a = 0$, přímka p je rovnoběžná s osou x, pokud platí $b = 0$, přímka p je rovnoběžná s osou y. Pokud platí $c = 0$, přímka prochází počátkem.

3.3.2 Parametrická rovnice

Pro přímku p je její parametrické vyjádření definováno vztahem $X = A + \vec{u}t$. V rovině tedy můžeme přímku zapsat rovnicemi

$$\begin{aligned} x &= x_0 + tu_1 \\ y &= y_0 + tu_2, \end{aligned} \tag{4}$$

kde $A = [x_0, y_0]$ je libovolný bod přímky, u_1, u_2 jsou konstanty určující směrový vektor $\vec{u} = (u_1, u_2)$ a t je parametr, pro který platí $t \in \mathbb{R}$.

3.4 Kružnice

Kružnice je množina všech bodů v rovině, které leží ve stejné vzdálenosti od pevně daného bodu. Tuto vzdálenost nazýváme *poloměr* a daný bod *střed* kružnice.

3.4.1 Středová rovnice

Pod pojmem středová rovnice kružnice rozumíme takovou rovnici, u které můžeme ihned určit střed a poloměr kružnice. Kružnice k se středem $S = [m, n]$ má středový tvar rovnice

$$(x - m)^2 + (y - n)^2 = r^2, \tag{5}$$

kde m, n jsou souřadnice středu kružnice a r je poloměr.

3.4.2 Obecná rovnice

Obecnou rovnici kružnice k získáme úpravou jejího středového tvaru a má tvar

$$x^2 + y^2 - 2mx - 2ny + p = 0, \tag{6}$$

kde $p = m^2 + n^2 - r^2$, m, n jsou souřadnice středu kružnice a r je poloměr. Platí, že ne každá rovnice v tomto tvaru je rovnicí kružnice.

3.4.3 Rovnice tečny

Tečnou kružnice k nazýváme přímku, která má s danou kružnicí právě jeden společný bod dotyku. Pokud existuje bod T, který leží na kružnici k, pak rovnice tečny v bodě T je

$$(x_0 - m)(x - m) + (y_0 - n)(y - n) = r^2, \tag{7}$$

kde x_0, y_0 jsou souřadnice bodu T, m, n jsou souřadnice středu kružnice a r je poloměr.

3.5 Elipsa

Elipsa je množina všech bodů v rovině, pro které platí, že mají stejný součet vzdáleností od dvou pevně zvolených bodů F_1, F_2, které nazýváme *ohniska*. Pro každý bod K elipsy e musí tedy platit

$$|XF_1| + |XF_2| = K, \tag{8}$$

kde K je konstantní číslo stejné pro všechny body elipsy.

Elipsu můžeme sestrojit zvolením středu elipsy $S = [m, n]$ a zadáním délky hlavní poloosy a a vedlejší poloosy b, pro které musí platit $a \neq b$. Pokud by platilo $a = b$, z elipsy by se stala kružnice.

3.5.1 Středová rovnice

Elipsu můžeme stejně jako kružnici zapsat středovou rovnicí. Pro elipsu se středem $S = [m, n]$, jejíž hlavní osa je rovnoběžná s osou x, má rovnice tvar

$$\frac{(x - m)^2}{a^2} + \frac{(y - n)^2}{b^2} = 1, \tag{9}$$

kde a je délka hlavní poloosy, b je délka vedlejší poloosy, m, n jsou souřadnice středu elipsy a x, y jsou souřadnice libovolného bodu elipsy.

Pokud by hlavní osa byla rovnoběžná s osou y, rovnice bude mít upravený tvar

$$\frac{(x - m)^2}{b^2} + \frac{(y - n)^2}{a^2} = 1, \tag{10}$$

kde b je délka hlavní poloosy, a je délka vedlejší poloosy, m, n jsou souřadnice elipsy a x, y jsou souřadnice libovolného bodu elipsy.

3.5.2 Obecná rovnice

Obecnou rovnici elipsy lze úpravami odvodit z jejího středového tvaru a obráceně. Obecná rovnice má tvar:

$$b^2x^2 + a^2y^2 - 2mb^2x - 2na^2y + b^2m^2 + a^2n^2 - a^2b^2 = 0, \tag{11}$$

kde a, b jsou délky hlavní a vedlejší poloosy a m, n jsou souřadnice středu elipsy.

Výše uvedený tvar rovnice se často používá s jinými koeficienty

$$px^2 + qy^2 + 2rx + 2sy + t = 0, \tag{12}$$

kde můžeme za koeficienty dosadit $p = a, q = b, r = mb^2, s = na^2$ a $t = b^2m^2 + a^2n^2 - a^2b^2$. Platí však, stejně jako u kružnice, že ne každá rovnice v tomto tvaru je rovnicí elipsy.

3.5.3 Parametrická rovnice

Parametrický popis elipsy se středem $S = [m, n]$ má tvar

$$\begin{aligned} x &= m + a\cos t \\ y &= n + b\sin t, \end{aligned} \tag{13}$$

kde a je délka hlavní poloosy, b je délka vedlejší poloosy a $t \in \langle 0, 2\pi \rangle$.

4 Použité algoritmy

V této kapitole stručně popíšu základní algoritmy analytické geometrie, které jsem použila při implementaci aplikace *AnaGeom2D*.

4.1 Operace s vektory

S vektory je možné v rovině provádět různé operace. Pomocí těchto operací můžeme určit například velikost vektoru, případně generovat vektory nové. Při definici níže uvedených pojmů a rovnic vycházím z textů [1, 2, 3].

4.1.1 Sčítání vektorů

Pro každé dva vektory v rovině $\vec{u} = (u_1; u_2), \vec{v} = (v_1; v_2)$ platí

$$\vec{u} + \vec{v} = (u_1 + v_1; u_2 + v_2). \tag{14}$$

Sčítání vektorů má podle textu [2] následující vlastnosti:

1. komutativita: pro každé dva vektory $\vec{u}, \vec{v}$ platí $\vec{u} + \vec{v} = \vec{v} + \vec{u}$,
2. asociativita: pro každé tři vektory u, v, w pak platí $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$,
3. neutrální prvek: platí $\vec{u} + \vec{0} = \vec{u}$, kde $\vec{0}$ nazýváme *nulový vektor*.

4.1.2 Násobení vektorů skalárem

Pokud násobíme nulový vektor reálným číslem k, výsledným vektorem je opět nulový vektor. Pokud vynásobíme nenulový vektor $\vec{u} = B - A$ číslem k, výsledným vektorem bude vektor $\vec{v} = C - A$, pro který platí

1. $|AC| = |k| \cdot |AB|$.
2. Je-li $k \geq 0$, leží bod C na polopřímce AB; je-li $k < 0$, leží bod C na polopřímce opačné k polopřímce AB.

4.1.3 Velikost vektoru

Výpočet velikosti vektoru je odvozen z výpočtu přepony trojúhelníku pomocí Pythagorovy věty. Velikost vektoru u značíme absolutní hodnotou $|u|$ a z textu [2] plyne, že spočítáme následovně

$$|u| = \sqrt{u_1^2 + u_2^2}. \tag{15}$$

Jestliže $|u| = 1$, nazývá se vektor $\vec{u}$ *jednotkový vektor*.

4.1.4 Skalární součin vektorů

Skalární součin je zobrazení, které dvojici vektorů přiřadí číslo, výsledkem tedy není vektor. Nejčastěji bývá definován na reálném vektorovém prostoru V jako zobrazení $V \times V \to \mathbb{R}$. Pokud existují dva vektory $\vec{u} = (u_1; u_2), \vec{v} = (v_1; v_2)$, pak jejich skalární součin je roven:

$$\vec{u} \cdot \vec{v} = (u_1 v_1; u_2 v_2). \tag{16}$$

Pokud existuje vektor $\vec{u} = (u_1; u_2)$ a druhý vektor, který je nulový, pak je jejich součin definován následovně:

$$\vec{u} \cdot \vec{v} = 0. \tag{17}$$

Z hlediska geometrického významu skalárního součinu platí:

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \, \|\mathbf{v}\| \cos \alpha, \tag{18}$$

kde α je úhel, který svírají vektory u, v. Pomocí tohoto vzorce můžeme zjistit velikost úhlu, který svírají dva vektory.

Z textů [1, 2] plyne, že pokud rovnici upravíme tak, abychom měli cosinus na levé straně rovnice, získáme vzorec pro odchylku dvou vektorů

$$\cos \alpha = \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}}{|\vec{\mathbf{u}}| \cdot |\vec{\mathbf{v}}|} = \frac{u_1 v_1 + u_2 v_2}{\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2}}. \tag{19}$$

4.1.5 Lineárně nezávislé vektory

Při zjišťování lineární závislosti vycházím z textu [1]. Abychom mohli definovat lineárně nezávislé vektory, musíme nejprve vysvětlit pojem lineární kombinace vektorů. Předpokládejme, že máme vektorový prostor V a vektory $\vec{u_1}, \vec{u_2}, \vec{u_3} \in V$. Dále máme číselné koeficienty vektorů $c_1, c_2, ..., c_n$. Pokud existuje vektor $\vec{v}$ takový, pro který platí:

$$\vec{v} = c_1 v_1 + c_2 v_2 + ... + c_n v_n, \tag{20}$$

pak je vektor $\vec{v}$ *lineární kombinací vektorů* $v_1, v_2, ..., v_n$ s koeficienty $c_1, c_2, ..., c_n$.

Lineárně nezávislý vektor pak je vektor, který nelze vyjádřit jako lineární kombinace ostatních vektorů z V. Při zjišťování lineární nezávislosti vektorů můžeme postupovat různými způsoby. První možností je z vektorů sestavit matici a upravit ji pomocí Gaussovy eliminační metody. Pokud po úpravě matice neobsahuje žádný nulový řádek, vektory jsou lineárně nezávislé.

$$\begin{bmatrix} \vec{v_{1x}} & \vec{v_{2x}} & \cdots & \vec{v_{nx}} \\ \vec{v_{1y}} & \vec{v_{2y}} & \cdots & \vec{v_{ny}} \end{bmatrix}. \tag{21}$$

Druhou možností je vynásobit vektory $v_1, v_2, ..., v_n$ koeficienty $c_1, c_2, ..., c_n$ a řešit soustavu rovnic s nulovým vektorem na pravé straně.

$$\begin{aligned} c_1 \vec{v_{1x}} + c_1 \vec{v_{1y}} &= 0 \\ c_2 \vec{v_{2x}} + c_2 \vec{v_{2y}} &= 0. \end{aligned} \tag{22}$$

Obecně však platí, že je tato metoda zdlouhavější a proto při větším počtu vektorů její užití není vhodné.

4.2 Vzájemná poloha a vzdálenosti geometrických objektů

Vzájemnou polohu dvou geometrických objektů je možné řešit pomocí rovnic jednotlivých objektů. Při definici níže uvedených pojmů a rovnic budu vycházet z textů [1, 2, 3].

4.2.1 Vzdálenost dvou bodů

Z textu [2] plyne, že vzdáleností $v(A, B)$ dvou bodů $A[a_1; a_2], B[b_1; b_2]$ v rovině rozumíme velikost vektoru $\vec{AB}$ a je dána vzorcem:

$$v(A, B) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}. \tag{23}$$

4.2.2 Vzdálenost bodu a přímky

Vzdálenost bodu $A[x; y]$ a přímky p zadanou obecnou rovnicí $p : ax + by + c = 0$ získáme dosazením do vzorce:

$$v(A, p) = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}. \tag{24}$$

Pokud po dosazení hodnot je $v(A, p) = 0$, bod A leží na přímce.

4.2.3 Vzdálenost bodu a úsečky

Podle textu [2] vzdálenost bodu X a úsečky AB spočítáme v několika krocích:

1. Pomocí libovolného bodu úsečky AB a a jejího směrového vektoru sestrojíme přímku p,
2. Pomocí bodu X a normálového vektoru přímky p sestrojíme přímku q, která bude na p kolmá,
3. Spočítáme průsečík I přímek p, q,
4. Pokud $I \in AB$, spočítáme vzdálenost mezi bodem X a přímkou p. Tím získáme vzdálenost $v(A, B)$.
5. Pokud $I \notin AB$, vzdálenost $v(X, AB)$ bude rovna menší ze vzdáleností $v(X, A), v(X, B)$.

4.2.4 Vzdálenost bodu a kružnice

Vzdálenost bodu $A[a_1; a_2]$ a kružnice k se středem $S[m; n]$ a poloměrem r spočítáme následovně:

1. Na body A, S aplikujeme algoritmus vzdálenosti dvou bodů,
2. Od výsledné vzdálenosti $v(A, S)$ odečteme poloměr kružnice k,
3. Výsledná hodnota je vzdálenost bodu A od kružnice k.

Dále platí, že pokud je vzdálenost bodů $v(A, S) > r$, bod A leží vně kružnice k. Pokud je $v(A, S) < r$, bod A leží uvnitř kružnice k. A pokud je $v(A, S) = r$, bod A leží na kružnici k.

4.2.5 Vzájemná poloha přímek

Máme dvě přímky p a q zadané parametrickými rovnicemi:

$$\begin{aligned} p &: X = A + t\vec{u} \\ q &: X = B + t\vec{v}, \end{aligned} \tag{25}$$

kde body $A = [a_1, a_2]$ a $B = [b_1, b_2]$ jsou počátečními body přímek p a q a vektory $\vec{u} = (u_1, u_2)$ a $\vec{v} = (v_1, v_2)$ jsou směrovými vektory přímek p a q.

V rovině mohou nastat tři vzájemné polohy přímek:

- $p \cap q = I$: Směrové vektory přímek nejsou lineárně závislé a existuje bod I, který náleží oběma přímkám. Takové přímky jsou *různoběžné*.
- $p \cap q = \emptyset$: Směrové vektory přímek jsou lineárně závislé a neexistuje žádný bod, který by měly přímky společný, přímky jsou *rovnoběžné*,
- $p = q$: Každý bod přímky p náleží i přímce q, pak jsou přímky *totožné*.

4.2.6 Vzdálenost přímky a kružnice

Vzdálenost přímky p a kružnice k se středem $S[m; n]$ a poloměrem r spočítáme následovně:

1. Na střed kružnice S a přímku p aplikujeme algoritmus pro vzdálenost bodu od přímky,
2. Od výsledné vzdálenosti $v(S, p)$ odečteme poloměr kružnice k,
3. Výsledná hodnota je vzdálenost přímky p od kružnice k.

Z výše uvedeného vyplývá, že pokud je vzdálenost $v(S, p) > r$, přímka p a kružnice k nemají žádný společný bod. Pokud je $v(S, p) < r$, přímka p je tečna kružnice k. A pokud je $v(S, p) = r$, přímka p je sečna kružnice k.

4.2.7 Vzájemná poloha přímky a elipsy

Máme přímku p a elipsu e zadané obecnou rovnicí.

1. Z rovnice přímky p vyjádříme jednu libovolnou neznámou a dosadíme ji do rovnice elipsy e,
2. Novou rovnici upravíme tak, abychom dostali rovnici kvadratickou,
3. Určíme diskriminant D kvadratické rovnice.

Podle hodnoty diskriminantu mohou nastat tři vzájemné polohy přímky a elipsy:

- $D > 0$: Přímka p s elipsou e mají dva společné body,
- $D = 1$: Přímka p s elipsou e mají právě jeden společný bod,
- $D < 0$: Přímka p s elipsou e nemají žádný společný bod.

4.3 Odchylka dvou geometrických objektů

V rámci této práce se budeme bavit primárně o odchylce vektorů, jelikož chceme-li spočítat odchylku přímek, počítáme vlastně odchylku jejich směrových vektorů.

4.3.1 Odchylka vektorů

Pro výpočet odchylky vektorů u, v využijeme vlastnosti skalárního součinu podle vzorce 19.

4.3.2 Odchylka přímek

Pro výpočet odchylky přímek p, q se směrovými vektory u, v využijeme vlastnosti skalárního součinu podle vzorce 19.

4.4 Transformace geometrických objektů

Pro popis transformací se v počítačové grafice často využívají matice, které se nazývají *transformační matice*. Složitejších transformací pak lze snadno dosáhnout skládáním jednotlivých transformací. Při popisu níže uvedených transformací jsem čerpala z textů [1, 2].

4.4.1 Translace

Translací v rovině rozumíme posun bodu $P = [x, y]$ o vektor $\vec{v} = (v_x, v_y)$, kde v_x je posun ve směru osy x a v_y je posun ve směru osy y. Souřadnice nového bodu získáme podle [2] sečtením původního bodu P s vektorem posunu v

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}. \tag{26}$$

4.4.2 Rotace

Pro rotaci bodu $P = [x, y]$ o úhel α kolem počátku souřadnicového systému vynásobíme bod P maticí rotace

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}. \tag{27}$$

Pokud bychom chtěli rotovat bod kolem jiného bodu než počátku, museli bychom použít upravenou matici rotace a souřadnice jednotlivých bodů rozšířit na $[x, y, 1]$. Mějme bod $R[r_x, r_y, 1]$, kolem kterého chceme rotovat. Matici rotace získáme složením matic následovně

$$M_{rot} = \begin{bmatrix} 1 & 0 & r_x \\ 0 & 1 & r_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -r_x \\ 0 & 1 & -r_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{28}$$

Výše použité matice jsou:

1. matice posunu ve směru bodu R,
2. matice rotace o daný úhel,
3. matice posunu zpět do počátku.

Výsledná matice rotace je tedy ve tvaru:

$$M_{rot} = \begin{bmatrix} \cos\alpha & -\sin\alpha & -r_x \cdot \cos\alpha + r_y \cdot \sin\alpha + r_x \\ \sin\alpha & \cos\alpha & -r_x \cdot \sin\alpha - r_y \cdot \cos\alpha + r_y \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

Nové souřadnice bodu P' poté získáme vynásobením původního bodu P s maticí rotace M_{rot}

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M_{rot} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{30}$$

4.4.3 Změna měřítka

Změna měřítka ve směru osy x je dána koeficientem s_x a ve směru osy y koeficientem s_y. Pro změnu měřítka bodu $P = [x, y]$ o koeficienty s_x, s_y vynásobíme bod P danými koeficienty

$$\begin{aligned} x' &= x \cdot s_x \\ y' &= y \cdot s_y. \end{aligned} \tag{31}$$

Pozorováním zjistíme, že změna měřítka u objektu ovlivňuje současně polohu i velikost transformovaného objektu ve směru souřadnicových os.[12] Pro hodnotu koeficientu s platí:

- $|s| \in (0, 1)$ dojde ke zmenšení a přiblížení objektu k počátku souřadnic,
- $|s| > 1$, dojde ke zvětšení objektu,
- $|s| < 0$, dojde ke zvětšení nebo zmenšení objektu v opačném směru.

5 Uživatelská dokumentace

Tato kapitola obsahuje základní uživatelskou dokumentaci pro aplikaci *AnaGeom2D*. Popisuji zde jednotlivé části uživatelského rozhraní a základní funkčnosti. Vzhledem k tomu, že aplikace je spustitelná jak na platformě Windows, tak macOS, je nutné v níže uvedených klávesových zkratkách využívat na platformě macOS místo klávesy *Ctrl* klávesu *Cmd*.

5.1 Spuštění programu

Aplikace *AnaGeom2D* je psaná v jazyce Java, je proto možné ji spustit na jakékoli platformě, na které je nainstalovaná Java. Aplikaci spustíme souborem *AnaGeom2D.jar* z adresáře, který se nachází na přiloženém DVD. Poté se otevře prázdná plocha s osami x a y, která slouží jako základní rovina pro tvorbu geometrických objektů.

Pro zajištění korektního chování programu je nutné ponechat spouštěcí soubor v archivu společně se složkou *lib*, který obsahuje veškeré importované knihovny. Bez těchto knihoven program neposkytuje veškerou funkcionalitu.

5.2 Hlavní okno programu

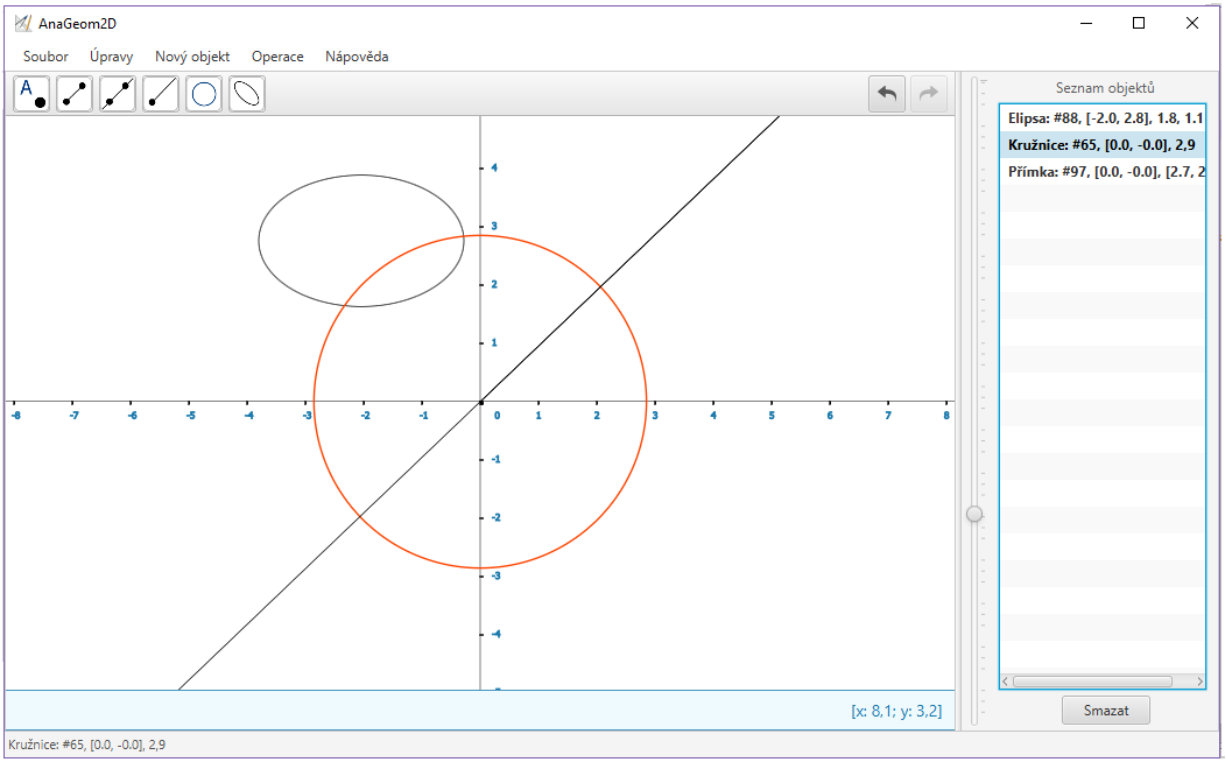
Hlavní okno programu je rozděleno do dvou částí. Ve větší části okna nalevo se nachází 2D plátno, do kterého se vykreslují jednotlivé objekty. V pravé části okna se nachází posuvník pro přibližování a oddalování plochy a vedle něj seznam již vytvořených geometrických objektů.

V horní části okna se nachází ovládací menu, které obsahuje položky:

1. *Soubor* - nabízí založení nového projektu, uložení a načtení plochy, export plochy do obrázku a ukončení programu,
2. *Úpravy* - umožňuje operace *Zpět*, *Vpřed* a vycentrování plochy,
3. *Nový objekt* - umožňuje vytvořit nové objekty - body, přímky, polopřímky, vektory a další geometrické objekty,
4. *Operace* - nabízí vybrané algoritmy,
5. *Nápověda* - nabízí uživatelskou nápovědu pro lepší orientaci v aplikaci.

Pod menu se nachází toolbar, který na levé straně obsahuje ovládací prvky pro vytváření objektů z plochy. Po kliknutí na tlačítko s ikonou příslušného objektu se aktivuje přidávání a v panelu výpočtů se zobrazí instrukce k postupu pro vytvoření daného objektu.

V dolní části programu se nachází stavový řádek, který poskytuje základní informace o právě provedených operacích. Tento řádek slouží pouze k lepší orientaci uživatele o aktuálním průběhu programu. Nad stavovým řádkem se nachází panel s výpočty, ve kterém se vypisují výsledky zadaných operací. Na rozdíl od stavového řádku neinformuje o aktuálních stavech objektů, ale je modifikován uživatelem na základě zadaného algoritmu. V



Obrázek 2: Hlavní okno programu

pravé části panelu s výpočty se dynamicky zobrazují aktuální souřadnice polohy kurzoru na ploše.

5.3 Manipulace s oknem

Vzhledem k použité technologii je omezena změna velikosti okna. Z tohoto důvodu je deaktivována možnost maximalizace a zároveň je nastavena fixní minimální velikost okna. Dále se mohou vyskytovat problémy na platformě Windows 10, kde přichytávání oken k okrajům obrazovky způsobuje nekonzistentní chování při vykreslování aplikace a je proto nezbytné se ho vyvarovat.

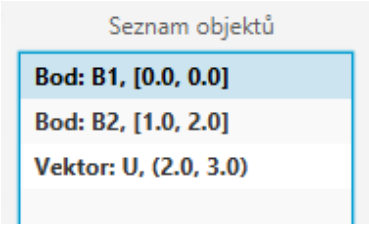
5.4 Vykreslovací plocha

Na této ploše se vykreslují objekty v závislosti na jejich souřadnicovém umístění, případně velikosti. Plochu lze přibližovat a oddalovat rotací středního tlačítka myši. Přiblížení či oddálení je možné realizovat i pomocí posuvníku, který se nachází na pravém okraji vykreslovací plochy. Plochu je možné posunovat tažením horizontálně i vertikálně při stisknutém levém tlačítku myši. Posunutou plochu lze jednoduše vrátit do původní pozice kliknutím

na položku *Úpravy/Vycentrovat* nebo stisknutím klávesové zkratky *Ctrl+Space*.

5.5 Seznam objektů

Na pravé straně hlavního okna se nachází seznam již vytvořených objektů na ploše. Každý objekt v seznamu má uveden svůj název a souřadnice na ploše. Při kliknutí na objekt levým tlačítkem myši se daný objekt na ploše zvýrazní červeně. Dvojitým kliknutím na objekt se zobrazí dialogové okno s detaily objektu a jeho možnými transformacemi. Každý objekt může být po jeho označení smazán kliknutím na tlačítko *Smazat* nebo pomocí klávesy *Delete*. Pro smazání více objektů najednou stiskneme *Ctrl* a kliknutím levým tlačítkem myši postupně označujeme vybrané objekty. Další možnost smazání jednoho či více objektů poskytuje jedna z položek kontextového menu.



Obrázek 3: Seznam objektů

5.6 Kontextové menu

Kontextové menu se vyvolává nad jedním či více označenými objekty. Pro označení jednoho objektu v seznamu klikneme na jeho pozici, pro označení více objektů postupujeme jako při výše zmíněném mazání objektů, tedy stiskneme *Ctrl* a levým tlačítkem myši klikneme na další objekt, který chceme označit. Kliknutím na pravé tlačítko myši nad jedním či více označenými objekty se zobrazí kontextové menu, s jehož pomocí lze rychle provádět některé vybrané operace nad objekty. Mezi tyto operace patří:

- *Skrýt výběr* - skryje všechny označené objekty na ploše,
- *Zobrazit výběr* - zobrazí všechny označené objekty na ploše,
- *Smazat výběr* - smaže všechny označené objekty,
- *Přebarvit výběr* - zobrazí dialog pro editaci barvy, změnu aplikuje na všechny označené objekty,
- *Určit polohu výběru* - do panelu výpočtů vypíše polohu a případné průniky dvou označených objektů.

Dále v případě označení dvou bodů v seznamu objektů se do kontextového menu přidají operace:

- *Vytvořit přímku* - vytvoří přímku pomocí dvou označených bodů,
- *Vytvořit polopřímku* - vytvoří polopřímku pomocí dvou označených bodů,
- *Vytvořit úsečku* - vytvoří úsečku pomocí dvou označených bodů.

Pokud není možné nad vybranými objekty provést nějakou operaci z kontextového menu, operace je označena jako neaktivní.

5.7 Přidání objektu

Přidání nového objektu na plochu provedeme kliknutím na položku v menu *Přidat objekt* a vybráním objektu z nabídky. Po zvolení objektu se otevře dialogové okno, které nás informuje o rovnici objektu. Zadávané parametry se liší v závislosti na konkrétním objektu, u každého objektu je pak navíc vyžadováno zadání názvu a barvy. Délka vstupu je omezena na 3 číslice, v případě zadávání desetinných čísel je možné zadávat s přesností na dvě desetinná místa. Místo desetinné čárky se zadává desetinná tečka. Další možností přidání objektu je využití již zmíněných ovládacích prvků nacházejících se v toolbaru.

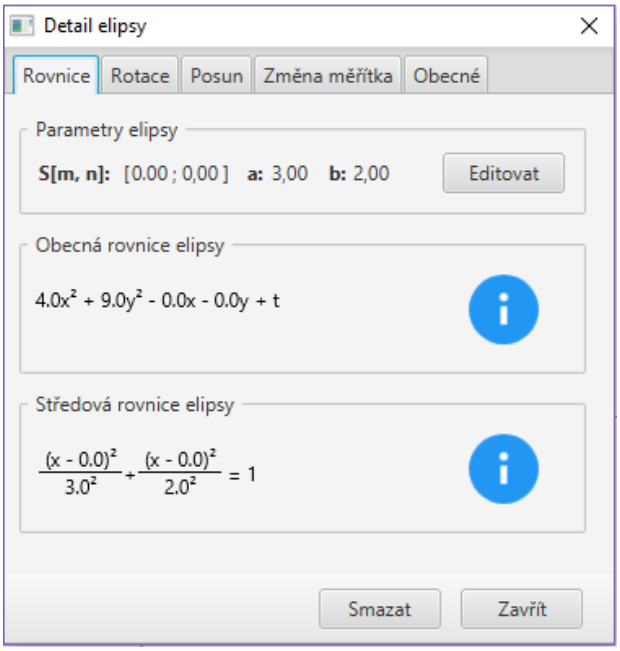
5.8 Editace objektů

Pro editaci a prohlížení vlastností objektů slouží dialog, který se zobrazí po dvojitém kliknutí na daný objekt v seznamu objektů. Tento dialog umožňuje na několika stránkách editovat jednotlivé vlastnosti objektu. Každý objekt poskytuje jiný dialog v závislosti na jeho konkrétních vlastnostech. Tento dialog může obsahovat například:

1. základní parametry objektu nebo rovnice objektu,
2. informace o elementárních transformacích objektu (rotace, posun, změna měřítka),
3. jméno a barvu objektu.

Zobrazované vlastnosti je možné editovat kliknutím na příslušné tlačítko. Vždy když dojde k editaci objektu, která změní jeho souřadnice, objekt změní pozici i na vykreslovací ploše.

Stránky dialogového okna upravující transformace objektu umožňují na objekt aplikovat transformace jako například posun, rotaci nebo změnu měřítka. U každé transformace je zobrazena transformační matice, do které se dynamicky doplňují zadávané hodnoty. Některé transformace se mohou lišit v závislosti na daném objektu.



Obrázek 4: Detail objektu

5.9 Uložení a načtení

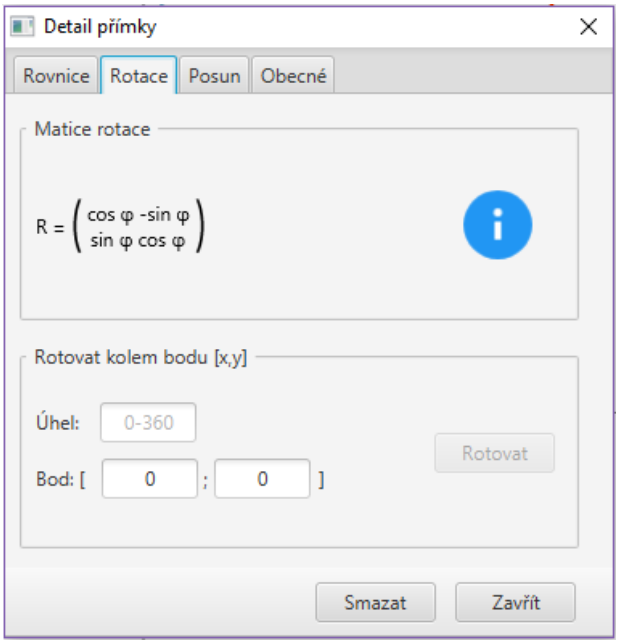
Plochu můžeme uložit kliknutím na příslušnou položku v menu, případně použitím klávesové zkratky *Ctrl+Shift+S*. Soubor můžeme načíst z menu nebo pomocí klávesové zkratky *Ctrl+O*. Po této volbě se otevře dialogové okno pro uložení či načtení, ve kterém zvolíme název ukládaného či načítaného souboru a jeho umístění.

5.10 Nová plocha

Novou kreslící plochu založíme z menu kliknutím na položku *Soubor/Nový* nebo klávesovou zkratkou *Ctrl+N*. Pokud plocha neobsahuje žádné vytvořené objekty, načte se nová plocha. V opačném případě program nabídne uložení aktuálního seznamu objektů do souboru.

5.11 Export do obrázku

Program umožňuje export plochy do souboru typu *.png*. Kliknutím na položku *Soubor/Export do obrázku* nebo stiskem klávesové zkratky *Ctrl+E* se zobrazí dialog pro uložení souboru, ve kterém je možné vybrat umístění a název ukládaného souboru a po potvrzení se provede export plochy s objekty do obrázku.



Obrázek 5: Transformace objektu

5.12 Uživatelská nápověda

Uživatelskou nápovědu otevřeme kliknutím na položku v menu *Nápověda/Zobrazit nápovědu* nebo stisknutím klávesové zkratky *Ctrl+H*.

5.13 Ukončení programu

Program ukončíme standardním způsobem, tedy kliknutím na křížek v pravém horním rohu okna nebo kliknutím na položku *Soubor/Zavřít*. Alternativou tohoto způsobu je klávesová zkratka *Ctrl+Q*. U všech těchto možností program nabídne možnost uložení aktuální plochy. Poslední možností ukončení programu je vynucené ukončení pomocí klávesové zkratky *Alt+F4*.

6 Programátorská dokumentace

Každé okno programu je po technické stránce složeno ze dvou hlavních částí. První je FXML soubor, který obsahuje samotný vzhled každého okna a jeho strukturu. Druhou částí je pak *controller* (dále jen kontroler), který se stará o inicializaci, interakci mezi okny a veškeré operace týkající se layoutu, který obsluhuje. Dále umožňuje měnit vlastnosti komponent a reagovat na jejich události.

Výsledný program je pak rozdělen do několika logických celků. V následujících podkapitolách uvedu stručný popis jednotlivých částí programu.

Vzhledem k tomu, že jsem při výběru technologií nenalezla vhodnější prvek pro vykreslování objektů, rozhodla jsem se využít plátno *canvas*, do kterého lze kreslit sadou grafických příkazů, které poskytuje třída *GraphicsContext*. Pro rozšíření použitelnosti jsem implementovala funkce, které budou popsány níže v textu. Jedná se zejména o funkce týkající se vykreslení a přepočítání souřadnicového systému.

6.1 Hlavní třídy

6.1.1 AnaGeom

AnaGeom je hlavní třída, která obsahuje metodu *start()* pro spuštění aplikace. Pomocí *FXMLLoaderu* načte FXML soubor a na základě jeho struktury vytvoří hlavní okno programu. Tato třída má přiřazený kontroler *LayoutController*, který má na starosti hlavní okno programu.

6.1.2 LayoutController

LayoutController obsahuje inicializaci hlavního okna a veškeré výpočty týkající se hlavního okna. Tato třída obsahuje sloty:

- `gc` – představuje grafický kontext, který po zavolání jeho getteru na *canvas* v inicializaci umožňuje používat její metody pro vykreslování objektů na plátno,
- `objectsList` – kolekce objektů *List<Object>*, obsahuje všechny aktuálně vytvořené objekty včetně všech jejich parametrů,
- `stringsList` – kolekce textových řetězců *ObservableList<String>*, obsahuje textovou podobu všech vytvořených objektů. Obsah *stringsList* se vypisuje v *lv*.
- `lv` – komponenta *ListView* pro prohlížení seznamu objektů,
- `statusLabel` – textový řetězec pro informaci o stavu ve stavovém řádku,
- `calculationsLabel` – textový řetězec pro zapsání výsledku výpočtu v panelu výpočtů,
- `undoStack` – zásobník typu *List<Object>*, obsahuje historii objektů dozadu,

- `undoStackStrings` – zásobník typu *ObservableList<String>*, obsahuje textovou podobu historie objektů,
- `redoStack` – zásobník typu *List<Object>*, obsahuje historii objektů dopředu,
- `redoStackStrings` – zásobník typu *ObservableList<String>*, obsahuje textovou podobu historie objektů.

Dále kromě getterů a setterů třída obsahuje výpočetní funkce. Mezi nejdůležitější patří přepočet souřadnicového systému a další funkce volané při inicializaci okna:

- `toX, toY` – Vzhledem k tomu, že *canvas* má počátek souřadnicového systému v levém horním rohu, ale my chceme pracovat s klasickým souřadnicovým systémem se středem uprostřed, kontroler obsahuje funkce *toX()*, *toY()*, které přepočítávají souřadnice standardního souřadnicového systému do souřadnic *canvasu*.
- `toXrev, toYrev` – Tyto funkce jsou reverzní k funkcím *toX()*, *toY()*.
- `drawCoords` – Vykresluje souřadnicové osy x a y,
- `redraw` – Funkce *redraw* překreslí *canvas* se všemi vytvořenými objekty. Tato funkce je volaná primárně při vytváření nových objektů a jejich editaci.
- `addCanvasScrolling` – Funkce vypočítá a nastaví *canvasu* scrollování.
- `addCanvasMoving` – Funkce vypočítá a nastaví *canvasu* možnost pohybu tažením.
- `deleteObject` – Smaže objekt ze seznamů *objectsList* a *stringsList*.

Každý nově vytvořený objekt je uložen v seznamu objektů *objectsList* a jeho textový řetězec do seznamu stringů *stringsList*. Při smazání objektu je vždy potřeba objekt odstranit z obou seznamů, stejně tak při každé změně objektu je nezbytná jeho editace v obou seznamech.

6.1.3 ObjectsController

ObjectsController obsahuje inicializace všech oken pro vytváření nových objektů. K hlavním FXML souborům, které má na starosti, patří:

- *Bod* – okno vytvářející objekt bod,
- *Usecka* – okno vytvářející objekt úsečku,
- *Poloprimka* – okno vytvářející objekt polopřímku,
- *Primka* – okno vytvářející objekt přímku,
- *Vektor* – okno vytvářející objekt vektor,

- *Kruznice* – okno vytvářející objekt kružnici,
- *Elipsa* – okno vytvářející objekt elipsu,
- *Ctverec* – okno vytvářející objekt čtverec,
- *Trojuhelnik* – okno vytvářející objekt trojúhelník.

Aby se otevřelo okno pro vytvoření daného objektu, nejdřív se musí načíst *FXMLLoader*, ze kterého je následně získán kontroler příslušného okna.

Zde je ukázka funkce, která načte okno vybraného FXML souboru.

```
1  private Stage createObjectModalWindow(String FXMLpath) throws IOException {
2        FXMLLoader loader = new FXMLLoader(getClass().getResource(FXMLpath));
3        AnchorPane ap = (AnchorPane) loader.load();
4        ObjectsController controller = loader.getController();
5        controller.setMainWindow(this);
6        controller.setSelected(lv.getSelectionModel().getSelectedItems());
7        Stage stage = new Stage();
8        stage.initStyle(StageStyle.UTILITY);
9        stage.initModality(Modality.WINDOW_MODAL);
10       stage.initOwner(img.getScene().getWindow());
11       Scene scene = new Scene(ap);
12       stage.resizableProperty().setValue(Boolean.FALSE);
13       stage.setScene(scene);
14
15       stage.setOnHiding(e -> {
16          ((Stage) (menu.getScene().getWindow())).toFront();
17       });
18       return stage;
19    }
```

Zdrojový kód 1: Vytvoření modálního okna

Po otevření okna se doplní parametry daného objektu a stiskne se tlačítko *OK*. Při stisknutí tlačítka se v dialogu vytváření nového objektu nejprve zavolá funkce *checkName*, která zkontroluje, jestli je zadaný název objektu unikátní, tedy jestli již není název použit některým z již existujících objektů. Poté se vyparsují hodnoty z textových polí a nastaví se jako parametry objektu. Výsledný objekt se uloží do seznamu objektů *objectsList* a jeho textový řetězec do seznamu stringů *stringsList*, který je v inicializaci *LayoutControlleru* nastaven jako obsah *ListView*.

6.1.4 DetailsController

DetailsController obsluhuje okna pro detaily stávajících objektů. Každý vytvořený objekt má své originální dialogové okno detailu. Tento kontroler v inicializaci nastavuje všechny

stránky dialogu včetně použitých rovnic a matic. Rovnice jsou tvořeny pomocí funkce *createEquation* a matice pomocí funkcí *createMatrix* a *createRotationMatrix*.

V kontroleru jsou funkce všech transformací objektů, které jsou vykonány při kliknutí na příslušné tlačítko. Všechny funkce mají podobnou strukturu, kterou pro větší přehlednost stručně popíšu u funkce pro rotaci bodu *rotateAndTranslatePoint*.

```
1  @FXML
2     private void rotateAndTranslatePoint(ActionEvent event) {
3        try {
4           Point2D p = new Point2D(Double.parseDouble(field1.getText()),
               Double.parseDouble(field2.getText()));
5           double angle = Double.parseDouble(field3.getText());
6           o.getStart().rotaceBoduKolemBodu(p, angle);
7           LayoutController.redraw();
8           double px = o.getStart().getX();
9           double py = o.getStart().getY();
10          label1.setText(prepareOutput(px));
11          label2.setText(prepareOutput(py));
12          LayoutController.editPoint(id, px, py);
13          reInitPoint(px, py);
14          LayoutController.setVypisText("Bod " + o.getJmeno() + " orotován a
               posunut o " + angle + ". "
15                + "Nové souřadnice bodu: " + o.getStart().toString();
16       } catch (NumberFormatException e) {
17       }
18    }
```

Zdrojový kód 2: Funkce *rotateAndTranslatePoint*

Při stisknutí tlačítka se nejprve vyparsují hodnoty z textových polí pro bod a úhel, kolem kterého chceme rotovat. Tyto hodnoty se předají jako parametry metodě *rotaceBoduKolemBodu*, která je zavolána na daný bod. Nové hodnoty bodu jsou předány funkci *editPoint* z *LayoutControlleru*, která v seznamu objektů přenastaví původní hodnoty na nové. Hodnoty jsou dále předány ještě funkci *reInitPoint* z *DetailsControlleru*, která reinicializuje hodnoty objektu v rámci dialogového okna detailu objektu.

6.1.5 EditController

EditController obsluhuje okna pro editaci objektů. Tento kontroler je speciální v tom, že na rozdíl od ostatních oken vyvolávaných z *LayoutControlleru*, je vyvolán z *DetailsControlleru*, tedy dialogového okna detailu objektu.

Při stisknutí tlačítka *OK* v editaci objektu se vyparsují nové hodnoty z textových polí a uloží se jako aktuální hodnoty. Stejně jako u výše zmíněného příkladu transformace objektu jsou tyto hodnoty nyní předány funkci *editObject*[1] v *LayoutControlleru*, která v seznamu

[1] místo pojmu *Object* je ve funkci vždy použit název konkrétního objektu, např. *Point*, *Line*.

objektů přenastaví původní hodnoty na nové. Hodnoty jsou dále předány ještě funkci *reInitObject*[1] v *DetailsControlleru*, která reinicializuje hodnoty objektu v rámci dialogového okna detailu objektu.

6.1.6 CalculationsController

CalculationsController obsahuje inicializace oken pro operace mezi objekty. Objekty pro výpočty je možné vybrat ze seznamu již vytvořených objektů nebo vytvořit objekty nové.

Při potvrzení výpočtu tlačítkem *OK* program nejprve rozhodne, jestli se vytváří nový objekt či nové objekty nebo jestli budou pro výpočet použity již existující objekty. Poté se na objekty zavolá příslušná funkce pro výpočet a pokud se při výpočtu tvořil nový objekt, je standardně přidán do *objectsList* a jeho textový řetězec do *stringsList*.

6.1.7 Calculations

V této třídě jsou všechny výpočetní funkce pro operace mezi objekty.

6.1.8 AuxiliaryFunctions

Tato třída obsahuje pomocné funkce a funkce pro ošetření vstupu.

6.2 Geometrické objekty

V této kapitole stručně popíšu třídy, které mají na starosti tvorbu geometrických objektů.

6.2.1 Point2D

Třída *Point2D* reprezentuje bod, který se dále používá pro konstrukci všech dalších objektů. Třída obsahuje hlavní dva sloty:

- `x` – souřadnice na ose x,
- `y` – souřadnice na ose y,

Kromě getterů a setterů tato třída obsahuje i metody pro transformace bodu:

- `rotaceBodu` – metoda, která rotuje bod vzhledem k počátku,
- `rotaceBoduKolemBodu` – metoda, která rotuje bod vzhledem k zadanému bodu,
- `posunBodu` – metoda, která posune bod ve směru zadaného vektoru.

6.2.2 Objects

Jedná se o výčtový typ obsahující všechny geometrické objekty. V případě implementace nového objektu je nutné jeho typ doplnit do tohoto výčtového typu.

6.2.3 Object

Třída *Object* reprezentuje veškeré geometrické objekty. Každý objekt obsahuje sloty:

- `typ` – výčtový typ identifikující geometrický objekt,
- `jmeno` – textový řetězec pojmenovávající geometrický objekt,
- `start` – objekt *Point2D*, nese informaci o hlavním bodu objektu (např. počáteční bod úsečky, střed kružnice),
- `end` – objekt *Point2D*, nese informaci o druhém hlavním bodu objektu (např. koncový bod úsečky),
- `center` – objekt *Point2D*, nese informaci o třetím hlavním bodu objektu (např. trojúhelník),
- `a` – hodnota typu *double*, nese informaci o velikosti hrany nebo poloměru objektu,
- `b` – hodnota typu *double*, nese informaci o velikosti druhého poloměru objektu,
- `clr` – barva geometrického objektu poskytovaná třídou *Color*.

Sloty objektů nejsou vždy inicializované všechny, inicializace jednotlivých slotů závisí vždy na konkrétním objektu.
Třída *Objekt* obsahuje všechny základní konstruktory objektů. Objekty *Bod*, *Elipsa* a *Trojúhelník* mají svůj jedinečný konstruktor.

```
//konstruktor bodu
Object(Objects typ, String jmeno, Point2D p, Color clr)

//konstruktor elipsy
Object(Objects typ, String jmeno, Point2D p, double a, double b,
   Color clr)

//konstruktor trojuhelniku
Object(Objects typ, String jmeno, Point2D p1, Point2D p2, Point2D p3,
   Color clr)
```

Některé další objekty jako *Přímka*, *Polopřímka*, *Úsečka* a *Vektor* používají stejný konstruktor a podle typu objektu konstruktor inicializuje dané sloty.

```
//kostruktor primky, poloprimky, usecky, vektoru
Object(Objects typ, String jmeno, Point2D p1, Point2D p2, Color clr)
```

Výjimku tvoří objekty *Kružnice* a *Čtverec*, které nerozlišují inicializaci slotů. *Kružnice* je zadaná středem a poloměrem a *Čtverec* levým dolním bodem a stranou.

```
//konstruktor kruznice, ctverce
Object(Objects typ, String jmeno, Point2D p, double a, Color clr)
```

Třída dále obsahuje kromě getterů a setterů metodu pro vykreslení objektů *draw*. Tato metoda vykreslí objekt podle jeho typu. Metoda požívá pro vykreslení třídu *GraphicsContext*, která poskytuje různé metody vykreslení objektů. V této metodě před vykreslením objektu převádíme souřadnice načtené z objektu do souřadnic *canvasu* pomocí výše zmíněných funkcí *toX()*, *toY()*.

6.3 Návrhy možných rozšíření

Vzhledem k tomu, že aplikace obsahuje implementaci základních rovinných obrazců, bylo by vhodné funkčnost dále rozšířit o další kuželosečky, případně objekty a operace ve 3D. Pro přidání kuželoseček by bylo vhodné využít funkcí *canvasu*, v tomto případě je však nutné implementovat potřebné algoritmy pro přepočet souřadnic a kreslení pomocí dostupných funkcí. Samozřejmostí by bylo také doplnění algoritmů pro operace s nově implementovanými objekty. Původním záměrem byla i implementace 3D scény a objektů, bohužel by zde bylo nutné využít zcela odlišný přístup než v rámci 2D scény.

Závěr

Výsledkem práce je aplikace *AnaGeom2D*. Tento program umožňuje zobrazovat vybrané geometrické objekty, kterými jsou v první řadě body, vektory, přímky, polopřímky, úsečky, elipsy, kružnice. Po vytvoření objektu je možné editovat jeho vlastnosti stejně jako transformace a sledovat změny v jeho vykreslení. Nástroj poskytuje algoritmy pro základní operace s vektory jako určení velikosti vektoru, sčítání vektorů, násobení vektoru skalárem, skalární součin vektorů nebo zjišťování lineární závislosti vektorů. Vybraným objektům je nástroj dále schopen určit jejich vzájemnou polohu, případně vzdálenost či průnik. Samozřejmostí je jednoduchá historie provedených akcí a možnost se v ní pohybovat.

Aplikace se všemi svými funkcionalitami by po dalším rozšíření mohla sloužit pro podporu výuky analytické geometrie a proto myslím, že mohu považovat cíl této práce za splněný.

A Pokyny pro instalaci a spuštění programu

A.1 Požadavky pro instalaci

1. Počítač s OS Windows/Linux/macOS
2. Nainstalováno JRE (Java Runtime Enviroment)

A.2 Pokyny pro spuštění

1. Vložíme DVD do mechaniky a přesuneme se do složky *app*.
2. Rozbalíme obsah *.zip* archivu na disk počítače.
3. Aplikaci spustíme pomocí *.jar* souboru.

B Obsah přiloženého DVD

Zde je uveden stručný popis obsahu přiloženého DVD.

`app/`
: V této složce se nachází *.zip* soubor obsahující spouštěcí soubor aplikace a složku s importovanými knihovnami.

`doc/`
: V této složce je uložen text této práce.

`src/`
: V této složce se nachází zdrojové kódy aplikace.

Literatura

[1] Krupka M., *Geometrie pro informatiky.* Univerzita Palackého v Olomouci, Přírodovědecká fakulta, 2008.

[2] Žára J., Beneš B., Sochor J., Felkel P., *Moderní počítačová grafika.* Computer Press, 2. vydání, 2005.

[3] Vektorový počet a analytická geometrie [online]. Copyright © [cit. 02.08.2017]. Dostupné z: http://homen.vsb.cz/ lud0016/M1_13/AG/Vektorovy_pocet_a_analyticka_geometrie.pdf

[4] About - GeoGebra. GeoGebra [online]. Copyright © 2017 [cit. 01.08.2017]. Dostupné z: https://www.geogebra.org/about

[5] Cabri. Pedagogická fakulta Jihočeské univerzity [online]. Dostupné z: https://www.pf.jcu.cz/cabri/cabri.htm

[6] The Java™ Tutorials [online]. Copyright © 1995, 2015. Oracle [cit. 01.08.2017]. Dostupné z: https://docs.oracle.com/javase/tutorial/

[7] Příručka JavaFX [online]. Copyright © [cit. 04.08.2017]. Dostupné z: https://java.vse.cz/wiki/uploads/4it115/priruckajavafx.pdf

[8] JavaFX Overview [online]. Copyright © 2008, 2014, Oracle [cit. 02.08.2017]. Dostupné z: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm

[9] Why Use FXML [online]. Copyright © 2011, 2014, Oracle [cit. 02.08.2017]. Dostupné z: https://docs.oracle.com/javase/8/javafx/fxml-tutorial/why_use_fxml.htm

[10] JavaFX Scene Builder User Guide [online]. Copyright © [cit. 04.08.2017]. Dostupné z: http://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm

[11] Working with the Canvas API [online]. Copyright © 2008, 2014, Oracle [cit. 04.08.2017]. Dostupné z: http://docs.oracle.com/javase/8/javafx/graphics-tutorial/canvas.htm

[12] Strachota P., *Geometrické transformace pomocí matic.* FJFI ČVUT v Praze, 2010.