VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

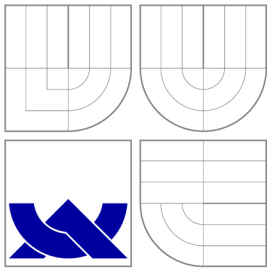
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PARAMETRICKÝ GEOMETRICKÝ NÁČRTNÍK

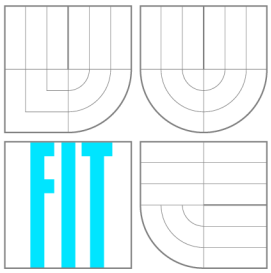
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN VALA
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PARAMETRICKÝ GEOMETRICKÝ NÁČRTNÍK

PARAMETRIC GEOMETRY SKETCH TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN VALA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PAVEL ZEMČÍK

BRNO 2012

Abstrakt

Cílem této Bakalářské práce je vytvoření návrhu a implementace parametrického geometrického náčrtníku. Práce rovněž obsahuje vymezení základních pojmů z oblasti dynamické geometrie. Shrnutí současného stavu, popis dnes existujících nástrojů následované vyhodnocením některých jejich vlastnosti. V závěru jsou shrnuty dosažené výsledky a je navrženo možné pokračování v práci.

Abstract

Purpose of this Bachelor's thesis is the design and implementation of Parametric geometry sketch tool. This thesis also includes basic terms of dynamic geometry. Summary of the state of the art, description of existing tools, followed by evaluation of their features. Results and possibilites of future expansion are summarized in the last chapter.

Klíčová slova

Parametrické nástroje, dynamická geometrie, 2D zobrazení, Euklidovská geometrie

Keywords

Parametric tools, dynamic geometry, 2D view, Euclidean geometry

Citace

Jan Vala: Parametrický geometrický náčrtník, bakalářská práce, Brno, FIT VUT v Brně, 2012

Parametrický geometrický náčrtník

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Pavla Zemčíka a že jsem uvedl všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Vala
14. května 2012

Poděkování

Rád bych poděkoval vedoucímu práce, Doc. Dr. Ing. Pavlu Zemčíkovi za rady, návrhy a připomínky, kterými mi pomohl k vypracování této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V současné době lze osobní počítače využívat k široké škále činností, jednou z nich je například počítačová grafika, zejména pak její využití při tvorbě geometrických konstrukcí. Pomocí počítačů lze realizovat výpočty a zobrazování jednotlivých prvků na obrazovce, to zpřístupňuje tyto kreslící aplikace prakticky komukoliv, protože uživatel je díky nim schopen vytvořit přesné nákresy, které může posléze například vytisknout. Parametrické kreslící nástroje ovšem pokračují ještě dále. Nejenže nabízejí výše zmíněné možnosti, ale poskytují i prostředky, kterými lze sledovat postup kterým jsme dosáhli určitého bodu při konstrukci. Dále pak umožňují sledovat, jaké dopady na zobrazenou konstrukci a na které její prvky bude mít změna některých parametrů. To znamená například posun některých bodů, změna vzdáleností rovnoběžek nebo poloměrů kružnic. Tato vlastnost je esenciální pro parametrické kreslící nástroje.

V první řadě je ovšem nutné vysvětlit, co se rozumí pod pojmem parametrický kreslící nástroj. Jedná se o aplikaci, která umožňuje přidávání jednoduchých grafických prvků do scény, jako jsou například body, přímky, úsečky, kružnice, atd. Mezi těmito prvky existují logické vazby, které odpovídají běžným geometrickým znalostem. Úsečka je například určena dvěma body, kružnice středem a poloměrem. Aplikace, která umožňuje vykreslování takovýchto objektů může dobře sloužit jakožto klasický kreslící nástroj. Pokud se však jedná o parametrický kreslící nástroj, jinak též prostředí s podporou dynamické geometrie, dochází k výraznému rozšíření tohoto chování.

Slovo parametrické, představuje možnost modifikace parametrů prvků ve scéně. Oproti obyčejnému kreslícímu nástroji se jedná o zcela zásadní změnu. Zatímco klasický kreslící nástroj po zakreslení objektu do scény tento objekt trvale zakreslil, parametrická nástroj umožňuje změnu parametrů, podle kterých jsou prvky vykreslovány. V případě bodu lze změnit jeho souřadnice, jejich změnou však dojde i ke změně dalších prvků, které jsou vázány na tento bod. Jako příklad lze uvést modifikace souřadnic jednoho z koncových bodů úsečky, při které dochází ke správnému překreslování ve scéně na základě nových souřadnic bodu.

V dnešní době existuje nepřeberné množství grafických kreslících nástrojů podporujících dynamic geometry. Jejich činnost spočívá v zásadě v tom, že geometrické útvary nelze pouze vytvářet, ale lze s nimi i dále manipulovat, což ovlivňuje výsledek jako celek. Například při přesunu bodu přímky dojde k posunu této přímky do nové pozice bodu. Počátky tohoto druhu software, jinak též *Dynamic geometry environment* (DGE) sahají do 80-tých let, dnes již však existuje široká škála aplikací zabývajících se touto problematikou. Ve zkratce se jedná o grafické aplikace, které umožňují manipulaci s vytvořenými grafickými primitivy (body, přímky, křivky), jejichž změnou dochází k transformacím dalších objektů na ně

vázaných. Díky tomu tedy uživatel může sledovat důsledky svojí činnosti.

Na první pohled je nejvíce zřetelným využitím těchto kreslících systémů školství. Poskytují totiž žákům snadno pochopitelný náhled do geometrie a díky svým technickým možnostem, které umožňují manipulaci s vytvořenými objekty, snadno přístupnou cestou rozšiřují znalosti a pochopení dané látky u žáků. Dalším teoretickým místem pro uplatnění těchto principů jsou CAD systémy, které by dokázaly těžit z vlastností poskytovanými DGE.

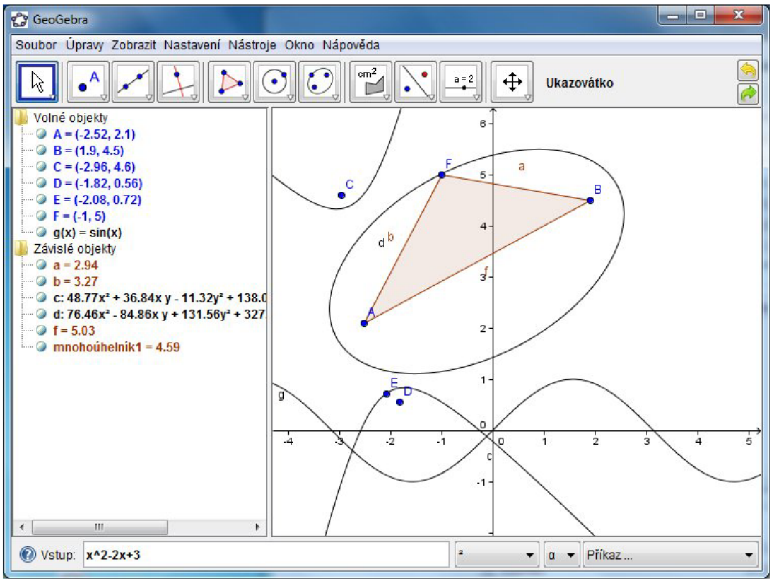
V kapitole *Základní pojmy z oblasti dynamické geometrie* jsou popsány principy euklidovské geometrie, včetně jejich limitů a využití v dnešní vědě. Jsou zde popsány její základní postuláty a jejich aplikace. Také jsou zde zmíněny jiné ne-euklidovské geometrie a jejich využití. Práce je strukturována do kapitol, které popisují současný stav problému, tedy popis technologií, které jsou v současné době k dispozici, a jejich omezení. Dále pak následuje vyhodnocení současného stavu kreslících nástrojů a popis návrhu a iplementace vlastní práce. V závěru je uvedeno kvalitativní zhodnocení práce, způsob splnění zadání a nastínění dalšího vývoje.

Kapitola 2

Základní pojmy z oblasti dynamické geometrie

Kapitola obsahuje představení software s podporou dynamické geometrie, stručný pohled do historie a popis některých klíčových funkcí a vlastností, které tyto aplikace poskytují. Dále je popsáno využití tohoto software v praxi, případně jeho alternativní aplikace.

Následující obrázek 2.1 zobrazuje vzhled uživatelského rozhraní běžného programu s podporou dynamické geometrie. Je zde vidět nástrojová lišta, dále seznam prvků, které se ve scéně nacházejí a ve spodní části příkazový řádek, kde lze zadávat instrukce pro konstrukci. Hlavní část obrazovky tvoří plocha se scénou. Toto uživatelské rozhraní se v různých variacích vyskytuje ve velkém množství tohoto druhu aplikací, určených pro běžné uživatele.



Obrázek 2.1: Ukázka prostředí podporujícího dynamic geometry. Jedná se o aplikaci GeoGebra [4].

Díky tomu, že konstrukce probíhá na počítači, je pak snadné provádět jednoduché operace jako je mazání některých částí konstrukce nebo zvrátit poslední provedenou akci (případně více akcí). Tyto výhody vyplívající ze samotné podstaty práce na počítači a

dále prohlubují výhody, které poskytují parametrické kreslící nástroje. Obecně pak platí, že čím sofistikovanější kreslící systém, tím existuje více možností jak kontrolovat tvorbu konstrukce.

2.1 Historie

Jak již bylo naznačeno v úvodu, počátky Dynamic Geometry sahají do osmdesátých let. V této době se také zákonitě objevily první nástroje podporující tento způsob konstrukce. První z nich byly *Geometer's Sketchpad* v roce 1989 a *Cabri Géometre* v roce 1988. Oba tyto produkty existují do dnešního dne, prakticky ve stejné podobě [1]. Vývoj ovšem pokračoval dále a v devadesátých letech a dnes již existují desítky aplikací, které se zabývají řešením problémů spojených dynamic geometry [10]. Dalším rozšiřováním se aplikace s podporou dynamické geometrie dostaly daleko za hranice geometrie samotné.

Dnes již není problém, aby tyto aplikace umožňovaly vykreslování libovolných konstrukcí případně zvládaly algebraické výpočty. V poslední době se taktéž čímdál časteji objevují aplikace tohoto druhu, které jsou určeny pro mobilní zařízení a kalkulačky. Jako příklad může například sloužit aplikace *Nspire* od firmy Texas Instruments, která kromě klasické verze pro osobní počítač nabízí i kapesní počítače s kreslícím software.

2.2 Použití

Již na začátku devadesátých let, tedy krátce po vzniku prvních aplikací tohoto typu, bylo velice rychle nalezeno uplatnění pro parametrické kreslící nástroje. Nejvýraznější z nich bylo použití ve školství. Jak žáci, tak učitelé mohli snadno vytvářet geometrické konstrukce, ale co je daleko důležitější, mohli sledovat jejich chování při provádění modifikací některých z parametrů konstukcí. Využití tohoto druhu aplikací je obzvláště užitečné na nižších stupních škol, kde poskytují určitou formu abstrakce pro různé formy grafických nebo geometrických transformací [11].

Další z možných případů využití jsou parametrické CAD systémy. Změnou některého z parametrů konstrukce v tomto případě můžeme sledovat velikost dopadu změn na konstrukci jako celek. Mnohem zajímavější je jiné využití těchto vlastností. Můžeme mít k dispozici jednoduchý prototyp konstrukce, který můžeme rychle s pomocí parametrů upravovat [10].

Zobrazení

V první řadě se zaměříme na dělení software dle jeho možností zobrazení, přesněji na možnost zpracovávání a zobrazování prostorových útvarů. To je všeobecně podstatně náročnější, jak co se týče zobrazení scény, tak manipulace s objekty, které se ve scéně nacházejí.

V případě, že zobrazujeme trojrozměrné objekty na dvojrozměrné zobrazovací zařízení (monitor), je třeba použít transformaci zvanou promítání [13]. Při promítání dochází ke ztrátě prostorové informace, tudíž se zobrazený objekt může jevit zkresleně. Volbou vhodného způsobu promítání můžeme zlepšit pochopení a čitelnost promítaných objektů.

Existují dva základní druhy promítání:

Rovinné
: Promítací paprsky jsou určeny směrem.

Středové
Promítací paprsky jsou určeny středem promítání.

Při rovinném promítání je zachovávána relativní velikost modelu, takže rovnoběžné přímky v modelu zůstávají rovnoběžkami i po zobrazení na průmětně. U středového promítání tomu tak však vždy být nemusí. Rovnoběžné promítání nachází úplatnění zejména v CAD systémech, kde je nutné, aby při zobrazování objektů nedocházelo ke zkreslení.

V případě, že zobrazuje dvojrozměné objekty (body, přímky, kružnice), není třeba promítání vůbec uvažovat, dvojrozměrné modely není problém zobrazit na dvojrozměrném zobrazovacím zařízení. Otázkou však zůstává způsob reprezentace těchto geometrických dat, jelikož existuje řada způsobů, jak se dají reprezentovat a uchovávat.

Dnes může být součástí tohoto druhu software i možnost doprovodu, například postupu konstrukce, vytvoření animace, která celý děj popisuje a jak již bylo zmíněno výše, jedná se o nedocenitelnou funkci v případě výuky geometrie [11].

Geometrické vlastnosti

Mezi základní vlastnosti naprosté většiny aplikací, které k dnešnímu dni existují je možnost sledovat jednotlivé kroky konstrukce a v případě potřeby se vrátit zpět. S tím souvisí i možnost sledovat změny, které způsobí například přesun bodu v reálném čase. Díky této vlastnosti lze například pochopit, jaký bude mít dopad posun jednoho z vrcholů trojúhelníku, jenž je vepsán kružnici. Uživatel ihned vidí důsledek své akce a může například přemýšlet nad příčinami, které vedly k tomuto výsledku. Velká část aplikací umí manipulace s body, přímkami, obecně křivkami a polygony. Výjimkou pak nemusí být možnost nastavení úhlu, který svírají například dvě přímky. V případě aplikací, které umožňují práci ve 3D ovšem vyvstávají problémy se zobrazením prostoru na dvojrozměrnou plochu.

2.3 Euklidovská a Ne-Euklidovské geometrie

Naprostá většina aplikací pracuje s Euklidovskou rovinou, ta představuje základ klasické geometrie a svou jednoduchostí a intuitivností obsáhne celou řadu oborů. Naproti tomu Neeuklidovské geometrie představují značně neintuitivní, a co se týče aplikace, i značně specializované.

Existují proto některé aplikace, jež se výhradně specializují na Neeuklidovskou geometrii, ovšem i systémy, které jsou schopny zpracovávat jak jak euklidovskou, tak neeuklidovské geometrie, jmenovitě parabolickou a hyperbolickou. [8]

Euklidovská geometrie, pojmenována po svém tvůrci, jenž byl řeckým matematikem, představuje nejstarší geometrii, jež našla uplatnění v mnoha oborech, například v klasické fyzice. Euklidovská geometrie je postavena na postulátech a obecných principech, které Euklides shrnul do publikace, dnes označované jako Základy. Pomocí těchto znalostí může být zbylá část geometrie logicky odvozena.

Postuláty euklidovské geometrie

Jak bylo výše zmíněno, euklidovská geometrie je postavena na postulátech (axiomech). Euklides uvádí pět postulátů, které tvoří logický základ geometrie.

1. Pro dva různé body existuje právě jedna přímka, která jimi prochází.

2. Prodloužením úsečky vznikne opět úsečka.

3. Je možné nakreslit kružnici s libovolným poloměrem a středem.

4. Všechny pravé úhly jsou si rovny.

5. K přímce a bodu, který na ní neleží je možno sestrojit právě jednu rovnoběžku, která prochází tímto bodem.

Základní rozdíl mezi euklidovskou a neeuklidovskou geometrií většinou spočívá v tom, že neeuklidovské geometrie nesplňují Euklidův pátý postulát. Termínem neeuklidovská geometrie (*Non-Euclidean geometry*) obecně označujeme geometrie, které nesplňují postuláty euklidovské geometrie.

Z pohledu moderní vědy jsou euklidovy postuláty nekompletní a poněkud zavádějící. Zatímco u prvních čtyř je snadné odvození jejich významu, u posledního to není tak zřejmé. Od doby, kdy Euklides poprvé zapsal tyto postuláty do dnešního dne, byla vytvořena celá řada pokusů odvodit tento postulát jako problém, výsledkem však vždy byl pouze další předpoklad [8]. (Euklidovská geometrie taktéž není dostatečná pro moderní, tj. relativistickou fyziku. Nicméně pro využití v klasické fyzice je stále dostatečná.)

Z ne-euklidovských geometrií rozpoznáváme eliptickou (Riemann) a hyperbolickou (Gauss-Boliay-Lobachevsky) geometrii.

Kapitola 3

Současný stav konstrukčních nástrojů

V této kapitole budou uvedeny některé vybrané kreslící nástroje, jejich použití a jejich možnosti. Zejména pak bude uvažováno jejich rozdělění podle funkcí, které jsou popsány v předchozích kapitolách, případně některých dalších vlastností, které oproti konkurenci nabízejí, nejedná se ovšem o úplný výčet všech těchto vlastností.

Obecný přehled

Existuje široká škála aplikací podporující parametrickou geometrii. Ať již se jedná o komerční produkty, či open-source, většina jich poskytuje základní prostředky a operace pro tvorbu a úpravu geometrických útvarů. Drtivá většina z nich pracuje s euklidovskou geometrií, jedná se o nejstarší část geometrie, jenž publikoval Euklides. Tento druh software, tedy parametrické kreslící aplikace, nabízejí poměrně širokou škálu uplatnění. Jedním z nich je například školství. V tomto případě je nutnou podmínkou mít k dispozici matematicky přesný nástroj, který žákům umožní velice snadné pochopení dané látky a rozvoj jejich znalostí.[11]

Zkoumaný software budeme dělit jednat dle jeho možností zpracovávat 2D nebo 3D, dále dle jeho dalších schopností (animace, důkazy, výrazy) a v neposlední řadě také, zda se jedná o komerční či svobodné aplikace. Cílem je sběr informací o aplikacích tohoto druhu a následného vyhodnocení jejich použitelnosti.

Vzhledem k zaměření práce na zobrazení dat v rovině, budou zde uvedeny aplikace, které vynikají zobrazením dvourozměrných dat a pro operace v prostoru poskytují buď omezené nebo žádné prostředky.

U všech placených aplikací zpravidla platí, že jsou poskytovány v široké škále jazyků, často plně lokalizovány. To souvisí především s hlavním účelem těchto aplikací, to jest poskytnout alternativní přístup k výuce geometrie, zejména její základy. Toho by se dalo pouze ztěží docílit kvůli jazykové bariéře, která je patrná například na základních školách. V případě specializovaných aplikací většinou tyto problémy nenastávají a není nutno je řešit.

Export dat

Existuje celá řada způsobů, kterými lze uchovávat zobrazení scény. Jednou z možností, kterou nabízejí vybrané aplikace je export do HTML. Povětšinou se jedná o webovou

stránku s appletem, který obsahuje uloženou scénu, umožňuje manipulaci, jako je například pohyb, jednotlivých elementů. Další možností je přidání i ovládacích prvků, takže uživatel může ve webovém rozhraní vytvářet vlatní konstrukce. U aplikací, k jejichž implementaci byl použit například jazyk Java je pak implementace těchto možností velice jednoduchá. Například u zkoumaného software Cinderella může být do webové stránky přidán například jednoduchý kalkulátor [10]. Tyto vestavěné funkce usnaďnují použití aplikace, zejména pokud jsou tyto prostředky využity v rámci školní výuky geometrie.

V neposlední řadě také aplikace tohoto druhu umožňují export scény ve do různých formátů. Nejběžněji PNG, BMP a další. V některých případech je podporován i export do vektorových formátů, například Scalable Vector Graphics (SVG) a v některých případech i do formátu Postscript(PS).

Skriptování a uživatelská makra

Využití uživateslkých maker je taktéž zajímavou možností tvorby konstrukcí v případě, že je daná aplikace podporuje. Jejich výhoda spočívá v tom, že lze snadným způsobem aplikovat dříve vytvořené předpisy pro reprodukci procesu. Pomocí maker může být předdefinováno velké množství jednoduchých konstrukcí, jejichž spojením lze rychle vytvářet složité konstrukce.

Další z vlastností, které některé z nástrojů poskytují, je tvorba skriptů a často je k tomuto účelu vytvořen vlastní skriptovací jazyk. Konstrukce tedy nelze vytvářet pouze pomocí uživatelského rohraní, nýbrž i pomocí textových instrukcí. Jako ukázkový příklad je na konec této kapitola zařazena aplikace *CLUCalc*, která mimo jiné pracuje s Geometrickou Algebrou [12].

Ve zbytku této kapitoly bude následovat popis vlastností vybraných konstrukčních nástrojů. Vybrány byly ty nástroje, které poskytují neobvyklou funkcionalitu, případně poskytují zajímavé rozšíření stávajících vlastností a metod. U každého nástroje bude k dispozici jednoduchá tabulka s funkcionalitou. Jednotlivé sloupce tabulky představují:

1. Podporu dané aplikace pro uživatelské skripty.
2. Rozsáhlost podpory práce v prostoru.
3. Možnosti exportu dat dané aplikace.
4. Podporu jiných geometrií. Zde se jedná hlavně o hyperbolickou.

3.1 Cabri II Plus

Mezi nejznámější aplikace zabívající se dynamickou geometrií patří pravděpodobně Cabri Geometry, specifičtěji pak Cabri II Plus pro práci v rovině nebo Cabri 3D pro práci v prostrou. Jedná se o komerční produkt, existuje ovšem zkušební verze, která je dostupná ke stažení a poskytuje základní funkcionalitu. Počátky vývoje software sahají až do roku 1986, kdy byl představen Cabri I, v roce 1994 pak jeho nástupce Cabri II, který byl již distribuován celosvětově. V roce 2000 vedoucí výzkumné skupiny, zodpovědné za vytvoření aplikace Cabri I založil společnost Cabrilog a byla zahájena práce na software Cabri II Plus a Cabri 3D, to jest dnešními aplikacemi [1].

Cabri II Plus obsahuje nástroje, které umožňují tvorbu a používání maker případně přidělení hesla makru. Cabri II Plus taktéž poskytuje funkce pro měření vzdáleností mezi

objekty, obsahy ploch nebo například pro výpočet směrnic přímek. Aplikace obsahuje zabudovaný kalkulátor se základními funkcemi. Další z poskytovaných funkcí je kontrola vzatů mezi jednotlivými objekty v nákresu. Například vyšetření vzájemné polohy dvou přímek, nebo zda body náleží danému objektu.

Další zajímavou funkcí je modifikace objektů na základě změny jejich rovnic. Přímky a kuželosečky jsou v aplikaci obecnými rovnicemi, ve kterých se dají upravit klíčové parametry ovlivňující jejich chování. V neposlední řadě se zde nachází funkce, která zapne sledování pohybu objektů ve scéně. Ve výsledku pak například bod, kterému byla změněna pozice, není vykrelen pouze v novém místě, nýbrž i na původní pozici a po celé trajektorii pohybu.

V neposlední řadě Cabri II plus obsahuje i vlastnost zvanou *Smart Lines*, přeloženo jako chytré čáry. Touto vlastností se rozumí zobrazení pouze relevantního úseku přímky pro danou úlohu. Při vysokém množství přímek poté nedochází k zahlcení obrazovky informacemi, které jsou pro uživatele zbytečné a odvádějí ho od práce. Pokud by ale například posunutím nějaké z těchto přímek mohl vzniknout průsečík s jinou funkcí, je viditelná část této čáry upravena odpovídajícím způsobem, aby nedošlo k zatajování informací.

Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Ano	pouze Cabri 3D	Html, PNG	Proprietární	Ne

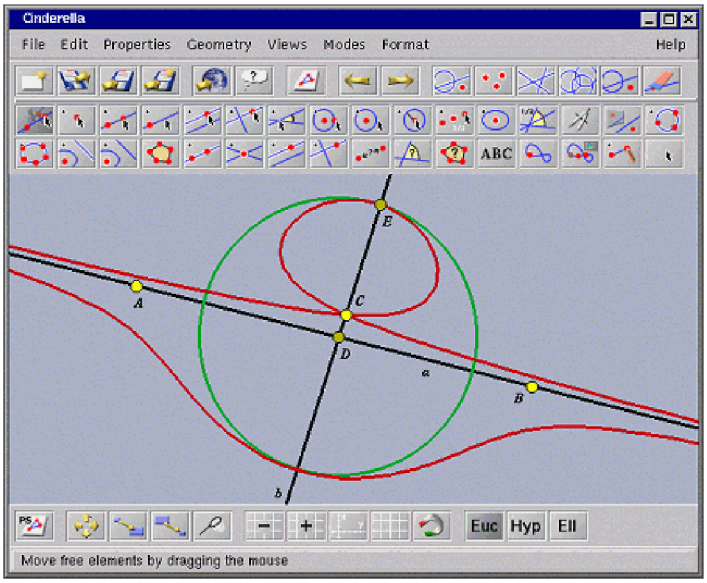
3.2 Cinderella

Dalším zástupcem je software Cinderella [3], poprvé publikován v roce 1998, v současné době napsán v jazyce Java. Cinderella poskytuje všechny základní funkce, které jsou od tohoto druhu software očekávané, zvláštností ovšem je, že poskytuje podporu nejen pro Euklidovskou geometrii, nýbrž i pro hyperbolickou, eliptickou a projektivní geometrii. Této značné rozmanitosti je dosaženo využitím poznatků o geometrii z devatenáctého století. Vzhledem k tomu, že vývoj software Cinderella započal již v devadesátých letech (Stejně jako v případě mnohých dalších aplikací zabývajících se podobnou problematikou), v současnosti je poskytována široká škála funkcí, nejen v oblasti geometrických konstrukcí, ale například i fyzikálních modelů a matematických důkazů.

Aplikace je navržena tak, aby poskytovala co největší modularitu, která zajišťuje jednoduchou rozšiřitelnost. Díky tomuto návrhu je možné, aby konstrukce vytvořená v hyperbolické geometrii byla zároveň zobrazena a modifikována v několika různých modelech. Další zvláštností je tvorba fraktálových obrazců, které lze s aplikací Cinderella snadno vytvářet.

Významným doplňkem jsou fyzikální modely, Cinderella obsahuje prostředí, určené pro tvorbu těchto modelů. Využití těchto modelů je nedocenitelné pro výuku, jelikož představují náhled do jinak abstraktních fyzikálních jevů. K těmto účelům Cinderella poskytuje vlastní skriptovací jazyk, CindyScript. S jeho pomocí pak Cinderella může provádět analýzu matematických funkcí, včetně jejich okamžitého vykreslování za vzniku výukového prostředí s širokým zaměřením.

Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Ano	Omezená	Html, PNG, BMP	Proprietární	Hyperbolická Eliptická



Obrázek 3.1: Aplikace Cinderella s prvky ve scéně a euklidovskou geometrií.

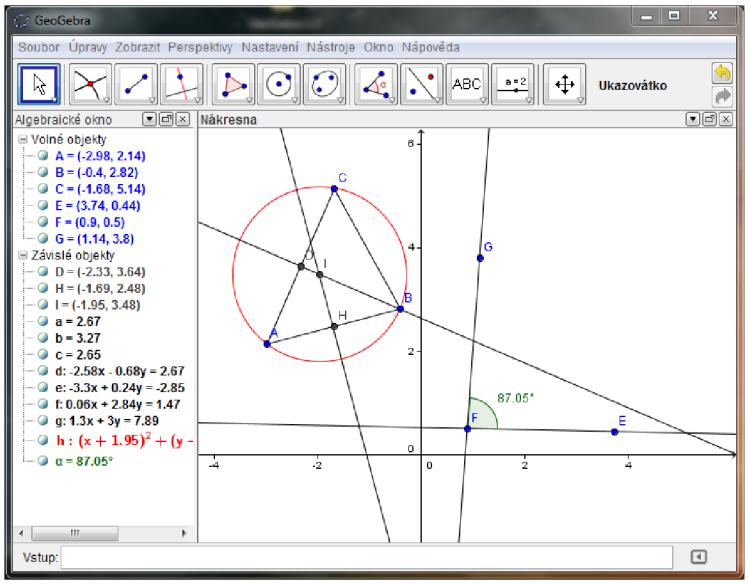
3.3 GeoGebra

GeoGebra je představitelem volně šiřitelného multiplatformního kreslícího nástroje s podporou dynamické geometrie. Podobně jako předchozí aplikace, je i GeoGebra napsána v jazyce Java a kromě spustitelného programu je na webových stránkách k dispozici spustitelný java applet. První verze, GeoGebra 1.0, byla vydána v roce 2002 a poskytovala jednoduché rozhraní pro tvorbu konstrukcí. V dalších verzích docházelo k rozšiřování stávající funkcionality, za přidání podpory algebry, tvorby tabulek a vykreslování uživatelem definovaných funkcí. Podobně jako tomu bylo u předchozích aplikací, i GeoGebra je primárně určena pro výuku. V následujících několika odstavcích budou zmíněny funkce, které aplikace poskytuje, případně její nedostatky [4].

GeoGebra, jako i jiné další nástroje umožňuje tzv. sledování pohybu, jedná se o vlastnost, která může být nastavena u jednotlivých prvků zobrazených ve scéně. Pokud je tato volba povolena, při přesunu daného objektu zůstává ve scéně vykreslena stopa jeho pohybu. Při povolení této funkce u většího množství objektů ovšem může vést ke značné nepřehlednosti scény.

Další z rysů aplikace GeoGebra je seznam objektů, které se ve scéně nacházejí. V seznamu jsou uvedeny názvy objektů, jejich pozice (v případě, že se jedná o body) nebo rovnice, které je popisují (například přímky a kuželosečky). Pro jednodušší orientaci v seznamu mají položky barvu objektů ze scény, tudíž lze při jejich velkém počtu poměrně snadno dohledat chtěný objekt, za předpokladu, že byl dříve označen. V seznamu jsou položky rozděleny do dvou skupin na volné a vázané objety. Zatímco s volnými objekty lze libovolně manipulovat, závislé objekty se řídí logikou, která jim byla přidělena při jejich vytváření. Příklady závislých objektů mohou být například střed úsečky nebo průsečík dvou přímek.

V neposlední řadě GeoGebra obsahuje velké množství zabudovaných funkcí, nejen pro konstrukci objektů ve scéně, ale i transformační, logické, pravděpodobnostní a mnoho dalších. Používání těchto funkcí je dostupné z příkazové řádky, kterou program obsahuje. Do této příkazové řádky lze ovšem zadat i libovolný předpis funkce, která bude zobrazena. S takto zobrazenými funkcemi lze dále manipulovat, případně na nich zobrazovat body a



Obrázek 3.2: Ukázka aplikace GeoGebra verze 4

zkoumat její chování nebo vliv na celkovou konstrukci.

Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Ano	Ano	PNG, SVG, PS	GPL	Ne

3.4 The Geometer's Sketchpad

Počátky tohoto nástroje sahají až do osmdesátých let, tedy do doby, kdy celý obor zabývající se dynamickou geometrií byl teprve ve svých počátcích. Jednalo se o projekt pod vedením Drs. Eugena Klotze a Dorise Schattschneidera se zaměřením na výzkum nových prostředků použitelných při výuce geometrie. V roce 1991 následovalo vydání první komerční verze programu následované v roce 1993 pro operační systémy Windows [5]. V následujících letech pokračovalo rozšíření z euklidovské geometrie do analytické. Ve čtvrté verzi vydané v roce 2001 následovalo rozšíření mimo oblast dynamické geometrie v rozšíření zaměřeném na algebru.

Jednou ze zajímavostí, kterou Sketchpad nabízí je dynamická změna os scény. Je možno měnit souřadnice počátku i jedtnolivé osy.

Geometer's Sketchpad je překládán do celé řady jazyků, zahrnujících například čínštinu, korejštinu nebo malajštinu. Vzhledem ke svému zaměření, obsahuje i funkce pro přidávání tlačítek do konstrukce nebo tvorbu animací s modelem.

Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Ano	Ano	PNG, Html	Proprietární	Ne

3.5 CaRMetal

Poslední zmíněnou aplikací bude program C.a.R. (Compass and Ruler) jehož autoři jsou Eric Hakenholz Rene Grothmann. Stejně jako v případě předchozích aplikací je i C.a.R. vyvíjen v jazyce Java a jeho počátky se datují do roku 1996. Software je licencován pod

GNU GPL. Od svých počátků, kdy se jednalo o velice jednoduchý nástroj, až do dneška prošel C.a.R. velkou řadou změn. Dnes poskytuje širokou škálu operací, včetně tvorby třírozměných modelů a podporu hyperbolické geometrie. Aplikace byla zamýšlena pro využití na nižších stupních škol, nicméně přidáním ne-euklidovské geometrie, se použitelnost rozrostla i pro vyšší stupně škol [2].

Podobně jako ostatní aplikace, i C.a.R. nabízí export do různých grafických formátů nebo pro LaTeX ve formě postscriptu. Samozřejmostí zůstává podpora maker jak ve 2D, tak v omezené míře i 3D. K větší přehlednosti modelů slouží nastavitelné barvy u jednotlivých prvků.

Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Ano	Značně omezená	PNG, SVG, PS	GPL	Hyperbolická

CLUCalc

Již pouze na okraj je uvedena aplikace CLUCalc, která k veškerým geometrickým výpočtům využívá geometrické algebry, jelikož autor této aplikace je i autorem knihy *Geometric Algebra with applications in Engineering* [12], která se zabývá využitím Geometrické algebry v praxi. Nástroj CLUCalc obsahuje interpret jazyka *CLUScript*, jenž slouží ke tvorbě konstrukce.

Geometrická algebra představuje silné prostředí pro reprezentaci a řešení nejrůznějších geometrických problémů. Své využití ve výpočetní technice zastává zejména v oblasti počítačové grafiky, robotiky a počítačového vidění [9].

V následující kapitole bude obsaženo shrnutí zjištěných poznatků, které byly získány studiem výše uvedených aplikací a stejně tak jejich zhodnocení, případně vyzdvyhnutí některých jejich jedinečných vlastností.

Kapitola 4

Zhodnocení současného stavu

Tato kapitola obsahuje shrnutí údajů, které byly získány studiem různých aplikací s podporou dynamické geometrie, jejich předností a použití. V tabulce je uveden výčet jejich funkcí a na závěr jsou shrnuty některé z jejich nedostatků. Tento výčet není úplný, nicméně je dostatečný vzhledem k povaze práce samotné.

4.1 Shrnutí současného stavu

Následující tabulka obsahuje shrnutí stávajících vlastností parametrických kreslících nástrojů a jejich zhodnocení.

	Skriptování	Podpora 3D	Export dat	Licence	Geometrie
Cabri II Plus	Ano	Značně omezená	PNG, SVG, PS	GPL	Hyperbolická
Cinderella	Ano	Omezená	Html, PNG	Proprietární	Hyperbolická Eliptická
GeoGebra	Ano	Ano	PNG, SVG, PS	GPL	Ne
Sketchpad	Ano	Ano	PNG, Html	Proprietární	Ne
CaRMetal	Ano	Značně omezená	PNG, SVG, PS	GPL	Hyperbolická

Aplikace, které byly uvedeny v předchozí kapitole představují poměrně široké spektrum parametrických kreslících nástrojů, nicméně z tabulky lze vyčíst jejich společné rysy. Jedním z nich je podpora skriptování v různé míře, dalším rysem jsou poměrně obsáhlé možnosti exportu, ať se již jedná o obyčejný obrázek, pdf nebo webovou stránku s java appletem.

Vzhledem k tomu, že uvedené aplikace se specializují na dvojrozměrnou geometrii, nelze očekávat velká podpora pro modelování v prostoru. Pokud by bylo potřeba právě toto modelování provádět, je lepším řešením zvolit specializovaný program, jelikož existuje pouze malé množství aplikací, které podporují konstrukce v rovině i prostoru, nicméně i tak některé ze studovaných aplikací, byť i jen v omezené míře, poskytovaly tyto nástroje.

Poslední sloupec tabulky označuje další druhy geometrií, pro které byla v aplikacích zavedena podpora. V tomto ohledu se u studovaných apliací projevil zajímavý úkaz. Zatímco systémy, které obsahovaly rozšířenou podporu práce v prostoru, jiné geometrie neuvažovaly, tak systémy s omezenými možnostmi práce v prostoru, umožňovaly práci i s jinými geometriemi. Zejména aplikace Cinderella obsahovala rozsáhlé možnosti ohledně práce s různými pohledy nad různými geometriemi.

Jedním z menších nedostatků některých aplikací zůstává například neúplná detekce nebo určování průsečíků. V případě že se jedná o průsečík dvou přímek, jedná se o jeden společný bod (Za předpokladu, že přímky nejsou totožné). Pokud se ovšem jedná o průsečík přímky s kuželosečnou nebo průsečík dvou kuželoseček, může na základě jejich vzájemné polohy být těchto bodů více. Záleží pak na daném software, jak se s touto situací vypořádá. Aplikace může buď najít všechny průsečíky, i když si to uživatel nemusel přát a tím pádem mu v konstrukci zůstávají objekty, které mu mohou překážet v jeho další činnosti. S tímto problémem se dá vypořádat například tak, že nechtěné body zůstanou skryty a uživatel si je může v případě potřeby zobrazit.

Dalším způsobem je vytvoření průsečíku pouze v místě, kde si to přál uživatel. Určení tohoto místa může být například vzdálenost místa kliknutí od některého z průsečíků. Výhodou tohoto řešení může být, že vzniká pouze průsečík, který si uživatel patrně přál. V případě, že ovšem hledal všechny průsečíky, bude muset tuto operaci opakovat pro každý z nich.

S průsečíky, ovšem nejen s nimi je spojen i druhý, nedostatek, který se vyskytoval u všech zkoumaných aplikací. Nutně se ovšem nemusí jednat o nedostatek, jelikož implementací této funkcionality by vzniklo neurčité chování objektů ve scéně. Jedná se o případ, ve kterém by uživatel chtěl manuálně provést přesun bodu, jehož chování je specifikováno jinými objety, nikoliv tedy jeho vlastními souřadnicemi. Příkladem takovýchto bodů jsou například průsečíky, středy úseček, úhlů nebo body ležící na přímkách, kuželosečkách, či dokonce obecných funkcích.

Zatímco v případě bodů, jejichž chování je dáno pouze jednou funkcí je manipulace s nimi poměrně snadná, zde stačí vyhodnotit funkční hodnotu pro danou souřadnici bodu, a všechny aplikace ji ovládají, pokud se ovšem jedná o průsečík přímek, situace se značně komplikuje. V dnešních aplikacích se v podstatě jedná o statické body, se kterými nelze manipulovat ve smyslu jejich manuálního přesunu. Pokud by je uživatel chtěl z nějakého důvodu přesunout na jinou pozici, musí tak učinit tím, že modifikuje postavení například jedné ze přímek.

4.2 Vypracování zadání

Pokud by existovala možnost libovolného přesunu, kteréhokoliv objektu v konstukci, značně by se rozšířila dynamičnost celé aplikace, ovšem pouze za předpokladu, že určující objekty, v případě průsečíku jsou to například dvě přímky, budou mít logické a očekávatelné chování pro uživatele. Jelikož se ovšem jedná o značně subjektivní záležitost, současné aplikace tuto funkcionalitu neposkytují.

Cílem mé práce je navrhnout a implementovat parametrický kreslící nástroj, který by poskytoval některé běžné funkce dnešních komerčních či svobodných aplikací. Chování těchto funkcí by mělo být pro běžného uživatele očekávatelné nebo přinejmenším pochopitelné. Dále bych se rád pokusil rozšířit chování těchto aplikací v oblasti interakce uživatele s objekty ve scéně, specifičtěji umožnění uživateli modifice souřadnic i bodů, které mají speciální chování, například střed úsečky, průsečík dvou přímek. Výsledný kreslící nástroj by měl splňovat následující body:

- Přidávání a mazání prvků do scény.
- Modifikace parametrů již vložených prvků.
- Překreslování na sobě závislých prvků scény v průběhu přesunu některého z nich.

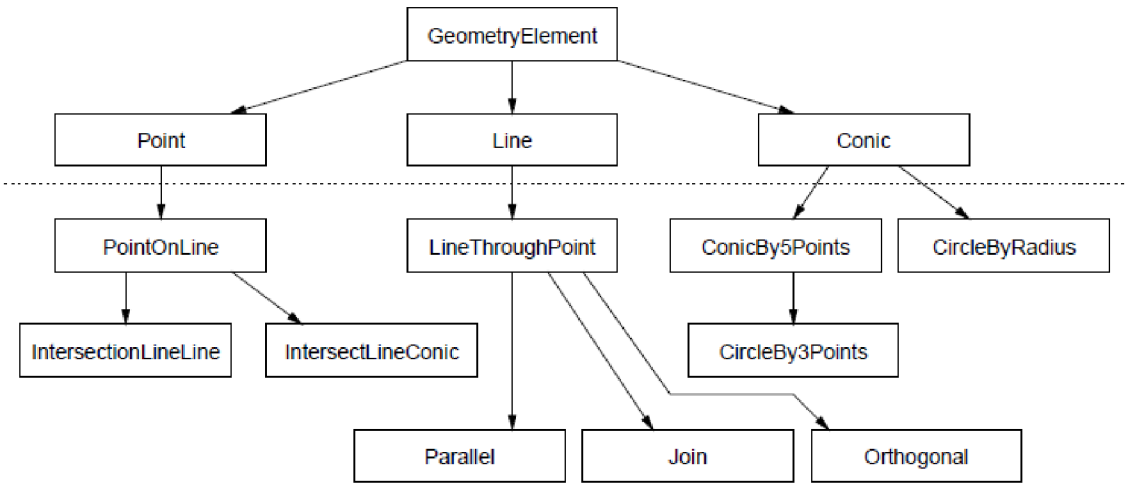
Kapitola 5

Návrh datových struktur aplikace

Volba vhodných datových struktur tvoří zásadní část návrhu aplikace a může zásadní měrou ovlivnit efektivitu jednotlivých operací a stabilitu celého systému. Vhodným návrhem lze dosáhnout snadno rozšiřitelné aplikace. Tato kapitola je věnována popisu různých alternativ návrhu datových struktur a chování aplikace.

Typický způsob uchování dat

Existují různé přístupy sloužící k reprezentaci grafických dat. V první řadě bude znázorněn klasický přístup k objektovému návrhu grafického geometrického software. Jedná se o způsob, který dnes používá většina aplikací zabývajících se dynamickou geometrií, princip je patrný z obrázku 5.1. Základním předpokladem této implementace je existence dědičnosti a polymorfismu. Existuje hierarchie tříd, které reprezentují jednotlivá grafická primitiva, jako jsou bod, přímka nebo křivka.



Obrázek 5.1: Hierarchie tříd při běžném způsobu použití dědičnosti a polymorfismu [10].

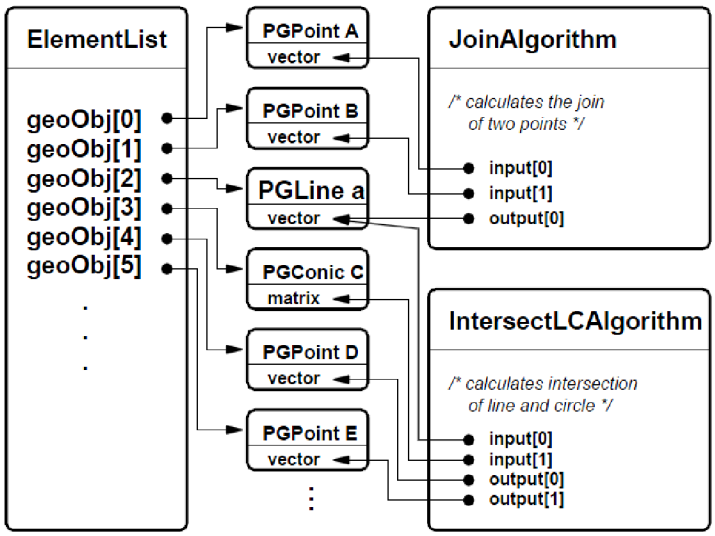
Pro případy, že se jedná například o bod na přímce, průsečík dvou přímek, apod. existují další odvozené třídy, které popisují toto chování. Tyto třídy přetěžují chování svých rodičů a přidávají další datové struktury potřebné k uchování nových informací. Při pohybu myší jsou uchopenému geometrickému elementu (elementu, který je určen k přesunu) zaslány

nové souřadnice. Dále je nutné aby byly upraveny souřadnice všech elementů, které jsou závislé změně pozice nebo jiné vlastnosti elementu, se kterým jsme původně manipulovali. Následuje úprava dalších elementů, které byly závislé na předchozích a tak dále. Obvykle existuje seznam, do kterého jsou vloženy všechny prvky, u kterých je potřeba vyšetřit dopady změn některého z elementů. Jelikož mezi prvky mohou existovat křížové závislosti, mohl by nastat případ, že dva na sobě navzájem závislé prvky by neustále požadovaly přepočet hodnot, je proto potřeba kontrolovat, zda již nebyl element, který požaduje přepočet upraven [10].

Základní třída, ze které dědí všichni potomci obsahuje metody, které budou používány potomky, dále může obsahovat například předpis pro obecné informace o objektu, jako je například název nebo barva, jedná se o abstraktní třídu. Z ní odvozené třídy `Point`, `Line` a `Conic` představují odpovídající elementy, které se nacházejí v prostoru. Tyto třídy tedy obsahují struktury potřebné k jejich konstrukci, u bodu jeho souřadnice, u přímky například její počátek a vektor. Z nich jsou odvozeny další třídy, které představují speciální případy těchto prvků. Jako jsou například průsečík dvou přímek představovaný třídou `IntersectionLineLine` nebo `CircleByRadius`.

Algoritmický přístup

Například program Cinderella nevyužívá výše popsaný způsob reprezentace dat, třídy které leží v hierarchii níže nahrazuje jiným způsobem reprezentace. Diagram hierarchie tříd je z části podobný hierarchii použité při klasickém přístupu zobrazeném na obrázku 5.1.



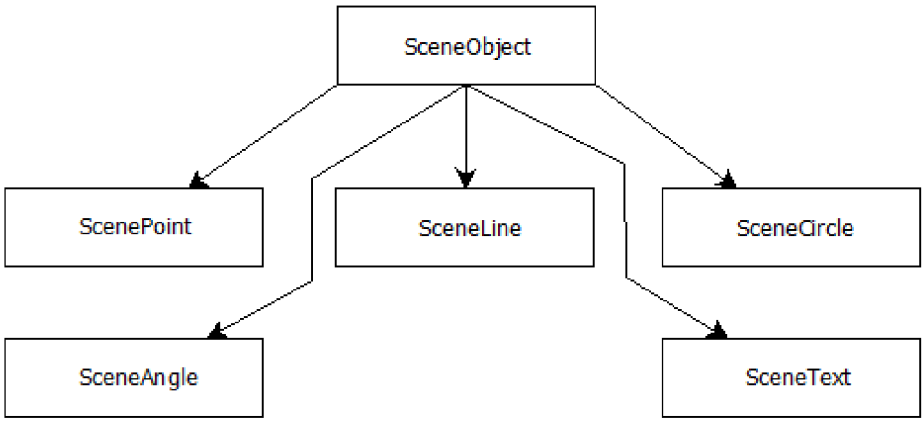
Obrázek 5.2: Algoritmický přístup k datům, použitý v aplikaci Cinderella [10].

Opět existuje jedna základní třída, reprezentující geometrický objekt. Od ní jsou odvozeny třídy, jež reprezentují základní prvky jako jsou bod nebo přímka. K dalšímu dědění jak tomu bylo v předchozím případě však již nedochází. Namísto toho, základní třída obsahuje objekt Algorithm, který obsahuje všechny informace o závislých geometrických prvcích včetně metod určených k jejich úpravě [10]. Cílem této změny oproti běžnému způsobu je zlepšení efektivity práce s datovými strukturami pro funkce, při kterých například dochází

k výpočtům průsečíků.

5.1 Způsob použitý v aplikaci

V této práci budu vycházet ze schematu, které je zobrazeno na obrázku 5.3 Základem je třída `SceneObject`, která obsahuje základní datové struktury pro popis nejobecnějších grafických prvků. Od ní pak dědí třídy, které představují bod, úsečku nebo kružnici. Oproti standardnímu postupu, který je zobrazen na obrázku 5.1, nastává v návrhu významná změna. Speciálního chování specifických objektů ve scéně není dosaženo dědičností z původních tříd, nýbrž alternativním způsobem, při kterém si každá třída, která reprezentuje daný geometrický element, uchovává i informace o jeho chování. Ve své podstatě se jedná o zjednodušenou formu algoritmického přístupu zobrazeného ve schematu 5.2, která je ovšem pro implementaci parametrického kreslícího nástroje dostačující.



Obrázek 5.3: Hierarchie tříd knihovny zajišťující dynamické chování objektů ve scéně.

Jádro knihovny tvoří třída, `SceneList`, která obsahuje seznam s jednotlivými prvky, které se vyskytují ve scéně. Třída obsahuje metody, pro přidávání jednotlivých prvků do scény a udržuje jejich stav. Mimo jiné také zajišťuje přidělování unikátních id.

5.2 Třída SceneObject

Třída `SceneObject` představuje základ pro všechny třídy reprezentující některý z grafických elementů. Kvůli serializaci objektu je nutné, aby objekty obsahovaly nejlépe číselný identifikátor, který by byl schopen je jednoznačně identifikovat. K tomuto účelu třída obsahuje položku *Id*, ta je nastavitelná buď pomocí konstruktoru nebo přístupové metody. Případný uživatel-programátor nemusí nastavení tohoto identifikátoru nijak řešit. V případě, že použije metody, které poskytuje třída `SceneList`, je unikátní Id objektu přiděleno při použití některé z metod `AddItem`.

Další položkou je *Name* typu `string`, jedná se o volitelné jméno grafického objektu, které může být zobrazeno například ve výčtu objektů scény. Tato položka je zcela volitelná a pokud není uživatele definováno jinak, všechny objekty budou nést název *unspecified*.

Důležitá je i datová položka *origin*, která obsahuje souřadnice objektu ve scéně a je využita v případě, že daný objekt je bodem. Následuje trojice ukazatelů *Source*, *Destination* a *Additional* na třídu `SceneObject`. Tyto položky jsou využity v případě, že grafický objekt

nemůže být popsán pouze pomocí svojí pozice, ale je určen jinými objekty. Takovým případem může být úsečky daná dvěma body nebo kružnice se středem a poloměrem. Položka Addtitional může sloužit například pokud je třeba určit kružnici třemi body.

`SceneObject` mimo jiné obsahuje položku dependencyList, která obsahuje ukazatele na všěchny objekty, které jsou závislé na daném objektu. Uchovávání této informace je nutné pro správné revalidaci objektů ve scéně v případě provedení nějaké změny.

K revalidaci stavu objektu pak slouží dvojice virtuálních metod `Revalidate` a `Execute`. Právě pomocí těchto metod je docíleno toho, že se všemi objekty ve scéně je možna interakce. Zatímco metoda `Revalidate` provede kontrolu pozice objektu a případně ji dle potřeby upraví, metoda `Execute` provede nad objektem přesně specifikovanou akci, která je dána parametrem typu `SceneAction`. Pro jeden objekt tedy může existovat několik druhů chování.

Přesuny objektů ve scéně

K popisu přesunů jednotlivých objetků ve scéně slouží třída `SceneAction`. V konstruktoru je specifikováno, o jaký druh akce se jedná a hodnoty, které specifikují akci. Například pokud se jedná o přesun, položka *offset* obsahuje hodnotu posunu ve směrech os.

Instance třídy je předána metodě `Execute` daného objektu, který danou akci provede a následně provede revalidaci objektů na něm závislých. Třída obsahuje datovou položku *object*, do které může být uložen ukazatel na objekt, kterého se akce týkala a celý tento záznam je uchován v historii pro případ, že by bylo nutno akci zvrátit. To ovšem zahrnuje pouze akce typu přesun a transformace, pro udržování kompletní historie by bylo potřeba tuto třídu rozšířit o další datové struktury.

5.3 Reprezentace bodu ve scéně

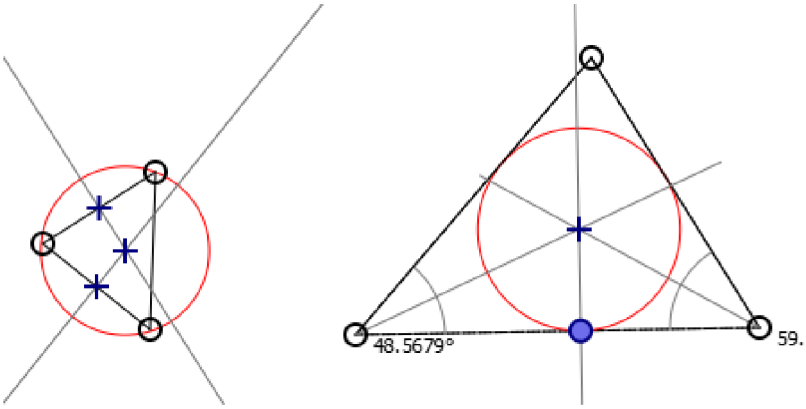
Základem většiny geometrických konstrukcí, ať se již jedná o konstrukci v počítačovém programu nebo na běžný papír, je bod. Ve vyvíjené aplikaci bod představuje fundamentální základ prakticky každé konstrukce, jelikož právě manipulací s bodem ve scéně vzniká jednoduchá metoda manipulace s ostatními prvky scény. Jako příklad lze uvést přesun úsečky, který se skládá z přesunu koncových bodů daným směrem a následného upravení hodnoty vlastní úsečky.

K reprezentaci všech druhů bodů, které se mohou vyskytnout ve scéně slouží třída `ScenePoint`. Ať se již jedná o volný bod nebo průsečík dvou přímek. Třída pro tento účel obsahuje položku *type*, jenž může nabývat hodnot daných výčtem `PoinFunction`. Ten obsahuje všechny druhy chování bodu, tedy volný bod, střed úsečky, průsečík nebo bod náležící některé z funkcí.

Z tohoto způsobu návrhu vyplývá, že jeden bod může mít pouze jednu speciální funkcim Může se buď jednat o střed úsečky nebo průsečík, nikdy však oboje. Rozšíření funcionality bodu na více druhů chovnání by však nebylo složité, otázkou zůstává jaká by byla využitelnost této vlastnosti. Aby bylo možno slučovat body různých funkcí, stačilo by zavést systém podbodů, které by sice měly vlastní chování z hlediska objektů, kterými jsou vázány, ovšem jejich pozici by určoval jeden rodičovský bod. Ten by v případě obdržení instrukce pro přesun tuto akci delegoval všem podbodům, které má pod svojí správou.

Tímto způsobem by bylo umožněno spojování bodů, kterými lze libovolně pohybovat, existují však i body, kterým uživatel může určovat pozici pouze ve směru některé z os. I

zde existuje několik cest, jak se s touto situací vypořádat. Buď lze zakázat, aby některý z podbodů byl typu `FunctionMember`, tím pádem by se vždy muselo jednat o rodičovský bod a nemohla by nastat situace, ve které by se tento bod ocitl mimo funkci, které náleží. Dalším způsobem je jednoduché zakázání jakéhokoliv vázání tohoto druhu bodů navzájem, tudíž tato problematická situace nikdy nenastane. Třetím způsobem je zavedení obousměrného řízení mezi bodem a jeho funkcí. Nejen, že souřadnice bodu by byly dány funkcí, ale i průběh funkce by mohl být ovlivněn pozicí bodu. Body by se pak mohly libovolně slučovat, nicméně pro složitější funkce by implementace této funkcionality byla prakticky nereálná. Bez ohledu na to, by se nejspíše nejednalo ani o očekávané chování pro uživatele. Ten by patrně očekával, že pokud bod leží na přímce, tak přímka vymezuje možné pozice tohoto bodu a nikoliv naopak.



Obrázek 5.4: Body ve scéně zobrazené různým způsobem, na základě hodnoty atributu type třídy `ScenePoint`. Způsob zobrazení má charakterizovat funkcionalitu bodu, dá se však libovolně měnit. Na obrázku jsou zobrazeny kružnice vepsaná a kružnice opsaná trojuhelníku.

Další z položek, obsahuje způsob vykreslení bodu ve scéně, jedná se o položku *Type*. Typ vykreslení bodu se dá kdykoliv ve vlastnostech bodu změnit, stejně jako další jeho atributy, nicméně implicitně je grafické zobrazení bodů děleno na základě jejich funkce. Cílem je zesnadnit uživateli orientaci mezi body a zejména aby na první pohled bylo zřejmé, že dva body, které leží v těsné blízkosti mohou mít různá chování.

Bod náležící funkci

Jelikož by knihovna měla podporovat i funkci bodů naležících jiné funkci, je potřeba vyřešit problém, jakým způsobem budou určovány funkční hodnoty těchto bodů.

Na první pohled se jedná o zcela triviální záležitost. Pokud máme k dispozici rovnici funkce a hodnotu x-ové souřadnice bodu, stačí nám pouhým dosazením zjistit funkční hodnotu. Pouze s touto znalostí si ovšem nevystačíme. Z obrázku je patrné, že pokud by nastal případ, že úsečka (přímka) by měla směrnici blízkou nekonečnu, pro nepatrné změny na ose x by docházelo k obrovským skokům ve funkčních hodnotách.

Řešení tohoto problému spočívá v rozhodnutí, která z hodnot bodu bude neznámá. V případě, že se jedná o úsečku (přímku), jsou nejprve vypočítány rozdíly mezi jednotlivými souřadnicemi bodů. Menší z nich bude určovat osu, pro kterou se bude vypočítávat souřad-

nice. Toto jednoduché řešení umožňuje libovolnou manipulaci s body ležícími na libovolné přímce.

Podobný problém nastává i u bodů, které leží na kuželosečkách. Zde mimo jiné dochází i k situaci, že pro jednu hodnotu, může existovat více funkčních hodnot. Jako nový bod je pak vybrán ten, který se nachází blíže k původnímu bodu. Tímto jednoduchým rozhodnutím se zabraňuje k tomu, aby při přesunu bodu po kružnici docházelo k nežádoucím skokům mezi jednotlivými alternativami. Zatímco u úsečky byl rozhodovacím faktorem rozdíl souřadnic koncových bodů, u kružnic k záměně os dochází vždy po $\pi/2$, s počátkem v $\pi/4$. Je tím docíleno poměrně intuitivního pohybu bodu po kružnici s pomocí počítačové myši. Stejným způsobem je možno určovat pozici bodu ležícího prakticky na libovolné funkci. Nicméně pokud budou vstupem funkce vyšších řádů, lze očekávat zpomalení výpočtu a s tím související zobrazení takovýchto funkcí a bodů ve scéně [7].

5.4 Přímky a úsečky

Další z důležitých prvků každé konstrukce jsou přímky a úsečky. K reprezentaci čar obecně slouží třída `SceneLine`.

Podobně jako tomu bylo u předchozí třídy, i `SceneLine` si v položce *function* uchovává funkci, kterou daná čára reprezentuje. Na základě hodnoty výčtu `LineFunction` se pak nahlíží na data uložená v položkách *source* a *destination*, podobným způsobem, jako tomu bylo v případě bodů.

Třída obsahuje tři základní druhy chování, může se jednat o úsečku danou dvěma body, kolmici k úsečce, procházející bodem, případně rovnoběžku k úsečce procházející bodem, kde právě jedna z těchto možností určuje způsob, jakým bude daná čára vykreslena, nebo osu úhlu. Nelze tedy zkonstruovat libovolnou přímku.

Stejně jako třída `ScenePoint`, i `SceneLine` implementuje obě funkce, *Execute* a *Revalidate*. Tím pádem je možno manipulovat s čarou jako s celkem. Třída sama zajistí, že klíčové i závislé body jsou správně pozměněny.

Instanci třídy lze nastavit vlastnost *Length*, která určuje délku dané úsečky. Pokud se jedná o jakýkoliv jiný typ, je toto nastavení ignorováno. K dosažení změny délky úsečky podle zadané hodnoty dochází na základě posunu koncového bodu, ten však i po změně leží na přímce, kterou určovaly původní body úsečky.

5.5 Kružnice

K realizaci kružnic ve scéně je k dispozici třída `SceneCircle`. V současnosti je povolena konstrukce zadáním středu a bodu ležícího na kružnici. Podobně jako u třídy `SceneLine` existovala metoda `setLength`, která nastavovala délku úsečky, nabízí `SceneCircle` metodu `setRadius`. Pomocí této funkce lze nastavit radius kružnice pomocí podobné logiky jako tomu bylo u `setLength`.

Ačkoliv třída `SceneCircle` dědí od `SceneObject`, k uchovávání dat nevyužívá struktury této třídy. Informace nutné pro sestrojení kružnice jsou uloženy v položkách `centerPoint`, obsahuje ukazatel na bod, který je středem kružnice, a `radiusPoint`, který obsahuje bod na kružnici. Vzdálenost mezi těmito dvěma body určuje poloměr kružnice. Důvodem, proč nejsou využity datové položky, které jsou zděděny od třídy `SceneObject`, je jejich využití pro jiné účely, jmenovitě modelování jiných druhů kuželoseček, pro které je potřeba uchovávat větší množství informací. V případě konstrukce kružnice to ovšem není potřeba.

Výpočet úhlů a zobrazení textu ve scéně

V neposlední řadě by bylo vhodné, aby do scény bylo možno přidat například odstavec textu. Ten může sloužit jako nápověda k zapsanému prvku konstrukce, popis, jakým konstrukce vznikla, nebo například úkol, který má uživatel, v tomto případě nejspíše student, splnit.

K jednoznačné reprezentaci libovolného textového pole je opět využito, vlasností třídy `SceneObject`, každý text, má tedy přidělený jednoznačný identifikátor a volitelné jméno. Samotný text ve scéně reprezentuje třída `SceneText`.

Vzhledem k využití třídy `SceneObject`, lze text ve scéně libovolně přesunovat nebo jej například skrýt před uživatelem.

V výpočtu úhlu, který je určen třemi body slouží třída `SceneAngle`, která umožňuje sledování a modifikace vybraných úhlů.

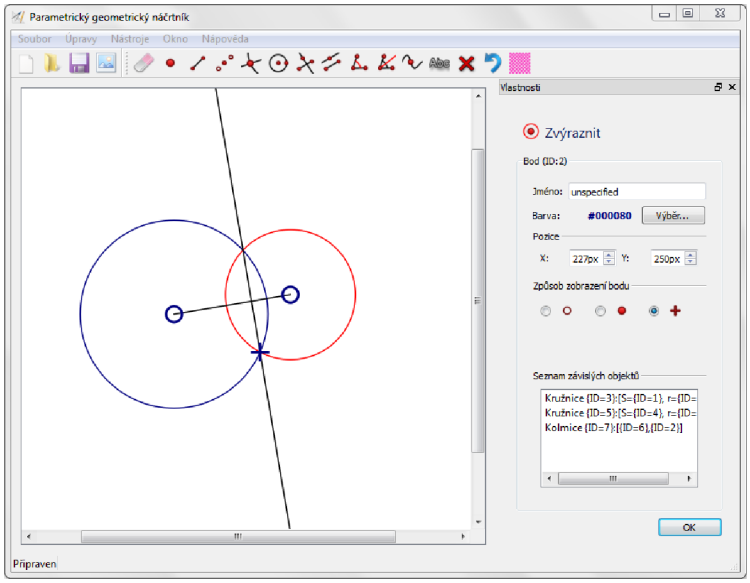
Narozdíl od řady jiných aplikací zabývajících se dynamickou geometrií, umožňuje tato třída kontrolu úhlu až do 360 °. Tímto rozšířením je každý úhel jednoznačně identifikovatelný. Třída taktéž obsahuje metodu, pomocí které lze úhel invertovat.

Kapitola 6

Návrh uživatelského rozhraní aplikace

Tato kapitola bude věnována popisu uživatelského rozhraní aplikace, ovládání a některých dalších fundamentálních vlastností, které umožňují snadno pochopitelnou podporu při tvorbě konstrukce s prvky dynamické geometrie.

V dnešní době není problém implementace chování dynamické geometrie. Existuje široká škála aplikací, které splňují požadavky uživatele, ať jsou jakékoliv. Hlavní důraz při vývoji by tedy měl být kladen na grafické uživatelské rozhraní, jelikož to je nedílnou součástí každé moderní aplikace. V případě parametrických kreslících nástrojů tato teze platí obzvlášť, jelikož manipulace s prvky ve scéně by měla být snadno pochopitelná a jednoznačná. Cílová skupina uživatelů jsou běžní uživatelé, je tedy nezbytně nutné, aby práce s aplikací byla možná například pouze pomocí myši.



Obrázek 6.1: Ukázka uživatelského rozhraní výsledné aplikace.

Na obrázku je vidět uživatelské rozhraní aplikace, která vznikla jako výsledek této práce. Jako u většiny těchto kreslících nástrojů, hlavní část okna zabírá kreslící plocha. V horní části okna je lišta s nástroji, pro zakreslení jednotlivých prvků do scény.

V této aplikaci je rovněž přidán formulář, který zobrazuje vlastnosti právě zvoleného objektu, a jejich modifikací dochází ihned k jejich správnému překreslení ve scéně.

Požadavky na uživatelské rozhraní

Cílem práce je vytvoření parametrického geometrického náčrtníku, z hlediska dobrého návrhu aplikace, by tedy bylo dobré, aby jednotlivé části aplikace zůstaly odděleny a relativně nezávislé na sobě. Pokud tedy v uživatelském rozhraní dojde k vyvolání akce, například vložení bodu do scény, je nejprve provedena změna v datových strukturách knihovny pro řízení zobrazení a po vyhodnocení této akce dojde k vykreslení scény na základě stavu prvků scény.

Jelikož cílem práce je plná interaktivita jednotlivých prvků ve scéně, musí být zajištěno odpovídající chování i pro body, které mají nějakou speciální funkci. Například tedy průsečíky nebo body, které jsou jinak závislé na více jiných objektech.

6.1 Zobrazení scény s konstrukcí

Aplikace, jejíž tvorba je cílem práce je logicky rozdělena na tři relativně samostatné části. První část tvoří knihovna poskytující chování aplikované v dynamické geometrii. Informace o této knihovně, jejích třídách a použití jsou popsány v předchozí kapitole. Druhou část aplikace zobrazovací zařízení, ve kterém je zobrazena scéna s konstrukcí a třetí částí pak uživatelské grafické rozhraní aplikace pro jednoduchou tvorbu konstrukcí.

Pro zobrazení scény je v aplikaci využito prostředků, které nabízí Qt Framework, přesněji třídy `QGraphicsScene`, která poskytuje prostředí pro zobrazení velkého množství grafických 2D elementů. V podstatě se jedná o úložiště objektů třídy `QGraphicsItem`, která tvoří společný základ pro všechny grafické objekty, které se v `QGraphicsScene` mohou vyskytnout [6].

Pro účely aplikace byla vytvořena třída `RenderArea`, která je právě potomkem `QGraphicsScene`. Tato třída zapouzdřuje seznam objektů dynamické scény v položce *sceneData*, která uchovává `SceneObject`, nicméně zobrazitelné objekty si uchovává ve vlastním seznamu. Existuje několik způsobů, kterými se dá provést mapování mezi odpovídajícími objekty `SceneObject` a `QGraphicsItem` objekty.

1. Využitím asociativního pole, které bude obsahovat jednotlivé dvojice objektů.
2. Zavedením třídy, která bude potomkem `QGraphicsItem` a například bude obsahovat referenci přímo na odpovídající objekt.

Ve vyvíjené aplikaci byla postupně použita obě řešení, přičemž v první fázi vývoje třída `RenderArea` skutečně obsahovala mapu mezi jednotlivými objekty. Zavedením druhého způsobu detekce objektů byla tato redundantní informace odstraněna a efektivita systému se zlepšila.

Třída `RenderArea` dále obsahuje metody, pro přidávání dynamických objektů do scény. Tyto metody jsou podobné původním, které nabízí třída `QGraphicsScene`, ovšem s tím rozdílem, že například při přidání úsečky nejsou jednotlivými parametry souřadnice koncových bodů, nýbrž již existující body ve scéně. To samozřejmě neplatí, pokud je do scény přidáván nový bod, u toho je nutné zadat jeho souřadnice.

Vhodným doplňkem, který by třída měla podporovat, je zobrazování grafických elementů, které zatím nebyly přidány do scény zcela nebo nebyly potvrzeny. Za příklad takového chování lze považovat konstrukci kružnice v momentě, kdy byl zvolen její střed, ovšem bod, který bude určovat poloměr kružnice zatím nikoliv. Při pohybu myší by se měla zobrazit kružnice, která by odpovídala, kdyby se hledaný bod nacházel právě na místě, kde se nyní nachází myš.

Další položky třídy `RenderArea` specifikují nastavení zobrazení scény, jako je nastavení antialiasingu nebo například zobrazení mřížky pro snadnější odhad vzdáleností mezi prvky ve scéně, případně další podobná nastavení. Všechna tato nastavení jsou uchovávána v konfiguračním souboru s nastaveními aplikace, v sekci Renderer. Tímto způsobem si program uchovává informace o nastavení v době ukončování programu a při dalším spuštění je právě z tohoto konfiguračního souboru načítá, pokud soubor není nalezen, je aplikace spuštěna s implicitními hodnotami nastavení.

Třída jako taková žádným způsobem nemanipuluje se souřadným systémem scény, všechny objekty jsou v seznam `SceneList` uloženy právě v tomto souřadném systému. Vhodnou transformací by sice bylo možné je převést do jiného vhodnějšího systému, nicméně tato funkcionalita není součástí třídy `SceneList`.

Tento druh operací spíše patří právě k zobrazovacímu zařízení, které určuje způsob, jakým budou interpretována data, která poskytuje knihovna pro dynamickou geometrii. Pokud by existovala volba na přepínání mezi jednotlivými souřadnými systémy, data o objektech ve scéně by se stejně musela ukládat v jednotné normalizované podobě, aby nedošlo ke ztrátě nebo poškození informací.

6.2 Zobrazení objektů ve scéně

Vzhledem k tomu, že je nutné zobrazovat grafické objekty a zároveň s nimi udržovat informace i informace o datových složkách jim odpovídajících, je potřeba rozšířit původní `QGraphicsItem`. Derivát této třídy, `GeneralItem`, již obsahuje datovou položku obsahující ukazatel na přiřazený `SceneObject`. `GeneralItem` také provádí předefinování virtuálních metod `hoverEnterEvent` a `hoverLeaveEvent`, jedná se o reakce na události, pokud k pohybu myší nad daným objektem. V tom případě se provede vyznačení uchopitelné oblasti daného objektu, pro lepší zpětnou vazbu s uživatelem.

Od této základní třídy jsou následovně odvozeny další, které již reprezentují specifický grafický objekt. V současné době se jedná o následující třídy:

- `PointItem`, reprezentuje všechny druhy bodů, které se ve scéně vyskytují, ať se již jedná o volný bod, pod náležící funkci, nebo například průsečík.
- `LineItem`, reprezentuje jakoukoliv úsečku nebo přímku.
- `EllipseItem`, která v současné době odpovídá pouze kružnici, nicméně může být využita pro vykreslení libovolné kuželosečky.

Společným rysem všech těchto tříd je, že každá implementuje metody `paint`, `boundingRect` a `shape` třídy, `QGraphicsItem`, jedná se totiž nutné minimum, které je potřebné k tomu, aby grafický objekt mohl být zobrazen ve scéně.

I když třída `QGraphicsItem` obsahuje datové struktury pro zobrazení objektu ve scéně, `GeneralItem` těchto služeb ze zjištných důvodů nevyužívá. V případě jejich použití by docházelo k redundanci a s tím spojením snížením rychlosti vykreslování objektů.

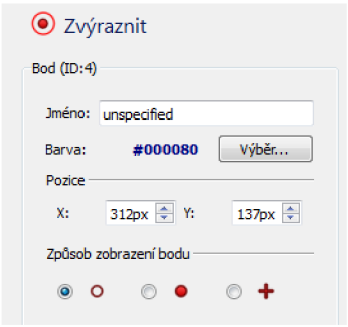
Třídu `GeneralItem`, lze využít i jinými způsoby. Jedním z nich může být například uchovávání stavu pera a štětce. I když lze jejich hodnoty určit dle objektu `SceneObject`, jejich uchováním lze předejít jejich zbytečnému vytváření při každém překreslení objektu. Pokud však nastane změna atributů ve třídě `SceneObject`, které ovlivňují jejich hodnotu, je nutné aby došlo k odpovídající revalidaci i na straně `GeneralItem`, k tomu mohou sloužit například události nebo v případě Qt signály a sloty [6].

6.3 Uživatelské rozhraní aplikace

Uživatelské rozhraní poskytuje základní prostředky pro práci s objekty ve scéně. Kromě scény samotné obsahuje také formuláře, které zobrazují vlastnosti právě vybraného objektu ze scény. Kdykoliv tak lze modifikovat jejich vlastnosti. Tyto vlastnosti jsou rozděleny do dvou skupin.

- Společné rysy, které obsahují všechny objekty scény, jako je jméno, barva, případně Id.
- Vlastnosti specifické pro daný objekt. Formulář dynamicky, dle vybraného objektu určí, které položky mají být zobrazeny. Za příklad může sloužit poloměr kružnice nebo délka úsečky.

Formulář také provádí vizualizaci položek Source, Destination třídy `SceneObject` a seznamu závislých prvků na daném objektu. Výběrem těchto položek lze procházet mezi jednotlivými prvky přímo vázanými na daný objekt.



Obrázek 6.2: Formulář s vlastnostmi vybraného prvku scény.

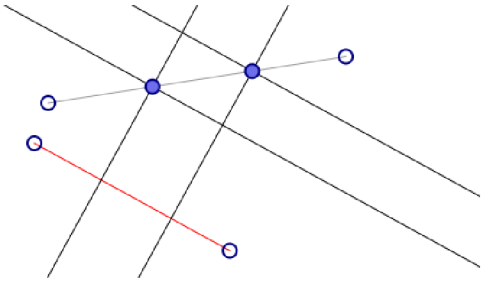
Aplikace taktéž obsahuje formulář, ve kterém jsou zobrazeny veškeré objekty, které se nachází ve scéně. Libovolný objekt lze vybrat a modifikovat. Zavedením tohoto formuláře dochází k odstranění jedné z nevýhod, kterou aplikace v současném stavu má a to je výběr prvků ve scéně, které se překrývají. Pokud bude na stejné souřadnici několik bodů a uživatel bude chtít zvolit některý z nich, vždy se mu podaří vybrat pouze ten bod, který byl zkonstruován jako poslední, to přináší celou řadu nevýhod a značně to může stížit některé konstrukce. Pomocí seznamu prvků, které se ve scéně nacházejí lze tento problém odstranit. Nevýhodou tohoto řešení zůstává, že uživatel musí znát, které body se na dané souřadnici nacházejí.

Prvky scény jsou v tomto seznamu zapsány tak, aby je bylo možné snadno identifikovat. V případě že nebylo zadáno jméno prvku, je zobrazeno jeho unikátní id, pokud však uživatel jméno zadal, bude místo id zobrazen právě tento název, zde ovšem nelze zaručit, že se v seznamu nebudou opakovat stejné názvy. To závisí zcela a pouze na uživateli.

6.4 Testování funkčnosti aplikace

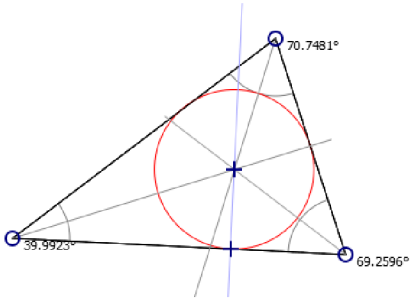
K ověření funkčnosti aplikace byla vybrána sada jednoduchých konstrukčních úloh, u kterých existoval předpoklad, že v průběhu konstrukce by mohlo dojít k selhání nebo by výsledek konstrukce neodpovídal očekávání, nejedná se tedy o testovací úlohy, které by měly zkoušet schopnosti uživatele používat ovládací prvky aplikace nebo jeho znalosti geometrie.

Jednou z takových úloh je například konstrukce kolmic k úsečce. Úloha spočívá v tom, že je dána úsečka a postupně jsou sestrojovány kolmice. Pomocí této úlohy byl v počátečních fázích vývoje aplikace odhalen nedostatek v návrhu, jehož výsledkem bylo, že kolmice sestrojená ke kolmici na úsečku nebyla s touto úsečkou rovnoběžná, což je zásadní chyba. Zároveň při manipulaci s některým z koncových bodů úsečky by mělo docházet ke správné rotaci všech kolmic.



Obrázek 6.3: Testovací konstrukce systému kolmic.

Dalším příkladem úlohy, kterou lze otestovat funkčnost aplikace, je například konstrukce vepsané trojuhelníku. Pomocí této úlohy lze ověřit správnost výpočtů úhlů a určování jejich os. V případě konstrukce kružnice opsané lze například otestovat chování kružnice v případě, kdy se její střed blíží k nekonečnu.



Obrázek 6.4: Testovací konstrukce kružnice vepsané trojuhelníku.

Pomocí těchto vybraných úloh se podařilo odhalit celou řadu vážných nedostatků, které mohly být vzápětí opraveny. To ovšem neznamená, že konstrukce musí probíhat zcela bez chyb, stejně jako tomu může být v jakémkoliv jiném parametrickém náčrtníku.

Kapitola 7

Závěr

Cílem práce bylo vytvoření návrhu a implementace parametrického kreslícího nástroje, který by pracoval na dnes známých principech. Funkčnost měla obsáhnout zobrazení v rovině, případně v prostoru. Tohoto cíle bylo dosaženo.

Druhá kapitola této práce shrnuje základní pojmy z oblasti dynamické geometrie, které tvoří obecný základ pro kreslící nástroje tohoto druhu. Nastudoval jsem některé z dostupných aplikací, které se zaobírají problematikou dynamické geometrie, a jejich vlastností, tyto jsou zapsány ve třetí kapitole. Jejich vyhodnocení je posléze provedeno ve čtvrté kapitole, tímto byl splněn první bod zadání. Pátá kapitola obsahuje návrh principů použitelných pro parametrické kreslící nástroje a dále návrh jejich implementace, splňující druhý a třetí bod ze zadání. Popis čtvrtého bodu zadání, tedy implementace knihovny pro dynamickou geometrii, včetně uživatelského rozhraní aplikace je následně popsán v šesté kapitole. Poslední bod zadání, možnosti dalšího pokračování v práci nebo návrhy rozšíření funkčnosti aplikace jsou uvedeny v následujících odstavcích.

Výsledkem práce je aplikace, která funguje jako parametrický geometrický náčrtník, který poskytuje základní funkcionalitu běžnou u řady dnešních aplikací. Pro správné použití aplikace se předpokládá, že uživatel bude mít základní znalosti z oblasti matematiky a geometrie. Zdaleka nejobtížnější se mi pak zdála implementace chování průsečíků funkcí, pokud se jedná například o průsečíky dvou kružnic a s tím spojené problémy. Zpočátku se také bylo obtížné docílit správného chování v případě, kdy docházelo například ke konstrukci kolmice k jiné kolmici, ovšem pozměněním původního návrhu byl tento problém snadno odstraněn.

Další pokračování vývoje aplikace, by se mohlo ubývat například rozšířením do jiných druhů geometrií, jako příklad lze uvažovat ne-euklidovské geometrie. Dalším možným rozšířením by mohlo být zavedení podpory skriptování. Pomocí jednoduchého skriptovacího jazyka by šla vytvořit podpora například pro uživatelská makra nebo přidávání objektů do scény pomocí textových příkazů.

Literatura

[1] *Cabri Geometry* [online]. [cit. 15. prosince 2011]. Dostupné na: <http://www.cabri.com/>.

[2] *C.a.R.* [online]. [cit. 26. března 2012]. Dostupné na: <http://zirkel.sourceforge.net/>.

[3] *Cinderella* [online]. [cit. 20. února 2012]. Dostupné na: <http://cinderella.de/>.

[4] *GeoGebra* [online]. [cit. 15. prosince 2011]. Dostupné na: <http://www.geogebra.org/>.

[5] *The Geometer's sketchpad* [online]. [cit. 26. března 2012]. Dostupné na: <http://www.dynamicgeometry.com/>.

[6] *Qt Online Reference Documentation* [online]. [cit. 30. dubna 2012]. Dostupné na: <http://doc.qt.nokia.com/>.

[7] Bartsch, H.-J. *Matematické vzorce*. 4. vyd. 2006. ISBN 80-200-1448-9.

[8] Coxeter, H. *Non-Euclidean Geometry*. 6. vyd. 1998. Dostupné na: <http://books.google.cz/books?id=usKZpDAH0WUC>. ISBN 0-88385-522-4.

[9] Dorst, L., Fontijne, D. a Mann, S. *Geometric algebra for computer science: an object-oriented approach to geometry*. 1. vyd. 2007. ISBN 978-0-12-369465-2.

[10] Kortenkamp, U. *Foundations of Dynamic Geometry*. Zurich: Swiss Federal Institute of Technology Zurich, 1999. Disertační práce. Dostupné na: <http://kortenkamps.net/papers/1999/diss.pdf>.

[11] Laborde, C. Dynamic Geometry Environments as a Source of Rich Learning Contexts for the Complex Activity of Proving. *Educational Studies in Mathematics*. 2000, roč. 44. S. 151–161. Dostupné na: <http://dx.doi.org/10.1023/A:1012793121648>. ISSN 0013-1954.

[12] Perwass, C. *Geometric Algebra with Applications in Engineering*. 1. vyd. 2009. ISBN 3-540-89067-X.

[13] Žára, J. *Moderní počítačová grafika*. 1. vyd. 2004. ISBN 80-251-0454-0.

Příloha A

Obsah CD

CD obsahuje následující soubory a složky:

`Src`	Složka obsahuje zdrojové soubory výsledné aplikace.
`Bin`	Složka obsahuje zkompilovanou aplikaci.
`Doc`	Složka obsahuje vygenerovanou dokumentaci zdrojových kódů aplikace.
`Report`	Složka obsahuje zdrojové kódy tohoto dokumentu.
`xvalaj01_bp.pdf`	Tento soubor obsahuje technickou zprávu v elektronické podobě.
`readme.txt`	Soubor obsahuje nápovědu ohledně spouštění aplikace a obsah CD.