STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor 18 – Informatika

AGCalc+

Program pro řešení úloh analytické geometrie

autor:	Martin Žák
	ufonzak@gmail.com
ročník:	septima
škola:	Gymnázium Jiřího Wolkera
	Kollárova 3, Prostějov 796 01
	http://www.gjwprostejov.cz
konzultant:	Josef Škurek

Rád bych poděkoval:

svému konzultantovi Josefu Škurkovi za skvěle podané rady a úvahy, které provázely naše diskuse nejen o projektu. Také spolužáku Petru Kalandrovi za drobné, ale nepostradatelné rady při implementaci projektu. A oběma za příjemné pracovní prostředí, které mě při práci obklopovalo.

pedagogům Mgr. Janě Koutné za její nekonečnou motivaci, Mgr. Františku Střídeckému za jeho čas při betatestingu a RNDr. Anně Průšové za netradiční výuku deskriptivní geometrie.

všem schopným autorům článků a příspěvku, ze kterých jsem čerpal především při implementaci.

mamince Marii Žákové za vytvoření loga programu.

a nakonec všem lidem, kteří se mě snažili podpořit nejen drobnými radami, ale i kladným slovem.

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V ……..... dne …… podpis: ……

Změny v práci (ChangeLog)

Zde uvádím všechny změny provedené v práci a programu od jejich prvního uveřejnění.

- 17. Dubna 2009 – fyzické rozdělení programu na dvě části: knihovnu s matematickým jádrem a samostatný program AGCalc+.
- 17. Dubna 2009 – úprava barev v legendě nástroje *Přehled pracovní plochy.*
- 17. Dubna 2009 – přidání klávesové zkratky pro aktivaci nástroje *Přehled pracovní plochy.*
- 17. Dubna 2009 – následkem rozdělení programu úprava kapitoly 4 a vygenerování nových diagramů tříd v příloze D.
- 19. Dubna 2009 – úprava číslování kapitol a odstavců.
- 19. Dubna 2009 – přidání data ověření platnosti URL adres v kapitole použité literatury a internetových zdrojů.
- 19. Dubna 2009 – nahrazení nesprávně použitého pojmu složený zlomek v kapitole 5.1 za pojem složené číslo.

Anotace

AGCalc+ je komplexní program určený k řešení praktických příkladů analytické geometrie, který si klade za cíl obsáhnout všechny výpočetní techniky v tomto oboru s důrazem na jeho praktické využitím jako pomůcky při středoškolské výuce jak pro studenty, tak pro pedagogy. Tato práce popisuje vznik programu spolu s úvahami vedoucími k jeho dnešní podobě.

Anotation

AGCalc+ is complex program invented to solve practical tasks in analytic geometry and which is aimed at any counting method in this domain as practical tool for high school students and teachers. This work describes creating this program including development and its actual form.

Klíčová slova

Analytická geometrie, Mongeova projekce, lineární transformace, C#, Microsoft Visual Studio, .NET Framework

Keywords

Analytic geometry, two-plane projection, linear transformations, C#, Microsoft Visual Studio, .NET Framework

Obsah

Seznam obrázků

1 Úvod

Při výuce analytické geometrie na naší škole nám byl zadán velmi rozsáhlý příklad, který přinejmenším přesahoval rámec hodiny. Zadáním byly tři vrcholy trojúhelníku s úkolem určit všechny analyticky dostupné atributy (střed a poloměr kružnice opsané i vepsané, těžiště, těžnice apod.). Když jsem se jím poté doma zabýval, napadlo mne, jak by bylo skvělé mít program, který by toto řešil prakticky pár kliknutími. Nejprve jsem vytvořil v jazyce C++ tyto matematické třídy: bod, vektor, přímku a k nim několik potřebných operací. Tyto třídy jsem poté sjednotil pod jednu konzolovou aplikaci, která všechny potřebné atributy trojúhelníku spočetla a vypsala v několika vyjádřeních. Se svou prací jsem byl spokojen, a proto jsem se této problematice začal více věnovat a rozhodl se pokračovat ve své práci. Uvědomoval jsem si, že již existují matematické programy, které tyto rovnice řeší, ne však dostupný program zabývající se přímo analytickou geometrií, a to z pohledu geometrických objektů. Svůj program jsem začal rozšiřovat s přibývajícím učivem a velmi rychle jsem si zvykl na ověřování výsledků úloh v této jednoduché konzolové aplikaci. Brzy již výpis z konzole přestal stačit, proto jsem přistoupil k vytvoření grafického uživatelského prostředí, a s touto částí se rychle začaly objevovat nové a nové požadavky na práci v programu. Tento moment lze považovat za zahájení mé práce, protože jsem se rozhodl svým programem pokrýt určitou skupinu uživatelů – ostatní středoškolské studenty a pedagogy.

Analytická geometrie zahrnuje rozsáhlou problematiku. Nabízí především mnoho technik, kterými lze řešit geometrické úlohy převodem na algebraické. V této práci se proto nebudu zabývat konkrétními technikami analytické geometrie ani jejich implementací, ale analýzou požadavků vzniklých při vývoji grafického prostředí, návrhem tohoto prostředí a také myšlenkami, které jsem vložil do tohoto programu.

2 Pojmy a zkratky v práci

Objekt – pojem jsem v práci omezil na objekt analytické geometrie (výčtem: bod, vektor, přímka, rovina, kuželosečky, kulová plocha), pro ostatní objekty (z informatického hlediska) jsem použil implementačního výrazu třída.

Funkce – funkcí se, kromě možnosti nebo účelu prvku, vždy myslí ryze informatická implementovaná funkce, nikoliv matematická funkce.

GUI – (*Graphical User Interface*) – grafické uživatelské rozhraní.

.NET – zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu dostupnou pro OS MS Windows, Web i další.

C# – objektově orientovaný jazyk vyvíjený firmou Microsoft, jeden z hlavních jazyků .NETu

MDI – (*Multi Dokument Interface*) – poměrně stará technologie implementovaná v .NETu, která umožňuje mít více oken (MdiChild) v rámci jedné aplikace otevřené v jednom hlavním okně (MdiParent), které pro ně vytváří volně expandující pracovní plochu.

XML – (*eXtensible Markup Language*) – *rozšiřitelný značkovací jazyk*; jazyk určený především pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

Flash – je grafický vektorový program v současné době vlastněný a vyvíjený firmou Adobe. Používá se především pro tvorbu (převážně internetových) interaktivních aplikací.

Mongeova projekce – dvojice pravoúhlých promítání na dvě vzájemně kolmé průmětny (půdorysnu a nárysnu); jedna ze základních technik deskriptivní geometrie.

Kótované promítání – pravoúhlé promítání do jedné průmětny, kde jsou body opatřeny kótami podle pozice vzhledem k průmětně.

3 Analýza

3.1 Úvod analýzy

Z pohledu analýzy jsem svůj software rozdělil na dvě pracovní části: Matematické jádro a samotný programu s GUI.

Matematické jádro je souhrn tříd, které se objevily již v první konzolové aplikaci. Tyto třídy zastupují všechny objekty analytické geometrie. Obsahují i matematické funkce s nimi související. Patří sem i třída, kterou mají všechny objekty poděděny. Je to třída *objekt analytické geometrie*, která obsahuje souhrnné vlastnosti a funkce. Při následné implementaci matematických operací se vyčlenily požadavky na funkce řešící různé algebraické operace (např. soustava rovnic o dvou neznámých) a ty jsem shrnul do jedné třídy (*algebraické operace*) v této části.

Program s GUI – obsluhuje matematické objekty a umožňuje uživateli pohodlnou manipulaci s nimi zahrnujíc jejich tvorbu, provádění matematických operací, grafického zobrazení atd.

Další dokumentaci analýzy jsem vyčlenil výběru platformy, vizuálnímu řešení programu a také požadavkům, které se vyskytly při jeho vývoji.

3.2 Výběr platformy

Jedním z faktorů hrajících roli při této volbě bylo i usnadnění, vlastně tedy urychlení mojí práce. Rozhodoval jsem se mezi klasickou desktopovou aplikací – moderním objektovým jazykem C# s knihovnami .NET Framework (pro GUI) a webovou aplikací – což by byl při tomto rozsahu projektu úkol pro Flash. Skutečnost kvalitní a rozsáhlé podpory jazyka C# a jeho přímého objektového zaměření, společně s příjemným vývojovým prostředím Visual Studia 2008 Express Edition pro mne ale bylo dostatečným důvodem pro výběr tohoto jazyka. Také designer GUI integrovaný v tomto vývojovém prostředí byl značným urychlením oproti vlastnímu návrhu a implementaci prvků GUI.

3.3 Vizuální řešení

3.3.1 Systém nástrojů

Když se rozhlédneme po moderních aplikací kolem sebe, ve většině z nich se uživatel setká s nějakým druhem nástrojů, ať už máme na mysli moderní grafické editory (např. editory firmy Adobe), nebo vyspělé textové procesory (OpenOffice Writer) a další. Tyto nástroje jsou obvykle začleněny v panelech, nebo se volně pohybují (tzv. floating) a různě k sobě přiléhají. Tento systém nástrojů mi připadal velmi interaktivní a funkční. Nechal jsem se jím inspirovat a svůj program vytvořil podobným způsobem. Navíc nástroje jako takové mi umožňují program velmi lehko rozšiřovat o nové možnosti.

V dalších částech vývoje svého programu jsem již vycházel z toho, že všechny možnosti a požadavky na funkce v programu budou vázány na nějaký nástroj.

3.3.2 Objekt a jeho reprezentace

Jelikož jsem chtěl zachovat určitou nezávislost matematického jádra na programu, nemohl jsem implementovat reprezentaci objektů na úrovni GUI přímo do matematických tříd. Z toho plyne, že k zachování této nezávislosti je potřeba vytvořit jistou vazbu mezi objektem a jeho reprezentací na úrovni GUI. Tato vazba by měla být pevná – především defaultně (standardní cestou) nenarušitelná, popř. narušitelná jiným, zabezpečeným a validní způsobem. Reprezentace na úrovni GUI bude pak jiná specializovaná třída, která zabezpečí vlastní vykreslení a další obslužné funkce.

3.3.3 Samotná vizualizace

Nástroje a grafickou reprezentaci objektů je potřeba nějakým způsobem zobrazit. U nástrojů byla volba snadná. Využil jsem plnou sílu .NET Frameworku a nástroje jsem pomocí standardní metody MDI kontejnerů vložil do hlavního okna aplikace. S objekty to pak bylo složitější. Programy (jako např. MathCad) obvykle preferují pracovní plochu v podobě dokumentu, kde mohou být objekty různě posouvány atd. Této možnosti jsem nemohl využít, protože samotné vytvoření systému pro vykreslování reprezentací objektů na pracovní plochu je velmi časově náročné a přináší spoustu problémů, které by byly zbytečným zdržením projektu. Proto jsem se opět spolehl na standard MDI: objekty jsou v něm volně pohyblivé – každý jako samostatný formulář.

3.4 Specifikace a analýza požadavků

Hlavní požadavek jsem si ujasnil již před napsáním první konzolové aplikace. V programu obsáhnu všechny druhy matematických operací, kterých je schopen člověk při řešení úloh v analytické geometrii. Z toho plyne také schopnost řešit i složité geometrické úlohy. Další požadavky jsou již velmi konkrétní a týkají se především GUI a jeho funkcí. Některé jsem si ujasnil, ještě než jsem začal navrhovat a implementovat GUI, další se objevily jako nové možnosti programu při jeho vývoji. Samozřejmostí by měla být intuitivnost a snadnost ovládání, stejně jako přehlednost samotného programu.

3.4.1 Požadavky na GUI a jeho funkce

3.4.1.1 Vytváření objektů dosazením přímo do obecné rovnice

Všechny úlohy začínají zadáním. V geometrické praxi jde většinou o několik bodů, které určují objekt, nebo stojí samostatně, v případě fyziky pak i vektory. Avšak při řešení úloh v analytické geometrii, zejména při hodinách matematiky, jsou objekty zadávány přímo rovnicemi, a i když lze body a vektory těchto objektů z rovnic přímo vyčíst nebo lehce vypočítat, jedná se o krok navíc. Krok, který chci uživateli značně zjednodušit. Proto jsem vytvořil nástroj, jenž je ve své podstatě výčtem všech objektů a jejich tvarů rovnic, do kterých lze přímo dosadit a tím vytvořit tento objekt. Tento nástroj jsem pojmenoval *Nástroje přímého zadání*.

3.4.1.2 Provádění matematických operací

Když je vytvořeno zadání pro úlohu, je potřeba provést potřebné operace s těmito objekty. Provedení těchto operací pak vyžaduje určitou interakci mezi objekty. Výsledkem pak bude nový objekt a nebo také hodnota, popř. obě tyto možnosti. Když jsem poprvé vytvářel potřebné interakce mezi objekty, vytvářel jsem je pro každý objekt zvlášť, výsledné řešení pak bylo rozsáhlé a nesjednocené. S jeho rozšiřováním projekt narůstal spíše exponenciálně, což mě nakonec vedlo k tomu, celý systém operací přepracovat a začít s úplně novou představou. S představou systému operací jako např. domácí pekárny na chleba. Uživatel se rozhodne podle kuchařky upéct nějaký druh chleba. Namíchá si podle kuchařky těsto, to pak vloží do pekárny a pekárna mu chleba upeče. Uživatel má však i druhou možnost: koupí si již hotové těsto (třeba bude vybírat se zavřenýma očima), pouze je vloží do pekárny a může se těšit na chleba; někdo mu totiž namíchal těsto a on se o ně nemusí starat. A já jsem se rozhodl systém operací takto udělat: uživatel si vybere operaci, ta mu ukáže svoje požadavky, uživatel je jako dítě doplní do skládačky a operace se provede. To vše na jednom místě. Jednoduché, funkční, rozšiřitelné. Možnost, ve které si uživatel náhodně vybere těsto jsem neopomenul – pro usnadnění polohových operací jsem vytvořil determinující mechanizmus, kterému stačí pouze zadat dva libovolné objekty a on určí jejich vzájemnou polohu a dle výsledku poté provede i související metrické a další operace. Např. určení vzájemné polohy přímky a bodu – dle polohy se určí parametr bodu na přímce nebo jeho vzdálenost. Scénáře *polohových operací* jsou samostatně vypsány v příloze A. Důležitou vlastností je, že objekty při žádné operaci nezanikají, pouze vznikají nové.

3.4.1.3 Výpis uživateli

V předchozím odstavci se zmiňuji o výsledcích operací jako hodnot. A tyto hodnoty je samozřejmě nutné uživateli sdělit jemu příjemným způsobem. Při běhu také program zaznamená spoustu událostí, které by uživatel chtěl znát. Také sám uživatel dělá chyby proti „systému“, o kterých musí být informovaný. Pro všechny tyto události jsem vytvořil záznamové (logovací) nástroje, jeden pro události, druhý pro výsledky. Chyby uživatele jsou pak rozděleny podle místa vzniku.

3.4.1.4 Grafické zobrazení objektů

Představa úlohy v analytické geometrii není nikterak jednoduchá a fakt, že velmi pomáhá k pochopení úlohy samotné, je dostatečným důvodem pro vytvoření tohoto nástroje. Diskutabilní zde je druh zobrazení v prostoru. Rozhodoval jsem se mezi kótovaným promítáním a Mongeovou projekcí. Kótované promítání je velmi jednoduché a představitelné, avšak dle mého názoru není při řešení polohových úloh tak názorné jako Mongeova projekce. Proto jsem zvolil na představivost náročnější Mongeovu projekci.

3.4.1.5 Dlouhodobé uchování objektů (přenositelnost dat)

Zde jsem zvolil formu tzv. *projektů*. Jelikož objektů je relativně velký počet i při řešení menších úloh, nepřipadá zde v úvahu možnost objektu jako samostatného souboru. Projekt analytické geometrie bude soubor slučující libovolný počet objektů. Pro soubor projektu jsem poté zvolil následující náležitosti – jméno, datum poslední změny a možnost uložení poznámek (např. pro instrukce atd.) Pro správu projektů jsem se rozhodl vytvořit samostatný nástroj *Prohlížeč projektů*.

3.4.1.6 Lineární transformace (maticí)

Lineární transformace nespadají již přímo do analytické geometrie, pro jejich začlenění jsem se rozhodl až později, kvůli rotaci, posunutí, škálovaní kartézské soustavy souřadnic. Matice jsou vhodný způsob, jak toto provést v čele s možností volně tyto operace kombinovat, a vytvářet tak plně transformované matice. Rozhodl jsem se přidat možnost matice ukládat s vlastními jmény a opět do celků zvaných *Sada matic*. Pro transformace jsem vytvořil samostatný nástroj.

3.4.1.7 Dostupná nápověda ke každému prvku

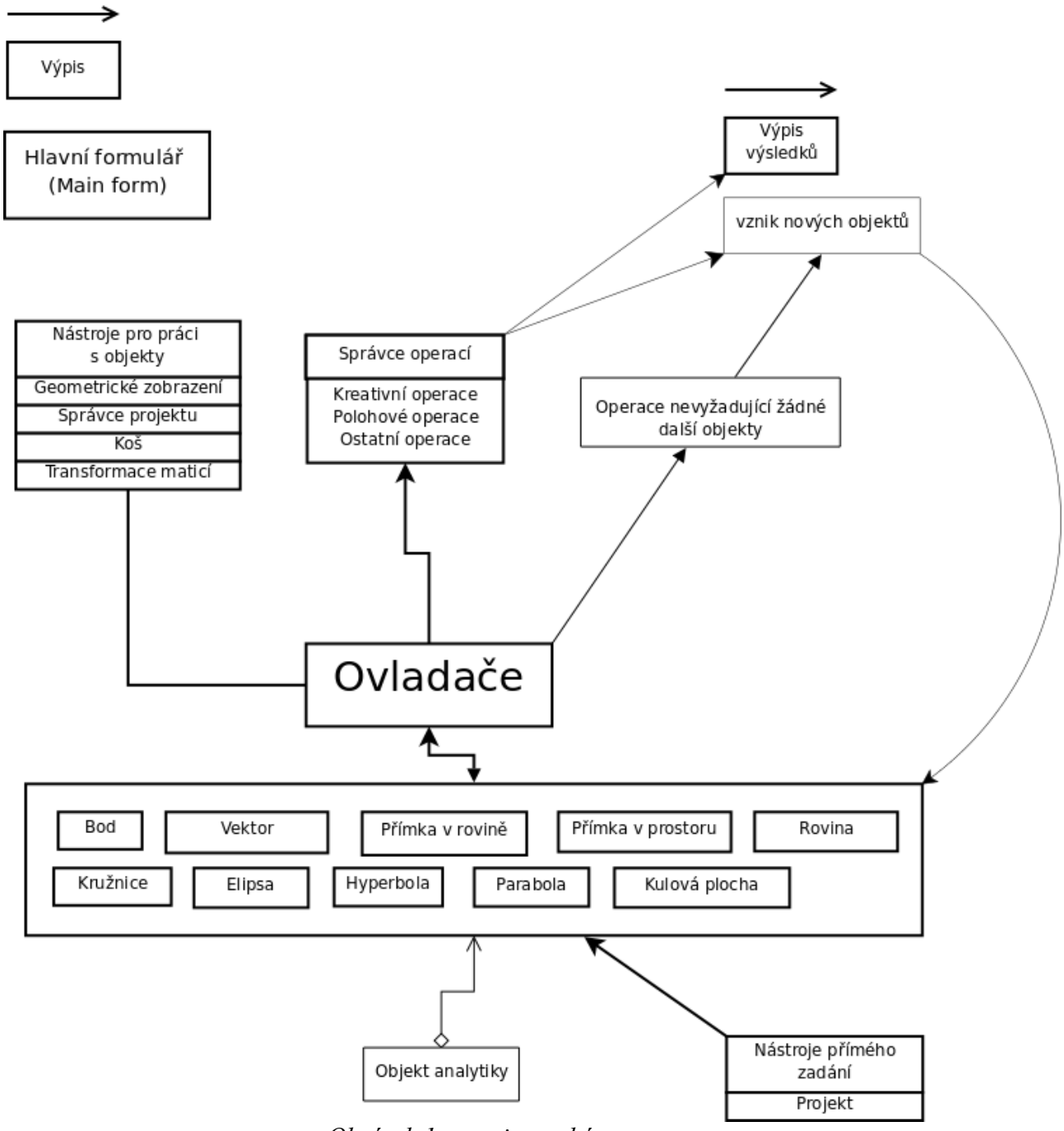
Nabízí se zde dvě možnosti již používané běžnými programy – formou vlastního nástroje vypisujícího nápovědu k prvku aktuálně vybraného uživatelem, nebo popisků (tzv. tooltipů) přímo v místě prvku. Obvykle programy tyto možnosti různě kombinují, avšak já jsem se rozhodl využít pouze druhé možnosti vzhledem k tomu, že další nástroj s nápovědou by byl zbytečným znepřehledněním pracovní plochy. Zbytek nápovědy již umístím standardním způsobem do samostatného souboru a dalších dokumentů nápovědy.

3.4.1.8 Možnost vrátit akci zpět

Tato možnost je již věcí uživatelského komfortu, nicméně o něj nechci uživatele ochudit. Jelikož objekty budou vznikat vždy nové a samostatné, je zde jedinou vratitelnou akcí smazání objektu. Pro tento požadavek se proto nejlépe hodí nástroj fungující na principu známého koše.

4 Návrh

Tato část je ve své podstatě nahlédnutím „pod slupku“ a její pročtení může i uživateli pomoci chápat děje probíhající v pozadí běhu programu, a vést ho tak k efektivnějšímu řešení úloh. Svůj návrh zde popíši rovnou na runtime schématu mého programu (Obrázek 1), které již obsahuje nástroje vyvozené v analýze.



Obrázek 1 - runtime schéma programu

Než se budu věnovat samotnému popisu běhu programu, vysvětlím prvky schématu, které se přímo neobjevily v analýze.

Ovladače – je to jedna z nejdůležitějších součástí programu, protože právě mezi ovladači a objekty se celý program dělí na knihovnu s matematickým jádrem a samotný program s GUI. Každý z ovladačů obsahuje veřejnou, ale mimo třídu nenastavitelnou proměnnou odkazující na objekt analytiky, který ovladač spravuje. Zpětná vazba, tedy z objektu na jeho ovladač, probíhá přes hlavní formulář, který obsahuje hashovací tabulku ovladačů; klíčem do tabulky je samotný objekt. Tuto oboustrannou vazbu jsem se nakonec rozhodl udělat nenarušitelnou a její vytvoření přenechat managerské části hlavního formuláře.

Ovladač má každý objekt, který má být viditelný pro uživatele (při operacích vznikají dočasné pomocné objekty apod.) Jeho hlavní funkcí je zobrazení objektu uživateli, je to tedy jeho reprezentace na úrovni GUI. Je to jeho nadřazený „pán“, hlídá si svůj objekt, mění ho a především jej spojuje se všemi ostatními třídami. Předává jej k zobrazení, přidání do projektu, při smazání také koši a především jej registruje správci operací. Jeden objekt může mít pouze jeden ovladač. Ovladač bez objektu zaniká a obráceně (je nemilosrdně smazán z paměti *Garbage Collectorem*).

MainForm – hlavní formulář stojí na schématu mimo, protože jeho funkce v programu je především managerská. Zprostředkovává vazbu mezi ovladači a nástroji i výše zmíněnou vazbu mezi objekty a jejich ovladači. Také jsem mu přenechal kompletní správu nástrojů. Nástroje vytváří, jsou z něj přístupné a hlídá, aby nebyl některý nástroj otevřen vícekrát apod. Jeho další funkce souvisí především s GUI programu: spravuje virtuální plochu a formuláře v ní.

Schéma popíši zespodu, kde běh programu začíná. Ve spodní části schématu je objektům nadřazená třída *Objekt analytiky* a třídy, jenž ji mají poděděny – jednotlivé objekty. Vedle něj stojí nástroj *Nástroje přímého zadání* a společně s projektem (pokud je zadání přeneseno souborem) jsou počátečními prvky, se kterými se uživatel setká. Vytvoří totiž prvotní objekty pro operace (při řešení úlohy slouží k tvorbě zadání). Nad těmito objekty stojí ovladače, které objekty vyobrazí uživateli a umožní mu provést s objektem, co zamýšlí (např. zobrazit jej do nákresu), popř. mu umožní přístup k jeho atributům a jejich zkopírování pro další práci (např. asymptota hyperboly, směrový vektor přímky). Ovladače také zpřístupňují operace, pro které nejsou potřeba další objekty, především pak dopočítávání bodů objektů. Nakonec se celé schéma zužuje k správci operací, ve kterém uživatel provede operace s více objekty – pokud to potřebuje. Po provedení operace vzniknou nové objekty a vypíší se výsledky. Kruh se uzavřel a celé schéma se provádí znovu zespodu. Mimo kruh stojí pouze výpis, který se děje celou dobu práce s programem.

Příloha D obsahuje diagram všech tříd a jejich veřejných funkcí (vnitřní mechanika nástrojů je již mimo přípustný rozsah tohoto dokumentu).

5 Popis programu

Za jeden z výsledků mé práce bych mohl považovat přímo samotný spustitelný program a samozřejmě je na místě tyto výsledky dokumentovat a sumarizovat. Těmto výsledkům dám formu zcela uživatelského pohledu na program s jeho hlavními možnostmi. Avšak i zdrojový kód (přes 8 000 negenerovaných řádků, 13 000 generovaných), který je v celé šíři okomentovaný, může být do jisté míry považován za jeden z výsledků mé práce.

5.1 Standardy v programu

V programu se ve většině případů zadávají čísla. Proto jsem implementoval tyto standardy (jak jsou běžné na kalkulačkách): zadávání zlomku ve tvaru čitatel/jmenovatel (např. „1/3“) a také složeného čísla ve tvaru celé_číslo/čitatel/jmenovatel (např. „2/3/4“). V místech, kde se zadávají stupně (pro rotaci apod.), se číslem zadávají stupně (např. „30“ pro 30°) a nebo se za číslo připíše „pi“ pro radiány. Tyto standardy je možné i kombinovat (např. „1/5pi“). Jakékoliv mezery jsou automaticky vyloučeny.

Dalším standardem je zaokrouhlování – veškeré hodnoty v ovladačích objektů jsou zaokrouhleny na tisíciny, v případě nutnosti přesnějšího výpisu má každý ovladač implementovanou možnost vypsání do výsledků, kde jsou všechny hodnoty včetně hodnot koeficientů obecných rovnic vypsány bez zaokrouhlení. Další výpis je pak možný přes atributy objektů (např. Střed kružnice).

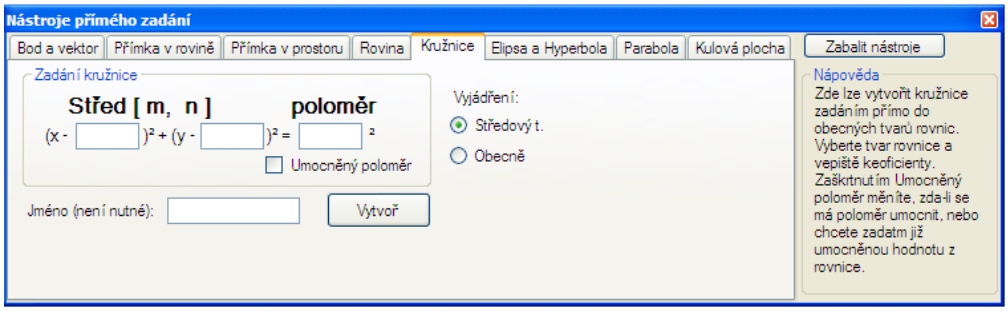
5.2 Hlavní okno

Primárně tento prvek (*MainForm*) zobrazuje virtuální pracovní plochu programu. Plocha se volně zvětšuje a opět zmenšuje dle objektů a nástrojů v ní zobrazených. Po virtuální ploše se lze pohybovat prostým uchopení a posunutím. Kromě toho má hlavní okno i další funkce – přes něj si program pamatuje nástroje otevřené při poslední práci s programem, pozice všech nástrojů a u některých i velikost. Také nástroje otevře kdykoliv by měl uživatel přijít o nějakou událost (např. když je vypsán výsledek a okno výsledku není otevřeno). V neposlední řadě přes kontextové menu pracovní plochy (myšleno programu) lze objekty srovnat do mřížky pro větší přehlednost.

5.3 Nástroje v programu

5.3.1 Nástroje přímého zadání

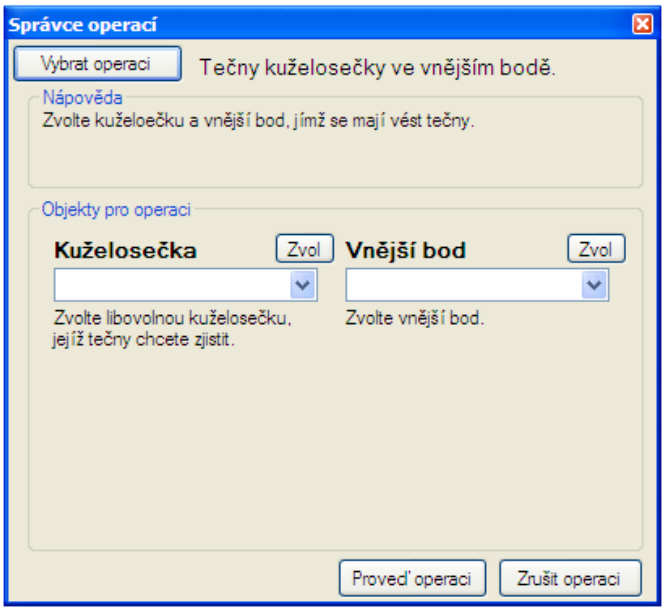
Nástroje lze zabalit, protože po vytvoření zadání úlohy již nemusí být bezprostředně použity. Všechny TextBoxy (vstupní pole) jsou opatřeny dodatečnými tooltipy (popisky) s popisem koeficientů. Jméno není nutné zadávat – bude vygenerováno automaticky. Zde jsem pro ilustraci vybral jednu ze záložek přímého zadání (Obrázek 2).



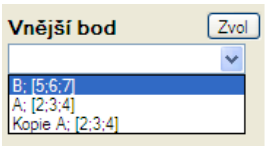
Obrázek 2 - Nástroje přímého zadání

5.3.2 Správce operací

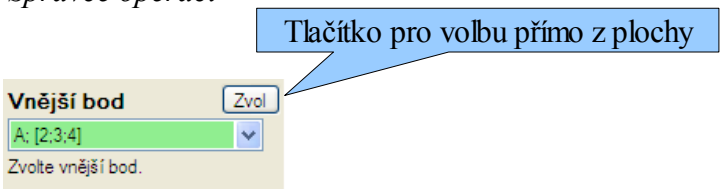
Prakticky nejdůležitější nástroj celého projektu. Při jeho tvorbě jsem se snažil zejména o intuitivní ovládání. Základem nástroje je rozsáhlé menu operací (schéma v příloze B). Po zvolení operace se objeví požadavky pro aktuální operaci a uživatel postupně zvolí z již existujících objektů ty, se kterými mají být provedeny operace. Buď tím, že objekt vybere ze seznamu, nebo přímo kliknutími na tlačítko *Zvol* a následně na objekt na pracovní ploše.



Obrázek 3 - Správce operací



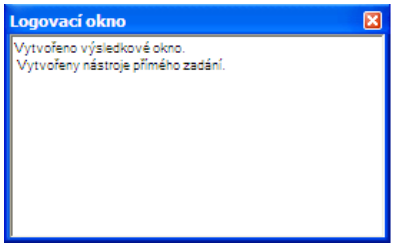
Obrázek 4 - výběr objektu pro operaci



Obrázek 5 - výběr objektu pro operaci 2

5.3.3 Výpis

Touto částí sloučím jak běžný výpis programu, tak i výpis výsledků, protože tyto nástroje jsou z uživatelského pohledu téměř identické. Při novém záznamu ve výpisu pozadí záznamového (logovacího) okna zabliká, aby byl o něm byl uživatel bezpečně informován.



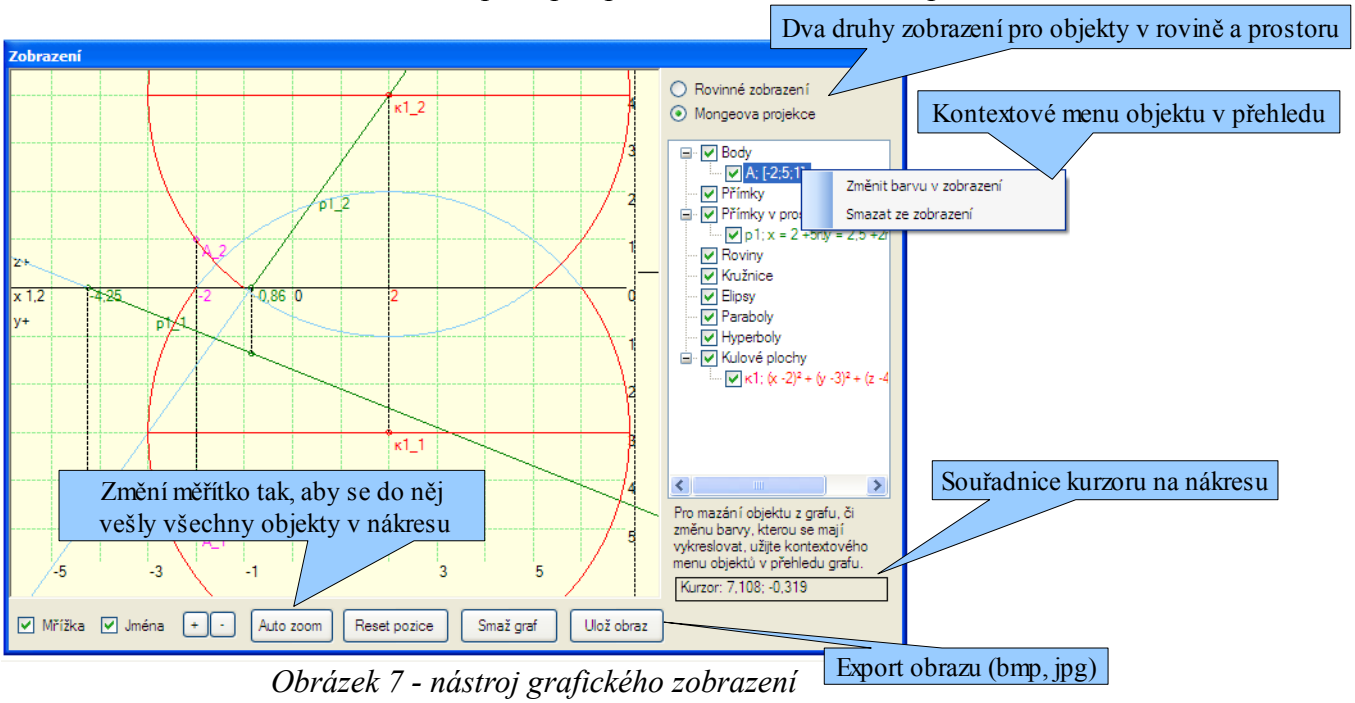
Obrázek 6 - záznamové (logovací) okno

5.3.4 Grafické zobrazení

Tento nástroj měl původně sloužit pouze jako doplněk, ale ukázal se jako velmi užitečný při jakékoliv úloze, proto jsem mu postupně implementoval další funkce, což spolu se schopností geometricky vykreslit všechny objekty způsobilo, že se tento nástroj stal technicky nejrozsáhlejším a také nejsložitějším.

Nástroj lze maximalizovat dvojitým kliknutím na jeho rámeček, tím vyplní celé hlavní okno (při prezentaci). Objektům v něm vyobrazeným lze změnit barvu využitím jejich kontextového menu v přehledu. Další funkce jsou také: aktuální souřadnice kurzoru, posouvání se po nákresu prostým uchopením, měnitelné měřítko atd.

Změna měřítka je podstatná a používaná funkce, což mě vedlo k vytvoření hned tří způsobů, jak změnu provést: stisknutím pravého tlačítka na myši a táhnutím ve vertikálním směru, tlačítkem *Auto zoom*, kterým se měřítko automaticky nastaví vzhledem k vyobrazeným objektům, a tlačítky „+“ a „-“. Doplňkovým způsobem je funkce postupného přibližování, která se provede dvojklikem do místa, ke kterému se uživatel chce postupně přibližovat; kliknutím se přibližování zastaví.

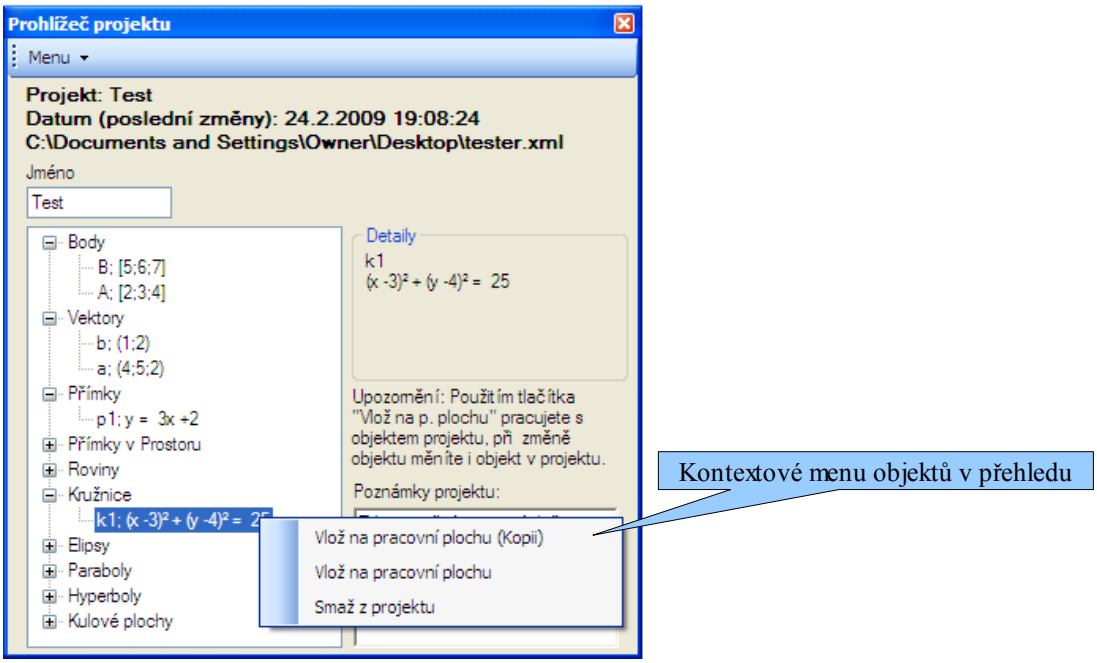


Obrázek 7 - nástroj grafického zobrazení

Pozn. Indexy objektů v Mongeově projekci jsou kvůli čitelnosti psány za podtržítkem.

5.3.5 Prohlížeč projektu

Nástroj *Prohlížeč projektu* (Obrázek 8) jsem vytvořil jako jeden z prvních hned po grafickém zobrazení, žádný program se totiž neobejde bez možnosti uchování dat. Přes kontextové menu lze objekty vložit na pracovní plochu (např. pro editaci), nebo zkopírovat pro další nezávislou práci. Jak je již zvykem v programech pracujících s vnějšími daty i zde je uživatel tázán na uložení projektu při zavření nástroje nebo programu. Soubory jsou bez jakéhokoliv kódování serializovány do souborů XML, a mohou tak být lehko upraveny a používány i mimo program.



Obrázek 8 - Prohlížeč projektu

5.3.6 Lineární transformace

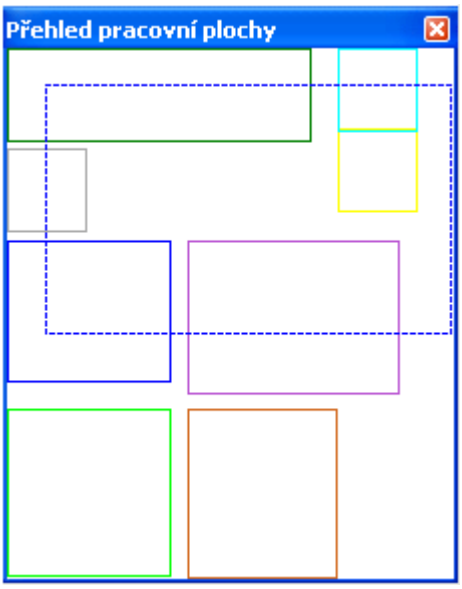
Možnost lineárně transformovat body a vektory maticí je již doplňková funkce, která přímo nespadá do domény analytické geometrie, proto jsem popis tohoto relativně složitého nástroje zařadil do přílohy C.

5.3.7 Koš

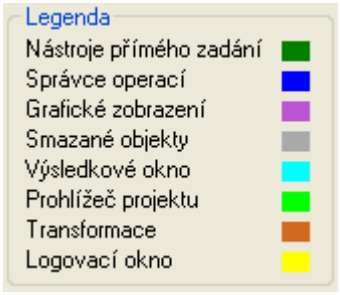
Nástroj, který jsem vytvořil velmi jednoduše. Po smazání objektu z pracovní plochy se objekt objeví v koši. Kliknutím na objekt v koši je objekt opět obnoven k používání. Koš lze přes kontextové menu vysypat.

5.3.8 Přehled pracovní plochy

Nástroj, se kterým jsem do analýzy nepočítal. Vznikl totiž na základě pozdější snahy celkového zpřehlednění programu. *Přehled pracovní plochy* zobrazuje plochu v měřítku a okna barevně rozlišuje, také reaguje na akce v oknech tím, že okno v přehledu zabliká. Při kliknutí na určitou část plochy v přehledu se pohled na plochu postupně přesune na toto místo. V okně lze také zobrazit legendu objektů. Okno je automaticky přichyceno k levému dolnímu rohu obrazovky.



Obrázek 10 - Přehled pracovní plochy

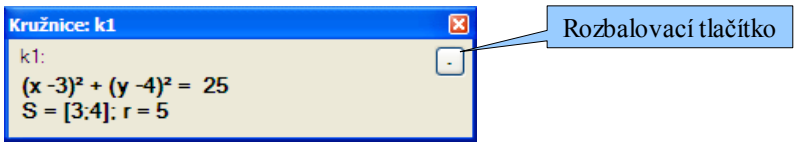


Obrázek 9 - legenda přehledu pracovní plochy

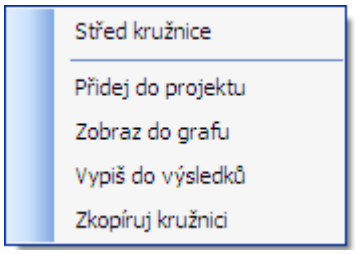
5.4 Ovladač objektu

Jednotlivé ovladače jsou velmi specifické pro dané objekty, proto jsem si vybral jeden vzorový. Vzorovým ovladačem bude ovladač kružnice se zadáním k1(S[3;4], r = 5).

Obrázek 11 vyobrazuje ovladač tak, jak jej uživatel uvidí při vytvoření takovéto kružnice. Obsahuje všechny důležité informace o kružnici a její analytickou rovnici.



Obrázek 11 - ovladač objektu



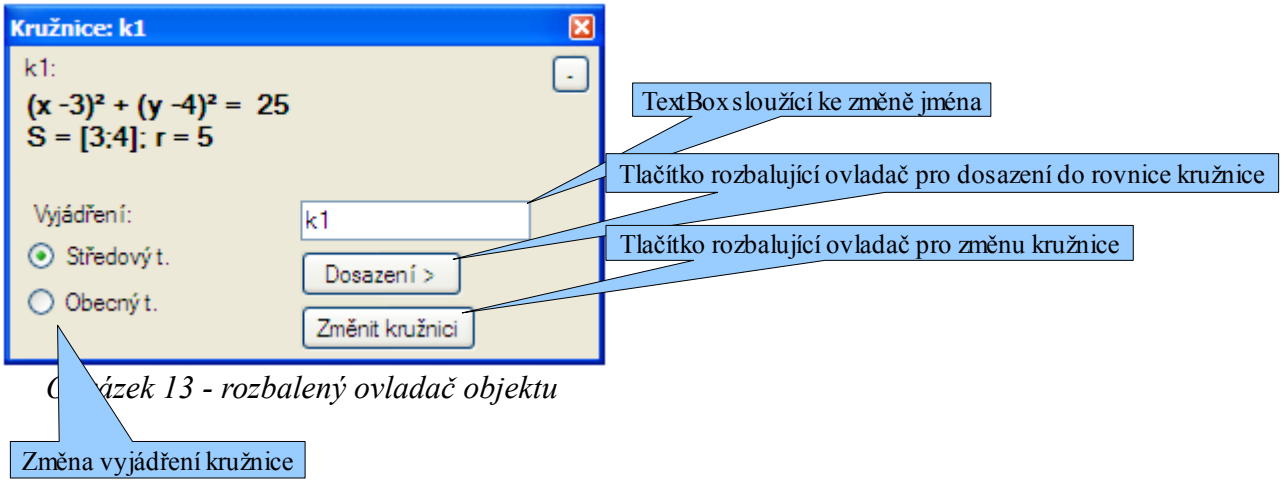
Obrázek 12 - kontextové menu ovladače objektu

Každý ovladač má své kontextové menu, které se vyvolá pravým kliknutím do prostoru ovladače. Přes toto kontextové menu lze provádět interakce s nástroji a také přistupovat k atributům objektů. Obrázek 12 vyobrazuje menu ovladače kružnice. Následuje krátký popis každé z položek menu.

Popis:

Střed kružnice – vytvoří nový nezávislý objekt Střed kružnice k1.
Přidej do projektu – předá objekt *Prohlížeči projektu*.
Zobraz do grafu – předá objekt grafickému zobrazení.
Vypiš do výsledků – vypíše objekt do výsledkového okna bez jakéhokoliv zaokrouhlení.
Zkopíruj kružnici – vytvoří nový, stejný, nezávislý objekt.

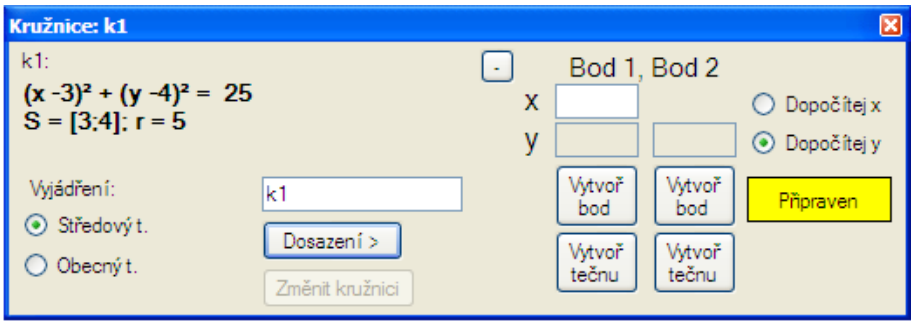
Po kliknutí na rozbalovací tlačítko na ovladači se ovladač rozbalí o další možnosti (Obrázek 13).



Obrázek 13 - rozbalený ovladač objektu

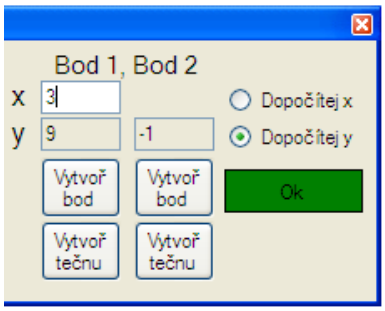
Při kliknutí na tlačítko *Změnit kružnici* se ovladač opět rozbalí o možnost změny hodnot v rovnici kružnice (podobné z jejího zadání v *Nástrojích přímého zadání*).

Po kliknutí na tlačítko *Dosazení* se ovladač rozbalí o možnost dopočítání bodů kružnice z její rovnice a následného vytvoření tohoto bodu jako samostatného objektu, popř. tečny kružnice v tomto bodě. (Obrázek 14-16)

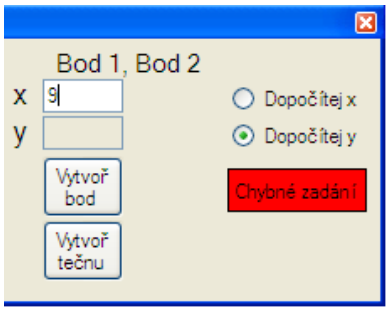


Obrázek 14 - ovladač rozbalený pro dosazení do rovnice

Po dosazení „3“ a „9“ jako x-ové souřadnice pro odpočítání bodu.



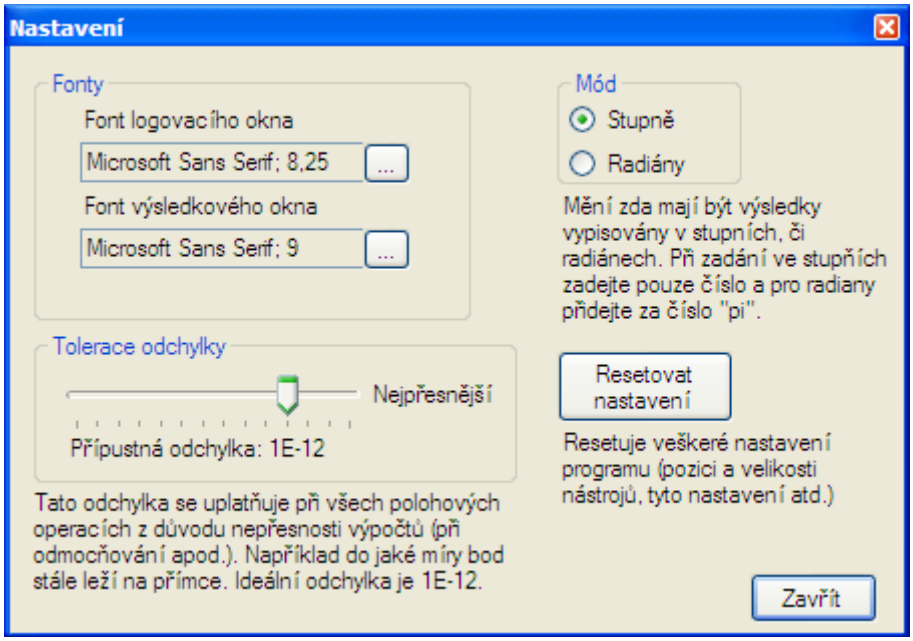
Obrázek 15 - dosazení do rovnice



Obrázek 16 - dosazení do rovnice 2

5.5 Nastavení programu

Nastavení programu je celé popsáno přímo na nastavovacím dialogu, proto uvedu pouze sejmutou obrazovku.



Obrázek 17 - okno nastavení programu

Pozn.: Běžně vzniklé odchylky jsou ve většině případu ošetřeny systémem odchylek, avšak někdy vznikají i odchylky, které jsou tak vysoké, že systémem odchylek nejsou zaznamenány až do přesnosti sta tisícin (povětšinou zpětné určení vytvořené tečny).

5.6 Klávesové zkratky

Pro funkce, které jsem hojně používal, jsem vytvořil i klávesové zkratky; zkratka se vztahuje na aktuálně aktivní okno v hlavním okně tj. má nezašedlý rámeček.

Zkratky:

–*Escape* – zavře libovolné okno (ať už nástroj, nebo smaže objekt analytiky)

–*Alt+Z* – vyobrazí objekt do geometrického zobrazeni

–*Alt+X* – přidá objekt do projektu

–*Alt+C* – otevře nebo označí nástroj *Přehled pracovní plochy*

6 Závěr

6.1 Souhrn

Záměrem projektu bylo vytvoření programu schopného počítat všechny úlohy s praktickým zadáním v analytické geometrii na středoškolské úrovni. Tohoto požadavku se podle mého názoru povedlo dosáhnout. Program byl také rozšířen o mnoho dalších možností usnadňujících práci a zlepšujících představu samotných úloh. Myslím si, že mnou vytvořené řešení je technicky dobře zpracované, a proto snadno rozšiřitelné o nové možnosti, přičemž samotné matematické jádro lze použít i pro další práci.

6.2 Summary

Aim of this project was to create a program able to count any task with practical submission in high school analytic geometry. This requirement was in my opinion reached. The program has also been advanced with many other accessibility features and features improving image of whole tasks. I think, that my solution is on good technical level, easy to expand with new features and mathematical kernel is usable for other future work.

6.3 Využití projektu

Projekt byl navržen tak, aby splňoval požadavky jak studentů, tak pedagogů při ověřování výsledků např. domácích úloh nebo písemných prací. Také by mohl být využit přímo při výuce v kombinaci s interaktivními tabulemi, které se stále více uplatňují téměř ve všech předmětech. Využití programu všeobecně nelze přesně vymezit, protože samotná analytická geometrie má velmi rozsáhlé užití při řešení velkého množství úloh.

Praktičnost využití byla potvrzena nasazením projektu pedagogům Gymnázia Jiřího Wolkera v Prostějově vyučujícím analytickou geometrii, kteří byli s využitím velmi spokojeni, především po stránce produktivity práce v programu a jednoduchosti samotného řešení GUI.

6.4 Budoucnost

I přesto, že je program odladěný, nelze jej považovat za zcela dokončený. Stále pracuji na usnadnění práce v programu, tvorbě nápovědy společně s uživatelskou příručkou.

Další kroky projektu se budou ubírat k nasazení projektu pedagogům, dle jejich rozhodnutí i studentům a možnému přímému zapojení do výuky na GJW Prostějov. Posléze chci tuto možnost nabídnout i dalším středním školám a gymnáziím. Dále bych rád vytvořil informační stránky projektu a plnil případné požadavky (*feedback*) pedagogů a dalších uživatelů.

Z hlediska dalšího technického řešení projektu, se zde nabízí vytvoření vlastního grafického prostředí, tedy nahrazení standartu MDI, což by mohlo pomoci lepšímu přizpůsobení GUI programu k jeho účelu. Také se nabízí rozšíření programu o nové druhy geometrických zobrazení, popř. i 3D modulu pro objekty v prostoru. Dle mého názoru ale tyto nové možnosti již vyžadují rozšíření vývojového týmu, aby bylo dosaženo potřebné dynamiky vývoje projektu.

7 Seznam literatury a internetových zdrojů

Zde uvádím seznam knih, ze kterých jsem vycházel pro teoretický základ svého projektu.

KOČANDRLE, Milan; BOČEK, Leo. *Matematika pro gymnázia Analytická geometrie*. 3. vyd. Praha: Prometheus, 2008. 220 s. ISBN 978-80-7196-163-5

POLÁK, Josef. *Přehled středoškolské matematiky.* Praha: Prometheus, 2008. 659 s. ISBN 978-80-7196-356-1

Při realizaci projektu jsem se omezil pouze na internetové zdroje, avšak nemohu zde uvést všechny konkrétní články a příspěvky na fórech, ze kterých jsem čerpal při implementaci, nicméně uvádím výčet mnou nejnavštěvovanějších webových stránek.

Microsoft Developer Network. c2009. URL: <http://msdn.microsoft.com>

CodeProject. c2009. URL: <http://www.codeproject.com>

Programujte. c2009. URL: <http://www.programujte.cz>

Programmers Heaven. c2009. URL: <http://www.programmersheaven.com>

C# Station. c2009. URL: <http://www.csharp-station.com>

Wikipedia. c2009. URL: <http://www.wikipedia.org>

8 Použitý software

Microsoft Visual Studio 2008 Express editon – vývoj programu.
URL: <http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx>

knihovny .NET Framework (2.0 – 3.5 SP1) – vývoj programu.
URL: <http://msdn.microsoft.com/en-us/netframework/default.aspx>

OpenOffice 3.0 – tvorba dokumentace. URL: <http://www.openoffice.org/>

DiaCze (Gtk) – tvorba diagramů. URL: <http://www.cze.cz/>

Model Maker 10 – tvorba API dokumentace, UML digramů (reverse engineering).
URL: <http://www.modemakertools.com>

GIMP (Gtk) – drobná grafika (sejmuté obrazovky apod.).

Adobe Acrobat Reader 8 – práce s formátem pdf. URL: <http://get.adobe.com/reader/>

Pozn.: Platnost všech URL byla ověřena dne 3. Dubna 2009.

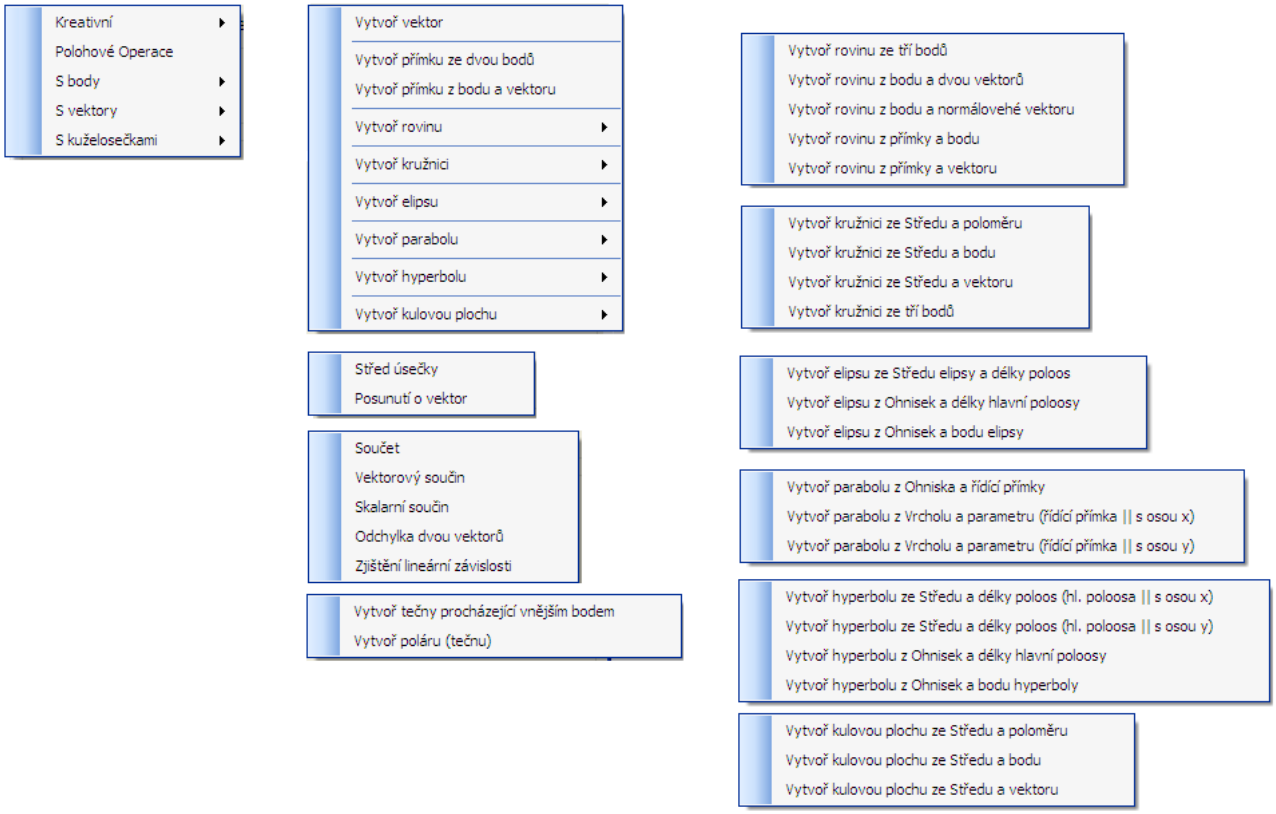
9 Přílohy práce

A. Polohové operace

Zde jsou zapsány všechny možnosti a scénáře polohových operací.

- bod, bod – body se porovnají a zjistí se jejich vzdálenost
- přímka v rovině, bod – v případě, že bod leží na přímce, určí se parametr, v obráceném případě se určí vzdálenost bodu od přímky
- přímka v rovině, přímka v rovině – určuje se průsečík přímek, v případě že jsou přímky různoběžné určí se i jejich odchylka, v případě rovnoběžnosti se určuje jejich vzdálenost
- přímka v prostoru, bod – v případě, že bod leží na přímce, určí se parametr, v obráceném případě se určí vzdálenost bodu od přímky
- přímka v prostoru, přímka v prostoru – určuje se průsečík přímek, v případě různoběžnosti se určuje i odchylka, v případě rovnoběžnosti vzájemná vzdálenost, v případě mimoběžnosti se neprovádí žádná další operace
- rovina, bod – v případě, že bod leží v rovině, určí se oba jeho parametry, v obráceném případě vzdálenost bodu od roviny
- rovina, přímka v prostoru – určuje se průsečík roviny s přímkou, pokud nejsou rovnoběžné určí se též odchylka, v případě rovnoběžnosti vzájemná vzdálenost
- rovina, rovina – určuje se průsečnice, v případě různoběžnosti i odchylka, v případě rovnoběžnosti vzájemná vzdálenost
- kuželosečka (kružnice, elipsa, hyperbola, parabola), bod – určuje se zda bod leží na kuželosečce, u kružnice i vzdálenost od středu
- kuželosečka, přímka v rovině – určuje se jeden nebo dva průsečíky
- kulová plocha, bod – určuje se zda bod leží na kulové ploše, v případě, že neleží určuje se vzdálenost od středu kulové plochy
- kulová plocha, přímka v prostoru – určuje se jeden nebo dva průsečíky
- kulová plocha, rovina – určuje se průsečík v případě, že je rovina tečnou rovinou, jinak vzdálenost roviny od středu kulové plochy

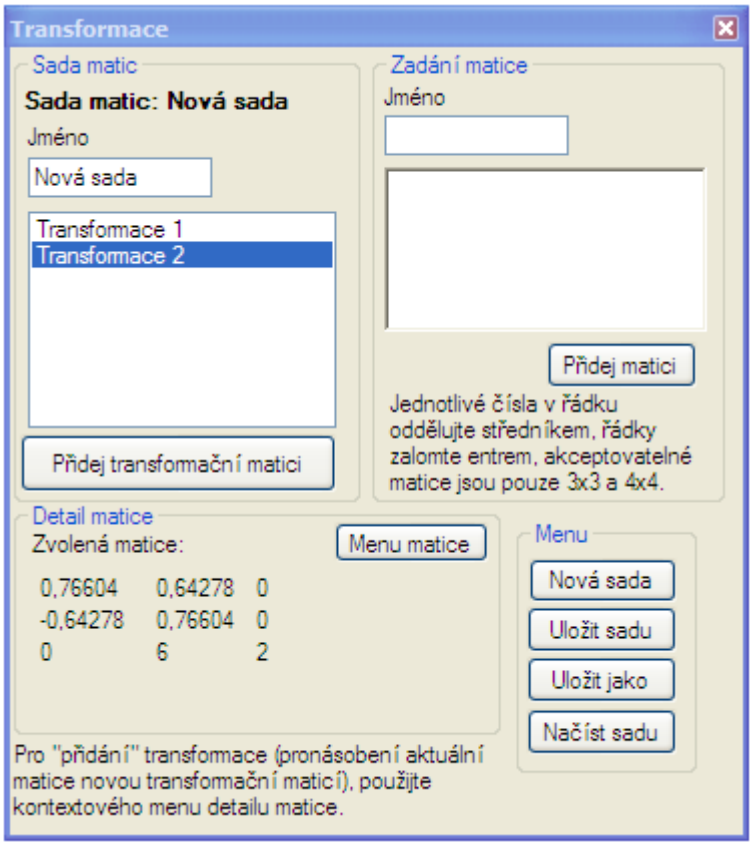
B. Schéma menu operací



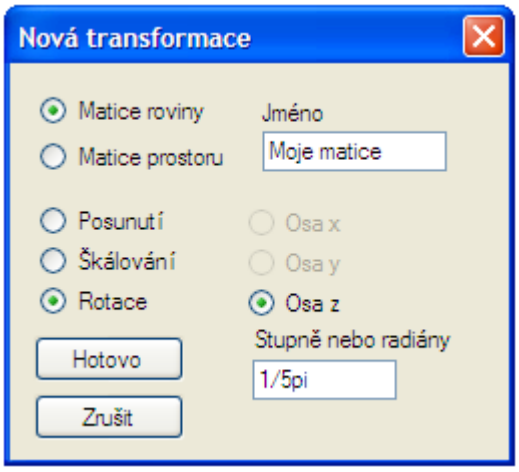
Obrázek 18 - schéma menu operací

C. Nástroj lineárních transformací

Zde je pro ilustraci uvedeno hlavní okno maticových transformací (Obrázek 19), na němž je patrná i jeho hlavní funkce – možnost vytvářet různé transformační matice, ty lze posléze násobit dalšími transformacemi, a vytvářet tak plně transformované matice. Matice lze zadávat i ručně, lze je pojmenovávat a ukládat do celku tzv. *Sady matic* a tuto sadu ukládat do samostatného souboru typu XML. Samotná transformace se poté provede přes kontextové menu bodů a vektorů.



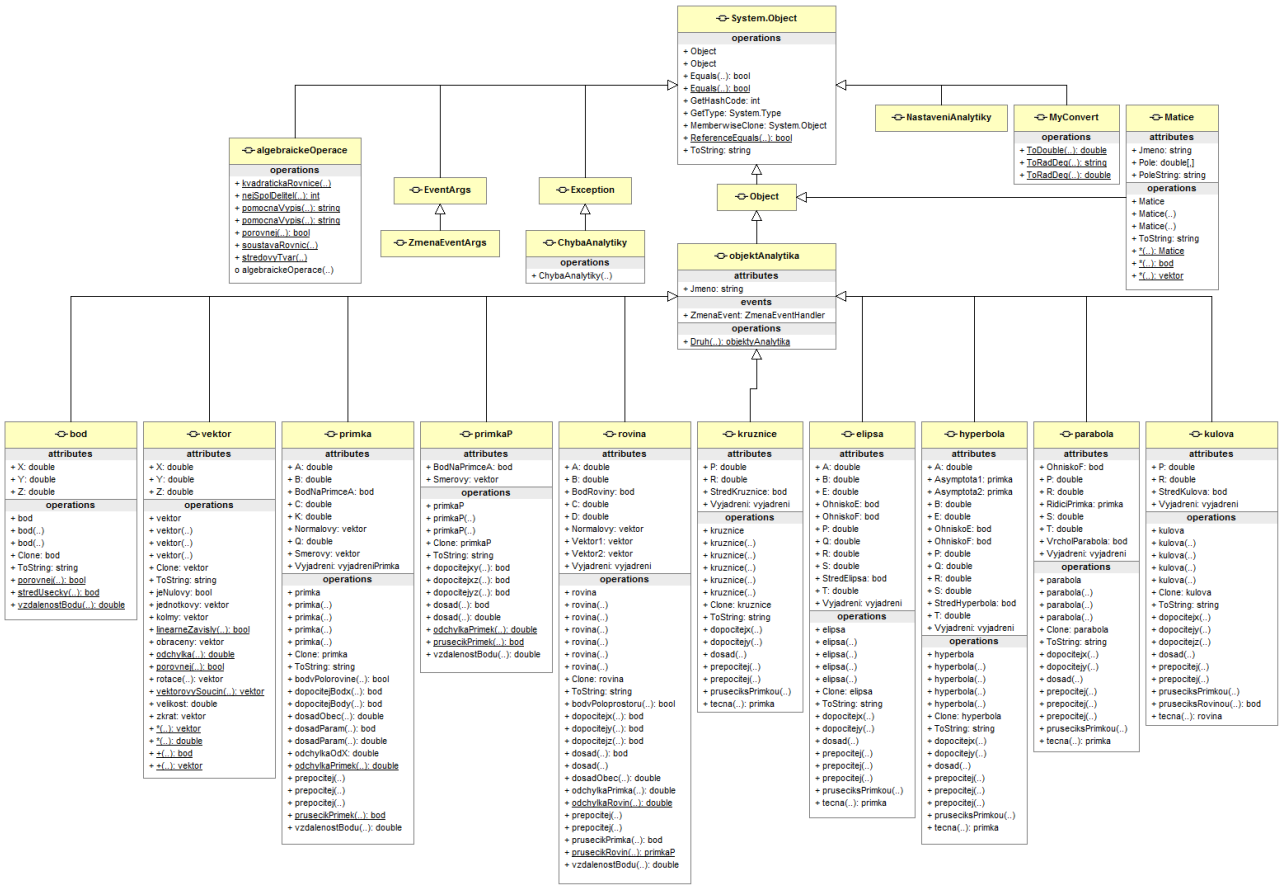
Obrázek 19 - okno nástroje lineárních transformací



Obrázek 20 - výběr nové transformace

D. Diagramy tříd

Digram tříd matematického jádra a jejich veřejných metod (API).



Digram tříd samotného programu s GUI a jejich veřejných metod (API). Na schématu nejsou vyobrazeny ovladače objektů (kromě spravovaného objektu nemají žádnou veřejnou proměnnou ani metodu).