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

CAD SYSTÉM PRO 2D KRESLENÍ

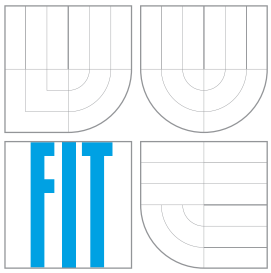
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ HÜBNER

BRNO 2010

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

CAD SYSTÉM PRO 2D KRESLENÍ

CAD SYSTEM FOR 2D DRAWING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ HÜBNER
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. PŘEMYSL KRŠEK, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Bakalářská práce se zabývá charakteristikami počítačem podporovaného projektování a specifikuje typické metody kreslení používané v CAD systémech. Praktickou část tvoří program pro 2D kreslení, který implementuje některé z těchto základních metod. V textu je popsán návrh a tvorba této aplikace s pomocí knihoven Qt a Open Scene Graph a následné zhodnocení jejích možností.

Abstract

This bachelor's thesis considers the characteristics of a computer aided design. It specifies the typical methods used for drawing in this sort of programs. Practical part is an aplication for 2D drawing, which implements some of the specified methods. There is described a concept and an implementation of the application with Qt and Open Scene Graph libraries.

Klíčová slova

CAD, vektorová grafika, 2D kreslení, Open Scene Graph

Keywords

CAD, vector graphics, 2D drawing, computer aided design, Open Scene Graph

Citace

Lukáš Hübner: CAD systém pro 2D kreslení, bakalářská práce, Brno, FIT VUT v Brně, 2010

CAD systém pro 2D kreslení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Přemysla Krška, Ph.D.

.....
Lukáš Hübner
19. května 2010

Poděkování

Děkuji vedoucímu mé práce panu Doc. Ing. Přemyslu Krškovi, Ph.D. za cenné připomínky a za nasměrování mých kroků k úspěšnému dokončení tohoto projektu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Lidé po celém světě se stále častěji setkávají s různými formami elektroniky. Výpočetní technologie postupně proráží, nebo již prorazily, do různých odvětví lidské činnosti. Odvětvím, které moderní technologie a počítače jistě výrazně poznamenaly, je obor projektování a technických návrhů. Ať už jde o architektonické studie domů, celých měst nebo o detailní návrhy nejmenších strojních součástek, většina jejich autorů si již dnes nedokáže představit jejich tvorbu bez využití moderních softwarových prostředků. Prakticky téměř každý výrobek prochází fází počítačového návrhu, nebo-li CAD (Computer Aided Design).

Základním technickým vyjadřovacím prostředkem jsou již od pradávna kreslené technické návrhy, které postupem času od uhlových skic přes inkoustem kreslené výkresy dospěly až do dnešní digitální podoby produkované počítačovými programy. Co vedlo k tomuto vývoji? Co bylo jeho příčinou? Bezesporu to byla vždy snaha navrhovat přesněji, rychleji a jednodušeji než dříve. Dnešní CAD systémy navazují na tento vývoj a dále zefektivňují proces projektování jak pomocí sofistikovaných prostředků pro kreslení, tak pomocí počítačem realizovaných výpočtů a simulací. Hlavním výrazovým prvkem technických projektů však stále zůstávají „nákresy“ (ať již dvourozměrné či 3D) a hlavním cílem systémů pro projektování zůstává zjednodušování jejich tvorby.

Cílem této práce je charakterizovat klíčové principy a funkce používané v CAD systémech pro podporu kreslení. Dále pak bude popsán návrh a implementace aplikace, která bude umožňovat kreslení dvourozměrných výkresů s využitím základních funkcí využívaných při počítačem podporovaném projektování.

Kapitola 2

Teoretická analýza

2.1 CAD - Počítačem podporované projektování

Každého jistě napadne široké spektrum odvětví, ve kterých počítačové systémy pomáhají s projektováním. Záběr CAD systémů je skutečně velký a existuje mnoho specializovaných programů, určených pro využití v specializovaných oborech. Jako příklady různých specializací lze uvést:

AEC - Architecture, engneering and construction Tyto aplikace jsou určeny pro architektonické návrhy, projektování budov a jiných staveb. Vzhledem k charakteru navrhovaných produktů bývají orientovány na návrh v 3D prostoru a bývají vybaveny knihovnami s předdefinovanými prvky, pro navrhování z normalizovaných částí (dveře, okna apod.).

CAM - Computer-aided manufacturing Systémy CAM se nezabývají pouze projektováním, ale zasahují i do samotného procesu výroby. Výsledný projekt pak může být zpracován obráběcím strojem (např. CNC frézou), která je na základě dat z návrhu schopna vyrobit např. dokonale přesnou strojní součástku.

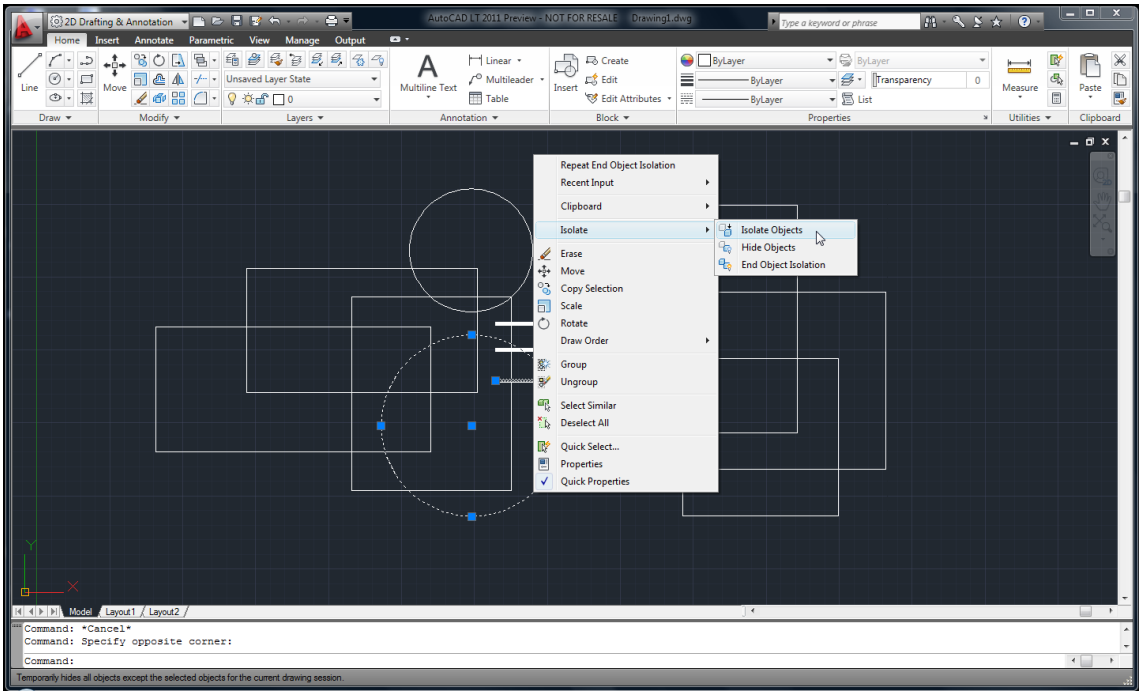
EDA - Electronic design automation Systémy pro návrh elektrických obvodů a součástek. Tyto programy jsou často spojeny se simulační částí, která je schopna simulovat chování navrženého obvodu.

GIS - Geographic information systems Geografické informační systémy a programy pro návrh map a plánů s CAD systémy velmi úzce souvisí, jejich využití je však spojeno s mnoha dalšími odvětvími informačních technologií.

Existuje mnoho dalších členění a specializací CAD systémů, není však cílem této práce je všechny obsáhnout. Uvedený seznam je zejména pro ilustraci toho, jak široký význam může mít pojem CAD.

2.2 Nejrozšířenější CAD systémy

Nabídka software pro počítačem podporované projektování se stále rozrůstá. Jsou vyvíjena rozšíření, knihovny i nové specializované programy. Mezi nejvýznamnější firmy na trhu s CAD software patří bezesporu firma Autodesk, jejíž produkt AutoCAD je uváděn jako



Obrázek 2.1: Obrazovka programu AutoCAD 2011 od společnosti Autodesk

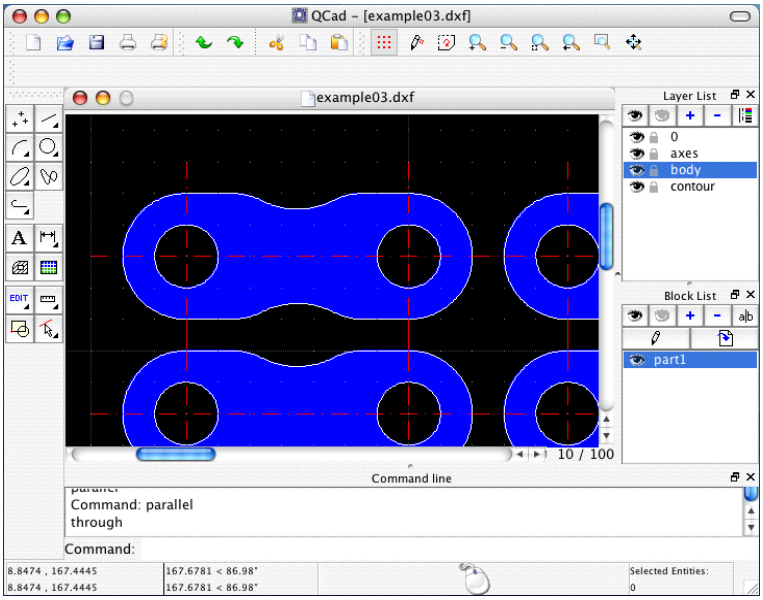
nejpoužívanější software tohoto druhu na světě. Program AutoCAD je považován za jakýsi nepsaný standard a např. podpora jeho formátu souborů je nezbytnou součástí všech rozsáhlejších CAD aplikací. V poslední době se také objevují volně šiřitelné, nebo dokonce „Open Source“ programy jako např. QCAD nebo Google SketchUp, které však zdaleka nedosahují kvality a rozsahu komerčních aplikací.

2.3 Charakteristiky CAD systémů

Přestože okruh použití těchto programů je velmi široký, existuje poměrně mnoho vlastností, které jsou pro všechny společné. V dnešní době se s CAD systémy setkáváme prakticky výhradně ve formě čistě softwarových produktů. Jejich použití většinou nevyžaduje speciální hardware, nebo jiné vybavení a běžně lze tyto programy provozovat na dnešních osobních počítačích. Přestože se pomocí CAD systémů daří výrazně usnadnit návrh určitého produktu, kvalifikovaný člověk jako obsluha je stále nezbytnou součástí projektování. Prostředkem pro obsluhu programů bývá často obyčejná myš, výstupem obraz na monitoru nebo tisk z tiskárny.

Většina CAD systémů vychází z principů vektorových grafických editorů. Kreslení technických výkresů má však svá určitá specifika, která je třeba respektovat a přizpůsobovat jim chování programu.

Přesnost Mezi nejdůležitější vlastnosti softwaru pro projektování patří bezesporu přesnost. Jestliže kreslíme např. omalovánky pro děti, není třeba se zabývat tím, zda bude mít kolo auta o milimetr navíc, důležitý je celkový vizuální dojem, který bude obrázek budit. Naopak, pokud vytváříme projektový výkres toho samého automobilu, hraje každý detail svou roli. Takový výkres pak může být použit např. dodavatelem náhradních dílů a i sebemenší nepřesnost může mít za následek výrobu nepoužitelných



Obrázek 2.2: Open source multiplatformní program QCAD spuštěný v Mac OS X

součástek. U technických výkresů není důležitá jejich vizuální stránka, ale přesnost a jednoznačnost informace, kterou nesou, proto je nutné, aby CAD systém umožňoval velmi přesný návrh (kreslení).

Podpora návrhu, realizace výpočtů, simulace Přesnost kreslení, která je při projektování nutná, často nejde ruku v ruce s jednoduchostí návrhu, proto tyto systémy poskytují funkce pro jeho usnadnění. Mezi ně bychom mohli zařadit uchopování významných bodů geometrických prvků, výpočty ploch, některé systémy umožňují i provádění simulací (např. crash testy automobilů).

Knihovny předdefinovaných prvků K dalšímu usnadnění návrhu pomáhají nejrůznější knihovny normalizovaných součástek a jiných opakujících se prvků. Objekty ze zakoupených nebo jinak získaných knihoven je možné využívat ve vlastních projektech a není třeba je „znovunavrhovat".

2.4 Vektorová grafika

Ve vektorové grafice je obraz tvořen pomocí geometrických entit. Škála těchto entit může být velmi široká, od primitivních geometrických tvarů až po mnohoúhelníky, křivky apod. V případě CAD systémů mohou být složitějšími entitami různé normalizované součástky a speciální prvky výkresů, např. kóty. Entity mohou mít různé vlastnosti jako barvu čáry, pozadí, typ čáry atd. U programů orientovaných na projektování jsou vlastnosti často přizpůsobeny technickým normám. V pokročilých CAD systémech dále mohou jednotlivé entity nést i jiné informace než pouze ty, týkající se vzhledu. Může se jednat např. o fyzikální, chemické či elektronické vlastnosti, které jsou dále použity při simulacích a výpočtech.

2.5 Specifika CAD kreslení

V následující části se pokusím popsat některá důležitá specifika, která sebou nese kreslení při počítačem podporovaném projektování.

2.5.1 Přesnost

Nejdůležitější a nejtypičtější pro kreslení technických výkresů je přesnost. Při návrhu jakéhokoliv výrobku nám záleží na jeho konkrétních rozměrech, na základě kterých je potom vyráběn on sám i další komponenty, které jsou na něm závislé. Je tedy nezbytné, aby CAD systém poskytoval možnosti pro velmi přesné kreslení, avšak zároveň tím co nejméně komplikoval práci projektanta. Mezi základní vlastnosti podporující přesné kreslení patří možnost zadávání souřadnic bodů, vzdáleností, poloměrů kružnic apod. Mezi další můžeme zařadit speciální módy kreslení a uchopování významných bodů a přichytávání k mřížce. Tyto charakteristické vlastnosti budou podrobněji rozebírány dále.

2.5.2 Uchopování

Uchopování je využíváno v případě kreslení pomocí myši. Pokud se kurzor myši nachází v blízkosti určitého významného bodu, pracuje se při události myši se souřadnicemi tohoto významného bodu a nikoliv se souřadnicemi, na kterých se nachází kurzor. Kurzor se k těmto bodům tedy jakoby „přichytává" nebo „uchopuje". Tato vlastnost je velmi důležitá pro podporu rychlosti a jednoduchosti návrhu. Není třeba ručně dopočítávat souřadnice specifických bodů jako jsou např. koncové body, středy nebo průsečíky, ale tyto body jsou uživateli automaticky nabídnuty. Mezi nejdůležitější body, které by měly být „uchopitelné" bezesporu patří:

- koncové body
- středy
- průsečíky

Uchopování se nemusí týkat pouze bodů, může se jednat také o jisté usměrnění právě kreslené entity. Jako příklad můžeme uvést kreslení tečné čáry ke kružnici. Kreslíme-li čáru dvěma body, známe polohu prvního bodu, ale chceme aby druhý bod byl tečný ke kružnici či oblouku, v takovém případě je na místě, když program při zadávání druhého bodu čáry nabídne polohu druhého bodu tak, aby výsledná úsečka byla tečná ke kružnici. Uchopování v průběhu kreslení je užitečné zejména pro:

- tečny
- kolmice

2.5.3 Módy kreslení

Program může také nabízet různé speciální módy kreslení. Mezi takové bychom mohli zařadit např. kreslení pouze ve směru souřadných os. Další možností je přichytávaní kurzoru k mřížce, nebo-li posun vždy o určitý nastavený krok. Tato vlastnost je využitelná např. pokud víme, že návrh bude v přesnosti na jednotky milimetrů a můžeme tedy nastavit minimální krok kurzoru na 1mm.

2.5.4 Entity

Kromě grafických primitiv jako je čára, kružnice, oblouk nebo křivka se v technických výkresech objevují různé specifické grafické entity. Typickou CAD entitou jsou například nejrůznější kóty, popisky nebo šrafování.

2.5.5 Možnosti zadávání entit

Zadávání a vkládání entit se většinou nespokojí pouze s kreslením pomocí myši. Je třeba aby program poskytoval možnosi zadání přesných souřadnic významných bodů entity nebo její jiné charakteristické vlastnosti (délka, odchylka atd.). Jednu entitu je proto často možné nakreslit hned několika způsoby, z nichž každý je vhodný pro jinou situaci a umístění entity.

Zadávání čáry

Čára je nejzákladnějším geometrickým prvkem a je při kreslení velmi často používána. Program pro počítačem podporovaný návrh by měl umožňovat kromě jejího zadání pomocí koncových bodů i jiné způsoby zadání, vhodné pro určité konstrukční případy. Vhodná je např. možnost zadání pomocí počátečního bodu, směru a délky úsečky. Pro kreslení os symetrických tvarů může být užitečné zadání pomocí středu, délky a směru čáry. Pro kreslení obrysů je vhodné mít k dispozici navazovanou „multičáru".

Zadávání kružnice

Základem pro zadávání kružnice je bezesporu její střed a poloměr. Existuje však několik dalších způsobů zadání kružnice, které mohou být pro projektování více či méně užitečné, např. zadání dvěma či třemi body, určení pomocí tečen atd.

Zadávání oblouku

Oblouk se zpravidla v technických výkresech vyskytuje poměrně často a jeho využití může být různé. Mezi nejčastější případy výskytu patří zaoblené hrany předmětů. Program by měl tedy, kromě základních možností kreslení oblouků zadáním poloměru a úhlů, umožňovat kreslení tečných oblouků nebo zaoblování v místě průsečíku dvou entit.

2.5.6 Hladiny

Entity jsou v CAD systémech umísťovány do hladin, nebo-li vrstev. Každá hladina většinou představuje určitou část nebo složku výkresu, má svou barvu, typ čáry a může být jednoduše zobrazena nebo skryta. Například v případě kompletního návrhu rodinného domu jsou různé složky návrhu kresleny do různých hladin, jsou vytvořeny hladiny pro zdivo, rozvody vody, elektřiny, plynu atd. Výsledný projekt pak představuje ucelený návrh, ale pomocí jednoduchého zobrazení či skrytí příslušných hladin je možné z něj vyčlenit informace důležité pro určité odvětví.

2.6 Interakce s ostatními programy

Bezesporu důležitým faktorem CAD systému je možnost jeho interakce s ostatními CAD programy. Z tohoto důvodu je vhodné, aby program umožňoval uložení dat do formátu,

který je „čitelný“ širokou škálou programů podobného zaměření. Zřejmě nejvhodnější k tomuto účelu je formát DXF (Drawing Exchange Format), vyvinutý firmou Autodesk pro výměnu dat s aplikacemi jiných výrobců. Formát DXF podporuje značná část současných CAD systémů.

Kapitola 3

Návrh a implementace

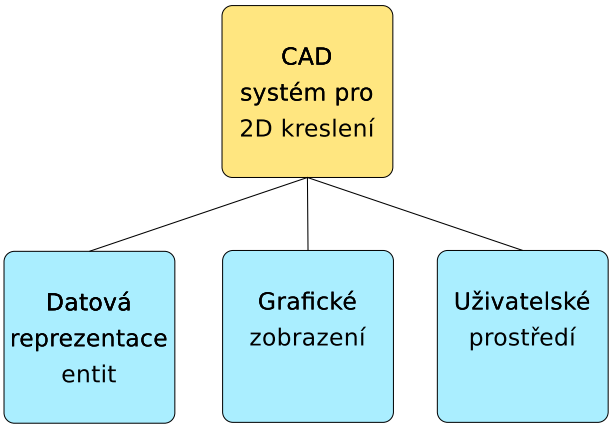
V zásadě se jedná o návrh programu pro vektorové 2D kreslení, avšak aplikace by měla zahrnovat důležité prvky CAD systémů, uvedené v kapitole věnující se teoretické analýze. Mělo by být také možné aplikaci v budoucnu co nejjednodušeji rozšířit o další funkce. Stejně tak je vhodné počítat i s rozvinutím pro prostorové 3D modelování.

3.1 Komponenty aplikace

Aplikace CAD systému se skládá z několika základních částí, ty navzájem komunikují a prolínají se. Z hlediska návrhu je možné CAD systém pro kreslení rozdělit do následujících základních komponent:

- datová reprezentace (grafických) entit
- grafické zobrazení výkresu (scény)
- uživatelské prostředí

V následující části budou popsány základní úlohy a vlastnosti těchto komponent.



Obrázek 3.1: Hlavní komponenty CAD systému pro 2D kreslení

3.1.1 Datová reprezentace entit

Tato komponenta představuje hlavní jádro programu. Jejím účelem je uchovávat data o vytvořených entitách a umožňovat práci s nimi. Mezi základní funkce, které by měla tato část programu provádět, patří:

- uchovávání dat pro zobrazovací část
- ukládání (načítání) dat do (ze) souboru
- výpočty nad grafickými entitami (průsečíky, uchopovací body atd.)

Základním stavebním prvkem této části programu budou jednotlivé grafické objekty a většina funkcí bude implementována metodami právě těchto objektů.

3.1.2 Grafické zobrazení výkresu

Jedná se o část, která převádí datové struktury grafických entit do formy „čitelné“ pro člověka, v tomto případě do formy obrazových dat. Jde o nejdůležitější součást grafického uživatelského prostředí, která má zejména tyto úkoly:

- zobrazovat grafické entity
- umožňovat uživatelský vstup myši (kreslení)

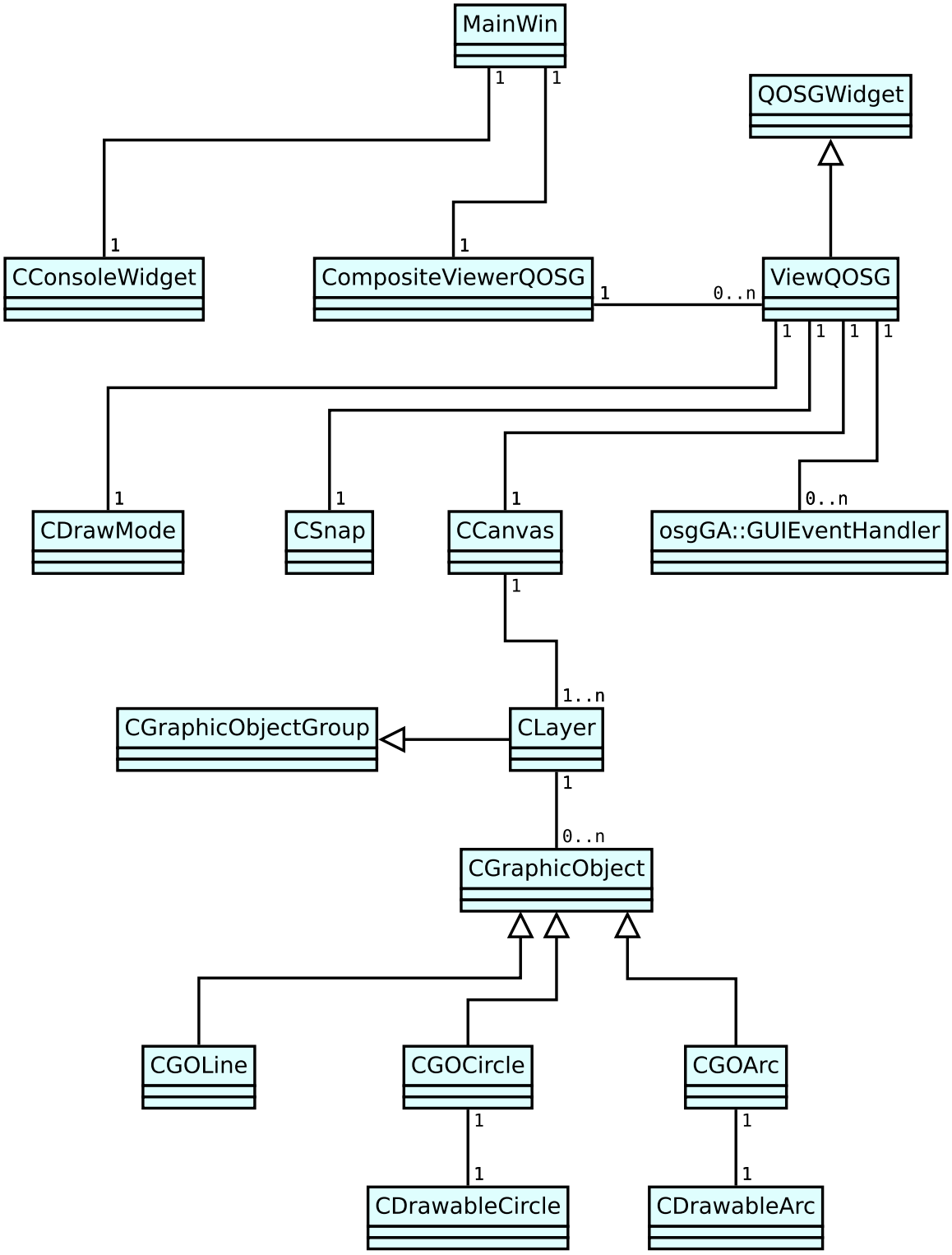
V praxi je tato komponenta představována zobrazovacím polem (widgetem). Widget by měl být především schopen zobrazit grafické entity uložené v datové části. Dále by měl poskytovat metody pro různé zobrazení grafických entit, např. zoom, přesun pohledu a metody pro obsluhu uživatelských událostí myši.

Obsluha událostí myši

Počítačová myš je základním prostředkem pro ovládání programu a zejména pro kreslení grafických objektů. Mezi události vyvolané myší můžeme zařadit stisk a uvolnění tlačítka, pohyb myši případně pohyb se stisknutým tlačítkem. Je třeba zajistit, aby na základě těchto vstupů uživatele bylo možné kreslit různými způsoby rozličné grafické entity. Komponenta pro grafické zobrazení by proto měla disponovat sadou obslužných funkcí (handlerů), které na základě sekvence vstupů z myši vytvoří daný objekt.

3.1.3 Uživatelské prostředí

Z hlediska principu programu je tato část méně důležitá. Uživatelským prostředím jsou myšleny ovládací panely, tlačítka a menu, které mění kontext programu a spouští jeho funkce. Protože u CAD systémů je vyžadována přesnost, kreslení samotnou myší je většinou zcela nedostatečné a proto je třeba mít možnost zadávat konkrétní rozměry. Důležitou součástí grafického prostředí by měl tedy být prvek (prvky) pro zadávání parametrů kreslených grafických entit.



Obrázek 3.2: Diagram tříd

3.2 Objektově orientovaný návrh

Systém je již dekomponován na základní stavební prvky, nyní je třeba tuto dekompozici ještě více zpřesnit. K tomu poslouží diagram tří, který definuje jednotlivé třídy programu a vztahy mezi nimi.

V následující části budou podrobněji rozebrány jednotlivé části objektového návrhu a popsány implementační detaily některých tříd.

3.3 Grafické objekty

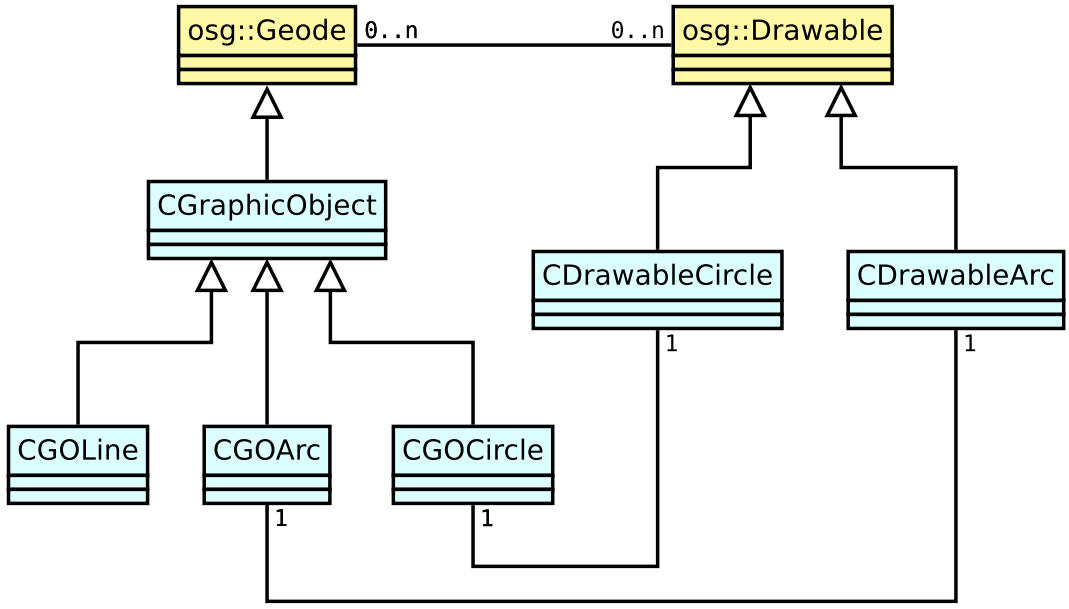
Grafické objekty, nebo-li prvky scény, mohou být reprezentovány objekty mnoha tříd. Všechny tyto třídy jsou však potomky jedné abstraktní třídy `CGraphicObject`, která definuje rozhraní pro třídy konkrétních grafických entit. Každý grafický objekt by měl nést informace, umožňující zejména jeho vykreslení, ale také možnosti provádět s ním potřebné operace. Mezi informace, které by měl každý grafický objekt být schopen poskytnout patří tyto:

- množina uchopovacích bodů objektu
- průsečík(y) s jiným objektem
- náležitost (kompletní i částečná) do obdélníkového výběru
- „blízkost" k určitému bodu
- export do formátu pro uložení
- vrstva, do které objekt přísluší

3.3.1 Implementace tříd grafických entit

Předkem všech tříd grafických objektů, je abstraktní třída `CGraphicObject`. Ta definuje rozhraní pro metody zajišťující funkce, které byly specifikovány výše. V každé třídě potomka jsou poté tyto metody implementovány na základě specifických vlastností dané grafické entity. Samotná třída `CGraphicObject` je potomkem třídy `osg::Geode` z knihovny Open Scene Graph (OSG), která slouží jako list grafu scény a zároveň jako kontejner pro vykreslitelné prvky (geometrické tvary). Dědičnost tedy umožňuje jakémukoliv potomku třídy `CGraphicObject` navázání do grafu scény a naplnění vykreslitelnými prvky, které jsou následně v programu zobrazeny. Konkrétními implementovanými potomky `CGraphicObject` jsou třídy:

- `CGOLine` - úsečka
- `CGOCircle` - kružnice
- `CGOArc` - oblouk



Obrázek 3.3: Vztahy mezi třídami grafických objektů v návaznosti na OSG

3.3.2 Vykreslitelné prvky

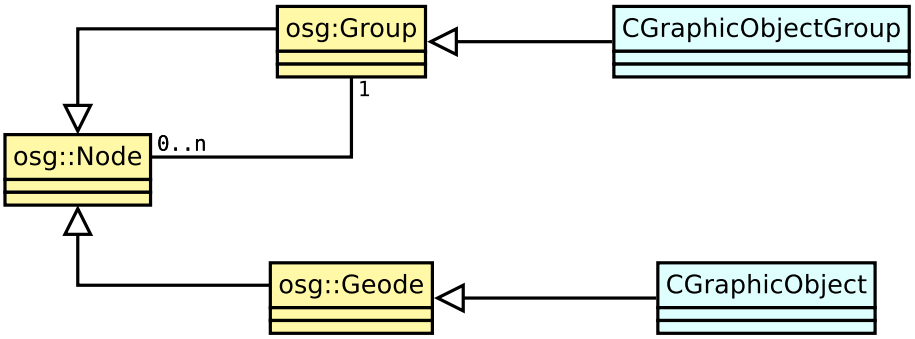
Zobrazitelné prvky v Open Scene Graph (agregované v objektech třídy `CGraphicObject`) jsou potomky abstraktní třídy `osg::Drawable`. Knihovna OSG poskytuje několik potomků třídy `osg::Drawable` pro vykreslování různých částí scény, mimo jiné třídu `osg::Geometry` pro tvorbu geometrických primitiv. Tato třída pracuje prakticky téměř na úrovni OpenGL a z polí bodů vykresluje základní grafické prvky jako čáry nebo plochy.

Třída `osg::Geometry` není však schopna jednoduše vykreslit například kružnici, oblouk či křivku. Z tohoto důvodu bylo nutné implementovat nové potomky třídy `osg::Drawable` pro vykreslení kružnice a oblouku. Kružnice je vykreslována jako n-úhelník, kde počet stran n-úhelníku závisí na detailu aktuálního zobrazení a je vypočítán tak, aby nebylo vykreslování příliš náročné avšak stále kvalitní a okem nerozeznatelné od kružnice vykreslené jednotlivými body. Kreslení oblouku je řešeno obdobně.

Při nakreslení malé kružnice a následném přiblížení, je patrná n-úhelníková skladba, protože scéna není při zoomování překreslována. Pokud by však bylo vyvoláno explicitní překreslení, bude i přiblížená kružnice správně zaoblená. Jako možnost ke zvážení pro další rozšíření programu se nabízí implicitní překreslování scény po překročení určitého kroku přiblížení, prakticky by se jednalo o dynamickou změnu „level of detail“.

3.4 Datové kontejnery pro uchovávání grafických objektů

Grafické objekty až na výjimky neexistují jen tak samy o sobě, jsou sdružovány do datových kontejnerů. Datové kontejnery pro grafické objekty jsou reprezentovány potomky třídy `CGraphicObjectGroup`. Jedná se zejména o hladiny třídy `CLayer`, dalším příkladem by mohl být blok grafických objektů. Speciálním případem kontejneru pro uchovávání grafických objektů je třída `CCanvas` reprezentující plátno (scénu). Tato třída ještě rozšiřuje vlastnosti a funkce „obyčejného“ úložiště o prostředky pro uchovávání speciálních prvků scény (dočasné a označené prvky) a pro manipulaci s nimi.



Obrázek 3.4: Třídy pro uchovávání grafických objektů v návaznosti na OSG

Třída `CGraphicObjectGroup` je implementována jako potomek `osg::Group`, což je třída knihovny Open Scene Graph pro agregaci prakticky libovolného počtu prvků scény (tvoří podstrom grafu scény). Celé schéma dědičnosti využívané pro uchovávání grafických objektů je patrné z obrázku 3.4. Třída `CGraphicObjectGroup` a její potomci jsou schopni agregovat objekty potomků `CGraphicObject` a to s využitím veškerých výhod knihovny Open Scene Graph. Mezi ně patří např. možnost nastavení barvy či typu čáry pomocí sestavy atributů `osg::StateSet` pro celý podstrom, který daný objekt třídy `CGraphicObjectGroup` představuje. Třída `osg::Group` by také měla zaručovat efektivní a rychlé vykreslování prvků grafu scény.

3.4.1 Hladiny

Hladina je speciálním případem kontejneru pro grafické objekty. Každá hladina má své jméno, barvu a mělo by být možné ji jednoduše zobrazit či skrýt. Tyto vlastnosti jsou implementovány v třídě `CLayer` která je potomkem třídy `CGraphicObjectGroup` a zároveň třídy `QObject` z knihovny Qt. Díky předku `QObject` je možné v třídě definovat Qt signály a sloty pro jednoduché napojení na grafické uživatelské prostředí.

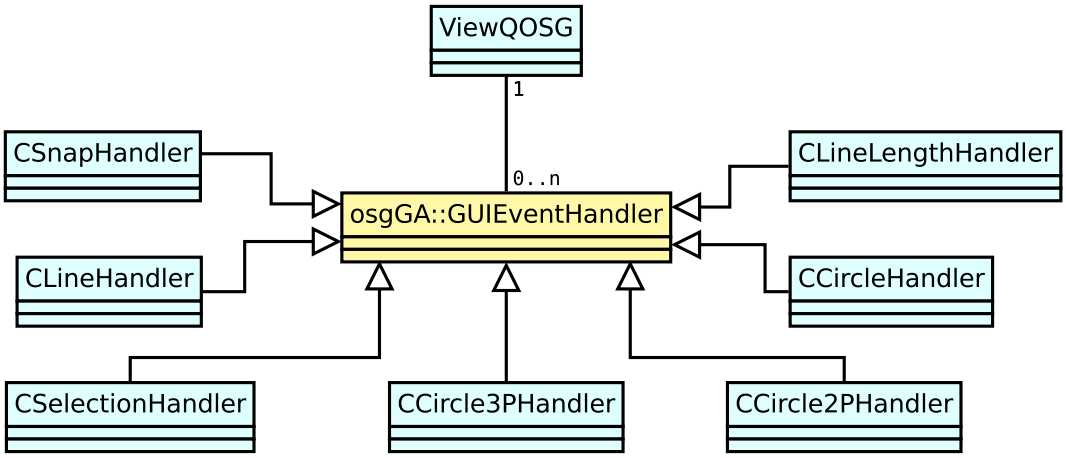
K zobrazení či skrytí hladiny je využíváno možnosti nastavení masky prvku grafu scény, což je možné díky zdědění z `osg::Group` (a `osg::Node`). Při zobrazování jsou pak vykresleny pouze ty podstromy (prvky) grafu, které mají vhodně nastavenou masku.

3.4.2 Scéna

Základní funkcí třídy celé scény je agregace všech hladin s jejich obsahem, tak aby je bylo možné jednoduše zobrazit. Je však také třeba mít možnost hladiny přidávat, mazat a editovat, umisťovat na scénu dočasné objekty, uchovávat informace o označených objektech atd. Třída `CCanvas` představující scénu, je potomkem `osg::Group` a agreguje prvky grafu scény. Kromě všech hladin zde náleží podstrom grafu s dočasnými prvky (např. právě rozkreslená čára, zvýrazněný bod). Třída také poskytuje metody pro označování objektů a práci s nimi.

3.5 Obsluha uživatelských událostí

Uživatel běžně ovládá program prostřednictvím myši a klávesnice, je proto nutné zajistit patřičné reakce na tyto události. O obsluhu uživatelských událostí se starají obslužné rutiny (handlery), objekty různých tříd, které jsou voleny podle toho, jakou chceme aby měl pro-



Obrázek 3.5: Implementace dědičnosti obslužných rutin

gram na události odezvu. Jako příklady různých zpracování stejných událostí (posloupností kliknutí myší) uveďme např.:

- kreslení čáry dvěma body
- kreslení čáry bodem a odchylkou
- kreslení kruhu dvěma body
- výběr objektů

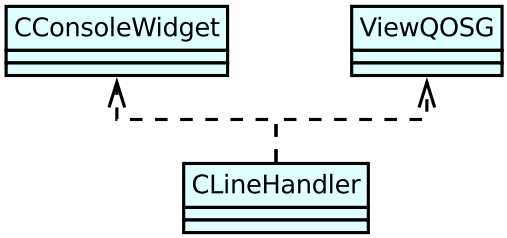
Zobrazovací komponenta má k dispozici sadu různých obslužných rutin, z nichž vybírá a aktivuje tu, která přísluší operaci právě prováděné uživatelem. Obslužné rutiny však nemusí reagovat pouze na události odchytávané zobrazovací komponentou, ale mohou zpracovávat vstupy i z jiných částí uživatelského prostředí např. z polí pro zadávání textu. Některé handlery jsou tedy propojeny i s více prvky GUI.

3.5.1 Implementace handleru

Samotná obslužná rutina by se dala popsat jako stavový automat a z toho vyplývá i její implementace. Jedná se o třídu, která je potomkem `osgGA::GUIEventHandler`, což je třída knihovny Open Scene Graph, určená pro obsluhu uživatelských událostí. Hlavní metodou třídy je metoda `handle`, která je volána při každé události vyvolané uživatelem. V této metodě se na základě aktuálního stavu a parametrů vstupní události provedou příslušné operace a nastaví se nový stav automatu. Takto se děje, dokud není handler zrušen (deaktivován), nebo dokud nedospěje stavový automat do konce. Díky zdědění z třídy `osgGA::GUIEventHandler` je možné handlery přímo „navázat” na zobrazovací widget OSG.

3.5.2 Napojení na prvky uživatelského prostředí

Grafické prvky mohou být kresleny myší, ale i zadáváním dat z klávesnice (do konzolového okna). Základní handler třídy `osgGA::GUIEventHandler` však nemá zajištěn přístup k těmto uživatelským prvkům. Zděděné obslužné rutiny pro kreslení grafických prvků mají proto k dispozici ukazatel na konzoli (viz. 3.6), mohou tak do ní psát a s využitím Qt slotů i reagovat na zadaná data.



Obrázek 3.6: Diagram závislosti obslužné rutiny pro kreslení úsečky (CLineHandler)

3.6 Uchopování a módy kreslení

Uchopování jako takové je vlastně upravováním vstupní souřadnice zadané uživatelem. Je tedy třeba, aby obslužné rutiny uživatelského vstupu měly přístup jak k „originálním“ souřadnicím, tak i k souřadnicím „uchopeným”. Komponenta pro uchopování představuje jakýsi filtr vstupních událostí, jeho princip je znázorněn v diagramu 3.7. Komponenta je tvořena třídami `CSnap` a `CSnapHandler`. První jmenovaná se stará zejména o nastavování módů a uchovávání stavu - souřadnic uchopeného bodu. Třída `CSnapHandler` je pak obslužná rutina uživatelského vstupu, která na základě nastavení v `CSnap` a obsahu scény filtruje souřadnice myši při každé jejich změně. Filtrování lze rozdělit podle toho, zda na něj má vliv obsah scény, nebo je na něm nezávislé. Mezi nezávislé uchopovací módy patří:

- přichytávání k mřížce
- kreslení ve směru souřadných os (ortogonální kreslení)

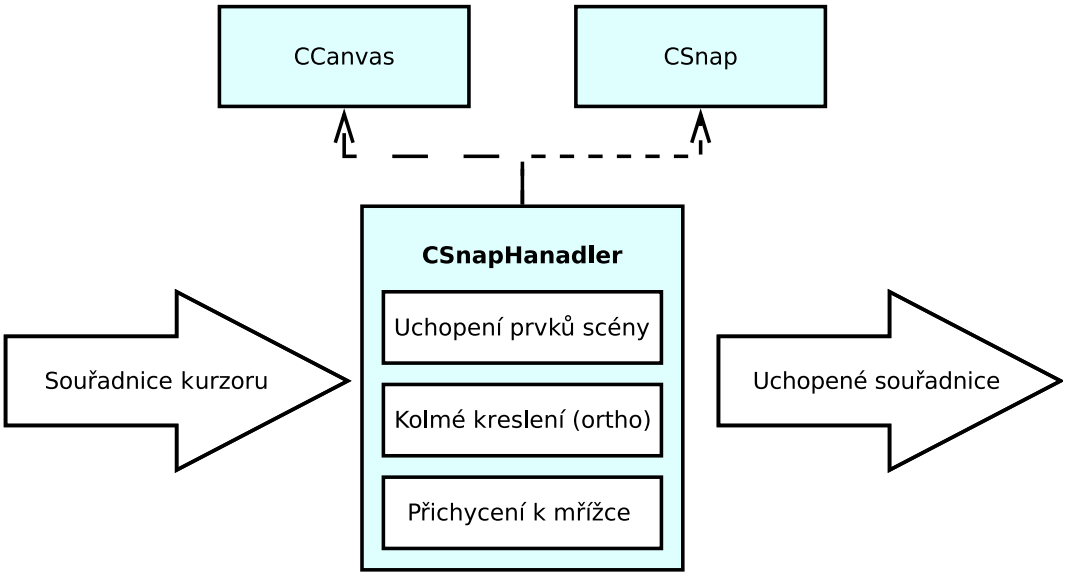
Jiné způsoby uchopování již pracují s obsahem scény a případně i s zadanými hodnotami právě kreslené entity. Přístup ke grafickým entitám je třeba při uchopování:

- významných bodů (koncové body, středy)
- kolmic a tečen
- průsečíků

3.6.1 Uchopování prvků scény

Protože uchopování probíhá prakticky při každém pohybu myši, je třeba klást důraz na rychlost provádění operací. Kritickým místem je v tomto případě práce s obsahem scény, kdy je třeba projít všechny potenciální kandidáty na uchopení a testovat je např. na významné body. Ještě kritičtější je pak situace při zjišťování průsečíků, kdy je třeba testovat objekty „každý s každým“ a složitost algoritmu tedy narůstá kvadraticky s počtem objektů. Navíc samotný výpočet průsečíku dvou entit je často matematicky poměrně složitý. Je proto žádoucí si okruh kandidátů na uchopení co nejvíce zúžit.

V programu jsou při každém pohybu myši procházeny všechny prvky scény a testuje se jejich poloha (vzdálenost) vůči souřadnicím bodu, nad kterým se nachází kurzor. Toto testování nemusí být zcela přesné, mělo by však co nejrychleji vyřadit co nejvíce grafických entit, kterých se uchopený bod nemůže týkat. K testování blízkosti je u potomků třídy `CGraphicObject` určena metoda `isNear`. Další průchody pro uchopení bodů se již provádí jen na zúženém seznamu kandidátních entit.



Obrázek 3.7: Princip činnosti a závislosti obslužné rutiny CSnapHandler

Zvolené řešení je poměrně jednoduché a testovat všechny prvky pouze na několik podmínek je rychlejší, než u všech počítat průsečíky. U rozsáhlejších výkresů by však zřejmě i toto řešení bylo pomalé a průchod celé scény by zabíral příliš mnoho procesorového času. Pro další vývoj systému by bylo vhodné implementovat dělení scény na části, kde by pro každou část existoval seznam ukazatelů (referencí) na entity, které se v ní nachází. Při zpracování vstupních souřadnic by se pak pracovalo pouze s částí scény, do které by daný bod příslušel. Nejvhodnějším řešením by bylo zřejmě dynamické dělení scény podle počtu objektů v určité části do struktury DL stromu.

3.6.2 Závislost uchopení na předchozím vstupu

Při kreslení tečny nebo kružnice je nutné znát výchozí bod kreslené čáry, abychom mohli stanovit bod dotyku tečny či průsečík kolmice. Pro tyto potřeby poskytuje třída `CSnap` možnost nastavení počátečního bodu. Pokud je tento bod nastaven a zároveň je povoleno uchopování kolmic nebo tečen, je možné vypočítávat např. potenciální tečné body a uchopovat je.

3.6.3 Implementace uchopování významných bodů

Jak již bylo zmíněno výše, významnými body jsou myšleny koncové body úseček, středy apod. K uchopení takovýchto bodů je třeba znát pouze obsah scény a aktuální bod, nad kterým se nachází kurzor myši. Po zúžení seznamu entit je pro každý objekt v seznamu volána metoda `snap`, která vrací, zda má daná entita vyhovující bod k uchopení a pokud ano, vkládá souřadnice tohoto bodu do seznamu. Po průchodu všech entit je z tohoto seznamu vybrán bod nejbližší k souřadnici kurzoru. Tento princip je shodný i pro uchopování tečen, kolmic a průsečíků, kde se však podstatně složitěji získávají souřadnice uchopovaného bodu.

3.6.4 Implementace uchopování tečen a kolmic

Pro uchopení tečen a kolmic nestačí znát pouze obsah scény, ale je třeba mít k dispozici i výchozí bod, z kterého kreslíme. Princip je však stejný jako u uchopování významných bodů. Nad všemi kandidátními (blízkými) entitami je volána metoda `ortho`, která plní seznam souřadnicemi vyhovujících bodů k uchopení. Tato metoda je implementována jak u čáry, tak i u kružnice a oblouku. U čáry vrací body pro kreslení kolmic a pro kružnice a oblouky body pro kreslení tečen.

Výpočty příslušných bodů nejsou příliš složité a vychází z analytických vzorců pro dané geometrické obrazce, které jsou vyučované v analytické geometrii, případně jsou dostupné na internetu [4]. Při implementaci byl však kladen důraz na efektivitu a rychlost výpočtů, protože vzhledem k jejich častému opakování, je rychlost kritická pro bezproblémový běh programu.

3.6.5 Implementace uchopování průsečíků

Uchopování průsečíků je výpočetně i algoritmicky nejsložitějším implementovaným uchopovacím módem. Jeho náročnost na prostředky spočívá zejména v nutnosti testovat objekty „každý s každým", což vnáší do problému kvadratickou složitost v závislosti na počtu testovaných objektů. Při průchodu seznamem entit je nad potomky `CGraphicObject` volána metoda `intersection`. Tato metoda je pro každý objekt postupně volána s každou kandidátní entitou jako parametrem, jedině tak lze zajistit výpočet všech průsečíků.

Protože metodě `intersection` může být jako parametr předán jakýkoliv potomek třídy `CGraphicObject`, měly by zde být řešeny výpočty průsečíků pro všechny možné potomky. Pokud by tyto výpočty byly implementovány přímo v těle metody `intersection`, docházelo by k opakování stejného kódu v různých třídách (např. v třídě úsečky by byl kód pro výpočet průsečíku s kružnicí a v třídě kružnice by byl ten samý kód pro výpočet průsečíku s čárou). Z tohoto důvodu je součástí programu třída `CIntersection` pro výpočty průsečíků, která je využívána v každé metodě `intersection`. Pro další rozšiřování je však ponechána možnost řešit výpočet průsečíků i přímo v těle metody `intersection`.

Výpočty průsečíků

Výpočty souřadnic průsečíků grafických objektů jsou realizovány ve třídě `CIntersections`. Prvním krokem výpočtu, je stanovení třídy (typu) obou entit, to je možné díky operátoru `dynamic_cast`, který C++ nabízí. Po stanovení typů, je pro danou kombinaci entit volána příslušná metoda (pokud existuje) pro výpočet průsečíků (např. `line_line()`, `line_circle()`, `circle_circle()` atd.).

Průsečíky jsou počítány na základě analytických vzorců příslušných geometrických obrazců z analytické geometrie [4]. U oblouku jsou využívány vzorce pro kružnici s následným testováním výsledných souřadnic, zda leží ve výseči oblouku.

3.6.6 Označování grafických objektů

Pro případnou editaci nebo mazání je třeba mít možnost grafické objekty označit. Jejich seznam je uchováván v objektu scény třídy `CCanvas`, samotné označování pak provádí handler třídy `CSelectionHandler`. Jedná se o běžnou obslužnou rutinu, která prochází všechny entity scény a testuje, zda se nachází v obdélníkovém výběru. Podle toho, zda jsou vybírány pouze objekty kompletně nebo alespoň částečně obsažené v obdélníkovém výběru, je

na entity volána metoda `isInFrame` nebo `isPartInFrame`. Implementace těchto metod je prakticky řešením viditelnosti ve 2D. Pro úsečky je použit modifikovaný algoritmus Cohen-Sutherland [1].

3.7 Ukládání a načítání dat

Pro ukládání dat jsem zvolil formát DXF, který je v oblasti CAD systémů používán spíše pro exporty a převody mezi různými programy, avšak pro účel tohoto projektu prozatím postačí i jako hlavní uchovávací formát a do budoucna může být jeho implementace využita pro účely exportu a importu. Každá třída grafické entity (potomek `CGraphicObject`) by měla mít implementovánu metodu `saveToDXF` pro export do DXF. Způsob uložení je tak v režii třídy entity.

Pro načítání z formátu DXF je implementována třída `CDxfReader`, která je potomkem třídy `DL_Adapter` z knihovny DXFLib. Toto řešení značně usnadňuje implementaci načítání dat, kde stačí pouze reimplementovat metody pro načtení příslušných entit.

3.7.1 Barvy ve formátu DXF

Prakticky jediným úskalím při ukládání do formátu DXF se ukázalo uchovávání informací o barvě. Barva zde totiž není reprezentována jako hodnota RGB, nebo jiného barevného modelu, ale jako index identifikující jednu z 256 pevně předdefinovaných barev. Z hlediska načítání toto není problém, stačí pouze na základě indexu přiřadit příslušnou hodnotu RGB, při ukládání je však nutné namapovat aktuální hodnotu RGB na nejvhodnější index. Informace o barvě hladiny jsou proto uchovávány v objektech třídy `CColor`, která je potomkem `QColor` z knihovny Qt a je rozšířena o metodu vracející nejvhodnější DXF index barvy a o metodu pro načítání zadáním indexu.

3.8 Uživatelské prostředí

Uživatelské prostředí CAD systému se skládá z mnoha různých komponent, panelů a dialogů. Z hlediska jeho podstaty jsou však pro CAD systém důležité zejména tyto dvě součásti:

- zobrazovací widget
- widget pro zadávání vstupů (konzole)

Není předmětem této práce zabývat se konstrukcí uživatelských prostředí v knihovně Qt a proto zde nebudou podrobněji rozebrány všechny jeho části ani základy a principy programování v Qt, které je možné nastudovat ze zdrojů k tomu určených [2]. V následujícím textu budou popsány pouze implementační detaily dvou výše uvedených stěžejních prvků GUI.

3.8.1 Zobrazovací komponenta

Knihovna Open Scene Graph poskytuje pro zobrazení scény komponentu `osgViewer`, která nabízí poměrně rozsáhlou škálu funkcí, včetně mechanismů pro zpracovávání uživatelských vstupů apod. Bylo tedy třeba tuto komponentu vhodně umístit do uživatelského prostředí v Qt a zajistit korektní předávání událostí do OSG. Vzhledem k rozšířenosti Qt i OSG se na internetu nabízí několik příkladů pro integraci `osg::Viewer` do Qt. Pro tento projekt

jsem využil příkladu z oficiálních webových stránek Open Scene Graph [3], který jsem však pro účely programu značně upravil.

Třídou, která tvoří jakýsi pomyslný most mezi Qt a Open Scene Graph, je `QOSGWidget`. Jedná se o potomka třídy `QWidget` a jsou zde reimplementovány metody zpracovávající uživatelské vstupy tak, aby předávaly příslušné události do grafického okna OSG. Při zpracování událostí se nemusí vždy jednat o pouhé kliknutí, nebo stisknutí klávesy, je třeba zpracovávat a předávat i informace o změně velikosti apod.

3.8.2 Vstupní konzole

Vstupní konzole představuje univerzální prostředek pro zadávání nejrůznějších dat pro kreslení objektů. Jejím hlavním úkolem je tedy „komunikace" mezi kreslící obslužnou rutinou a uživatelem. Nástrojem komunikace uživatele je zadávání dat z klávesnice a čtení textu vypsaného na obrazovku. Je tedy třeba, aby obslužná rutina měla možnost vypisovat text do konzole a zároveň přijímat data zadaná uživatelem. Protože knihovna Qt nenabízí přímo komponentu konzole, bylo třeba implementovat vhodný ovládací prvek úpravami stávajícího widgetu pro editaci neformátovaného textu.

Konzole je implementována jako potomek třídy `QPlainTextEdit`, což je třída knihovny Qt pro zadávání neformátovaného textu. V zásadě nebylo nutné provádět mnoho úprav, bylo třeba zejména zajistit, aby již potvrzený text a text nezadaný uživatelem nebylo možné editovat. Objekt si z tohoto důvodu uchovává pozici posledního „editovatelného" znaku a v případě pokusu o editaci znaku na nižší pozici této akci „zamezí". Dalším rozšířením bylo přidání Qt signálu, který je emitován při každém potvrzení („odenterování") vstupních hodnot. Tento signál s sebou nese vstupní hodnotu ve formě textového řetězce, tak aby bylo na něj možné napojit příslušný slot pro zpracování vstupních hodnot.

Komponenta konzole může být v dalším vývoji programu rozšířena o některé užitečné funkce, jako např. nabízení předchozích hodnot nebo rozšíření možností vstupů uživatele. Vhodná by byla např. implementace zpracování relativních a polárních souřadnic.

3.9 Volba knihoven a implementačních prostředků

Není efektivní (ani žádoucí) programovat funkce, které již byly někdy naprogramovány. Je tedy vhodné použít prostředky dostupných knihoven, zejména pro akcelerované vykreslování grafických entit a pro tvorbu grafického uživatelského prostředí (GUI). V dnešní době existuje poměrně široká škála volně dostupných knihoven, které nabízí rozsáhlé množství funkcí a multiplatformní podporu.

3.9.1 Vykreslování grafických entit

Pro přímé vykreslování se v současné době nabízí dvě základní knihovny, a to DirectX vyvíjený firmou Microsoft nebo open source projekt OpenGL. Vzhledem k požadavku multiplatformnosti cílové aplikace je volba jasná, protože DirectX je určen zejména pro systémy Windows. Jako vhodná knihovna pro vykreslování tedy zůstává OpenGL, která však sama o sobě poskytuje pouze nejnutnější „low-level" prostředky pro vykreslování. Existuje proto několik nástaveb, které přidávají OpenGL přívětivější rozhraní a poskytují mnohem více prostředků pro implementaci aplikace. Z těchto knihoven jsem pro tvorbu aplikace vybral Open Scene Graph.

3.9.2 Uživatelské prostředí

Vzhledem ke zvolenému programovacímu jazyku C++ existuje několik možností výběru knihovny pro implementaci grafického uživatelského prostředí. Pokud budeme trvat na multiplatformnosti, nabízí se zejména knihovny WxWidgets a Qt. Vzhledem k mým předchozím zkušenostem jsem pro implementaci zvolil knihovnu Qt (verzi 4.5), která je v současné době stále více populární. Za velkou výhodu Qt považuji dostupnost mnoha příkladů použití a velice kvalitní dokumentaci.

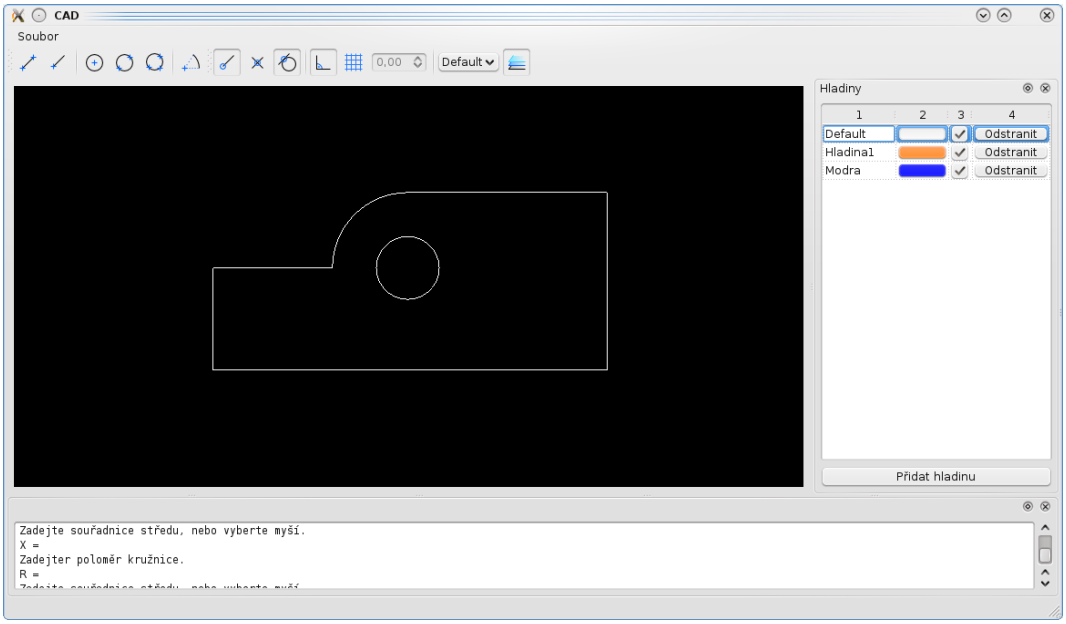
3.9.3 Ukládání dat

Jak již bylo zmíněno dříve, jedním z nejrozšířenějších formátů pro přenos mezi dat mezi programy CAD je DXF. Existuje několik knihoven, které poskytují rozhraní pro čtení a zápis souborů v tomto formátu. Pro tento projekt jsem se rozhodl využít open source knihovnu DxfLib. Tato knihovna poskytuje poměrně jednoduché rozhraní pro čtení a ukládání souborů. Přestože nepokrývá všechny možnosti formátu DXF, je myslím dostačující i pro rozsáhlejší projekty než je tato práce (je využívána například v projektu QCAD).

Kapitola 4

Zhodnocení výsledků

Na základě popsaného návrhu se podařilo implementovat aplikaci, která je schopna kreslit základní geometrické útvary s využitím principů používaných v CAD systémech. V následující kapitole budou přesněji popsány funkce a možnosti, které program nabízí.



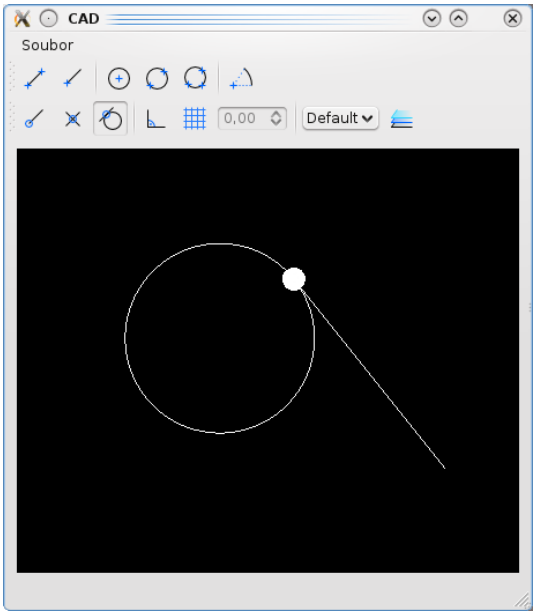
Obrázek 4.1: Snímek výsledného programu

4.1 Možnosti kreslení

Program umožňuje kreslit tři základní geometrické útvary:

- úsečku
- kružnici
- oblouk

Tyto tvary je možné kreslit několika způsoby a to jak myší, tak zadáváním přesných hodnot pomocí klávesnice. Při kreslení je možné využívat několika možností uchopování



Obrázek 4.2: Kreslení úsečky tečné ke kružnici

a podpory kreslení. Kreslené objekty jsou umísťovány do hladin, mohou být mazány a přesouvány mezi hladinami. Hladiny je možné přidávat, editovat a nastavovat jejich název, barvu a viditelnost. V souhrnu aplikace implementuje principy vektorového grafického editoru s charakteristickými znaky CAD systémů.

4.1.1 Úsečky

Kreslení úseček je možné dvěma základním způsoby:

- zadáním počátečního a koncového bodu
- zadáním počátečního bodu, délky a odchylky (úhlu)

V obou případech lze všechny údaje zadat buď zadáním dat do konzole nebo přímo nakreslit myší, případně oba způsoby zadání kombinovat. Při kreslení lze využít možností uchopování významných bodů, průsečíků, kolmic i tečen. Při aktivaci přichytávání kolmic a tečen jsou při volbě druhého bodu, odchylky nebo délky úsečky uchopovány body, které vedou ke kreslení úsečky kolmé na blízkou úsečku nebo tečné ke kružnici či oblouku.

Další možností, kterou je možné při kreslení úseček využít, je ortogonální (kolmý) mód kreslení. Úsečky je poté možné kreslit pouze rovnoběžně se souřadnými osami, což je užitečné zejména při kreslení úseček zadáním délky. Tento mód umožňuje např. rychlé nakreslení čtverce či obdélníku.

Při dalším vývoji programu by bylo vhodné rozšířit možnosti kreslení čáry o navazovanou „multičáru“, nebo možnost kreslení zadáním středu a koncového bodu, což by bylo užitečné např. pro kreslení os kružnic.

4.1.2 Kružnice

Kružnice je možno kreslit třemi základními způsoby:

Zadáním středu a poloměru Jedná se o základní způsob zadání kružnice, kdy je zvolen nejprve střed a poté poloměr kružnice. Obě hodnoty lze zadat buď myší, nebo z konzole.

Zadáním dvou bodů kružnice Zadání pomocí dvou bodů je vlastně zadáním průměrem. Kružnice je vykreslena tak, že její střed leží v polovině pomyslné úsečky mezi dvěma zadanými body a její průměr se rovná délce této úsečky. Oba body je možné opět zadat buď myší, nebo souřadnicemi z konzole.

Zadáním tří bodů kružnice Jde o nejkomplikovanější implementované zadání kružnice, kdy je kružnice zadána pomocí tří bodů, které na ní leží. Souřadnice bodů je možné zadat myší, nebo z konzole.

Při všech způsobech kreslení kružnice je možné využívat celou škálu uchopovacích módů. Je tak možné kreslit kružnice, které mají střed ve „významném bodě", nebo např. kružnice tečné k úsečkám. Pro kružnici je méně významný ortogonální mód kreslení, který lze sice použít, ale vzhledem k charakteru kružnice nemá značné využití.

V případě dalšího vývoje se nabízí mnoho možností, jak volby tvorby kružnic rozšířit. Velmi užitečné by bylo kreslení formou „tečna - tečna - rádius", kdy jsou nejprve vybrány dva prvky (např. úsečky), ke kterým bude kružnice tečná, a poté je zvolen její rádius.

4.1.3 Oblouk

Kreslení oblouku je možné pouze jedním základním způsobem, a to zadáním středu, poloměru, počátečního a koncového úhlu. Lze opět kombinovat zadání myší a z konzole, úhly jsou zadávány ve stupních. Oblouk je vždy vykreslován proti směru hodinových ručiček, je třeba na tuto skutečnost při zadávání počátečního a koncového úhlu myslet. Je možné využít všech možností uchopování, díky tomu lze jednoduše kreslit tečné oblouky, uchopovat úhly atd.

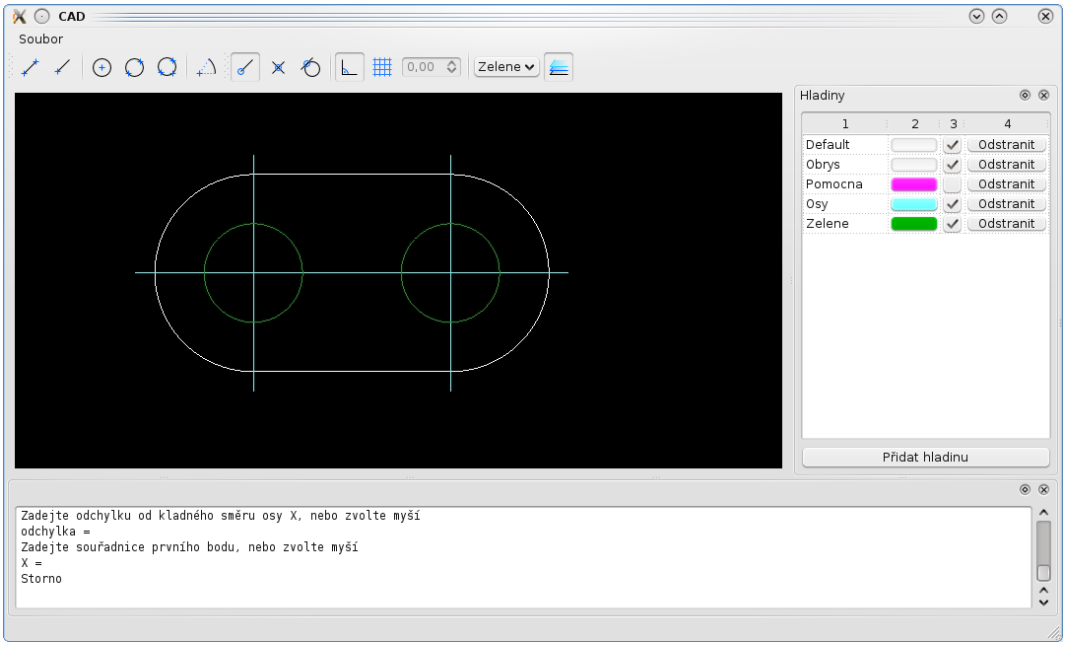
Narozdíl od kružnice, u oblouku získává na významu ortogonální mód. Při jeho aktivaci je možné jednoduše kreslit oblouky v násobcích devadesáti stupňů.

Oblouk je v technických výkresech poměrně často používaný u zaoblených hran apod. Z tohoto důvodu by bylo vhodné možnosti jeho zadávání dále rozšířit, zejména o možnost zaoblování hran, kdy by bylo možné zaoblit daným poloměrem úhel svíraný dvěma objekty (úsečkami).

4.1.4 Hladiny

Výsledný program umožňuje základní práci s hladinami, které jsou charakteristickou součástí každého CAD systému. Hladiny je v aplikaci možno přidávat, mazat i editovat, každé hladině lze nastavit barvu, název a viditelnost. Tyto základní funkce umožňují např. vytvoření hladiny pro pomocné konstrukce, která je pro zobrazení výsledného výkresu skryta. Možnost nastavení různých barev hladin umožňuje rozlišení druhu konstrukce, složky výkresu apod.

Další vlastností, která by měla být pro hladiny v budoucnu implementována, je nastavení typu čáry, kterým se budou objekty v dané hladině vykreslovat. V současné podobě program umožňuje pouze kreslení souvislými čarami. Rozšíření např. o čáry čárkované a čerchované by bylo velmi vhodné.



Obrázek 4.3: Ukázka využití hladin

4.1.5 Uchopování

Možnosti uchopování byly v zásadě popsány u jednotlivých typů grafických objektů. Jen pro zopakování, v aplikaci lze uchopovat:

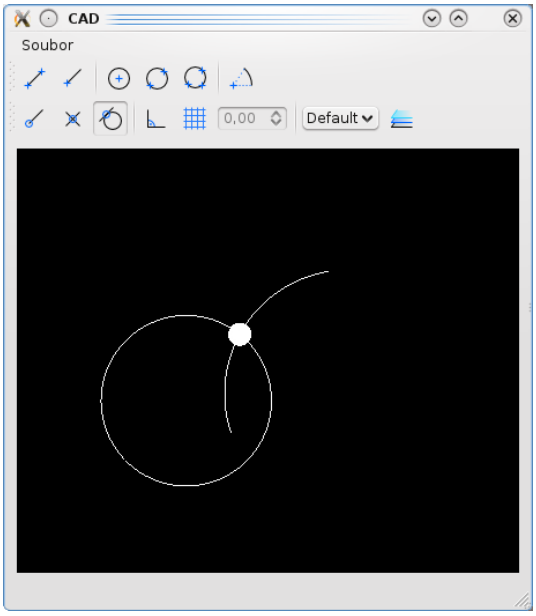
- koncové body úseček
- středy úseček
- středy kružnic
- tečny
- kolmice

V zásadě program nabízí užitečné uchopovací možnosti, které významně usnadňují kreslení. Kromě uchopování objektů scény lze používat i přichytávání k mřížce, jejíž hustota je jednoduše nastavitelná.

4.2 Editace, výběr objektů

Z hlediska editace není aplikace příliš rozsáhlá, avšak i některé pokročilé CAD systémy sázejí spíše na systém „vymaž a udělej znovu". Důležitým předpokladem pro další možnosti editace je možnost označování (vybírání) objektů scény. Program umožňuje několik způsobů výběru:

- výběr jednoho objektu (kliknutím myši)
- výběr objektů kompletně obsažených v obdélníkovém výběru



Obrázek 4.4: Uchopení průsečíku kružnice a oblouku

- výběr objektů alespoň částečně obsažených v obdélníkovém výběru

Vybrané objekty lze mazat, případně měnit hladinu, do které náleží. Bylo by možné dále rozšířit možnosti editace o přesun objektů, kopírování či změnu velikosti. Realizované způsoby označování budou také užitečné při implementaci dalších kreslících metod, např. při zaoblování vybraných hran (úhlů).

4.3 Efektivita ovládání

Základními ovládacími prvky programu jsou události myši a vstupní konzole. Ovládání myší je dostatečně názorné, při kreslení jsou zobrazovány výsledné obrazce tak, jak budou vypadat v případě potvrzení bodu, nad kterým je kurzor. Jsou také dostatečně znázorňovány uchopené body. Konzole umožňuje základní zadání hodnot a souřadnic. Značnou výhodou je možnost kombinování kreslení pomocí myši a zadáním z konzole, je tak možné např. střed kružnice zvolit myší, ale přesný poloměr uvést do konzole. Díky kombinaci kreslení myší a pomocí konzole, je možné s programem kreslit poměrně rychle a přesně.

Pro další zvýšení produktivity práce by bylo vhodné dále rozšířit možnosti zadávání dat do konzole, např. o nabídku předchozích hodnot, relativní souřadnice atd. Z hlediska ovládání by tedy bylo vhodné zabývat se zejména implementací dalších vylepšení konzole.

4.4 Možnosti ukládání a načítání dat

Aplikace je schopna ukládat a načítat data ve formátu DXF v rozsahu odpovídajícím rozsahu implementovaných funkcí. Je tedy možné ukládat a načítat úsečky, kružnice, oblouky a informace o hladinách. Formát DXF není příliš vhodným hlavním formátem pro ukládání dat z CAD systému, byl navržen a je používán jako exportní formát. Vzhledem k rozsahu aplikace jsou však jeho možnosti zatím dostatečné i pro výchozí ukládání a v dalším vý-

voji je pak možné jej využít pouze jako prostředek pro export a import. Uložená data jsou ve validním formátu DXF a jsou zobrazitelná např. v prohlížečích tohoto formátu.

4.5 Programová koncepce a rozšiřitelnost

Implementace opravdu komplexního a konkurenceschopného CAD systému daleko přesahuje rozsah bakalářské práce. Z tohoto důvodu byla aplikace navrhována tak, aby implementovala charakteristické funkce CAD systému, ale zejména aby bylo možné ji dále rozšiřovat. Byl tedy kladen důraz na systematickou koncepci celého návrhu, naopak k implementaci byli vybráni vždy jen určití zástupci z dané skupiny funkcí (je možné kreslit např. „jen“ 3 grafická primitiva).

V průběhu vývoje aplikace docházelo postupně k rozšiřování jednotlivých funkcí a bylo tedy možné si rozšiřitelnost jaksi „otestovat”. Pro snadné přidávání dalších komponent je hojně využíváno dědičnosti tříd a polymorfismu. Aplikace je snadno rozšiřitelná o další grafické prvky, je také velmi snadné doprogramovat další metody jejich kreslení. Toto považuji za splnění zásadních předpokladů pro další úspěšný vývoj, protože právě přidáváním dalších grafických entit a možností jejich kreslení program získá na využitelnosti.

Kapitola 5

Závěr

Jako výsledek práce vznikla multiplatformní aplikace pro 2D kreslení s orientací na počítačem podporované projektování. Program umožňuje provádět základní úkony, které jsou pro CAD kreslení charakteristické a stěžejní, což bylo základním cílem této práce. Výslednou aplikaci nelze brát jako komplexní reálně využitelný CAD systém, ale spíše jako studii návrhu tohoto typu programu a jádro pro další rozšiřování.

Celý projekt byl přínosem zejména z hlediska nových znalostí a zkušeností, které jsem během jeho realizace nabyl. Cennou zkušeností byla práce s dvěma rozsáhlými knihovnami Qt a Open Scene Graph, kde použití Open Scene Graph se ukázalo pro projekt tohoto rozsahu jako zbytečně „megalomanské“ a zcela jistě by postačilo použití pouze OpenGL. Jako hlavní nedostatek OSG se ukázala špatná dokumentace této knihovny a s tím spojené větší úsilí potřebné k nastudování jejích principů a možností. Implementace pomocí „čistého” OpenGl by tedy zřejmě zjednodušila a zkrátila vývoj. Přestože použití OSG pro tento projekt bylo spíše kontraproduktivní a jeho výhody nejsou v tomto rozsahu práce příliš vidět, věřím že výběr této knihovny byl vhodný z hlediska dalšího vývoje programu.

Mým cílem při řešení projektu bylo vytvoření zejména jednoduše rozšiřitelné aplikace, která bude připravena na postupnou implementaci dalších funkcí. Pro zvýšení využitelnosti je třeba implementovat zejména kreslení dalších grafických entit, přidat možnost jejich kopírování, přesunu a jiných transformací. Dále je nutné rozšířit možnosti zadávání pomocí konzole alespoň o relativní souřadnice a nabídku předchozích (výchozích) hodnot. Dalším důležitým krokem je vyřešení možnosti tisku společně s nastavením typu a tloušťky čar. Do budoucna bude také vhodné rozšířit implementaci uchopování o dynamické dělení scény na části, aby při uchopování nebylo třeba testovat všechny existující entity.

Vzhledem k tomu, že výsledkem projektu je funkční aplikace, která nabízí mnoho dalších možností vývoje a nekončí ve slepé uličce, splnila práce mé cíle a považuji ji za úspěšnou.

Literatura

[1] Kršek, P.: *Základy počítačové grafiky - Studijní opora*. FIT VUT v Brně, 2006-11-08.

[2] Nokia Corporation: Dokumentace knihovny Qt. `http://doc.trolltech.com/`, 2010.

[3] OSG Community: OpenSceneGraph Wiki.
`http://www.openscenegraph.org/projects/osg/wiki/`, 2010.

[4] Vojáček, J.: Analytická geometrie.
`http://maths.cz/mapa-webu/analyticka-geometrie.html`, 2010, iSSN: 1803-7615.

Dodatek A

Obsah CD

/bin	spustitelné soubory
/src	zdrojové kódy programu
/doc	zdrojové texty dokumentace v LaTeX
/doxygen	dokumentace zdrojového kódu programu
bp.pdf	projektová dokumentace ve formátu PDF
readme.txt	informace k obsahu CD