VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

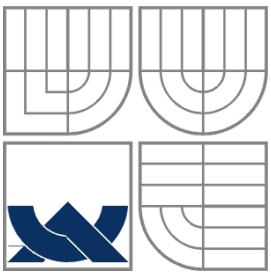
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# CAD SYSTÉM PRO 2D KRESLENÍ

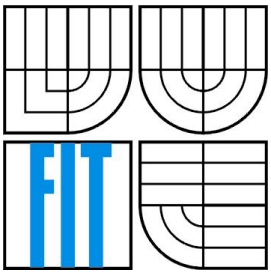
**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS

AUTOR PRÁCE PETR LANGR
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# CAD SYSTÉM PRO 2D KRESLENÍ

CAD SYSTEM FOR 2D DRAWING

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

AUTOR PRÁCE — PETR LANGR
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. PŘEMYSL KRŠEK Ph.D.
SUPERVISOR

BRNO 2012

# Abstrakt

Tato bakalářská práce se zabývá problematikou CAD systémů pracujících ve dvojrozměrném prostoru. Seznamuje nás s definicí CAD systémů a s požadavky na ně kladenými. Práce se dále věnuje návrhu a tvorbě aplikace demonstrující navržené řešení. Na závěr práce jsou shrnuty dosažené výsledky a nástin možností budoucího rozšíření aplikace.



# Abstract

This thesis deals with CAD systems operating in two-dimensional space. It acquaints us with the definition of CAD systems and the requirements placed on them. The thesis also deals with design and implementation applications that demonstrate the proposed solution. At the conclusion summarizes the results and outline possible future expansion of the application.



# Klíčová slova

CAD systém, 2D kreslení, vektorová entita, uchopovací mód, OpenGL



# Keywords

CAD system, 2D drawing, vector entity, snap mode, OpenGL



# Citace

Petr Langr: CAD systém pro 2D kreslení, bakalářská práce, Brno, FIT VUT v Brně, 2012

# CAD systém pro 2D kreslení

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Přemysla Krška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Langr
15. května 2012

## Poděkování

Tímto bych rád poděkoval vedoucímu mé práce doc. Ing. Přemyslu Krškovi, Ph.D, za poskytnuté rady a konzultace během tvorby této práce.

# Obsah

Obsah ..... 1
1 Úvod ..... 3
2 Rozbor CAD systémů ..... 4
2.1 Charakteristika ..... 4
2.1.1 Požadavky ..... 5
2.1.2 Hladiny ..... 6
2.1.3 Základní entity ..... 6
2.1.4 Modifikace ..... 10
2.2 Uchopovací mód ..... 10
2.2.1 Uchopovací body ..... 11
2.3 Ortogonální a polární mód ..... 11
2.4 Výstupní soubor ..... 12
2.5 Grafické uživatelské rozhraní ..... 12
3 Návrh a implementace řešení ..... 14
3.1 Cíle řešení ..... 14
3.2 Reprezentace dat ..... 14
3.2.1 Reprezentace entit ..... 15
3.2.2 Reprezentace hladin ..... 17
3.2.3 Navázání entity na hladinu ..... 18
3.2.4 Datové struktury ..... 18
3.3 Uchopovací mód ..... 19
3.3.1 Uchopovací mřížka ..... 19
3.3.2 Selection buffer ..... 20
3.3.3 Modifikace selection bufferu ..... 20
3.3.4 Použité způsoby uchopovacího módu ..... 21
3.3.5 Výběr uchopovacího bodu ..... 21
3.4 Krokovací manager ..... 21
3.5 Grafické uživatelské rozhraní ..... 22
3.5.1 Uživatelské rozhraní hladin ..... 24
3.5.2 Pracovní plocha ..... 25
3.5.3 Souřadnicový systém ..... 26
3.6 Práce se souborem ..... 26
3.6.1 Příklad souboru ..... 27
3.6.2 Ukládání výkresové dokumentace ..... 28

3.6.3 Načítání výkresové dokumentace....28
3.7 Jádro aplikace....28
3.8 Implementační nástroje....29
3.8.1 OpenGL....29
3.8.2 Knihovna JOGL....30
3.8.3 Správa verzí....31
3.8.4 Reprezentace výkresové dokumentace....31
3.8.5 Grafické uživatelské rozhraní....31
4 Výsledky....32
4.1 Dosažené výsledky....32
4.1.1 Neimplementované cíle....33
4.2 Entity a hladiny....33
4.3 Modifikace....34
4.4 Výběrový mód....34
4.5 Uchopovací mód....34
4.6 Příkazová řádka....35
5 Závěr....36
Literatura....37
Seznam příloh....38
Příloha 1. Obsah přiloženého CD....39

# 1 Úvod

Doby kdy se architekti učili na školách kreslení čar jsou pryč a na místo toho se učí kreslit za pomoci počítačových kreslících nástrojů, kterými jsou CAD systémy. Hlavní výhodou kreslení pomocí CAD systémů je oprava případně vzniklé chyby. Představte si, že kreslíte rozsáhlý výkres. Jste téměř u konce, ale najednou vám ujede ruka a celý výkres je tím znehodnocen. Jediné řešení je začít znovu. Toto se vám užitím CAD systémů nemůže stát. Jejich další výhodou je snadná orientace v rozsáhlých výkresech a schopnost mít veškeré podklady k výkresu na jednom místě. Jedním takovým příkladem je kompletní nákres budovy, který může obsahovat půdorys, bokorys, schéma rozvodů elektřiny a vodoinstalace. Systémy v tomto případě pak dokáží zobrazit jen požadovanou část.

Cílem práce je seznámení a analýza problematiky CAD systémů, zaměřená především na problematiku dvourozměrného kreslení. Dále pak návrh a realizace řešení CAD systému. Výstupem této práce by měla být samostatná aplikace realizující navržené řešení.

# 2 Rozbor CAD systémů

V této kapitole se budeme zabývat úvodní problematikou CAD systémů (co to jsou CAD systémy, k čemu slouží a kde nacházejí uplatnění). Uvedeme si jejich základní rozdělení. Dále se zaměříme na požadavky od těchto systémů a částečně nastíníme řešení těchto požadavků. V neposlední řadě se budeme zabývat základními kreslícími prvky. Nakonec pár slov o grafickém uživatelském rozhraní.

## 2.1 Charakteristika

CAD systémy jsou moderní nástroje pro konstruování. Oblast CAD je jen jednou součástí nasazení výpočetní techniky v průmyslu. Souhrnně je toto nasazení označeno CA. Zkratka CA znamená „*Computer Aided*“ – *počítačová podpora*[1]. CAD systémy jsou především používány v prvotních etapách výrobního procesu za účelem efektivního a přesného návrhu. Nacházejí využití v mnoha oborech jako je automobilový a letecký průmysl, strojírenství, architektura, ale také se uplatňují při tvorbě různých animací, kde pomocí křivek umožňují definovat dráhu pohybu různých těles. Vzhledem k jejich veliké rozšířenosti byli také hnací silou pro vývoj počítačové grafiky. CAD systémy do procesu konstruování vnášejí různé výhody jako je snadná spolupráce mezi zainteresovanými spolupracovníky, snadná tvorba velkého počtu variant a modifikací návrhu.

Podle zaměření lze CAD systémy rozdělit na:

- **Otevřené** – Umožňují flexibilní použití systému pro různé oblasti.
- **Specializované** – Jsou přizpůsobeny pro určitou oblast zaměření (ORCAD - elektrotechnika, ArchiCAD – architektonika, Invertor – strojírenství)

Podle způsobu práce v ploše, nebo v prostoru se CAD systémy dělí na:

1. **2D** – Základním konstrukčním prvkem u těchto systémů je úsečka, ale také poskytují kompletní knihovnu geometrických entit. Jejich schopnostmi je provádět šrafování specifických uzavřených oblastí a kótování definovaných objektů.
2. **3D** – Tyto systémy mají v dnešní době převahu nad čistými 2D systémy. Jejich využití může být i pro 2D modelování, primárně je však každý objekt reprezentován prostorovými souřadnicemi. Výstupem 3D CAD systémů není už jen výkresová dokumentace, ale kompletní 3D model součásti. Každý 3D systém zpravidla obsahuje prostorové modelování, které může provádět jedním ze způsobů:
   - Drátový model – Jedná se o kostru objektu. Tento model neobsahuje plochy, je vytvořen pouze z bodů, úseček a křivek definujících hrany objektu. Tento typ modelování může

být časově náročný, protože každý objekt, z něhož se drátový model skládá, musí být nezávisle nakreslen.

- Plošné modelování – Výsledný model nedefinuje pouze obrysové hrany, ale také plochy, které tyto hrany představují.
- Objemové modelování – Při tomto modelování vytváříme výsledný model pomocí 3D útvarů (kvádr, kužel, válec,..). Tyto útvary můžeme kombinovat a vytvořit složitější objemová tělesa.

Grafické aplikace se na základě principu práce s grafickými prvky dělí na rastrové a vektorové editory. Všechny CAD systémy patří do skupiny programů využívající vektorovou grafiku. Rastrové editory pracují vždy jen s grafickým prvkem bod a ukládají si jeho pozici a barevnou hloubku. Naopak vektorové editory využívají jako základní prvek úsečku, o které si uchovávají souřadnice počátečního a koncového bodu, barvu a její styl vykreslení.

V současnosti všechna grafická zařízení pracují s rastrovým zobrazením. Proto je potřeba pro zobrazení vektorové grafiky provádět tzv. *rasterizaci*. Rasterizace je proces převodu vektorové reprezentace dat na jejich rastrovou formu s cílem dosáhnout maximální možné kvality a rychlosti výsledného zobrazení [3].

### 2.1.1 Požadavky

CAD systémy slouží pro zjednodušení práce návrhářům, architektům, animátorům aj. Proto jsou na ně kladeny různé požadavky. Společnými požadavky pro většinu systémů jsou:

- **Přesnost** – V procesu konstrukce je důležitým parametrem, který se snažíme maximalizovat. Dosažení přesného návrhu lze přímým zadáváním bodů do výkresu. To je umožněno např. pomocí příkazové řádky, která mimo přímého zadávání bodů může navíc sloužit jako průvodce procesem tvorby výkresu a zjednodušit uživateli orientaci v systému. Dalším způsobem může být tzv. uchopovací mód, který vybírá již existující body ve výkresu.
- **Organizace** – Jelikož v dnešní době jsou CAD systémy využívány v mnoha oborech, ve kterých požadujeme čím dál tím rozsáhlejší dokumentace, vyvstává nám potřeba jisté organizace návrhu. Proto většina systémů podporuje vrstvení výkresu pomocí tzv. *Hladin*.
- **Editace** – Nedílnou součástí každého CAD systému je proces editace entity, nebo modifikace kolekce entit. Modifikace nám slouží pro jednoduché, přesné a rychlé kreslení jinak složitých objektů.

### 2.1.2 Hladiny

Hladiny slouží k organizaci a orientaci ve velkých konstrukčních projektech. Lze si je představit jako pauzovací papíry naskládané na sebe, které leží rovnoběžně s obrazovkou. Každá hladina může obsahovat různé informace a lze ji kdykoli zneviditelnit k účelu zlepšení orientace ve výkresu a ke zrychlení jeho vykreslování. Uživatel pak také pracuje se zjednodušenou verzí své práce a nemůže omylem zasáhnout do neviditelné hladiny.

Důležitými atributy hladin jsou:

- **Jméno** – Slouží jako identifikátor, podle kterého se dají hladiny vyhledávat, setřiďovat, nebo nastavit hladinu jako aktivní (právě používána pro nově zadávané entity).
- **Viditelnost** – Vypnutí viditelnosti hladiny způsobí, že entity v ní obsažené nebudou vykreslovány. Systémy s těmito hladinami nadále pracují. Budeme-li prohledávat z nějakého důvodu databázi entit výkresu, budou se prohledávat i neviditelné hladiny.
- **Zmrazení** – Stejně jako neviditelné hladiny i hladiny zmražené nejsou na plochu vykreslovány. Podstatným rozdílem je, že systém tyto hladiny úplně vyřadí. Prohledávání databáze výkresu pak bude probíhat mnohem rychleji.
- **Zámek** – Slouží pro možnost povolení, nebo zakázání editace entit uvnitř hladiny. Pokud je hladina zamčená můžeme do ní přidávat nové entity, ale nemůžeme tyto entity editovat.
- **Barva, styl a tloušťka čar** – Tyto atributy jsou využity pro rozpoznání hladin na pracovní ploše. Jejich hodnota je přednastavená, později je možné je editačními příkazy upravit.

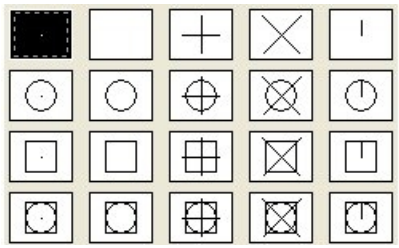
CAD systémy musí zajistit, aby se každý objekt výkresové dokumentace nacházel právě v jedné hladině. Proto každý systém po vytvoření nového výkresu definuje základní, nemazatelnou hladinu pevného jména (Autocad hladina „0“, 4MCAD definuje „0“ a spoustu jiných předdefinovaných).

### 2.1.3 Základní entity

Základními entitami rozumíme takové prvky pomocí niž můžeme definovat větší, rozsáhlejší celky. Entita je často používaný pojem pro souhrnné označení kreslícího nástroje. Každá entita je přiřazena k nějaké existující hladině. Nelze tedy nakreslit entitu bez jejího zařazení do hladiny. S tímto se také pojí vykreslení entity, podle vlastností předepsaných v příslušné hladině. Atributy vykreslení každé entity jsou převzaty z hladiny v níž je entita definována. Důležitou vlastností entit a hladin je možnost přemístit jakoukoli entitu do jiné hladiny a tím změnit atributy vykreslení entity.

Základními entitami jsou:

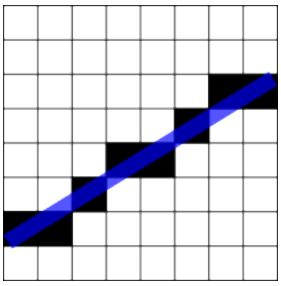
- **Bod**: Je tou nejjednodušší entitou CAD systémů, je definován pouze souřadnicemi sebe samého. Jeho vykreslení už ale není tak jednoduché, protože pokud bychom vykreslili pouze jeden rastrový bod obrazovky, bylo by obtížné kvůli jeho velikosti s ním dále pracovat. Proto ve většině CAD systémů se vykreslí pomocí definované značky.



Obrázek 2.1: Příklad vykreslení bodu v programu AutoCAD

- **Úsečka**: Je definována dvěma koncovými body, nebo může být definována jedním koncovým bodem, její délkou a sklonem vůči nulovému úhlu. Vykreslení úsečky do rastru lze provést pomocí následujících algoritmů [3]:
    1. DDA algoritmus
    2. Error control DDA
    3. Bresenhamův algoritmus

  Při rasterizaci úsečky dochází k nežádoucímu jevu skokové změny zobrazení úsečky v rastru viz Obrázek 2.2. Odstranění tohoto nežádoucího jevu je možné nastavením vyššího rozlišení zobrazovacího zařízení, nebo úpravou rasterizačních algoritmů. Zvýšení rozlišení je limitováno počtem zobrazovacích bodů zařízení. Proto se používají principy, pro upravování jasu pixelů tvořících úsečku.



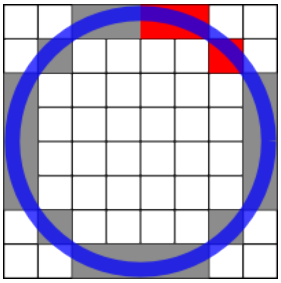
Obrázek 2.2: Rasterizace úsečky

- **Kružnice**: Je definována jejím středovým bodem a poloměrem, nebo může být definována dvěma protilehlými body na kružnici jejichž vzdálenost udává průměr. Poloměr a souřadnice

středu jsou pak dopočítány z těchto bodů. Pro zjednodušenou práci s kružnicí se v CAD systémech přidávají další editační body na kvadratických bodech kružnice. Pro rasterizaci kružnice lze použít tyto algoritmy [3]:

1. Varianta DDA pro kružnici – vykreslení kružnice jako N-úhelník
2. Midpoint

Na obrázku 2.3 je ukázán příklad rasterizace, kde stačí vypočítat pouze osminu bodů v prvním kvadrantu (červeně znázorněné). Zbylé body získáme prohozením souřadnic[3].



Obrázek 2.3: Rasterizace kružnice

- **Elipsa**: Je dána středovým bodem, hodnotami hlavní a vedlejší poloosy a sklonem vůči ose x. Nebo může být definována obdélníkem, který je elipse opsaný. Souřadnice středu se v tomto případě dopočítají z protějším vrcholů, délka hlavní a vedlejší poloosy je dopočítána ze vzdálenosti středového bodu od příslušných bodů dotyku elipsy s obdélníkem. V tomto případě elipsa má sklon stejný jako je sklon obdélníku. Výpočet rastrových bodů elipsy se provádí pouze v jednom kvadrantu, podobně jako tomu bylo u kružnice. Zbylé body lze získat prohozením souřadnic.
- **Oblouk**: Může se vyskytovat ve dvou verzích kružnicový a eliptický oblouk. U eliptického oblouku jsou navíc parametry délka hlavní, nebo vedlejší poloosy a sklon elipsy. Oblouk může být zadán mnoha způsoby středem, poloměrem, počátečním a koncovým úhlem, dále také koncovým bodem, bodem ležícím v polovině oblouku a druhým koncovým bodem.
- **Obdélník**: Obdélník je základní entitou i když jej můžeme vykreslit pomocí dílčích úseček. Toto je proto, že požadujeme práci s obdélníkem jako s celkem. V systémech zadáme obdélník pomocí souřadnic protilehlých vrcholů.
- **Polygon**: Slouží k jednoduchému kreslení pravidelných mnohoúhelníků. Polygon můžeme nakreslit za pomoci kružnice opsané, nebo vepsané se specifickým počtem vrcholů. Stejně jako s obdélníkem i s polygonem požadujeme pracovat jako s celkem.
- **Křivka**: Je jedním z velmi důležitých komponent moderních výkresů. Křivky nám umožňují definovat font písmen, povrchy ploch a také nacházejí uplatnění v počítačových animacích, kde definují trajektorii různých objektů.

Základními druhy křivek jsou:

– Interpolační: křivka přímo prochází svými řídícími body.

– Aproximační: křivka neprochází řídícími body, ale je k nim pouze aproximována.

Další dělení křivek na [11]:

– Racionální: Každý řídící bod obsahuje váhový koeficient určující přimknutí křivky k bodu.

– Neracionální: Tvar křivky je ovlivněn pouze polohou řídících bodů.

Jednotlivé křivky můžeme za sebe spojovat, čímž vzniká tzv. *Spline* (spline je po částech polynomiální křivka [11]). Při spojování křivek určujeme jak k sobě dílčí segmenty přimykají pomocí krajních bodů segmentů a derivací v těchto bodech.

Požadavky na modelování křivek v CAD systémech jsou [1]:

– Plně řídit tvar křivky jednoduchými nástroji.

– Možnost vytváření ostrých hran.

– Zajištění spojitosti

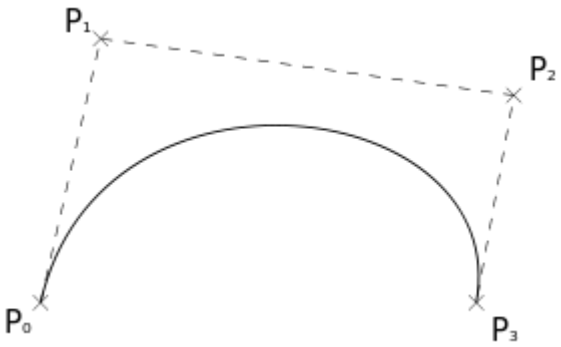
Nejznámější autoři v oblasti křivek:

– Ferguson: Zavedl interpolační křivku danou dvěma koncovými body a tečnými vektory v těchto bodech.

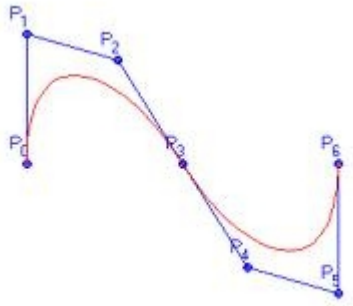
– Beziér: Zavedl aproximační křivku zadávanou sítí bodů [1]. Snadné intuitivní ovládání.

– Coons: Měl hlavní význam pro teorii spline křivek.

V CAD systémech je nejpoužívanější křivkou *NURBS*, které mají popis založený na náročném matematickém aparátu. Výhodou je naprosto přesný popis tvaru [1].




Obrázek 2.4: Kubická Beziérova křivka



Obrázek 2.5: Napojení Beziérových křivek

- **Text**: Je dán souřadnicemi vrcholových bodů opsaného obdélníku, obsaženým textovým řetězcem, stylem a velikostí písma. Typ písma je buďto převzat z typů definovaných v operačním systému, nebo používají vlastní typ, který je vykreslen pomocí úseček.

### 2.1.4 Modifikace

Modifikace jsou operace prováděné nad jednou, nebo nad více entitami. Slouží pro zrychlení a zjednodušení práce koncovým uživatelům. Jedná se především o různé úpravy již existujících entit. Výsledkem modifikace může být i entita nová. Mezi typické modifikace patří:

- **Odstranění –** Provede odstranění požadovaných entit z výkresu.
- **Kopie –** Slouží pro vytvoření kopie vybraných entit a jejich následný posun na pozici danou zvoleným referenčním bodem.
- **Posun –** Modifikace umožňující přesný posun vybraných entit na pozici vůči zvolenému referenčnímu bodu.
- **Rotace –** Provádí natočení zvolených entit o daný úhel vůči zvolenému středu rotace
- **Měřítko –** Změní velikost vybraných entit o zvolený faktor měřítka vůči danému referenčnímu bodu.
- **Zrcadlení –** Slouží pro vytvoření zrcadlené kopie zvolených objektů vůči definované přímce.
- **Ořez –** Modifikace, která provede oříznutí zvolené entity o její část mezi body průsečíků s definovanými hraničními objekty.
- **Prodloužení –** Je to opak modifikace ořez. Provádí prodloužení zvolené entity po body zdánlivých průsečíků.
- **Protažení** – Přesune množinu entit avšak zachová vazbu přesouvaných prvků na prvky ostatní.
- **Pole** – Vytvoří mnohonásobné kopie objektů uspořádané do pravoúhlého pole (zadává se počet řádků, sloupců a vzdálenost mezi nimi), nebo do kruhového pole (zadává se střed, počet prvků a vyplňovaný úhel).
- **Zaoblení** – Provede spojení dvou úseček, oblouků, nebo kružnic pomocí oblouku o zadaném poloměru.
- **Ekvidistanta** – Vytvoří k zadané křivce paralelní čáru v zadané vzdálenosti na zadané straně.

## 2.2 Uchopovací mód

Uchopovací mód CAD systémů slouží pro výběr existujících bodů ve výkrese. Systém musí prohledat okolí kursoru pro nalezení nejbližší entity, u které pak vyhledá požadovaný, nebo nejbližší uchopovací bod.

### 2.2.1 Uchopovací body

Slouží pro rychlé a přesné kreslení pouze za použití myši. Umožňují přichytávat řídící body entit na pozici tohoto bodu a tím ulehčují práci se zadáváním souřadnic do příkazové řádky. Navíc některé z těchto bodů je zapotřebí dynamicky počítat, čímž odpadá nutnost dopočítávat souřadnice ručně. Nastavením výčtu uchopovacích bodů může uživatel specifikovat typy bodů, které budou vybírány. Typickými uchopovacími body jsou:

- **Koncový –** Jedná se o okrajové body úsečky, nebo oblouku a vrcholy polygonů a obdélníků.
- **Polovina –** Bod nacházející se v polovině úsečky, nebo oblouku.
- **Střed –** Středový bod oblouků, kružnic a elips.
- **Kvadrant –** Body ležící na kružnici, oblouku, nebo elipse oddělující její kvadranty (0°, 90°, 180°, 360°). Dají se dopočítat přičtením/odečtením poloměru.
- **Kolmice –** Pata kolmice z posledního zadaného bodu na požadovaný objekt. Kolmici lze tímto bodem pouze spustit nikoli vztyčit.
- **Tečna –** Váže se na bod ležící na kružnici, elipse, nebo oblouku tak, že jeho spojnice s posledním zadaným bodem je tečnou k dané entitě.
- **Průsečík –** Průsečík dvou objektů.
- **Zdánlivý průsečík** – Fiktivní průsečík dvou entit. Bod, ve kterém by se entity proťali, kdyby byli delší.
- **Protažení** – Pomyslný bod po protažení úsečky, nebo oblouku.
- **Nejblíže** – Bod, který leží na ukázané entitě a je nejblíže bodu ukázání. Používá se k přesnému ukázání na čáru.
- **Rovnoběžně** – Umožňuje nakreslení rovnoběžky s již existující úsečkou.

## 2.3 Ortogonální a polární mód

V těchto režimech CAD systémy zadávají souřadnice bodů po kroku definovaném pomocí úhlu. Tyto režimy lze kdykoli vypnout a zase zapnout podle potřeby. V ortogonálním módu systém fixuje zadávání bodů, tak aby spojnice vstupních bodů byla rovnoběžná s jednou ze souřadnicových os. Naopak v polárním módu systémy umožňují nastavení krokovacího úhlu. Některé systémy v tomto módu nefixují pohyb, ale pouze při přiblížení se tomuto úhlu dojde k přitažení.

## 2.4 Výstupní soubor

CAD systémy produkují jako výstup výkresovou dokumentaci definovanou v nějakém podporujícím souborovém formátu. Tento formát byl měl být co nejlépe přenositelný mezi více systémy. Tato kapitola vychází z literatury [6].

Standardem pro přenos dat mezi CAD systémy je formát DWG, od společnosti AutoDesk. Slouží pro přenos dat ve 2D a částečně i 3D systémech. Jedná se o nativní soubor, kde data jsou zapsána v binární podobě pro rychlejší přístup.

Dalším z vhodných formátů je DWF. Je to moderní formát pro sdílení a integraci dat v CAD systémech. Podporuje 2D i 3D výkresy, distribuované zpracování a work-flow výkresové dokumentace.

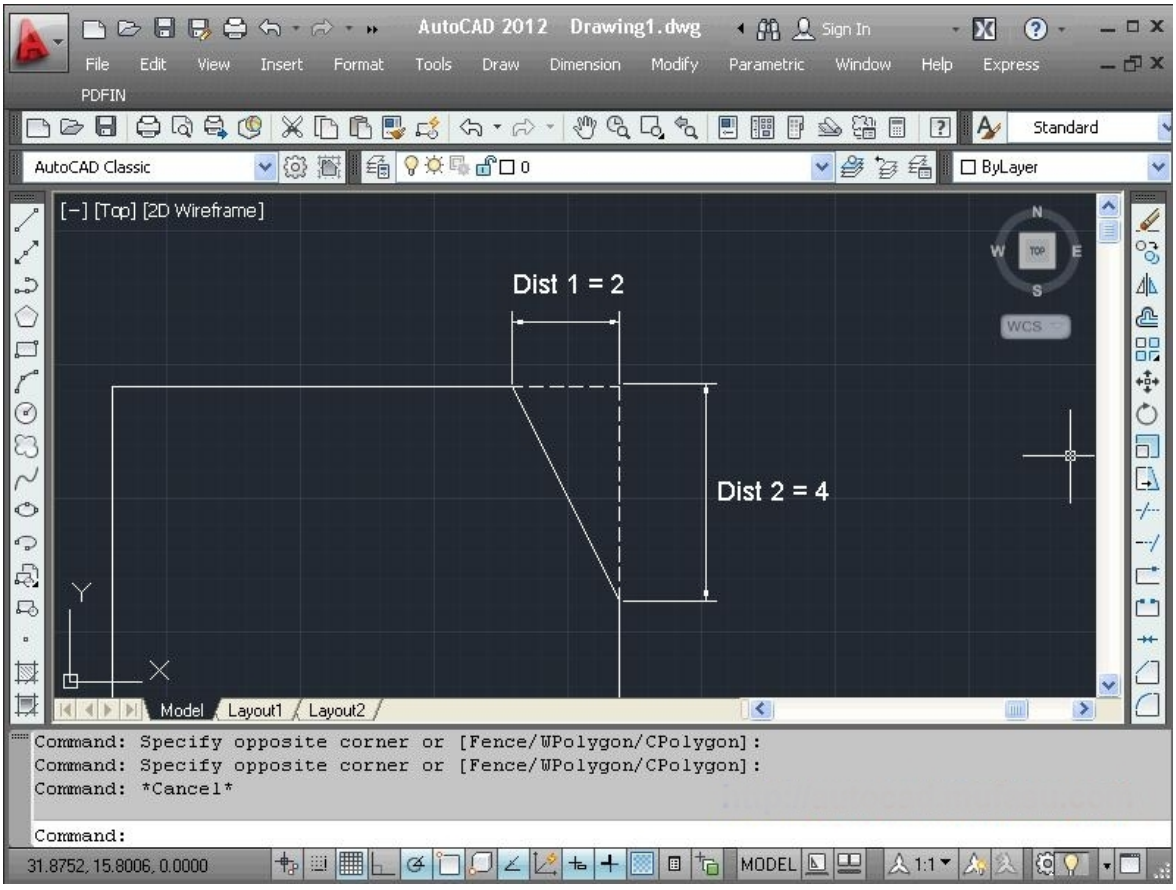
## 2.5 Grafické uživatelské rozhraní

Jako každý vektorový editor i CAD systémy potřebují pro snadnou práci efektivní, intuitivní a uživatelsky přístupné grafické rozhraní (zkráceně GUI – *graphical user interface*). Toto rozhraní musí uživatele dostatečně informovat o stavu aplikace. Jelikož dobré GUI urychluje práci se systémem je zapotřebí specifikovat co nesmí chybět. Kapitola vychází z materiálu k předmětu Vizualizace a CAD, Fakulty informačních technologií VUT v Brně [5].

- **Pracovní plocha** – Je nejdůležitějším prvkem GUI CAD systémů, s jejíž pomocí jsou entity obsažené ve výkrese zobrazovány uživatelovi. Logicky zabírá největší část. Pro urychlení zobrazení by plátno mělo být softwarově, nebo hardwarově akcelerované. Také ovladatelnost scény musí být zjednodušená. Posuv scény většina systémů provádí typem „*drag and drop*“ (uchop, posuň a pusť), změnu rozměrů pohledu přiblížení/oddálení scény zas definuje jako posuv směrem blíže/dále k pozici kursoru za pomoci kolečka na myši.
- **ToolBar** – Jedná se o panel, který obsahuje různé grafické komponenty. Jeho výhodou je snadná změna umístění v okně aplikace, čímž dovolujeme uživateli přizpůsobit si toto rozhraní. Především slouží ke zjednodušení přístupnosti velmi často používaných ovládacích prvků. CAD systémy využívají toolbary k efektivnímu přístupu velmi často používaných tlačítkek pro výběr entity, práci s hladinami, modifikace, uložení dokumentace aj.
- **Příkazová řádka** – V systému by nemělo chybět přesné zadávání souřadnic od uživatele. Toho se dociluje pomocí příkazové řádky, která často navíc informuje uživatele o aktuálním kroku. Jejím primárním účelem je zadávání příkazů a parametrů entit.

- **Editační panel** – Další často používanou grafickou komponentou pro editaci entit. Obsahuje veškeré editační parametry každé entity a jeho obsah se liší podle typu entity.
- **StatusBar** – Neopomíjenou vlastností dobrého GUI je tzv. *Stavový řádek* (nebo-li *StatusBar)*, který informuje uživatele o stavu jednotlivých akcí např. došlo ke správnému uložení, či otevření výkresu, nebo pro zobrazení souřadnic pozice kursoru ve výkresové oblasti.
- **Klávesové zkratky** – Jsou důležitou součástí rychlého a efektivního ovládání programů. Klávesové zkratky jsou použity pro často prováděné operace jako je výběr entity, otevření a uložení souboru a jiné další.

Standardními ovládacími prvky GUI jsou klávesnice a myš. Pomocí myši provádíme výběr entity, zadávaní bodů přímo na pracovní ploše, editaci entit aj. Klávesnice především slouží k zadávání přesných hodnot souřadnic bodů, jména hladiny, pro potvrzení výběru, odstranění entit, nebo pro ukončení zadávání bodů. Rozšířenými ovládacími prvky jsou pak navigátoři a piloti. Ukázka GUI CAD systému AutoCAD 2012 je na obrázku 2.6.




Obrázek 2.6: Ukázka GUI (AutoCAD 2012)

# 3 Návrh a implementace řešení

V této kapitole se budeme zabývat požadavky na cílovou aplikaci. Uvedeme si možná řešení jednotlivých problémů. Poslední uvedené řešení bude použito ve výsledné aplikaci. Rozebereme si jak reprezentovat datové objekty v paměti, jak provádět jejich rozpoznání a výběr na obrazovce. Provedeme návrh uživatelského rozhraní. Vysvětlíme si jak fungují funkce zpět a vpřed. Nakonec nás čeká vysvětlení, jak pracuje jádro aplikace a použité implementační prostředky.

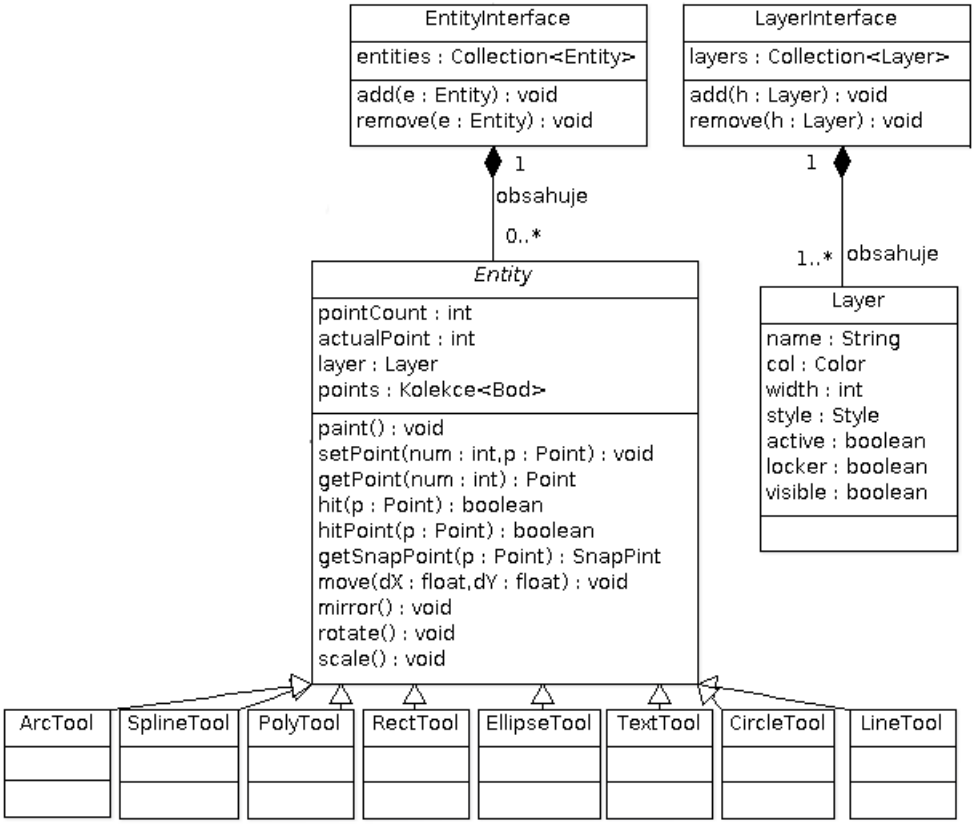
## 3.1 Cíle řešení

Našim cílem je navrhnout 2D CAD systém, který bude přinejmenším splňovat základní vlastnosti popsané v kapitole 2. Tedy bude se jednat o plně funkční grafický editor umožňující práci se základními entitami výkresu tj. přidání entity do výkresu, její výběr a editaci. Systém bude podporovat následující entity – úsečka, kružnice, oblouk, elipsa, obdélník, polygon, křivka a text. Bude zobrazovat námi požadovanou část výkresu za pomoci hardwarově akcelerovaného plátna s podporou oddálení, přiblížení a posuvu zobrazeného výkresu. Dále by měl obsahovat organizaci výkresu pomocí vrstvení do hladin a umět přidat novou hladinu, odstranit a také upravit již existující hladiny výkresu. Za parametry hladin budeme považovat jméno hladiny, sloužící jako identifikátor pro výběr, barvu, styl a tloušťku čáry, viditelnost a zámek. Styly čáry převezmeme z technického kreslení a to konkrétně plnou, čárkovanou, čerchovanou a dvojitě čerchovanou. Program bude implementovat jednotlivé modifikace entit rotace, kopie, posun, zrcadlení, změna měřítka a ořezávání. Dalším požadavkem aplikace je uchopovací režim, který by měl podporovat základní uchopovací body jako jsou koncové body entit, polovina úsečky a oblouku, střed kružnice či elipsy, kvadrant kružnice a elipsy, tečna a průsečík. Dále by také neměla chybět možnost krokovat nazpět/vpřed a uložení/načtení výkresu ze souboru. Samozřejmostí je také příkazová řádka aplikace.

## 3.2 Reprezentace dat

Datovou reprezentaci v programu budeme volit dle dříve specifikovaných požadavků. Našim nejdůležitějším požadavkem je přesnost, proto budeme volit 32-bitová čísla s plovoucí čárkou, což je dostačující pro splnění požadavku. Abychom mohli přistupovat k entitám nezávisle na jejich typu vytvoříme si abstraktní třídu pro definici společných vlastností všech entit. Tím také umožníme snadné rozšíření programu o nový typ entity, pro kterou stačí vytvořit vlastní třídu. Entity jsou definovány souřadnicemi jejich řídících bodů a případně vzdáleností. Konkrétní typ entity vytvoříme pomocí dědičnosti abstraktní třídy a implementací požadovaných metod.

Pro zajištění organizace výkresu je potřeba vytvořit třídu, která nám bude představovat jednotlivé hladiny. Každé přidáme atributy dle již dříve specifikovaných požadavků hladin. Na obrázku 3.1 je ukázka diagramu tříd popisující reprezentaci dat. Třída `Entity` reprezentuje abstraktní třídu pro entity. Její znázorněné metody jsou abstraktní, které musí každý typ entity definovat. Dále pak je vidět třída rozhraní pro práci s entitami `EntityInterface`. Toto rozhraní definuje kolekci objektů výkresové dokumentace. Další znázorněnou třídou je třída `Layer` a její rozhraní `LayerInterface`.




Obrázek 3.1: Reprezentace dat

### 3.2.1 Reprezentace entit

Základní datovou jednotkou v naší aplikaci je nějaký typ vektorové entity, který je potomkem abstraktní třídy. Tato třída bude disponovat statickými metodami především pro výpočet průsečíků entit. Hlavním důvodem zavedení této třídy je implementace metod pro nastavení společných vlastností entit a jednotný přístup ke všem entitám. Vlastnosti abstraktní třídy jsou:

- Definovat proměnné reprezentující společné atributy entit a metody pro jejich čtení, případně i zápis. Např. proměnná pro určení zda je entita vybrána, velikost zobrazovaných madel atd.

- Pomocí datového typu `enum` produkovat výčet všech typů entit, toho se bude využívat především v jádru aplikace. Ale také každý potomek této třídy bude obsahovat tuto hodnotu určující její typ a metodou pro získání hodnoty. Proto při tvorbě nového typu je nutné tuto hodnotu definovat nastavením proměnné.
- Uchovávat si index právě zasaženého bodu entity metodou `hitPoint`, pro jeho pozdější nastavení pomocí metody `setHitPoint`. Toto nám umožní dynamické úpravy souřadnic řídících bodů entit.
- Další její funkcí bude získání opsaného obdélníku, pro určení rozměrů na pracovní ploše. K tomu využijeme metodu, která bude procházet všechny řídící body entity a specifikovat velikost opsaného obdélníku. Tato vlastnost entit bude sloužit pro určení velikosti výkresu.
- Některé typy entit, především pro účel podpory jejich zadávání, potřebují vědět jestli je toto zadávání již hotovo. K tomu bude sloužit proměnná `done` a metoda `settingDone`.
- V neposlední řadě bude určovat abstraktní metody, které musí každý její potomek dodefinovat v závislosti na konkrétním typu entity.

Protože výsledná aplikace bude využívat grafické knihovny OpenGL (popsané v kapitole 3.8.1), musíme tomu přizpůsobit návrh entit. Hlavně způsob jejich vykreslování, který si každý typ entity určuje sám. Každá konkrétní entita si bude uchovávat výčet všech uchopovacích bodů kterými disponuje.

Nyní si popíšeme jednotlivé typy entit v naší aplikaci:

- **Úsečka** – Bude definována dvěma koncovými body a bodem v její polovině. Uchopovací body úsečky jsou koncový, polovina a kolmice. Úsečku budeme kreslit zadáním dvou bodů na ploše, nebo do příkazové řádky.
- **Kružnice** – Bude definována třídou `CircleTool` pomocí středového bodu a poloměru. Její zadávání bude výběrem dvou bodů na plátně, nebo pomocí příkazové řádky, kde od nás budou očekávány souřadnice středového bodu a poloměr. Nebo také lze využít kombinace obou dvou způsobů. Protože OpenGL neumí přímo vykreslit kružnici, budeme provádět vykreslení jako mnohoúhelník. Souřadnice bodů tohoto mnohoúhelníku vypočteme pomocí matematických funkcí a uložíme si je do *vertex array*, což je pole vrcholů sloužících především pro snížení počtu volání OpenGL funkcí a zrychlení přenosu dat do grafické karty. Kružnice disponuje třemi typy uchopovacích bodů středovým, kvadrantem a tečnou. Tečné body vůči jednomu vztažnému bodu může mít kružnice dva.
- **Oblouk** – Bude reprezentován třídou `ArcTool` pomocí středového bodu, poloměru, počátečního a koncového úhlu. Jeho vykreslení budeme provádět proti směru hodinových ručiček od počátečního úhlu po koncový. A podobně jako u kružnice využijeme vykreslení

mnohoúhelníkem. Oblouk budeme zadávat pomocí tří bodů na plátně konkrétně středovým bodem, bodem udávajícím poloměr a zároveň počáteční úhel a nakonec bodem specifikujícím koncový úhel. Nebo také pomocí příkazové řádky, či kombinací obou možností. Uchopovacími body oblouku jsou koncové body, středový bod, polovina oblouku a také tečna.

- **Elipsa** – Elipsu budeme chtít kreslit přímo pod určitým sklonem. Tím nám odpadá nutnost dodatečné rotace. Budeme ji definovat ve třídě `EllipseTool` pomocí souřadnic středového bodu a bodů na kvadrantech. Zadání provedeme v pořadí středový bod, bod určující šířku a sklon elipsy, a nakonec bod pro výšku. Nebo pomocí příkazové řádky. Vykreslení elipsy je stejný problém jako vykreslení kružnice popsané výše. Obsahuje také úplně stejné uchopovací body.
- **Obdélník** – V systému bude definován pomocí souřadnic vrcholových bodů ve třídě `RectTool` a bude vykreslen jako uzavřený objekt pomocí úseček. Zadávání budeme provádět dvěma protilehlými body. Při editaci jeho souřadných bodů nebude docházet k dopočítávání zbylých souřadnic vrcholů, ale pouze budeme upravovat pozici editovaného bodu, podobně jako ostatní CAD systémy. Uchopovacími body obdélníku jsou koncový a kolmice.
- **Polygon** – Je entitou sloužící pro kreslení pravidelných mnohoúhelníků, definovaných počtem vrcholů, středovým bodem a souřadnicemi vrcholových bodů. Zadání počtu vrcholů budeme provádět pomocí příkazové řádky. Pak budeme zadávat střed a poloměr opsané kružnice. Uchopovací body polygonu jsou koncový (body ve vrcholech), střed a kolmice.
- **Text** – Bude definován bodem určující levý horní roh textového okna, obsaženým textem, velikostí a fontem písma ve třídě `TextTool`.
- **Spline** – Bude definován souřadnicemi jeho řídících bodů ve třídě `SplineTool`. Spline budeme implementovat pomocí Beziérových křivek čtvrtého řádu.

### 3.2.2 Reprezentace hladin

Pro zajištění organizace dat slouží hladiny definované ve třídě `Layer`. Dvoustavové atributy hladiny budeme reprezentovat typem `boolean`. Barva hladiny bude reprezentována pomocí OpenGL *vertex array*, což je pole pro zrychlení přenosu dat do grafické karty. Pro styl čar hladiny vytvoříme samostatnou třídu, která bude obsahovat dvě různé definice typů. Jeden typ bude využíván pro zobrazení v dialogovém okně hladin dle specifikací použitého programovacího jazyka. Druhý typ pak bude sloužit pro vykreslení pomocí OpenGL. Tento typ obsahuje dvě čísla. Prvním je tzv. *Pattern* typu `short`. Pattern v binární podobě určuje kdy bude čára a kdy bude volné místo. Nejlépe nám to popíše příklad, kde pattern je 0x0f0f v binární podobě „0000 1111 0000 1111“. Pak na místech kde je

„1“ bude kreslit čáru a na místech kde je „0“ zas nebude kreslit. Druhé číslo je *factor* typu `int` a určuje délku jednoho segmentu. Oba tyto typy čar pak zapouzdříme do třídy `LineTypes`.

### 3.2.3 Navázání entity na hladinu

Nyní už máme vytvořenou reprezentaci pro hladiny a entity, budeme řešit problém, jak připojit tyto hladiny k jednotlivým entitám. Hned z počátku se nám nabízejí dvě metody. Jednou takovou metodou je přidat kolekci entit ke každé hladině. Třída reprezentující celkový výkres pak bude obsahovat pouze seznam všech hladin. Pokud bychom prováděli operaci změny hladiny nad entitou, budeme muset odstranit entitu z kolekce v jedné hladině a přidat ji do té druhé. Další metodou je přidat každé entitě atribut, určující do jaké hladiny náleží. V tomto případě bude celkový výkres obsahovat dvě kolekce jednu pro hladiny druhou pro všechny obsažené entity. Pokud bychom u tohoto způsobu chtěli změnit hladinu entity, stačí pouze upravit hodnotu atributu.

V naší aplikaci použijeme druhý způsob využívající atributu hladiny u každé entity. Tento způsob se jeví jako lepší, protože nám dovoluje pracovat s entitami nezávisle na její hladině. Jedním z nedostatků tohoto způsobu je nutnost kontroly viditelnosti hladiny při vykreslování.

### 3.2.4 Datové struktury

V této kapitole se podíváme blíže na možnosti vhodných datových struktur pro kolekci entit a hladin. Ze všech možných nejlépe vyhovují pole, seznamy a seznam implementovaný polem. Pro výběr nejvhodnější struktury musíme nejprve specifikovat, jaké operace jsou nejčastější nad danými kolekcemi, potom se budeme moci rozhodnout jaké typy použít. Určitě nejčastější operací nad kolekcí entit bude sekvenční procházení při vykreslování, potom budeme často potřebovat vyhledat příslušný prvek pro jeho editaci, nebo odstranění a nakonec přidání a odstranění prvku z kolekce.

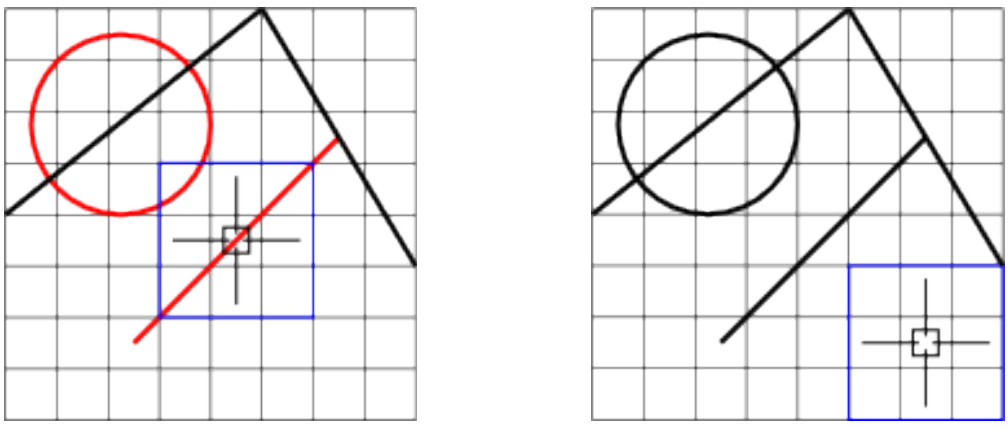
1) **Pole –** Při použití pole získáme rychlé sekvenční procházení kolekce a přímý přístup k prvku použitím indexu. Jedinou jeho nevýhodou v našem případě je mazání prvků, při němž dochází k posuvu všech elementů za odstraněným prvkem.
2) **Seznam –** Je vhodný pro časté odstraňování prvků z kolekce, kde dochází pouze k přepisu ukazatelů. Ale nedisponuje možností přímého přístupu k prvkům.
3) **Seznam implementovaný polem –** Jakýmsi kompromisem mezi kolekcí sezname a pole. Jedná se o pole jehož prvky obsahují ukazatele na jiný další prvek, navíc obsahuje pole indexů odstraněných prvků, tudíž při odstranění si pole pouze vnitřně uloží informaci o uvolněné pozici (nedochází k posuvu). Pokud každý prvek v tomto poli bude obsahovat atribut číslo indexu, na které pozici se nachází v tomto poli, můžeme použít i přímí přístup

známe-li tento index. Tato kolekce nejlépe vyhovuje při užití uchopovacího režimu popsaném v kapitole 3.3.3.

## 3.3 Uchopovací mód

Jak již bylo zmíněno v požadavcích na přesnost CAD systémů, uchopovací mód nám pomáhá zadávat body, které se již ve výkrese vyskytují. Toto není jedinou jeho funkcí dokonce ani jeho hlavní funkcí. Kterou je schopnost rozpoznat a vybrat požadované objekty v obraze. Protože se jedná o aplikaci ovládanou pomocí myši, pomocí níž provádíme výběr objektů, dochází k rozpoznání pouze těch objektů nacházející se nejblíže pozici kursoru. Nebo v případě výběru pomocí výběrového obdélníku hledáme objekty nacházející se uvnitř obdélníku. Proto musíme řešit problém jak tyto objekty rozpoznat. Naivním řešením je sekvenční procházení objektů a kontrola vzdálenosti objektu od kursoru. Toto řešení je časově náročné a neefektivní. Možnou modifikací tohoto způsobu je zefektivnění kontroly pozice kursoru vůči objektu. Toho lze dosáhnout využitím algoritmů pro ořezávání např. algoritmus *Cohen-Sutherland*[4]. Tento algoritmus je postaven na zefektivnění testování pozice úsečky vůči oknu za použití označení koncových bodů úsečky binárními kódy. Operací *and* nad těmito kódy určíme jestli objekt protíná hranici okna. V našem případě oknem rozumíme obdélník okolí kursoru, nebo výběrový obdélník. Tento algoritmus funguje dobře pro úsečku, ale v našich požadavcích máme i jiné entity.

### 3.3.1 Uchopovací mřížka



a) b)

Obrázek 3.2: Uchopovací mód pomocí mřížky a) červené vybrány b) nevybráno

Jedná se o efektivnější řešení hledání blízkých objektů, které provádí rozdělení obrazu na stanovený počet sektorů. Každý tento sektor ponese informaci o objektech nacházejících se uvnitř,

např. použijeme identifikační číslo pro každý objekt. Sektor pak definuje pole těchto identifikátorů. Takto si zúžíme prohledávanou oblast objektů.

Pro vyhledání je potřeba ještě definovat jak daleko od kursoru se může objekt nacházet. Při pohybu kursoru zjistíme nad jakým sektorem se nachází. Tento a okolní sektory prohledáme pro nalezení blízkých objektů. Na obrázku 3.2 je znázorněno rozdělení obrazu na sektory. Obdélník definující okolí pozice kursoru je znázorněn modrou barvou, červenou barvou jsou znázorněny objekty nacházející se v definovaném okolí.

### 3.3.2 Selection buffer

Dosud zmíněná řešení stále nejsou nejefektivnější. Jednou z možností GUI je použití hardwarově akcelerované plátno. K použití se nám nabízí OpenGL, které disponuje tzv. *selection bufferem.* Selection buffer je pole, které obsahuje informace o objektech vykreslených do požadované části obrazu. Toto vykreslení probíhá na pozadí tzn. že na obrazovku se nic nevykresluje ale pouze do bufferu na pozadí. Velikost a umístění požadovaného okna na obrazovce je předem nastavena, pak se provede vykreslení. Nejedná se o obyčejné vykreslení objektů, ale místo barevné složky se do bufferu na pozici pixelu uloží jednotlivé identifikátory objektů. Do bufferu jsou vykresleny pouze ty objekty, které zasahují do nastaveného okna. Pro dokončení se naplní selection buffer hodnotami identifikátorů a z-souřadnicemi skutečně vykreslených objektů. Více o tom jak provést nastavení selection bufferu pro režim výběru v literatuře [2].

Toto řešení je již efektivní, ale pro náš případ se příliš dobře nehodí, protože při každém požadavku na výběr objektů dochází k překreslení selection bufferu. Od našeho systému očekáváme prohledávání okolí kursoru při zadávání souřadnic, což vede k častému volání této metody.

### 3.3.3 Modifikace selection bufferu

K odstranění předchozího problému použijeme stejný princip vykreslení objektů na pozadí do bufferu. Pouze s jediným rozdílem takovým, že vykreslovaná oblast nebude už jen okno definované velikosti, ale zaplní nám celý prostor zobrazený uživateli na obrazovce. Překreslení tohoto okna bude prováděno na požádání, takže odpadá nutnost překreslení vždy na požadavek výběru. Tato metoda není součástí OpenGL a pro její implementaci využijeme *color buffer*. Do tohoto bufferu budeme vykreslovat entity, tak že do jejich barevné složky zakódujeme identifikační číslo entity. Barva je složena ze tří složek pro červenou, zelenou a modrou, kde každá je definovaná jako 8-bitové číslo. Při kódování identifikátoru na barvu provádíme jeho rozdělení na tři 8-bitová čísla, kde nejméně významných 8 bitů přiřadíme modré složce, dalších 8 bitů zelené složce a nejvýznamnějších 8 bitů červené složce. Tímto kódováním omezujeme maximální počet entit výkresu na $2^{24}-1$. Nutností této metody je rozlišení pozadí od objektů, proto budeme indexovat objekty od 1 a pozadí bufferu

bude mít barvu černou tedy hodnotu 0. Při výběru se provádí pouze operace průchodu pole na požadované pozici a zápis identifikátorů do selection bufferu. Více o tom jak kreslit do vlastního framebufferu v literatuře [2].

Tento způsob prohledávání požadovaného prostoru entit je již vhodný pro naši aplikaci. K jeho implementaci budeme potřebovat podporu od pracovní plochy. Princip podpory plátna pro tuto operaci je vysvětlen v kapitole 3.5.2.

### 3.3.4 Použité způsoby uchopovacího módu

V naší aplikaci budeme využíváme dva uchopovací režimy. První použijeme při výběru obdélníkem. K tomu slouží selection buffer z kapitoly 3.3.2. A druhý používáme pro rychlý výběr uchopovacích bodů. K tomu slouží modifikace selection bufferu z kapitoly 3.3.3. Můžete se ptát proč budeme využívat obě možnosti. Důvod je prostý, protože modifikace selection bufferu nedovoluje výběr entit při změně scény (např. potřebujeme vybrat větší část než je právě zobrazená a musíme posunout scénou), což nás omezuje pouze na výběr objektů přímo zobrazených. Na druhou stranu použití selection bufferu pro výběr uchopovacích bodů je příliš náročné na vykreslování.

### 3.3.5 Výběr uchopovacího bodu

Princip určení uchopovacího bodu je následující. Při pohybu kursoru dochází k vyhledávání entit v definovaném okolí pomocí dříve specifikovaných principů. Pokud je nalezena entita, ptáme se ji na uchopovací body, ze kterých vybereme nejbližší kursoru. Pokud je nalezeno více entit, našim prvotním cílem bude získat body týkající se více entit. Teprve v případě, že žádný takový bod nebyl nalezen, provedeme dotaz nad první vybranou entitou na její uchopovací body.


Obrázek 3.3: Návrh značek uchopovacích bodů aplikace

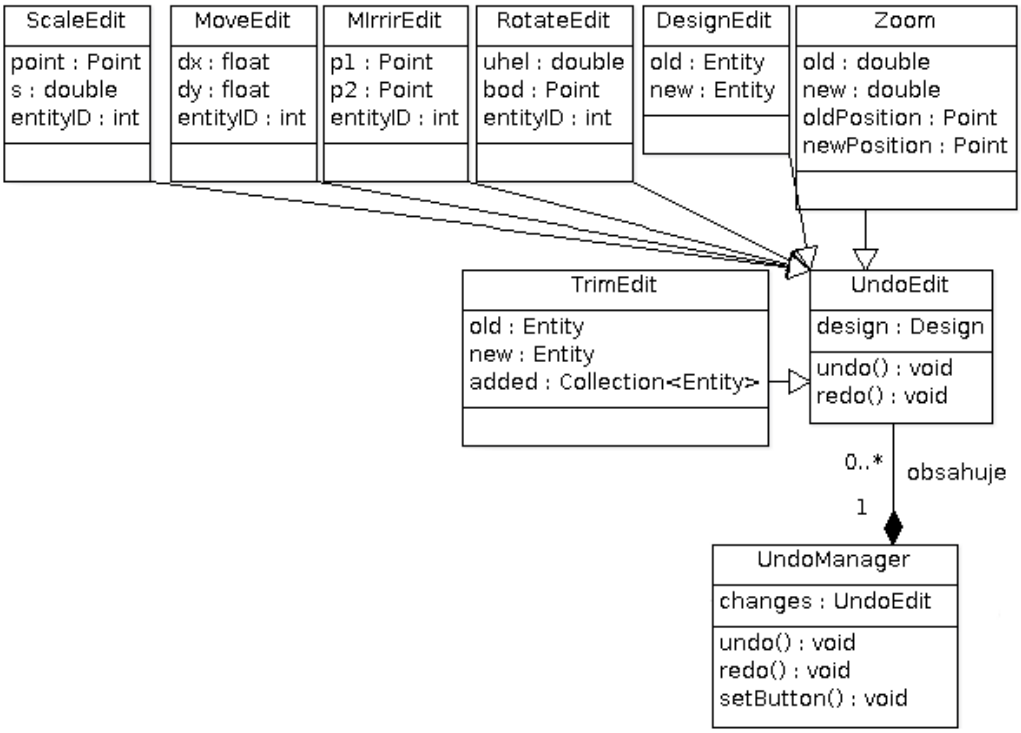
## 3.4 Krokovací manager

Součástí každého dobrého editoru je možnost vracet se nazpět v editačních krocích, nebo také se při vracení posunout znovu vpřed. K tomuto slouží tzv. *undo manager*, neboli krokovací manager. Tento manager si ukládá jednotlivé důležité[1] editační kroky do zásobníku editací. Při požadavku vpřed, nebo zpět vybere editaci z vrcholu zásobníku a zavolá příslušnou metodu pro provedení kroku.

1 to, které editační kroky jsou důležité určují zákazníci.

Pokud manager používá zásobník jako strukturu pro ukládání změn, bude obsahovat dva tyto zásobníky jeden pro funkci zpět a druhý pro funkci vpřed. Při provedení změny se tato změna přesune z jednoho zásobníku na druhý, čímž docílíme očekávaného chování pořadí změn. Další možností je použít dvousměrný seznam, ve kterém by směr postupu změn byl odlišní pro funkci zpět a vpřed. V tomto případě každá změna obsahuje obě funkce zpět a vpřed.

V našem CAD systému budeme chtít ukládat změny týkající se přidání, odstranění a editaci jedné nebo více entit, změny provedené pomocí modifikací, změny hladin entit a změny týkající se zobrazení. Pro definici změn vytvoříme abstraktní třídu, která bude mít dvě abstraktní metody. Jednu pro funkci zpět a druhou pro funkci vpřed. Každý typ modifikace je dán různými parametry, proto každé vytvoříme vlastní třídu pro změnu. Obrázek 3.4 ukazuje reprezentaci změn. Třída `UndoEdit` je naší abstraktní třídou pro uložení změn.




Obrázek 3.4: Krokovací manager

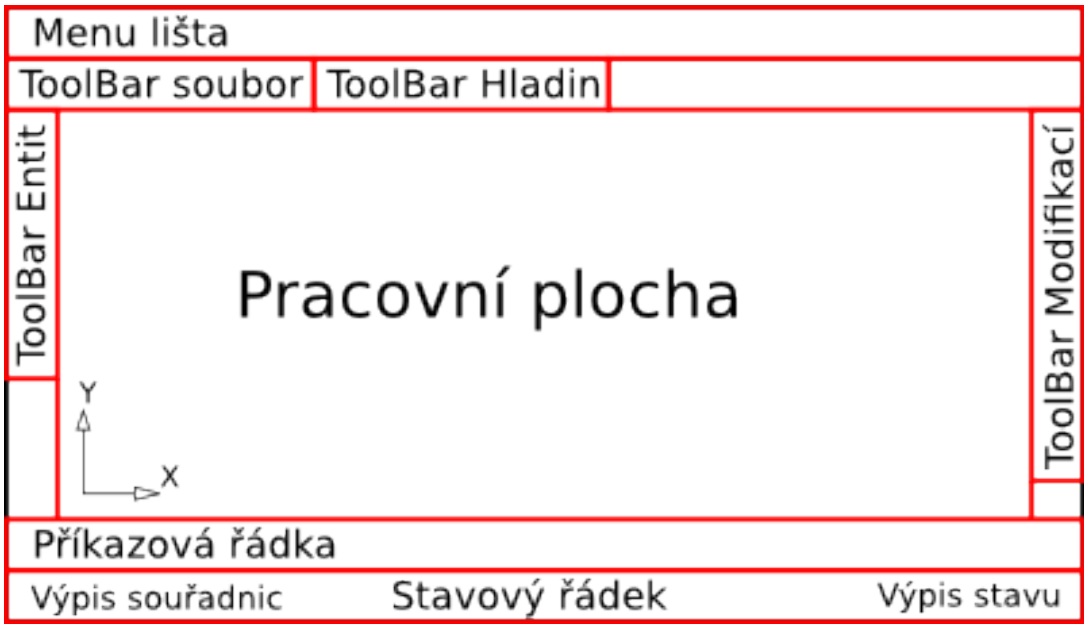
## 3.5 Grafické uživatelské rozhraní

Pro veškeré grafické editory, jimiž CAD systémy bezpochyby jsou, je důležité věnovat pozornost uživatelskému rozhraní. Největším problémem dnešních systémů je právě tato kapitola. Pokud tedy aplikace zvládá spoustu užitečných funkcí, ale nemá pro tyto funkce vytvořené efektivní rozhraní, stává se méně cennou. Naší snahou proto bude dodržet požadavky na dobré GUI, které byli

specifikovány v kapitole 2.3. V této kapitole navrhneme celkový vzhled rozhraní aplikace, rozhraní pro práci s hladinami a pracovní plochu se souřadnicovým systémem.

Na obrázku 3.5 je ukázka návrhu uživatelského rozhraní naší aplikace. Okno obsahuje menu nabídku ve vrchní části aplikace, tak jak jsme zvyklí. Nabídka se skládá z podnabídek pro práci s výkresovou dokumentací, krokovacími funkcemi zpět/vpřed, pracovní plochou, kreslícími a modifikačními nástroji a nakonec nastavení aplikace.

To jsme měli menu, které je nezbytnou součástí každého dobrého GUI. Z požadavků známe mnohé prvky, které nesmí chybět. Našim problémem je, jak tyto prvky rozvrhnout do okna. Mají být vloženy postupně za sebou, nebo snad pod sebe a když ano v jakém pořadí? Ne nic takového provádět nebudeme. Rozvržení volíme s největším důrazem na pracovní plochu. Okno rozdělíme na pět částí. První částí bude centralizovaný obdélník uprostřed našeho okna a bude obsahovat pracovní plochu. Další části se budou nacházet v jeho okolí na pravé, levé, vrchní a spodní straně, do nichž umístíme požadované komponenty, jak je znázorněno na obrázku 3.5.




Obrázek 3.5: Návrh GUI aplikace

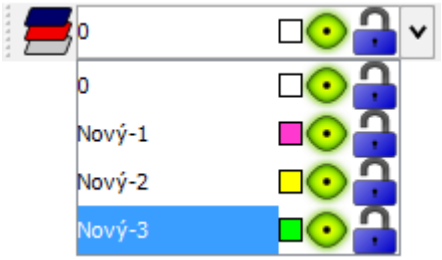
Jedním z požadavků je zpřístupnit uživateli často používaná tlačítka. Tento požadavek je řešen použitím toolbarů umožňující uživateli měnit jejich pozici v okně. Použití toolbarů neznamená, že původní tlačítka se zruší, ale dochází k rozšíření možností výběru volby. Toolbar soubor bude obsahovat tlačítka pro vytvoření nového výkresu, uložení stávajícího výkresu a otevření nějakého existujícího výkresu. Toolbar entit bude obsahovat veškerá tlačítka pro výběr nové entity. Stejně tak i toolbar pro modifikace.

Nakonec na dno okna umístíme poslední požadované části příkazový řádek a stavový řádek. Příkazový řádek bude vstupní textové pole, kam uživatel bude moci zadat příkaz, nebo souřadnice bodu. Taky bude sloužit jako průvodce programem. Např. uživatel zvolí nový objekt úsečka, v tomto případě se do řádky vypíše *„Úsečka určete první bod:“*. Aplikace v tomto případě očekává zadání souřadnic bodu. Stavový řádek bude sloužit k zobrazení stavů aplikace a aktuální souřadnice kursoru.

### 3.5.1 Uživatelské rozhraní hladin

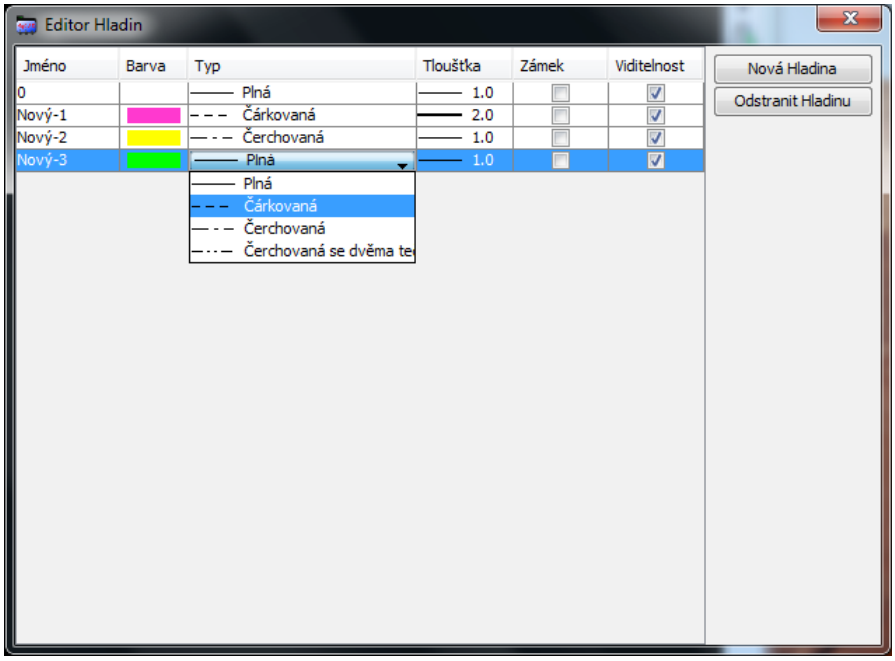
Hladiny jsou důležitou funkcí CAD systémů, pro jejich efektivní využití potřebujeme vytvořit intuitivní a rychlé rozhraní. Od rozhraní hladin očekáváme možnost úprav jednotlivých hladin, přidání nové, nebo odstranění staré hladiny. Pro efektivnost rozdělíme rozhraní hladin na dvě části. První část bude editor pro kompletní práci s hladinami. Druhou část zabalíme do toolbaru pro rychlý přístup k editaci důležitých atributů. Proto si znovu rozebereme jaké atributy hladiny obsahují. Prvním atributem je jméno, které použijeme pro výběr aktivní hladiny, pak barva, styl čáry, tloušťka čáry, viditelnost a zámek. Pro dvoustavové atributy jako zámek a viditelnost připravíme ikony reprezentující jejich aktuální stav.

- **Toolbar hladin –** Bude obsahovat rozbalovací seznam, pomocí něj budeme provádět výběr aktivní hladiny. Dále tento seznam bude sloužit jako rychlá editační volba pro změnu barvy, viditelnosti, nebo stavu zámku hladiny. Zároveň v seznamu uživatel uvidí jaké má možné hladiny a jaká je jejich barva pro rozpoznání entit. Ukázka toolbaru hladin je na obrázku 3.6.




Obrázek 3.6: Toolbar hladin a rozbalovací seznam

- **Editor hladin –** Bude obsahovat už kompletní definici hladin. Vytvoříme jej jako modální dialog, obsahující tabulku hladin, tlačítka pro vytvoření nové hladiny a pro odstranění vybraných hladin. Tabulka umožňuje měnit pozici sloupců dle libosti, ale standardně bude první sloupec se jménem, druhý s barvou. Po kliknutí na druhý sloupec s barvou se otevře další dialog pro výběr barvi. V dalších dvou sloupcích tabulky budou rozbalovací seznamy se všemi možnými typy a tloušťkami čar. Nakonec postačí zaškrtávací políčka pro dvoustavové atributy viditelnost a zámek. Příklad dialogu pro hladiny je znázorněn na obrázku 3.7.




Obrázek 3.7: Příklad editoru hladin

### 3.5.2 Pracovní plocha

Bezpochyby nejdůležitější část, kterou budeme hardwarově akcelerovat pomocí OpenGL, což je programové rozhraní pro práci s grafickým hardwarem (více v kapitole 3.8.1). OpenGL plátno je pouze plocha, na kterou lze kreslit. Našim požadavkem na pracovní plochu je jeho schopnost zobrazit požadovanou část výkresové dokumentace. Proto si vytvoříme třídu, která bude zapouzdřovat akcelerované plátno a bude ovládat práci s ním. Jejím primárním úkolem bude posouvat scénou a také přibližovat pohled. Protože pro uchopování použijeme metody OpenGL popsané v kapitole 3.3.3 a kapitole 3.3.2, bude také umět provádět tuto práci nad akcelerovaným plátnem.

Důležitou součástí OpenGL plátna je jeho Renderer *(hardwarový vykreslovač)*. V naší aplikaci to bude třída `GLRenderer`. Tento renderer se bude moci nacházet ve čtyřech stavech:

- **Render** – Stav kdy provádí normální vykreslení. Nejprve inicializuje počátek souřadnicové systému scény a pak zavolá metodu pro vykreslení dokumentace.
- **Select** – V tomto stavu provádí výběr pomocí selection bufferu z kapitoly 3.3.2. Nejprve nastaví selection buffer, potom inicializuje zásobník jmen na hodnotu -1, dále provede inicializaci velikosti výběrového okna a zavolá metodu pro vykreslení dokumentace pomocí jmen. Nakonec přečte získané hodnoty a naplní nimi náš selection buffer.
- **Snap** – Je stav, kde renderer provádí výběr pomocí modifikace selection bufferu popsaném v kapitole 3.3.3. Konkrétně prohledává pole color bufferu na specifikované oblasti, při tom naplňuje selection buffer nalezenými hodnotami.

- **SnapUpdate** – Poslední stav, ve kterém aktualizuje color buffer pro výběr pomocí modifikace selection bufferu. Provádí stejné akce jako při normálním vykreslování pouze před vykreslením nastaví vykreslování do našeho framebufferu na pozadí a volá metodu pro vykreslení dokumentace pomocí kódování identifikátoru do barvi.

Pracovní plocha pro schopnost měnit zobrazení scény bude obsahovat objekt pro převod mezi souřadnicovými systémy popsané v kapitole 3.5.3. Jak se ve zmíněné kapitole dozvíme, tento objekt obsahuje souřadnice středu systému výkresové dokumentace v souřadnicovém systému plátna. A protože renderer nastavuje střed souř. systému pomocí tohoto objektu, můžeme jednoduchým posuvem této souřadnice posunout celou scénou. Na podobném principu je založeno také přiblížení k pozici kursoru a zobrazení celé výkresové dokumentace.

### 3.5.3 Souřadnicový systém

Souřadnice pozic grafických komponent v GUI každé aplikace je dána počtem pixelů vzdálených od levého horního rohu nadřazeného okna, nebo panelu. Tato souřadnice platí i pro pozici kursoru. Dalším souřadným systémem vyskytující se v aplikaci je souřadný systém modelu výkresové dokumentace. Jedná se o klasický kartézský souřadný systém jak jej známe. Tyto dva systémy se od sebe liší a to především tak, že souřadnice kursoru na obrazovce je se středem v levém horním rohu okna a y souřadnice roste se vzdáleností od tohoto rohu. Protože chceme, aby uživatel mohl zadávat body přímo za pomoci myši na pracovní ploše, musíme provádět převod této souřadnice na odpovídající souřadnici v modelu. Pro tento účel vytvoříme třídu uchovávající souřadnice počátku souřadnicového systému modelu v souřadnicovém systému pracovní plochy a velikost zvětšení, která hraje důležitou roli při převodu mezi systémy. Dále tato třída bude obsahovat metody pro převod souřadnic bodu mezi těmito dvěma systémy.

## 3.6 Práce se souborem

Obecným požadavkem aplikací při načítání je jejich stálá odezva při probíhající práci. Z tohoto důvodu provádíme načítání a ukládání ve vlákně na pozadí. V naší aplikaci pro tento účel vytvoříme třídu `Loader`, která bude provádět načítání i ukládání výkresu. Při načítání je nutné zablokovat některé funkce pracující nad objektem výkresové dokumentace. Budeme tedy muset zablokovat jádro aplikace pro nastavení nástroje, nebo modifikace a zajistit zrušení reakcí na vstup od klávesnice a myši.

Objekt `Loader` bude pracovat ve dvou režimech pro načítání a ukládání. Pokud tedy chceme provádět načítání, či ukládání musíme nejprve nastavit příslušný režim a až potom budeme moci

spustit vlákno. Po spuštění vlákna už jen čekáme na jeho dokončení. Pro zajištění získání zprávy od vlákna informující, že došlo k dokončení akce, vytvoříme vlastní rozhraní. Toto rozhraní bude implementovat obslužnou metodu pro příchod takovéto zprávy. V této metodě si přečteme výsledný stav akce objektu, na jehož základě informujeme uživatele.

### 3.6.1 Příklad souboru

V našem případě pro popis výkresové dokumentace použije vlastní formát souboru. Pro jeho popis použijeme značkovací jazyk XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<mycad version="1.0">
  <design name="screwM10">
    <layers count="2">
      <layer active="true" color="0 0 0" factor="1" lock="false" name="0"
         pattern="-1" visibility="true" width="1.0"/>
      <layer active="false" color="255 255 255" factor="2" lock="true" name="Osa"
         pattern="6399" visibility="false" width="1.0"/>
    </layers>
    <entities count="2">
      <line>
        <points count="3">390.0 137.0 390.0 100.0 390.0 63.0 </points>
        <layer>Osa</layer>
      </line>
      <arc>
        <points count="4">
          286.64966 167.5 264.79218 150.0 264.79218 185.0 258.64966 167.5
        </points>
        <layer>0</layer></arc>
    </entities>
  </design>
</mycad>
```

Kořenovým elementem souboru je `<mycad>`, který definuje soubor vytvořený našim programem. Obsahuje jeden atribut verzi programu. Jeho jediným vnořeným elementem je `<design>`, který popisuje obsah jako celek, jeho atribut obsahuje jméno výkresové dokumentace. Následují elementy popisují výkres. Prvním takovým je element `<layers>`, který značí sekci pro definici hladin. Jeho jediným atributem je počet obsažených hladin. Obsahem jsou jednotlivé definice hladin v elementu `<layer>`, atributy tohoto elementu jsou atributy hladin. Při ukládání hladin ukládáme i základní hladinu „0“. Dalším elementem popisujícím výkres je `<entities>`, který také

obsahuje atribut s počtem obsažených entit. Obsažené elementy jsou nazvány podle typu entity. V příkladě je ukázán element `<line>`, který definuje úsečku a `<arc>` pro oblouk. Každý element popisující entitu obsahuje dva základní elementy `<points>`, určující řídící body entity, a `<layer>`, určující hladinu entity.

### 3.6.2 Ukládání výkresové dokumentace

Pro možnost ukládání objektů budeme potřebujeme rozhraní, díky kterému nás pak nezajímá jaký objekt ukládáme. Toto rozhraní pojmenujeme `Saveable`. Prvním krokem pro uložení výkresové dokumentace bude volání metody nad objektem jádra aplikace `SimpleDriver`. V této metodě se vytvoří základ, který určuje soubor vytvořený právě naší aplikací. Potom jádro nastaví režim `Loaderu` a spustí vlákno. `Loader` provede zapsání názvu dokumentace a bude pokračuje vytvořením sekce hladin, do které následně zapíše všechny hladiny. To samé provede ještě pro všechny entity. Pro dokončení `Loader` zašle jádru aplikace zpráva o konci tohoto vlákna. Jádro provede uložení datových struktur XML do souboru a informuje uživatele.

### 3.6.3 Načítání výkresové dokumentace

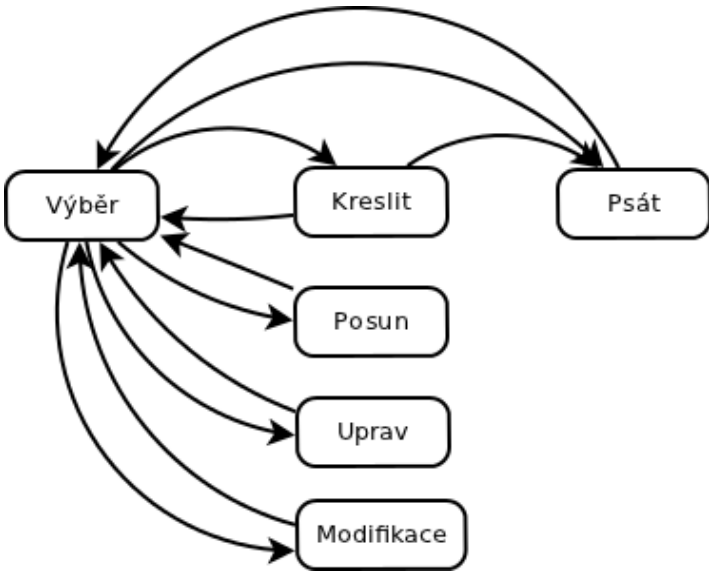
Načítání bude znovu začínat voláním metody jádra aplikace, kde v parametru předáme cestu k souboru. Jádro provede rozbalení a zkontroluje zda se jedná o podporovaný soubor. Potom provede nastavení objektu `Loader` a sebe sama na režim načítání. Nakonec spustí *Loader* ve vlákně, kde se provede načtení jména výkresu, hladin a entit. `Loader` zároveň provádí kontrolu počtu hladin a entit. V případě nějaké chyby nastaví výsledek své práce, který si jádro po dokončení přečte a informuje uživatele.

## 3.7 Jádro aplikace

Důležitou částí aplikace bude její řídící jádro. Toto jádro bude vykonávat reakce na uživatelem vyvolané akce, řídit proces načítání a ukládání. Jeho hlavním úkolem bude práce s výkresovou dokumentací. Bude se jednat o stavový automat, jehož stavy a přechodový diagram je znázorněn na obrázku 3.8.

Stav `výběr` je počátečním stavem. V tomto stavu jádro provádí výběr entit na pracovní ploše. Automat je možné nastavit vnějším impulsem do stavů `kreslit`, nebo `modifikace` výběrem kreslícího nástroje, nebo zvolením jedné z možných modifikací entit. Ve stavu `kreslit` očekává zadávání bodů pomocí myši, ale i pomocí příkazové řádky. Po dokončení zadávání souřadnic se znovu ocitne ve stavu `výběr`. Stav `posun` slouží k posunu vybraných entit typem „*drag and drop*“.

Do tohoto stavu se automat dostane pokud uživatel klikne na jednu z vybraných entit a začne ji táhnout na novou pozici. Dalším stavem je stav `uprav` sloužící pro editaci řídících bodů entit. Do tohoto stavu se jádro dostane při kliknutí na řídící bod vybrané entity. Posledním zatím nezmíněným stavem je `psát`. Tady automat zadává znaky do textové entity. Ze všech možných stavů lze automat přepnout do počátečního stavu za pomocí tlačítka escape. V případě, že automat se již nachází v provádění akce, provede operaci zpět, které vrátí vše do původní polohy.




Obrázek 3.8: Přechodový diagram jádra aplikace

## 3.8 Implementační nástroje

Kapitola se věnuje použitému programovacímu jazyku a knihovnám, které jsou v aplikaci využity. Systém bude implementován v objektově orientovaném jazyce Java. Již v návrhu jsme se zmínili o rozhraní pro práci s grafickým hardwarem OpenGL. V jazyku Java existuje knihovna pro OpenGL nazvaná JOGL. Co je to OpenGL a JOGL bude vysvětleno v této podkapitole.

Pro implementaci byl zvolen programovací jazyk Java, protože nabízí spoustu volně dostupných knihoven, pro nejrůznější použití. Je to jazyk objektově orientovaný, který dodržuje dokumentaci. Každá knihovna je dokumentovaná a při použití vývojového prostředí je přímo k dispozici. Nejdůležitější je však jeho jednoduchá přenositelnost mezi operačními systémy.

### 3.8.1 OpenGL

V této kapitole si přiblížíme knihovnu OpenGL, která byla zvolena z důvodu přenositelnosti aplikace mezi různými operačními systémy. Kapitola vychází z literatury [2].

OpenGL je programové rozhraní pro práci s grafickým hardwarem. Bylo vyvinuto firmou Silicon Graphics, Inc., a je dostupné na většině operačních systémů. Nejedná se o programovací

jazyk, ani o okenního správce, ale je určeno pro optimalizované zobrazování a manipulaci s grafickou kartou. Naopak pro svoji práci potřebuje být zobrazeno pomocí nějaké okenního správce. Nachází uplatnění v CAD systémech, počítačových hrách, vědeckotechnických vizualizacích, virtuální realitě apod.

OpenGL je spíše procedurální, než deklarativní. Namísto popisu scény a jak by měla být vykreslena, se definují kroky nezbytné pro vykreslení požadovaného vzhledu. Tyto kroky lze popsat více jak 200 příkazy a funkcemi. Příkazy především slouží pro vykreslení grafických primitiv jako je bod, úsečka a polygon. OpenGL dále disponuje funkcemi pro osvětlení, stínování, texturovaní a mnoha dalšími. Lze jej charakterizovat jako stavový automat, který disponuje mnoha stavy definujícími proces vykreslování. Tento automat vykresluje za pomoci svého stavu tak dlouho dokud není změněn, potom vykresluje podle nové definice stavu. Nastavíme-li barvu modrou je použita až do té doby dokud nenastavíme barvu jinou. Vykreslování provádí do pomocných bufferů celkově označených jako *framebuffer*. Mezi tyto buffery patří *depth buffer* (slouží pro zapsání z souřadnice vykresleného pixelu), *color buffer* (obsahuje barevnou složku) a další.

V naší aplikaci používáme OpenGL pro vykreslování 2D výkresové dokumentace. Největším důvodem pro použití v našem systému je uchopovací mód, který pomocí identifikačních čísel objektů provádí rozpoznávání. V tomto módu se využívá vykreslování do framebufferu na pozadí což znamená, že žádná informace se nezobrazuje na obrazovku.

### 3.8.2 Knihovna JOGL

Pro užití OpenGL knihovny v Javě byla použita knihovna JOGL, jejíž popis vychází z literatury [7] a [8]. JOGL (*Java OpenGL*) je Java API, které poskytuje vazbu OpenGL knihoven pro virtuální stroj. To nám zpřístupňuje objektově orientované nástroje pro využití hardwarové akcelerace 2D a 3D grafiky. JOGL umožňuje přístup k většině OpenGL funkcím dostupných v programovacím jazyku *C* pomocí JNI *(Java Native Interface, česky Java rozhraní pro nativní knihovny).* Tyto nativní knihovny jsou k dipozici na operačních systémech Windows, Linux, Solaris, Mac OS. Knihovnaje podporována v Javě 1.4 a novější.

JOGL se od ostatních vazeb OpenGL na Javu (např. LWJGL, GL4Java, Java 3D aj.) liší především tím, že pouze odhaluje procedury stejné jako v klasickém OpenGL a nesnaží se tyto funkce převést na objektově orientované paradigma. Většina JOGL kódu je automaticky generována z *C* hlavičkových souborů pomocí nástroje *GlueGen*, čímž je usnadněná tvorba JOGL. Přímé mapování OpenGL *C* funkcí na Java metody zjednodušuje konverzi. Tenká vrstva abstrakce poskytuje JOGL poměrně efektivní běh. Protože většina kódu je automaticky generována, změny provedené v OpenGL jsou jednoduše aplikovány v JOGL.

Jedná se o knihovnu OpenGL, proto potřebuje grafickou komponentu, ve které provedeme zobrazení. V Javě existují dva základní přístupy k tvorbě GUI AWT *(Abstract Window Toolkit)* a Swing. JOGL je podporován v obou těchto GUI knihovnách. Pro AWT existuje komponenta `GLCanvas`, která podporuje hardwarovou akceleraci a pro Swing existuje `GLJPanel` s podporou softwarové akcelerace.

### 3.8.3 Správa verzí

Jedním z cílů implementace systému bylo využití nástroje pro správu verzí aplikace, který slouží pro tvorbu větších projektů. Tento nástroj běží na serveru, kam má každý pracovník, pracující na projektu, přístup. Protože výsledná aplikace je napsána v programovacím jazyce Java, bylo pro vývoj použito prostředí NetBeans. Toto prostředí také disponuje různými klienty pro správu verzí. Za tímto účelem bylo použito systému SVN, běžícím na školním serveru *merlin*.

### 3.8.4 Reprezentace výkresové dokumentace

Výkresová dokumentace je reprezentována třídou `Design`. Tato třída implementuje rozhraní `EntityInterface, LayerInterface`. Dále obsahuje metody `paintDesign`, pro normální a výběrové vykreslení, a metodu `paintColorSelect` pro vykreslení při výběru pomocí barvy. Nedílnou součástí výkresu je jeho uchopovací manager třída `SnapManager`. Tento manager obstarává veškeré operace nad vybranými entitami, uchovává si pomocné pole referencí na vybrané entity, provádí výběr entit a uchopovacích bodů aj.

### 3.8.5 Grafické uživatelské rozhraní

GUI systému je vytvořené na základě grafické knihovny v Javě *Swing*. Tato knihovna podporuje tzv. „*look and feel*“, což nám umožňuje nastavit vzhled aplikace dle prostředí, na kterém je spuštěna. Čímž dodržujeme konvence pro tvorbu správného GUI. Navíc *swing* knihovna obsahuje spoustu pro nás vhodných komponent jako je toolbar.

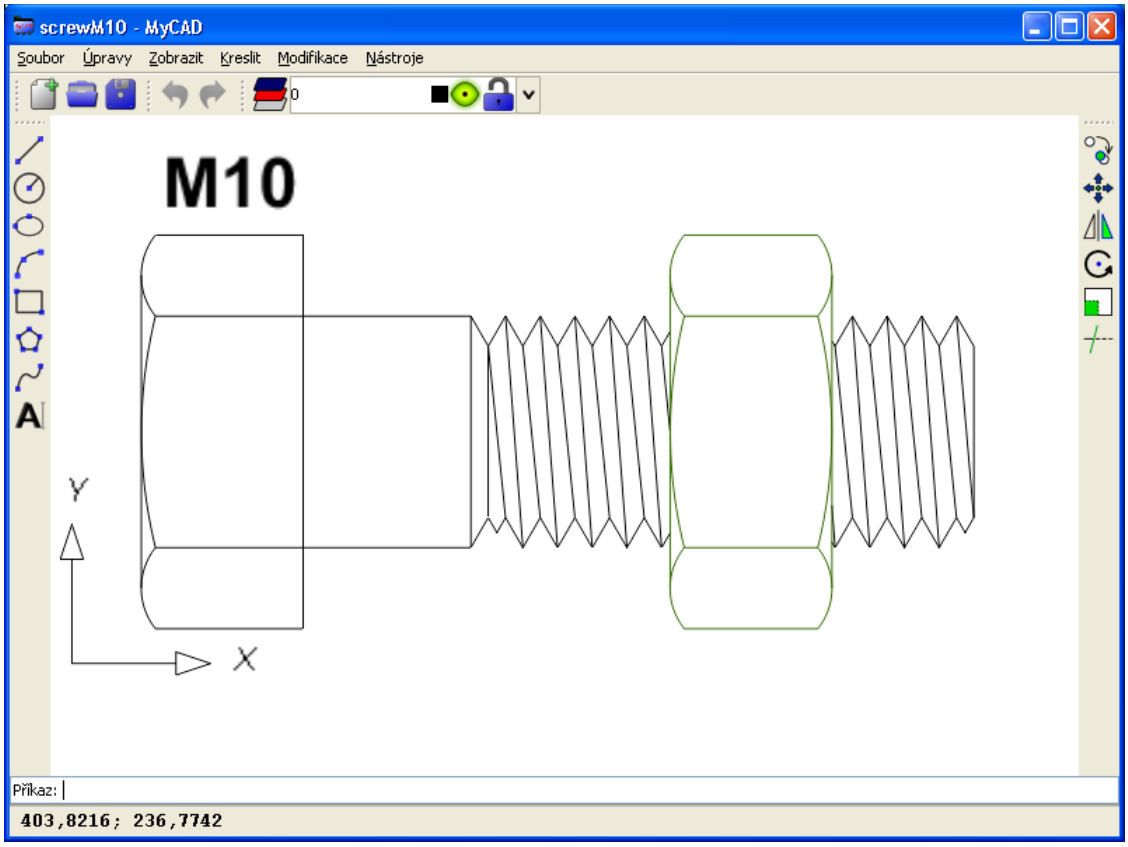
Plátno aplikace je implementováno hardwarově akcelerovanou komponentu `GLCanvas`. Tento prvek není z knihovny `Swing` a navíc se jedná o „*heavyweight prvek*“. Proto musíme při inicializaci okna aplikace nastavit prvky rolovací nabídky na „*lightweight*“.

# 4 Výsledky

Cílem této práce bylo analyzovat problematiku CAD systémů, užívaných v dnešní době, navrhnout a zkusit implementovat jeden takovýto systém. Tyto cíle už máme za sebou a zbývá nám poslední cíl zhodnocení výsledků naší výsledné aplikace a stanovit další vývoj projektu.

## 4.1 Dosažené výsledky

Našim implementačním cílem bylo dosažení základních požadavků na CAD systémy, kterým jsme se věnovali v kapitole 3.1. Dosáhli jsme funkční aplikace s uživatelsky jednoduchým, ale efektivním grafickým rozhraním.




Obrázek 4.1: Ukázka výsledné aplikace

Toto rozhraní bylo navržené pro co největší efektivitu práce s aplikací. Všechny prvky jsou dostupné a rozložené tak, aby byli jednoduše ovladatelné. Jedním důležitým přínosem aplikace je příkazová řádka, které při akcích informuje uživatele jaké požadavky jsou od něj očekávány. Některé akce mají svoji klávesovou zkratku. Jedná se především o zkratky pro práci se souborem, jehož ikony jsou také umístěné v toolbaru. Tyto akce dovolují uživateli ukládat (*CTRL + S*) a otevřít (*CTRL + O*)

soubor, provádět akci zpět (*CTRL + Z*) a vpřed (*CTRL + Y*), nebo také ukončit program (*CTRL + Q*). Aplikace také disponuje jednoduchým nastavením přístupném přes menu nabídku. V tomto nastavení můžeme změnit barvu plátna a velikost terčíku pro uchopovací režim. Ovládání plátna provádíme pomocí myši. Prostředním tlačítkem můžeme posouvat scénou a kolečkem pak přibližovat, nebo oddalovat směrem ke kursoru. Ukázka aplikace je na obrázku 4.1.

Aplikace je předpřipravená pro globální rozšíření. Je zde použito slovníku, který definuje jednotlivé hlášky zobrazené uživateli v jazyku českém a anglickém. Podle nastaveného prostředí dojde k výběru patřičného jazyka.

### 4.1.1 Neimplementované cíle

Aplikace neobsahuje uložení změny pohledu do krokovacího manageru. Je to z toho důvodu, že nebylo přesně specifikováno kdy k tomu má dojít. Dalším důvodem bylo, že těchto změn by bylo uloženo v manageru mnohem více než těch důležitých a při funkci zpět/vpřed by z větší části došlo k provádění změny pohledu.

## 4.2 Entity a hladiny

Naše aplikace podporuje základní vektorové komponenty požadovaných po CAD systémech:

- Úsečka
- Kružnice
- Oblouk
- Elipsa
- Obdélník
- Polygon
- Křivka
- Text

Možnosti jejich zadávání jsou popsány v kapitole 3.2.1. U textové entit můžeme změnit její písmo, nebo ji dále editovat. Toho dosáhneme pomocí kliknutím pravého tlačítka myši nad entitou, kde se zobrazí rolovací menu s výběrem písma, nebo editací.

Aplikace umí pracovat s organizací výkresu do hladin. Hladiny zvládají základní možnosti pro viditelnost, můžeme měnit její jméno i barvu, typ a tloušťku čar. Pomocí rolovací nabídky umístěné v toolbaru hladin můžeme provádět změnu hladiny vybraných entit. Na obrázku 4.1 je ukázka organizace do hladin, kde matka leží v jiné hladině než šroub.

## 4.3 Modifikace

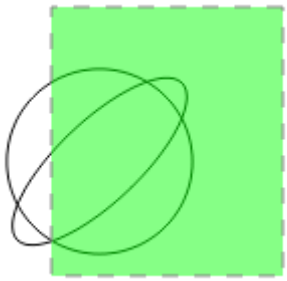
Výsledná aplikace už zvládá i modifikace entit, nebo množiny vybraných entit:

- kopie
- posuv
- zrcadlení
- rotace
- měřítko
- ořezávání

Ořezávání entit je zatím jen v počáteční fázi. Ořezat můžeme entity úsečka, kružnice, oblouk, obdélník a polygon. Při ořezávání obdélníku, nebo polygonu dojde k nahrazení těchto entit pomocí úseček spojující zbylé vrcholy.

## 4.4 Výběrový mód

Aplikace obsahuje výběrový mód, pomocí kterého můžeme provádět výběr objektů přímo zobrazených na pracovní ploše pouhým kliknutím, nebo za pomoci výběrového obdélníku. Tento obdélník je dvojího typu, který se rozlišuje dle jeho zadaných bodů. Pokud budeme zadávat výběrový obdélník směrem vlevo bude se zobrazovat zelenou barvou a vybírat veškeré objekty do něj zasahující. Na druhou stranu (tedy vpravo) se bude zobrazovat modrou barvou a vybírat ty objekty, jejichž veškeré řídící body se nacházejí uvnitř. Při stisknuté klávese *shift* bude provádět odebírání objektů z výběru.



Obrázek 4.2: Ukázka výběru obdélníkem

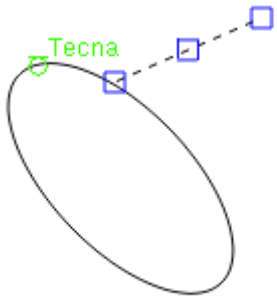
## 4.5 Uchopovací mód

Aplikace také disponuje uchopovacími body:

- Koncový

- Polovina
- Střed
- Tečna
- Kolmice
- Průsečík
- Kvadrant

Zvládá výpočet průsečíků veškerých entit, kromě průsečíku křivky s jakoukoli jinou entitou. Dále zvládá výběr požadovaného typu uchopovacího bodu. Tento výběr provedeme pomocí klávesy *shift* a pravého tlačítka myši. Po této volbě se nám zobrazí rolovací menu pro výběr konkrétního uchopovacího bodu. Aplikace pak vyhledává pouze jen tento bod dokud se nenastaví nový typ. Pro možnost znovu hledat nejbližší možný zvolíme volbu *nejbližší*, jak je znázorněno na obrázku 4.4. Na obrázku 4.3 je znázorněné vyhledání tečných bodů.




Obrázek 4.3: Ukázka výběru uchop. bodu


Obrázek 4.4: Rolovací menu uchop. bodů

## 4.6 Příkazová řádka

Příkazová řádka umožňuje zadávat přímo souřadnice bodů jak pomocí absolutní, tak i relativní souřadnice. Pro zadání relativní souřadnice uvedeme jeden ze znaků r, R, nebo @ před první souřadnici. Dále pak vypisuje informaci o očekávaném zadání parametrů uživatele. Jednotlivé příkazy pro výběr entit jsou v příslušném jazyku systému. Příklad pro češtinu/angličtinu:

- úsečka/line
- kružnice/circle
- oblouk/arc
- elipsa/ellipse
- obdélník/rectangle
- polygon/polygon
- text/text
- spline/spline

# 5 Závěr

V této práci jsme se seznámili s principy práce použitých v CAD systémech a jaké problémy je potřeba řešit při jejich návrhu. Výsledkem práce je vlastní aplikace, která byla implementována s důrazem na multiplatformní použití. Z tohoto důvodu byla aplikace implementována v jazyce Java. O tomto jazyku je znám především jeho pomalý běh aplikací. Vytvořená aplikace ovšem ukazuje právě opak tohoto problému. Jedná se sice o aplikaci, které obsahuje pouze základní požadavky na CAD systémy, ale tyto požadavky uspokojivě splňuje a má naději na další vývoj. Knihovna JOGL se ukázala jako nejlépe vyhovující z důvodu její nízké abstrakce.

Tvorba tohoto systému mi přinesla úplně nový pohled na CAD systémy. Přispěla k lepšímu pochopení principů těchto systémů, především jejich uchopovací režim. Další přínos shledávám v práci s OpenGL v Javě a také větší rozšíření znalostí o tomto jazyku.

Výsledný systém poskytuje jen zlomek toho co profesionální systémy. Z tohoto důvodu uvedeme směr dalšího vývoje aplikace. Začneme rovnou u entit systému. Např. by bylo vhodné rozšířit možnosti zadávání některých entit konkrétně oblouk, který lze zadat bodem na prvním konci, bodem v polovině oblouku a bodem na druhém konci. Křivku by určitě bylo lepší implementovat aspoň jako plně modifikující spline s úpravou návaznosti segmentů. V budoucnu by jsme ale s touto možností neměli počítat a přímo implementovat tuto entitu jako NURBS křivku. Dále pak dodělat ořezávání entity křivka a elipsa, pro kterou budeme muset vytvořit novou entitu eliptický oblouk. Budoucím doplněním hladin by mělo být načtení typu čar z externího souboru, protože v nynějším stavu jsou tyto typy definovány přímo v kódu aplikace. Možno také přidat další atributy zmrazení, průhlednost. Co se týče rozhraní pro hladiny do budoucna dodělat možnost filtrování hladin, protože výkres může obsahovat až stovky hladin. Pro další vývoj aplikace je možné do nastavení, které je také k dispozici v této verzi, přidat seznam pro výběr požadovaných uchopovacích bodů. Mezi chybějícími body je např. protažení, rovnoběžně, nebo prodloužený průsečík, který občas je možné získat již nyní. Modifikace by jsme měli doplnit o prodloužení, protažení, pole aj. Dále by bylo vhodné přidat ortogonální a polární režim.

# Literatura


[1] Bejček, V.: *Základy CAD/CAM*. 2002, [online], Fakulta strojního inženýrství, VUT v Brně URL http://drogo.fme.vutbr.cz/opory/pdf/CAD.pdf

[2] WRIGHT, Richard S. a Michael SWEET. *OpenGL SuperBible. Second Edition*. USA: Waite Group Press,1999. ISBN 1-57169-164-2.

[3] Kršek, P. Španěl, M.: *Základy počítačové grafiky: Rasterizace objektu ve 2D*. 2011, [online], Fakulta informačních technologií, VUT v Brně.
URL https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_rasterizace_print.pdf

[4] Kršek, P. Španěl, M.: *Základy počítačové grafiky: Ořezávání objektů ve 2D*. 2011, [online], Fakulta informačních technologií, VUT v Brně.
URL https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_orezavani_print.pdf

[5] KRŠEK, Přemysl. *Vizualizace a CAD: Uživatelské rozhraní v CAD systémech* [online]. Fakulta informačních technologií, VUT v Brně, 2008 [cit. 2012-04-20]. Dostupné z: https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/viz_slide_gui_print.pdf

[6] KRŠEK, Přemysl. *Vizualizace a CAD*: *Sdílení dat v CAD systémech* [online]. Fakulta informačních technologií, VUT v Brně, 2008 [cit. 2012-04-20]. Dostupné z: https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/xviz_slide_sdileni_dat_print.pdf

[7] DAVISON, Andrew. *Pro Java 6 3D game development: Java 3D, JOGL, JInput, and JOAL APIs*. New York, NY: Distributed to the book trade worldwide by Springer-Verlag, c2007, 498 s. ISBN 978-159-0598-177.

[8] WOLFF, David. *Using OpenGL in Java with JOGL* [online]. Department of Computer Science and Computer Engineering, Pacific Lutheran University, 2005 [cit. 2012-04-23]. Dostupné z: http://coitweb.uncc.edu/~krs/courses/4120-5120/handouts/jogl-tutorial.pdf

[9] WEISSTEIN, Eric. *Wolfram MathWorld* [online]. [cit. 2012-04-29]. Dostupné z: http://mathworld.wolfram.com

[10] HUGHES, Gary B. a Mohcine CHRAIBI. *Calculating Ellipse Overlap Areas* [online]. [cit. 2012-04-29]. Dostupné z: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1016&context=stat_fac

[11] KRŠEK, Přemysl a Michal ŠPANĚL. *Základy počítačové grafiky: Křivky v počítačové grafice* [online]. Fakulta informačních technologií, VUT v Brně, 2011 [cit. 2012-05-06]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_krivky_print.pdf

# Seznam příloh

Příloha 1. Obsah přiloženého CD

# Příloha 1. Obsah přiloženého CD

| | |
|---|---|
| /MyCAD | Adresář se spustitelnými soubory aplikace. |
| /src | Adresář se zdrojovými texty aplikace. |
| /doc | Dokumentace zdrojových kódů v JavaDoc |
| /Documents | Úplná dokumentace (uživatelská příručka, diagram tříd) |
| thesis.odt | Zdrojový tvar písemné zprávy v LibreOffice |
| thesis.pdf | Technická zpráva ve formátu PDF |
| Readme.txt | Návod pro instalaci a spuštění aplikace |