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

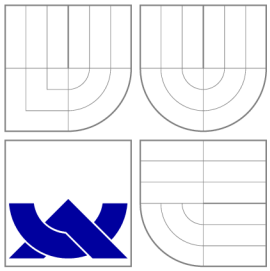
CAD SYSTÉM PRO 2D KRESLENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

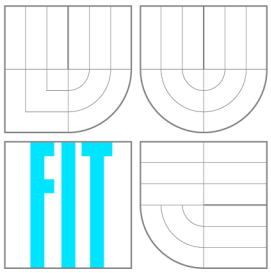
AUTOR PRÁCE
AUTHOR

VOJTĚCH BRNICKÝ

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
AUTHOR

VOJTĚCH BRNICKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. PŘEMYSL KRŠEK, Ph.D.

BRNO 2010

Abstrakt

Tato práce se zabývá tvorbou CAD systému pro 2D kreslení. Obsahuje seznámení s obecnou problematikou a charakteristikou dnešních CAD systémů. V práci jsou analyzovány požadavky na systém, jsou popsány návrh a následná implementace systému. V závěru práce jsou zhodnoceny dosažené výsledky a stanoveny možnosti dalšího vývoje.

Abstract

This thesis deals with creating a CAD system for 2D drawing. It contains an introduction to the genaral issues and characteristics of today's CAD systems. The thesis analyzes the system requirements, describes the design and subsequent implementation of the system. In the conclusion of the thesis, the achieved results are evaluated and the potential for further development is defined.

Klíčová slova

CAD systém, 2D kreslení, vektorová entita, vektorový objekt, výkresová dokumentace, uchopovací mód

Keywords

CAD systems, 2D drawings, vector entity, vector object, design, snap mode

Citace

Vojtěch Brnický: 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. a k jejímu vypracování jsem využil zdroje uvedené v seznamu použité literatury.

.....
Vojtěch Brnický
17. května 2010

Poděkování

Na tomto místě bych rád poděkoval doc. Ing. Přemyslu Krškovi, Ph.D. za cenné připomínky a odborné rady, kterými přispěl k vypracování této bakalářské práce.

Obsah

Kapitola 1

Úvod

V dnešní době se čím dál více zvyšují požadavky na vývoj a výrobu nových výrobků. Jsou kladeny nároky na zkrácení doby, snížení nákladů na výrobu a zvýšení kvality výrobků. Proto jsou ve výrobě stále častěji využívány počítačové systémy. CAD systém je počítačový program umožňující snazší, rychlejší a přesnější tvorbu dokumentací potřebných pro výrobu jednotlivých výrobků.

Cílem této práce je vytvořit CAD systém pro 2D kreslení. CAD systémy charakterizuje druhá kapitola. Vysvětluje, co to CAD systém je, kde se používá a na jaké kategorie se rozděluje. Kapitola také popisuje, jaké jsou dnešní požadavky na takový systém.

Další kapitola se zabývá podrobnější analýzou požadavků a popisuje návrh vytvářené aplikace. Jsou zde stanovené cíle, kterých má tato práce dosáhnout, a rozebrány možné způsoby, jak těchto cílů dosáhnout. Kapitola také rozděluje CAD systém na jednotlivé části a popisuje, jakým způsobem budou tyto části ve výsledném systému obsaženy.

Čtvrtá kapitola se věnuje samotné implementaci systému. Nejprve jsou rozebrány nástroje zvolené pro implementaci systému. Poté kapitola popisuje způsob, jakým jsou vytvořeny a propojeny jednotlivé části systému. Kapitola se také zabývá tím, k jakým změnám v návrhu došlo v průběhu vývoje systému.

Předposlední kapitola shrnuje, jakých cílů se podařilo resp. nepodařilo dosáhnout ve výsledném systému. Dále jsou zde zmíněna možná další rozšíření, která by bylo vhodné do systému přidat.

Samotný závěr práce obsahuje celkové shrnutí výsledků této práce a zmiňuje přínos celého procesu vývoje CAD systému.

Kapitola 2

CAD systém

Tato kapitola se zabývá základní problematikou CAD systémů, především pak systémů pracující s 2D prostorem. Kapitola dále pojednává o reprezentaci dat v CAD systému a popisuje, jaké jsou hlavní nároky na uživatelské rozhraní. Obsah a logická stavba kapitoly vychází z materiálů k přednáškám z předmětu Vizualizace a CAD na Fakultě informačních technologií VUT v Brně [10], [7], [8], [9].

2.1 Charakteristika

Pro přiblížení se problematice si nejprve definujeme pojem CAD systém. CAD je zkratka anglických slov *„Computer Aided Design“*[2], což můžeme přeložit jako *počítačem podporované konstruování*. CAD systém se používá v mnoha oborech (strojírenství, stavebnictví, medicína a další) pro vytvoření výkresové dokumentace, usnadnění a zrychlení práce a snížení výrobních nákladů. CAD je součástí skupiny více systémů pro počítačovou podporu výroby, jako jsou CIM, CAM, CAE a mnoho dalších, ale těmi se v této práci zabývat nebudeme.

CAD systémy můžeme obecně rozdělit do tří skupin:

1. **Malé systémy** – umožňují konstruování pouze ve 2D prostoru a generování výkresových dokumentací.
2. **Střední systémy** – na rozdíl od malých systémů obsahují i nástroje pro práci s trojrozměrným modelem. Pracují s menšími sestavami ve 3D prostoru.

3. **Velké systémy** – jsou plně trojrozměrné. Výkresovou dokumentaci generují na základě trojrozměrného modelu.

2.2 Požadavky

Jelikož v dnešní době jsou CAD systémy používány ve výrobě, stávají se tak plnohodnotnými konstrukčními nástroji. To však vyžaduje větší požadavky na přesnost modelu. V první řadě musí CAD zajistit dostatečnou numerickou přesnost. Dále je důležité, aby systém popisoval vytvořený model přesným geometrickým popisem a nalezl správné analytické řešení geometrických operací s modelem. V kvalitním systému nesmí pro zjednodušení práce chybět možnost zadávání přesných souřadnic uživatelem, např. příkazový řádek nebo tzv. *uchopovací mód* pro výběr již existujících souřadnic.

Pracujeme-li se složitějšími modely, potřebujeme je často rozdělit na menší logické celky. CAD systémy nabízejí různé nástroje, které to umožňují. Mezi takové nástroje patří tzv. *hladiny*. Hladiny mohou obsahovat jednotlivé prvky modelu a definovat jejich vlastnosti. Umožňují model rozdělit na prvky se shodnými vlastnostmi nebo na části, z kterých se model skládá. Některé systémy rozdělují trojrozměrný model do stromově uspořádané struktury, kterou nazýváme *sestavou*. Jednotlivé části sestavy jsou propojeny geometrickou vazbou.

2.3 Reprezentace dat

V této části si popíšeme, v jakém formátu CAD systémy reprezentují data při konstruování v dvojrozměrném prostoru. V podkapitole 2.2 jsme se zmínili, že pro větší přesnost systém popisuje model geometrickým popisem. V případě dvojrozměrného prostoru se tedy model skládá z vektorových entit (úsečky, kružnice, křivky, apod.). Jsou vhodné pro přesnou práci s modelem při editaci jednotlivých entit i při různých transformacích modelu. V některých případech potřebujeme přenést data z jednoho systému do druhého. Reprezentace dat pomocí vektorových entit je poměrně jednoduchá a jednoznačná a umožňuje relativně snadné sdílení dat mezi systémy.

2.4 Uživatelské rohraní

CAD systém je grafický nástroj, tudíž by se mělo jednat o systém s grafickým uživatelským rozhraním (zkráceně GUI). Jaké jsou požadavky na grafické rozhraní? Systém by měl umožňovat rychlou a efektivní práci. Uživatel pak nemusí neustále opakovat stejné postupy, není-li to opravdu nutné. Pro rychlou práci s rozsáhlejšími modely potřebuje systém dostatek paměti a pro rychlé zobrazování je vhodné použit softwarovou či hardwarovou akceleraci. Systém by měl rovněž poskytnout snadné a intuitivní ovládání. Z toho plyne, že je potřeba zajistit přehlednost a jednoduchost používání nástrojů a funkcí systému. V systému by také neměl chybět způsob zadávání příkazů přímo od uživatele. Může např. obsahovat příkazový řádek, kde uživatel zadává parametry funkcí nebo celé příkazy, například pomocí skriptovacího jazyka. Základní ovládání CAD systému je z pravidla zajištěno počítačovou myší a klávesnicí. Může zde být využito i jiných ovládacích prvků, jako jsou různé tablety a tzv. *piloti a navigátoři*, kteří mohou práci značně usnadnit a urychlit.

Kapitola 3

Návrh aplikace

V této kapitole se budeme věnovat kompletnímu návrhu aplikace. Pro začátek si uvedeme, co bude od aplikace očekáváno. Dále si vysvětlíme, jakým způsobem budou v aplikaci obsaženy datové informace o modelu výkresové dokumentace. Popíšeme si jednotlivé části grafického uživatelského rozhraní. V závěru kapitoli si rozebereme podrobnější návrh dalších důležitých částí aplikace. Během vývoje aplikace vyvstaly nové požadavky, se kterými původní návrh nepočítal. Nebudeme se jimi zde zabývat, ale popíšeme si je v kapitole 4 věnované samotné implementaci aplikace.

3.1 Cíle

Než se pustíme do návrhu celého projektu, ujasníme si, co ve výsledku budeme od aplikace očekávat a jaké na ni budou kladeny požadavky. Těmito informacemi se řídí návrh i vývoj aplikace.

Cílem práce je vytvořit jednoduchý CAD systém pracující v dvojrozměrném prostoru. Je ovšem nutné podotknout, že se nebude jednat o převratný systém v oblasti konstruktérských nástrojů. V Podkapitole 2.2 je popsáno, jaké požadavky jsou v dnešní době na CAD systémy kladeny. Návrh i implementace se soustředí na splnění hlavních požadavků. Konkrétně by výsledný systém měl splňovat:

- Jedná se o aplikaci s grafickým uživatelským rozhraním, které umožňuje snadnou a rychlou práci. Přesný návrh GUI si popíšeme později.
- Aplikace se ovládá pomocí myši a klávesnice.

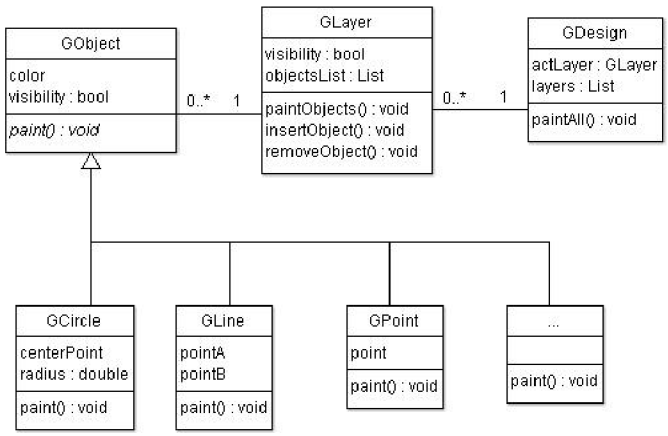
- Návrh výkresové dokumentace je reprezentován vektorovými entitami.
- Jednotlivé entity jsou uspořádány do hladin.
- Parametry vektorových entit umožňuje systém zadávat i pomocí příkazové řádky.
- Systém bude vybírat existující body za pomocí uchopovacích módů.
- Na vykreslování je použita hardwarová akcelerace.
- Je možno uložit výkresovou dokumentaci.

3.2 Reprezentace dat

Jak bylo již zmíněno, data v modelu výkresové dokumentace musí mít patřičnou přesnost a musí být zajištěno i přesné řešení geometrických operací. Elementární jednotkou modelu bude tedy vektorová entita. Vektorové entity budou zadány pomocí bodů a vzdáleností. Abychom dodrželi potřebnou přesnost budou tyto hodnoty reprezentovány alespoň jako 32-bitové číslo s desetinnou čárkou. Pro začátek budeme počítat s nejzákladnějšími entitami pro CAD systém – bodem, úsečkou, kružnicí, obdélníkem. Pro každou entitu vytvoříme samostatnou třídu. Vytvoříme abstraktní třídu nad entitami, která nám bude reprezentovat libovolný typ vektorového objektu. Třída bude obsahovat vlastnosti a operace společné pro jednotlivé typy objektů. Aplikace pak bude pracovat s těmito objekty bez ohledu na jejich typ. To nám umožní případné pozdější rozšiřování aplikace o nové druhy vektorových entit. Vytvoříme pro ně pouze novou třídu.

CAD systém by měl poskytnout jistý druh organizace dat. V našem případě implementujeme třídu, která bude reprezentovat nám již známé hladiny. Každý vektorový objekt bude patřit právě do jedné hladiny, která bude uchovávat seznam svých objektů. Hladinám budeme moci nastavit jejich viditelnost a barvu v modelu. Tyto vlastnosti budou aplikovány na všechny objekty obsažené v hladině. To nám umožní seskupovat je do logických celků a odlišit je barvou. Návrh počítá i s možností, kdy pro jednotlivé objekty v hladině je definována barva a viditelnost jiná než u hladiny. Hladiny budou tvořit celkový návrh výkresové dokumentace, pro niž bude vytvořena třída nesoucí informaci o hladinách obsažených v celkovém návrhu.

Objektový návrh pro reprezentaci dat znázorňuje diagram tříd na obrázku 3.1. Třída `GDesign` představuje celkový model výkresové dokumentace a obsahuje seznam všech vrstev modelu. Atribut `actLayer` označuje aktivní hladinu, do níž se vloží nově vytvořené



Obrázek 3.1: Diagram tříd popisující reprezentaci dat

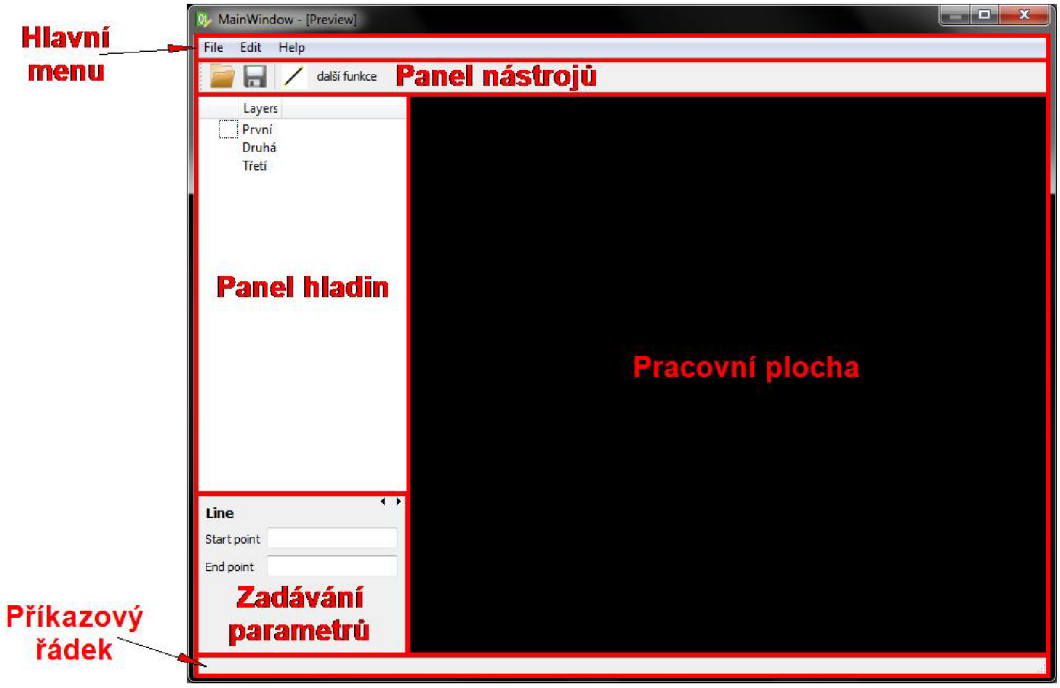
objekty. Aktivní je vždy právě jedna hladina, tu zadává uživatel. Pro vykreslení celého modelu obsahuje třída metodu `paintAll()`, která zajišťuje postupné vykreslení všech viditelných hladin v modelu. `GLayer` pak znázorňuje třídu reprezentující hladinu. Obsahuje seznam objektů v dané hladině a zajišťuje jejich vykreslení. Abstraktní třída pro vektorové objekty je nazvána `GObject` a od ní jsou odvozeny třídy pro jednotlivé druhy vektorových entit. Základními grafickými prvky obsaženými ve většině programů pro kreslení v rovině jsou úsečky, lomené čáry, kružnice, elipsy, mnohoúhelníky, křivky a textové řetězce.[1] Třída označená třemi tečkami znázorňuje ostatní zmíněné vektorové objekty, popřípadě umožňuje přidávání ještě dalších typů. `GObject` definuje metodu `paint()`, která je pak přetížena u všech typů vektorových entit. Každá entita se totiž vykresluje odlišným způsobem, ale jejich vykreslení je voláno stejným způsobem. Každý typ entit je zadán jiným geometrickým popisem. Např. přímka (GLine) je zadána dvěma body a kružnice (GCircle) jedním bodem a hodnotou udávající její poloměr.[1]

3.3 Návrh grafického uživatelského rozhraní

Před tvorbou aplikace je důležité, jaké rozhraní bude použito a jaké možnosti ovládání bude uživateli nabízet. V případě CAD systému navrhneme grafické rozhraní. Jak by mělo vypadat správné GUI pro dnešní CAD systém je popsáno v podkapitole 2.4. Zde si rozebereme podrobněji návrh rozhraní použitého v této práci. Bude se jednat o okenní aplikaci.

Na obrázku 3.2 můžeme vidět strukturu hlavního okna aplikace. Okno obsahuje hlavní

[1] Převzato z kapitoly 3 z knihy [12]



Obrázek 3.2: Okno aplikace

menu, které se skládá z nástrojů pro práci s oknem aplikace a s vytvářeným modelem výkresové dokumentace a poskytuje informace o aplikaci a nápovědu k jejímu používání.

Uživatelské rozhraní by mělo umožňovat rychlou práci při vytváření modelu. Z tohoto důvodu je dobré umístit často používané funkce a nástroje do panelu nástrojů. V tomto CAD systému budou do takového panelu umístěny nástroje pro práci se souborem a funkce pro vytváření nových vektorových objektů. S vývojem aplikace bude přibývat počet funkcí v nástrojovém panelu. Způsobeno to může být přidáváním nových typů entit nebo nových možností jejich zadávání. V tom případě je pak vhodné nástrojový panel rozdělit na panely dva. První panel umožní pracovat se souborem a druhý uživateli poskytne rychlou a přehlednou tvorbu výkresové dokumentace.

Všechny vektorové entity obsažené v modelu budou organizovány do jedné nebo více hladin. V jedné části okna aplikace bude seznam hladin. Pomocí něj budeme moci do modelu přidávat nové hladiny nebo je z něj umazávat. Seznam také bude umožňovat definovat viditelnost a barvu hladin. Pro tyto vlastnosti budou do seznamu přidány jednoduché ikony znázorňující aktuální stav hladiny a pomocí těchto ikon bude možné jejich stav změnit. Pokud na název hladiny dvakrát poklepeme tlačítkem myši, vybereme ji jako aktivní hladinu, do které se budou vkládat následně vytvořené objekty.

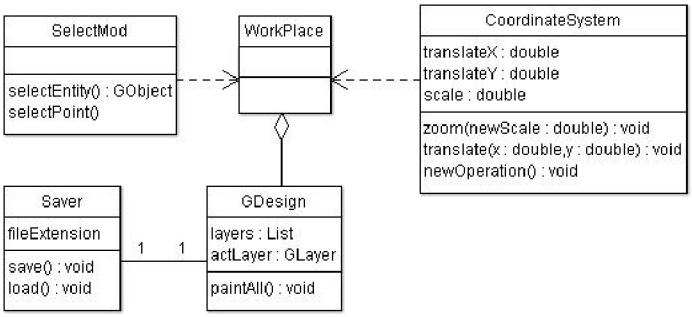
V levé dolní části okna bude vytvořen dynamický panel, který bude měnit svůj obsah na základě právě zvolené funkce. Jeho účelem je zadávání přesných parametrů objektů uživatelem pomocí klávesnice. V tomto případě aplikace umožňuje zadávat pouze jednotlivé parametry. Aplikace by mohla být rozšířena o zadávání celých příkazů pomocí maker psaných např. v některém ze skriptovacích jazyků. Tento návrh však implementaci maker neuvažuje.

Hlavní částí uživatelského rozhraní je pracovní plocha, na které bude zobrazen aktuální stav návrhu výkresové dokumentace. Bude umožňovat interakci s uživatelem, kdy uživatel bude ovlivňovat zobrazení návrhu pomocí transformací (přiblížení, oddálení, posunutí). Dále umožní pomocí myši zadávat souřadnice jako parametry operací prováděné nad návrhem. Pro přesné zadávání zde bude využito uchopovacího módu. Aplikace by měla umožnit tvorbu i rozsáhlých výkresových dokumentací, které mohou obsahovat velké množství objektů. Pro zajištění rychlého vykreslování většího počtu objektů a lepší práci při transformacích obrazu využijeme v této části okna aplikace hardwarové akcelerace. Návrh a implementace uchopovacího módu a hardwarové akcelerace budou popsány níže.

Poslední částí okna je stavový řádek. Do stavového řádku budou vypisovány informace o aplikaci a vytvářeném modelu. V jedné části stavového řádku budou při pohybu myši nad pracovní plochou dynamicky vypisovány souřadnice. Udrží se tím lepší přehled o aktuální pozici v modelu a alespoň přibližná informace o souřadnicích při zadávání parametrů pomocí myši.

3.4 Důležité části aplikace

Tato podkapitola se zabývá ostatními částmi CAD systému, které je nutné implementovat pro splnění původně daných cílů a jejichž návrh doposud nebyl popsán. Jedná se o části důležité pro správnou práci s pracovní plochou aplikace a daty reprezentujícími výkresovou dokumentaci. Objektový model těchto částí můžeme vidět na obrázku 3.3. Třída `WorkPlace` představuje pracovní plochu. Do této plochy jsou vykreslována data z třídy `GDesign`. O struktuře dat v této třídě pojednává kapitola 2.3. `Saver` má na starost ukládání těchto dat do souboru v zadaném formátu. Zvětšením, zmenšením nebo posunutím vykreslované oblasti v pracovní ploše je ovlivněna třída `CoordinateSystem`. Uchovává informace o transformacích nad pracovní plochou a jejím úkolem je zajistit správný převod hodnot mezi souřadným systémem okna a souřadným systémem celého modelu výkresové dokumentace. Na základě aktuálně prováděných akcí v pracovní ploše je zvolen typ uchopovacího módu (třída `SelectMod`). Rozhoduje se, zda se budou uchopovat celé objekty či pouze jejich body a jaké druhy bodů se budou brát v úvahu.

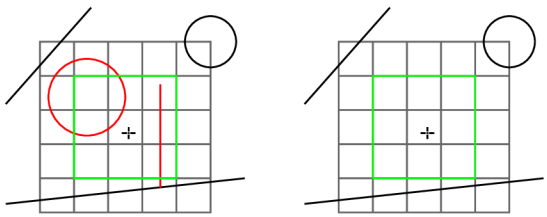


Obrázek 3.3: Zjednodušený diagram tříd pro práci s pracovní plochou

Nyní si popíšeme podrobněji zmíněné části a způsoby jejich možné implementace.

3.4.1 Uchopovací mód

Základním předpokladem uchopovacího módu je schopnost rozlišit objekty v obraze na základě uživatelovy aktivity, kterou může být například pohyb myši nad pracovní plochou. Při každém pohybu se prohledá datová oblast a zjistí se nejbližší objekt. V případě velkého počtu objektů se může stát prohledávání časově náročné a neefektivní. Zavedeme pro to redukci počtu prohledávaných objektů pouze na ty nejblíže kurzoru myši. Data se neprohledávají, když uživatel pohybuje kurzorem v oblasti, kde v jeho blízkosti nejsou žádné objekty. Tento návrh bere v úvahu více možností implementace. Změna způsobu výběru entit poblíž kurzoru by měla být umožněna novou implementací pouze třídy uchopovacího modu.



Obrázek 3.4: Ukázka výběru objektů pomocí mřížky.

Jedním způsobem je použití pevné mřížky, rozdělující pracovní plochu na několik sektorů. Mřížka může být skryta, uživatel by tedy nemusel mít ponětí o její existenci, nebo může být zobrazena a využita pro zlepšení možnosti vkládání přesných souřadnic. Uživatel by

v tomto případě definoval měřítko a krok mřížky. Každý sektor by nesl informaci o objektech, které v něm jsou obsaženy. Uchopovací mód by pak uvažoval objekty obsažené v sektoru pod pozicí kurzoru, popř. v několika úrovních okolních sektorů. Pro lepší pochopení slouží obrázek 3.4. Zeleně je vyznačena výběrová oblast (osmi-okolí kolem sektoru pod kurzorem). Obrázek 3.4(a) ukazuje případ výběru objektů v oblasti, jsou zvýrazněny červeně. Obrázek 3.4(b) znázorňuje pohyb kurzoru v oblasti bez objektů.

V této práci již bylo zmíněno využití akcelerovaného grafického výstupu. Uchopovací mód může využít nástrojů pro výběr prvků v obrazu, které knihovny pro práci s akcelerovaným grafickým výstupem nabízejí. V našem případě využijeme možnost použití tzv. *selection bufferu* z rozhraní openGL. Jak se s ním pracuje a jak je použit při výběru objektů pro uchopovací mód, je blíže popsáno v kapitole 4.1.1.

Uchopovací mód však nebude mít na starost pouze výběr objektů v obraze. Jeho dalším úkolem bude výběr speciálních bodů patřících vybraným entitám. Nejprve se naleznou objekty v určené vzdálenosti od kurzoru. Třída uchopovacího módu bude mít přístup k těmto objektům a nad každým zavolá metodu, která nalezne všechny speciální body pro daný objekt. Budou se brát v úvahu pouze do definované vzdálenosti od kurzoru. Ze všech nalezených bodů, které vyhovují i vzdálenostně, se vybere ten nejblíže kurzoru. Vybraný bod se mění na základě pohybu kurzoru nad pracovní plochou.

Typy uchopovacích bodů

Pokud v obraze nalezneme alespoň jeden objekt, je z něj možné určit speciální typy bodů patřící objektům. Uživatel pak může nastavením uchopovacího módu určit, které typy bodů se k výběru budou nabízet. Základem jsou body určující parametry objektů. U přímky to mohou být její koncové body. Dalším typem může být středový bod přímky nebo kružnice. Často se setkáváme s body rozdělujícími kružnici na kvadranty. Body leží na kružnici a získáme je přičtením či odečtením poloměru kružnice ve směru os souřadné soustavy. Další typy bodů pak závisí na výběru více objektů. Přiblíží-li se kurzor svým výběrovým okolím ke dvěma nebo více objektům zároveň, zjistíme vzájemnou polohu těchto objektů v rovině, a má-li to smysl, určíme jejich společné body. Mohou jimi být průsečíky nebo tečné body. Těchto a dalších bodů můžeme využít při dynamickém kreslení pomocí kurzoru. Na obrázku 3.5 je tento postup naznačen pro vložení kolmé (3.5(a)) a tečné (3.5(b)) úsečky. Po přiblížení se kurzoru k objektu se tyto body dopočítají a kreslená úsečka se k nim přichytí (znázorněno zelenou barvou).

(a) kolmost (b) tečnost

Obrázek 3.5: Příklad přichycení bodu při kreslení přímky.

3.4.2 Souřadný systém

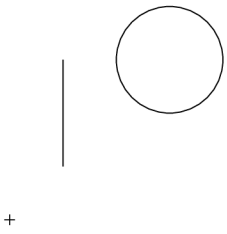
Aplikace používá několik odlišných souřadných systémů. V našem případě nás budou zajímat dva z nich. Vprvní řadě potřebujeme mít informaci o pozici kurzoru v okně aplikace a to především v případě, pokud kurzorem pohybujeme nad pracovní plochou. Tento systém se bude měnit pouze při změně velikosti pracovní plochy, jinak bude neměnný a počátek bude ve stejném místě (roh plochy). Druhý souřadný systém bude patřit modelu výkresové dokumentace. Souřadnice těchto dvou systému se nemusí shodovat a ve většině případů se ani shodovat nebudou. Jelikož uživatel zadává parametry objektů pomocí kurzoru myši, je nutné souřadnice kurzoru na obrazovce převést na odpovídající souřadnice v modelu. Stejně tak potřebný je i opačný převod, kdy uživatel parametr zadává přes příkazovou řádku a je třeba objekt před jeho vytvořením dynamicky vykreslit na správném místě na pracovní ploše. Odlišnost souřadných systémů je dána provedenými transformacemi zobrazení pracovní plochy. Pro přepočet souřadnic nebo jiných hodnot (např. vzdálenosti) využijeme zpětné transformace. Vytvoříme třídu uchovávající informace o posunutí a zvětšení obrazu, které potřebujeme pro převod mezi souřadnými systémy. Pro správné fungování převodu musí být každá změna zobrazení pracovní plochy v této třídě zaznamenána.

3.4.3 Práce se souborem

Výstupem CAD systému je výkresová dokumentace. Doposud jsme se zabývali její tvorbou, ale často výsledek tvorby potřebujeme uložit, znovu otevřít nebo přenést do jiného systému pro další zpracování. Systém by měl umožnit export do více formátů. Pokud není účelem přenesení do jiného systému, uložíme výsledek práce v CAD systému do vlastního formátu. Ukládat budeme pomocí značkovacího jazyka XML. Využijeme struktury reprezentace dat v systému a data v této struktuře uložíme do souboru s příponou xml. Navržený formát podrobněji popisuje následující příklad.

Příklad uložení ve vlastním formátu XML

```
<?xml version=,,1.0‘‘ encoding=,,UTF-8‘‘?>
<project program=,,cad_system‘‘ version=,,1.0‘‘>
    <layer name=,,nová‘‘ color=,,#ff0000‘‘ visibility=,,false‘‘/>
    <layer name=,,Default‘‘ color=,,#ffffff‘‘ visibility=,,true‘‘>
        <gpoint color=,,#ffffff‘‘ visibility=,,true‘‘>
            <point x=,,0‘‘ y=,,0‘‘/>
        </gpoint>
        <gline color=,,#ffffff‘‘ visibility=,,true‘‘>
            <pointA x=,,10‘‘ y=,,100‘‘/>
            <pointB x=,,10‘‘ y=,,10‘‘/>
        </gline>
        <gcircle color=,,#ffffff‘‘ visibility=,,true‘‘>
            <centerPoint x=,,100‘‘ y=,,100‘‘/>
            <radius value=,,50‘‘/>
        </gcircle>
    </layer>
</project>
```



Obrázek 3.6: Ukázka grafické reprezentace souboru uloženého ve vlastním formátu xml (bod, úsečka a kružnice)

Hlavní elementem souboru je `<projekt>` reprezentující třídu `GDesign`[2]. Atributy určují verzi programu, ve kterém byl soubor vygenerován. Projekt je rozdělen na hladiny. Příklad představuje model se dvěma hladinami, které tvoří elementy označené `<layer>`. Jejich atributem je unikátní jméno, barva a stav viditelnosti. Hladina pojmenovaná „nová“ neobsahuje žádné objekty, druhá hladina pak v sobě nese 3 objekty: bod (`<gpoint>`), úsečku (`<gline>`) a kružnici (`<gcircle>`). Samotné objekty mají také jako atribut barvu a viditelnost. Vlastnosti jsou zděděné od hladiny, ve které jsou uloženy, ale uživatel bude mít

[2] viz. obrázek 3.1

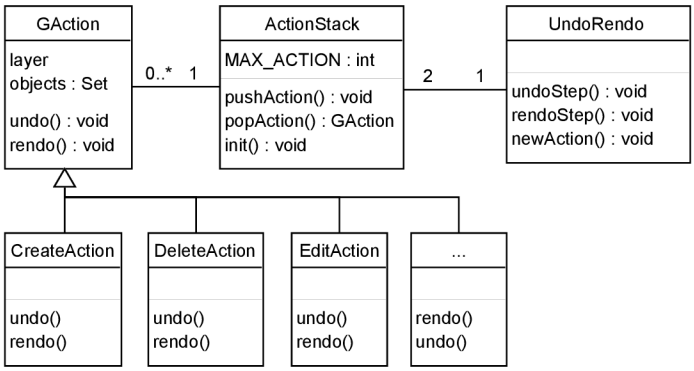
možnost tyto vlastnosti pro jednotlivé objekty změnit. Synovskými elementy jsou pak hodnoty nebo souřadnice, pomocí nichž jsou objekty zadány (přímka dvěma body, kružnice středem a poloměrem, apod.). Jak vypadá grafické zobrazení tohoto příkladu, můžeme vidět na obrázku 3.6.

3.4.4 Funkce zpět a vpřed

Při práci systému může nastat situace, kdy se uživatel potřebuje vrátit k nějakému předešlému stavu tvorby jeho návrhu výkresové dokumentace. Návrh se mezitím mohl hodně změnit a opakovat postup jeho tvorby by bylo pracné a časově náročné. V takovém případě je vhodné do systému naimplementovat funkce vrácení kroku zpět. Funkce by umožnila vrátit do návrhu smazané objekty a hladiny a vracela by změny provedené při editaci vlastností a parametrů jednotlivých prvků návrhu. Další funkce by umožňovala krok v opačném směru, tedy vpřed. Uživatel by se mohl libovolně pohybovat mezi různými stavy návrhu.

Možností jak implementovat funkce zpět a vpřed je více, přičemž vždy je potřeba vytvořit vhodnou datovou strukturu. První z možností je obousměrně provázaný seznam akcí. Pohyb mezi stavy návrhu by se pak shodoval s pohybem v seznamu ve dvou směrech. Další možností je implementovat funkci jako dva zásobníky. Každá nová akce by se vkládala na vrchol jednoho zásobníku, který by reprezentoval zásobník akcí pro funkci zpět. Při každém kroku vrácení by se akce odstranila z vrcholu tohoto zásobníku a vložila se na vrchol druhého zásobníku zastupujícího funkci vpřed. Nevýhodou takového řešení je práce se dvěma zásobníky a neustálé přehazování akcí z jednoho zásobníku na druhý. Pro správný chod funkcí je ovšem nutné při vytvoření každé nové akce nenávratně zahodit všechny akce náležící funkci vpřed, která se v takovém případě stane neaktivní. V tomto případě je výhoda v použití dvou zásobníků, kdy pouze vyprázdníme zásobník akcí pro funkci vpřed.

Obrázek 3.7 znázorňuje objektový model funkcí zpět a vpřed. Akce prováděné v návrhu jsou zde reprezentovány třídou `GAction`. Obsahuje informaci o objektech a hladinách, nad kterými byla akce provedena. Metody `undo()` a `rendo()` pak definují chování akcí při kroku zpět a vpřed. Jiné chování bude mít vymazání objektu a jiné změna parametrů hladiny. Z toho důvodu bude třída `GAction` abstraktní a pro každý druh akce z ní bude odvozena nová třída. V obrázku jsou reprezentovány objekty `CreateAction`, `DeleteAction` a `EditAction`. Tři tečky znázorňují možnost přidání dalších typů prováděných akcí. Třída `ActionStack` reprezentuje zásobník akcí. Konstanta `MAX_ACTION` udává maximální možný počet akcí na zásobníku, tzn. počet možných kroků zpět (vpřed). `UndoRendo` je pak třídou starající se o celý chod funkcí zpět a vpřed. Obsahuje právě dva zásobníky akcí. Stará se



Obrázek 3.7: Objektový model návrhu funkcí zpět a vpřed

o vytváření nových akcí a jejich následné vložení na zásobník. Řídí přenos akcí z jednoho zásobníku na druhý a mazání zásobníku reprezentujícího funkci vpřed při vytvoření nové akce.

Kapitola 4

Implementace

Kapitola popisuje implementaci návrhu CAD systému. Nejprve si stručně popíšeme nástroje využité při implementaci aplikace. Dále si uvedeme, jak vypadá výsledné grafické uživatelské rozhraní. Výsledná aplikace se skládá z několika logických celků. Kapitola popisuje jejich tvorbu a práci v systému. Zejména se pak zaměříme na odlišnosti od návrhu, ke kterým došlo při vývoji aplikace.

4.1 Implementační nástroje

V této části si přiblížíme základní informace o nástrojích použitých při implementaci CAD systému. Systém bude implementován v objektově orientovaném jazyce C++ s využitím knihoven Qt. Co je Qt a jak bylo využito při tvorbě systému, se dozvíme později. Pro práci s grafickým výstupem je použito rozhraní openGL. Obecné pojednání o openGL vychází především z knihy [11]. Základní popis Qt je převzat z internetové stránky [1], kde můžete naleznout mnoho dalších informací.

4.1.1 OpenGL

OpenGL je programové rozhraní grafického hardwaru. Toto rozhraní se skládá z asi 250 příkazů, které se používají pro určení objektů a operací potřebných k vytvoření interaktivních trojrozměrných aplikací. Neobsahuje žádné příkazy pro práci s okny aplikace či získávání uživatelských vstupů. Proto je nutné použití jiného systému pracujícího s okny podle hardwaru, který používáme. OpenGL také neobsahuje příkazy pro popisování troj-

rozměrných objektů. Složitější objekty se modelují za pomocí skládání jednoduchých geometrických útvarů, přičemž se používá bodů, úseček, trojúhelníků, obdelníků, rovinných polygonů a rastrových objektů (dvojrozměrné obrázky, bitmapy a fonty). Těmito vlastnostmi zajišťuje nezávislost na použitém hardwaru a použitelnost na mnoha systémových platformách.

OpenGL si můžeme představit jako stavový automat. Obsahuje spoustu stavových proměnných, které určují vlastnosti vykreslovaného obrazu a objektů v něm. Každá vlastnost je použita se stejnou hodnotou do doby, než je tato hodnota změněna. Nastavíme-li například barvu objektu na modrou, budou se všechny objekty vykreslovat modře. Jakmile změníme tuto barvu na jinou, budou se objekty dále vykreslovat touto novou barvou. Další stavové proměnné řídí aktuální pohled na vykreslované objekty, transformace jejich projekce, vzory a módy kreslení objektů, pozice a charakteristiky světel a spoustu dalších.

V této práci je využito OpenGL pro nastavení projekce a vykreslení 2D scény reprezentující návrh výkresové dokumentace tvořené v systému. Významnou částí systému je uchopovací mód. Pro jeho realizaci je využit mechanismus pro výběr objektů v OpenGL. Nejprve se vykreslí aktuální scéna. Poté se přepneme pomocí funkce `glRenderMode` s parametrem `GL_SELECT` do módu výběru a scénu překreslíme. Jakmile opustíme mód výběru, OpenGL vrací seznam základních objektů, které se protínají s objemem pohledu. Vrácený seznam základních objektů je ve formě pole celočíselných *jmen* a souvisejících dat. V okamžiku vydávání příkazů kreslení v módu výběru ukládáme celočíselná jména objektů na tzv. *zásobník jmen*. Navrácený seznam jmen můžeme tedy použít k určení těch základních objektů, které mohly být uživatelem na obrazovce vybrány.

4.1.2 Qt

Qt je multi-platformní knihovna často používaná pro vytváření aplikací s grafickým uživatelským rozhraním. Je možné ji ovšem použít i pro vytváření konzolových aplikací. Knihovna zajišťuje přenositelnost bez nutnosti přepisování zdrojového kódu mezi mnoha různými distribucemi systémů Mac OS, Linux a Windows. Také podporuje vývoj aplikací na platformách pro vestavěné systémy (Embedded Linux, Windows CE, Maemo, Symbian). Vývoj knihoven Qt započal v roce 1996 jako produkt firmy známé pod názvem Trolltech. Kvůli zlepšení své multi-platformní strategie přešel Trolltech v červnu 2008 pod společnost Nokia, čímž se vytvořila vývojová skupinu *Qt Development Frameworks*.

Pro vývoj této práce bylo využito volně dostupného balíčku Nokia Qt SDK, který obsahuje veškeré knihovny Qt verze 4.6, integrované vývojové prostředí Qt Creator IDE verze 1.3.0

a další vývojové nástroje. Pro lepší pochopení práce s knihovnami Qt pak bylo využito rozsáhlé nápovědy integrované přímo do vývojového prostředí Qt Creatoru a nástroje Qt Demo, obsahujícího několik příkladů využití knihoven Qt. Celou nápovědu a popis příkladů lze také naleznout na internetových stránkách [4]. Všechny třídy začínající na velké písmeno *Q*, o kterých je v této práci zmínka, patří do prostředí knihoven Qt.

4.1.3 Správa verzí a dokumentace zdrojových textů

Jedním z cílů této práce bylo vyzkoušet si práci na větším projektu. K tomu patří i používání nástroje pro správu verzí aplikace. Za tímto účelem bylo využito systému SVN hostovaným na serveru indefero.net [3]. Vývojové prostředí Qt Creator nabízí vlastního klienta SVN. Kromě tohoto klienta bylo ještě využito volně dostupného nástroje pro správu verzí aplikace jménem TortoiseSVN [5]. Nástroj je určený k práci s SVN v operačních systémech Windows a jeho výhodou je včlenění přímo do Průzkumníka Windows.

Během vývoje celé aplikace byly zdrojové texty komentovány. Komentáře určené k generování programové dokumentace byly psány stylem JavaDoc a ke generování dokumentace byl použit nástroj Doxygen [6].

4.2 Grafické uživatelské rozhraní

Pro tvorbu grafického uživatelského prostředí bylo využito modulu `QtGui` z prostředí knihoven Qt. Balíček Qt SDK využívaný při vývoji aplikace obsahuje nástroj *Qt Designer*, který umožňuje tvorbu vzhledu GUI z komponent knihovny Qt. Jedná se o tzv. WYSIWYG[1] editor, který umožňuje vytvořit okno aplikace na základě jeho vzhledu. V editoru lze také vytvářet základní funkčnost a chování GUI.

Základním stavebním kamenem uživatelského rozhraní Qt je třída `QWidget` reprezentující jednotlivé rámce okna aplikace. Tvoří stromovou strukturu a oknem je nazýván nejvrchnější uzel struktury, který nemá žádného předchůdce. V této práci je hlavní okno reprezentováno třídou `MainWindow` odvozeného od třídy `QMainWindow` z modulu `QtGui` určené právě pro zastoupení hlavního okna v aplikaci. Pomocí této třídy je do aplikace vloženo hlavní menu, stavový řádek a panely nástrojů. Hlavní menu v sobě obsahuje veškeré funkce pro práci se souborem, objekty a hladiny návrhu výkresové dokumentace, dále pak nabízí nástroje pro editaci pracovní plochy a zobrazení nápovědy programu. Stavového řádku je využito pro

[1] zkratka anglické věty „What you see is what you get“ v překladu „co vidíte, to dostanete“

zobrazení souřadnic pozice kurzoru v návrhu a nastavení uchopovacích bodů pomocí zaškrtávacích políček. Aplikace obsahuje dva panely nástrojů. První je určen pro práci s návrhem výkresu a najdeme v něm funkce zpět a vpřed, uložení, načtení návrhu a vytvoření nového prázdného návrhu. Druhý panel pak obsahuje veškeré funkce pro tvorbu a editaci vektorových objektů návrhu. Prostor hlavního okna je rozdělen do tří rámců. Největším z nich je pracovní plocha popsána v kapitole 4.3. Další důležitou částí je seznam hladin obsažených v návrhu výkresu. Jak jsou v něm uloženy data a jak se s nimi pracuje, zmiňuje kapitola 4.4. Poslední rámec slouží k zadávání parametrů vytvářených objektů. Mění svůj obsah na základě aktuálně vybrané funkce. Je implementován třídou `QStackedWidget` fungující jako jakýsi zásobník svých podrámců, přičemž v daný okamžik zobrazuje právě jeden z nich. Parametry se zadávají do formulářových textových polí. Ve většině případů je požadováno zadání bodu ve formátu X, Y, kde X a Y jsou souřadnice bodu zadané reálným číslem v libovolném zápisu číselných konstant v jazyce C++. Desetinná čárka je značená tečkou a je možný i exponenciální zápis čísla. Bílé znaky jsou při zadávání ignorovány.

4.3 Pracovní plocha

V této části práce si popíšeme, jak byla implementována hlavní část aplikace. Pracovní plocha nese největší podíl na interakci s uživatelem. V první řadě zobrazuje výstup celé aplikace, kterým je aktuální návrh výkresové dokumentace. Plocha také slouží jako vstupní prvek aplikace. Uživatel především za pomocí myši nastavuje pohled na vykreslovaný návrh a vkládá do něj nové objekty zadáním souřadnic – opět pomocí myši. Na rozdíl od původního návrhu aplikace došlo ke sjednocení třídy reprezentující pracovní plochu a třídy obsahující kompletní návrh výkresové dokumentace. Výhodou sjednocení je jednodušší přístup k samotným objektům návrhu při vykreslování a výběru pro uchopovací mód za pomocí OpenGL.

Součástí prostředí Qt je modul usnadňující práci s rozhraním OpenGL. Aplikace využívá třídy `QGLWidget` a z ní vytvořené podtřídy `GLWidget`. Jedná se o rámec v okně pro vykreslování grafické scény v OpenGL. Obsahuje tři virtuální metody pro vykreslování scény, které jsou v nově vytvořené podtřídě implementovány pro potřebu aplikace. První metoda `initializeGL()` je volána pouze jednou, a to při samotném vytvoření rámce. Jejím úkolem je zinicializovat scénu před prvním vykreslením. Následně je volána další metoda pojmenovaná `resizeGL()`. Jak název napovídá, je volána při každé změně velikosti rámce. Má dva parametry udávající novou šířku a výšku rámce a slouží pro nastavení nového pohledu na scénu a její projekce. Poslední a nejdůležitější metodou je `paintGL()`. Pomocí ní se

vykresluje celá scéna. V případě našeho systému v této funkci dochází k nastavení transformací a následnému vykreslení návrhu výkresové dokumentace. Její datová reprezentace se drží návrhu popsaného v kapitole 2.3. To znamená, že jsou vykreslovány všechny vektorové prvky obsažené v hladinách nastavených jako viditelné. Jelikož aplikace pracuje pouze ve dvojrozměrném prostoru, můžeme využít pro vykreslení jednoduchých 2D primitiv pomocí třídy `QPainter`.

Rámec reprezentující pracovní plochu zpracovává události zaslané na základě uživatelovy aktivity s myší a klávesnicí. Události ovlivňují stav a vykreslování návrhu dokumentace, ale způsobují i změny v částech systému související s pracovní plochou. Jak jednotlivé části související s pracovní plochou fungují a jak jsou realizovány v systému, si popíšeme níže.

4.3.1 Transformace obrazu a souřadný systém

Již při návrhu aplikace jsme si řekli, že v pracovní ploše budou obsaženy dva souřadnicové systémy. Souřadný systém určující pozici kurzoru v ploše je uživateli skryt a on nemusí mít vůbec ponětí o jeho existenci. Naopak souřadný systém, který je uživateli určený, je systém patřící návrhu dokumentace. Aplikace musí zajistit správný převod mezi souřadnými systémy. K tomu slouží třída `CoordinateSystem`, která nese informaci o aktuálních transformacích. Uživatel mění pohled scény pomocí kolečka myši nebo funkcí k tomu určených. Pootočením kolečka myši směrem nahoru obraz přiblíží a směrem dolů oddálí. Při těchto akcích je obraz posunut tak, aby pod kurzorem zůstal stejný bod návrhu. Pohled na obraz může uživatel zvětšit pomocí funkce *Zoom In* (*Ctrl* + +) nebo zmenšit funkcí *Zoom Out* (*Ctrl* + −).[2] K posunutí obrazu dojde tahem myši při stisknutém prostředním tlačítku myši nebo přidržené klávese *Shift* a stisknutém levém tlačítku myši. Při každé změně obrazu se mění stav transformací v třídě `CoordinateSystem`, ty jsou pak použity při překreslování pracovní plochy. Třída dále nabízí obousměrný převod souřadnic bodů i číselných hodnot (vzdáleností v obraze).

4.4 Reprezentace dat

Základní datovou jednotkou návrhu výkresové dokumentace je nějaký druh vektorové entity. Zde si podrobněji popíšeme, o jaké entity se jedná a v jaké datové struktuře jsou v aplikaci uloženy. Aplikace v nynějším stavu obsahuje tyto druhy vektorových entit:

[2] V závorkách jsou uvedeny klávesové zkratky pro dané funkce

- **Bod** – je definován vlastními souřadnicemi. V aplikaci je zobrazován jako malý křížek na pracovní ploše. Je to jediný objekt s neměnnou velikostí, a proto je při vykreslování jeho velikost podělena aktuálním zvětšením. Bod reprezentuje třída `GPoint` a je zadáván funkcí pojmenovanou *Point*.
- **Úsečka** – je reprezentována třídou `GLine`. Zadává se dvěma body pomocí funkce jménem *Line*.
- **Lomená čára** – nemá vlastní třídu, jedná se pouze o funkci (*PolyLine*) pro vykreslení jednotlivých na sebe navazujících úseček. Ty jsou pak v návrhu reprezentovány jako samostatné úsečky.
- **Kružnice** – v systému je definována středovým bodem a poloměrem ve třídě `GCircle`. Kružnici je možno zadat funkcí *Circle*, a to výběrem dvou bodů v pracovní ploše pomocí myši. První bod definuje střed kružnice a druhý libovolný bod ležící na kružnici. Místo druhého bodu je možno zadat hodnotu poloměru do políčka *Radius* v panelu pro zadávání parametrů objektů.
- **Obdelník** – zadává se pomocí dvou úhlopříčných bodů (funkce *Rectangle*) a je reprezentován třídou `GRectangle`.

Všechny typy entit jsou odvozeny od třídy `GObject` a pouze přetěžují metody pro vykreslování, počítání vzdáleností a hledání a úpravu svých specifických bodů. Pro uložení informace o číselných hodnotách byl zvolen datový typ s desetinnou čárkou `qreal`. Jedná se o datový typ definovaný v knihovně Qt, který reprezentuje standardní datový typ `double` jazyka C++ na všech platformách kromě těch založených na architektuře ARM (pro ty ale není tato aplikace určena). Pro uložení informace o souřadnicích bodu nebo vektoru byla využita třída z knihovny Qt `QPointF`. Definuje bod s přesností na desetinnou čárku pomocí nám již známého datového typu `qreal` a umožňuje snazší práci při matematických operacích s body a vektory.

Vektorové entity jsou organizovány do jednotlivých hladin návrhu. Pro tyto hladiny je vytvořena třída `GLayer`. Hladiny definují viditelnost a vykreslovací barvu objektu a řídí jejich vykreslování v pracovní ploše. Návrh výkresové dokumentace obsahuje jednu a více hladin, které jsou uvedeny v seznamu v pravé části okna aplikace. Pro zobrazení rámce se seznamem bylo využito třídy `QTreeWidget`. Každý řádek seznamu reprezentuje jednu hladinu návrhu a skládá se ze tří sloupců. První sloupec obsahuje ikonu značící stav viditelnosti. Žlutá ikona (●) znamená, že je hladina viditelná, a šedá ikona (●) definuje opačný stav. Ke změně stavu pak dochází při kliknutí na ikonu tlačítkem myši. Následuje sloupec se jménem hladiny. Jméno je v návrhu vždy unikátní a pomocí něj je hladina v systému zastoupena.

V poslední sloupci je u každé hladiny zobrazena zvolená barva hladiny. Implicitně je tato barva nastavena na bílou barvu. Změnit ji je možná kliknutím na ikonu zastupující danou barvu, čímž se otevře dialogové okno pro výběr nové barvy. Při změně barvy hladiny se objekty obsažené v hladině překreslí na nově zvolenou barvu. Jednotlivé hladiny organizuje a pracuje s nimi třída návrhu výkresové dokumentace a pracovní plochy `GLWidget`. Nově vkládané objekty do návrhu se ukládají do aktuálně vybrané hladiny. Vybrat hladinu lze dvojklikem na název hladiny v jejich seznamu. Vybraná hladina je pak v seznamu barevně zvýrazněna.

4.4.1 Uložení do souboru

Implementace této práce nabízí uložení textového souboru ve formátu XML. Pro uložení souboru jsou v aplikaci dvě funkce *Save* a *Save As*. Pracují na stejném principu, jaký známe z běžných aplikací. V návrhu aplikace v kapitole 3.4.3 jsme si uvedli příklad možného uložení do tohoto formátu. Zde si popíšeme, k jakým změnám v uložení souboru došlo ve výsledném systému. Malá změna byla učiněna v názvu atributu pro označení viditelnosti – z původního `visible` na `visibility`. Změna není nijak významná a název atributu byl zvolen podle názvu proměnné v programu nesoucí tuto informaci. K významnější změně pak došlo při uložení parametrů vektorových entit. V původním návrhu byly uloženy v elementu pojmenovaném shodně s název parametru entity (`<point>`, `<radius>`, atd.). Výsledná aplikace je však ukládá do elementů určujících jejich datový typ (např. `<qreal>` nebo `<qpointf>`) a jejich název je pak uložen v elementu pomocí atributu `name`. Takže například kružnice bude uložena následovně:

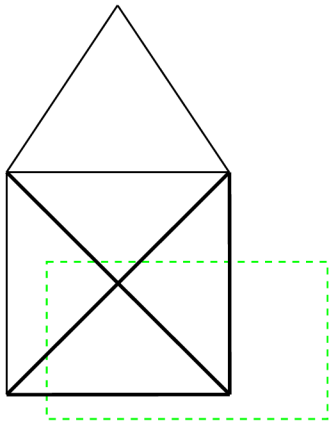
```
<gcircle color=,,#ffffff‘‘ visible=,,true‘‘>
            <qpointf name=,,centerPoint‘‘ x=,,100‘‘ y=,,100‘‘/>
            <qreal name=,,radius‘‘ value=,,50‘‘/>
</gcircle>
```

4.5 Uchopovací mód

O uchopovacím módu bylo už mnohé v této práci napsáno. Nyní se podíváme, jak je v aplikaci implementován a jak ve skutečnosti pracuje.

Třídou pro výběr elementů z obrazu je `DragMode`. Ve výsledné aplikace je výběr implementován pomocí mechanismu obsaženého v rozhraní OpenGL. Kapitola 4.1.1 osahuje popis, jak

tento mechanismus pracuje. Nejdříve je vytvořen indexovaný seznam všech objektů, které se ve scéně vykreslují. Indexy objektů slouží jako jejich jména, která se uloží na zásobník jmen při jejich vykreslování v módu výběru. Vykreslovaných objektů může být ve scéně více, než je maximální velikost zásobníku jmen. Aby nedocházelo k přetečení zásobníku, je výběr rozdělen do několika cyklů v závislosti na počtu objektů. Počet cyklů vypočteme, podělíme-li celočíselně počet objektů velikostí zásobníku jmen a přičteme-li jedničku. Velikost zásobníku se liší podle implementace OpenGL, ale jeho minimální velikost je 64 jmen.[3] Výběrový mód nám ve výsledku vrátí indexy objektů, které se vyskytují ve výběrové oblasti. Výběrová oblast je volena podle typu požadovaného výběru. Vybíráme-li objekty na základě pozice kurzoru v pracovní ploše, pak je výběrovou oblastí zvoleno čtvercové okolí o velikosti $30x30$ pixelů. Aplikace také nabízí výběr objektů za pomocí obdélníkového výběrového okna, pak výběrová oblast odpovídá tomuto oknu. Vybrány jsou tedy objekty, které do výběrového okna zasahují alespoň nějakou svojí částí. Příklad výběru obdélníkovým oknem je znázorněn na obrázku 4.1.



Obrázek 4.1: Příklad výběru za pomocí obdélníkového okna – zeleně je znázorněno výběrové okno a tučně vybrané objekty

V třídě `GLWidget` určené k vykreslování objektů pomocí rozhraní OpenGL je pro výběrový mód implementována metoda `paintGLForSelect()`. Metoda pracuje s indexovaným seznamem objektů a má jediný parametr určující index objektu od kterého začne vykreslování. Metoda nastaví okno na výběrovou oblast a vykreslí objekty od zadaného indexu, ale maximálně tolik, jako je zvolená velikost zásobníku jmen. Před vykreslením objektu je na zásobník uložen jeho index. Pokud chceme vybrat nejbližší objekt ke kurzoru, tak po získání objektů ve výběrové oblasti se vypočítá vzdálenost každého nalezeného objektu od pozice kurzoru v pracovní ploše a vybere se objekt jehož vzdálenost je nejmenší. Za tímto

[3] kapitola 13 str. 476 v [11]

účelem je ve třídě reprezentující všechny typy vektorových objektů (`GObject`) vytvořena virtuální metoda `distance()`, která počítá vzdálenost k bodu předaného v parametru této metody.

Ukázali jsme si, jak systém vybírá požadované objekty, a nyní si popíšeme, jak z vybraných objektů získáme jejich specifické body důležité pro uchopovací mód. Třída `GObject` obsahuje virtuální metodu `getSpecificPoints()`, kterou voláme nad každým vybraným objektem pro nalezení požadovaných specifických bodů. Jedním parametrem funkci předáme hodnotu bitového výčtového typu `POINT_TYPE` definující jaké druhy bodů se budou vybírat. Jednotlivé typy bodů jsou zastoupeny binární logickou hodnotou 1 v patřičném bitu. Použité typy bodů a jejich binární hodnoty zobrazuje tabulka 4.1.

Název typu	**Hodnota**	**Popis**
NO_POINT	0x00	žádný bod není pro výběr zvolen
END_POINT	0x01	koncové body úseček, rohové body obdelníku
CENTER_POINT	0x02	středový bod (kružnice, úsečka)
INTERSECTION_POINT	0x04	průsečík entit (pozn. neimplementováno)
ON_ENTIT_POINT	0x08	bod ležící na entitě (pozn. neimplementováno)
QUADRANT_POINT	0x10	body na kružnici oddělující její kvadranty (0°, 90°, 180°, 270°)
RECT_CENTER_POINT	0X20	průsečík úhlopříček v obdelníku

Tabulka 4.1: Tabulka hodnot výčtového typu `POINT_TYPE`

Jaké typy specifických bodů se budou brát v úvahu, definuje uživatel pomocí zaškrtávacích políček ve stavovém řádku hlavního okna aplikace. Po získání specifických bodů od všech vybraných objektů se zvolí ten nejblíže kurzoru myši. Uchopovací mód je využit při vytváření objektů, ale i jejich editaci (posun vrcholu a objektů). Najde-li v těchto případech uchopovací mód specifický bod a je-li v jeho blízkosti stisknuto (uvolněno) tlačítko myši pro zadávání bodů, pak jsou souřadnice nalezeného bodu zadány jako parametr aktuálně zvolené funkce.

4.6 Funkce zpět a vpřed

V hlavním menu aplikace nebo v panelu nástrojů máme možnost použití funkcí emphZpět ($Ctrl + Z$) a emphVpřed ($Ctrl + Y$). Hlavní okno aplikace zasílá signál při každé aktivaci těchto funkcí třídě `UndoRendo`, která řídí jejich chod. Každé funkci je přiřazen jeden zásobník reprezentován třídou `QStack`. Prvky zásobníků jsou akce provedené uživatelem v systému

při práci s návrhem výkresové dokumentace. Libovolný typ provedené akce je implementován třídou `StackAction`. Třída obsahuje dvě virtuální metody, které definují chování akce při kroku zpět a kroku vpřed. Jednotlivé typy akcí jsou pak vytvořeny odvozením od této třídy. Aplikace obsahuje čtyři zásobníkové akce. První je použita v případě vytváření objektu (třída `CreateAction`), další při mazání jednoho nebo více objektů (`DeleteAction`) a další dvě akce reprezentují posunutí bodu (třída `MoveVertexAction`) a posunutí objektů (`MoveObjectsAction`). Pokud uživatel provede v systému nějakou z těchto aktivit, je v systému vytvořena nová akce a vložena na vrchol zásobníku zastupujícího funkci *Zpět* a druhý zásobník je vyprázdněn. V třídě `UndoRendo` je definována konstanta určující maximální možný počet akcí na zásobníku. Konstanta je implicitně nastavena na hodnotu 100. Pokud je potřeba vložit novou akci a zásobník je plný, bez možnosti návratu se odstraní akce na dně zásobníku. Zde se ukazuje výhoda využití třídy `QStack` z knihovny Qt odvozené od třídy `QVector`. `QVector` reprezentuje dynamické pole, což umožňuje přístup k jeho libovolnému prvku, čehož je využito při získání prvku na dně zásobníku.

Kapitola 5

Výsledky

Úkolem této práce bylo správně pochopit problematiku dnešních CAD systémů, analyzovat a správně navrhnout vlastní CAD systém pro kreslení v dvojrozměrném prostoru s jeho následnou implementací. Nezbývá než zhodnotit výsledek celé práce a popsat možnosti dalšího vývoje programu.

5.1 Dosažené cíle

V kapitole 3.1 byly definovány cíle, kterých měla tato práce dosáhnout, a vlastnosti, které by měla výsledná aplikace obsahovat. Podařilo se vytvořit aplikaci s grafickým uživatelským rozhraním, která obsahuje alespoň základní funkčnost částí, které by v CAD systému neměly chybět. Aplikace umožňuje snadné ovládání za pomocí myši a klávesnice. Při tvorbě GUI byl kladen důraz na jednoduché a intuitivní ovládání více než na vzhled aplikace. Přesnost práce je zajištěna přesným výběrem bodů a objektů za pomocí uchopovacího módu a reprezentací dat vektorovými entitami umožňujícími přesný výpočet dalších potřebných prvků. Aplikace nabízí pouze několik základních vektorových entit pro tvorbu návrhu výkresové dokumentace a jako způsob jejich organizace byly zvoleny hladiny, které zpřehledňují práci v systému za pomocí viditelnosti a barev. Rychlost vykreslování a transformací obrazu rozsáhlejší výkresové dokumentace je v aplikace zajištěna použitím grafické hardwarové akcelerace pomocí rozhraní OpenGL.

5.2 Možnosti dalšího rozvoje aplikace

Podařilo se vytvořit CAD systém pouze se základní funkčností, a proto se nabízí ještě hodně možností jeho rozšiřování. Návrh a implementace některých částí s případnými rozšířeními počítaly. V této části si popíšeme způsob rozšíření takových částí a rozebereme nedostatky funkčnosti a implementace obsahu aplikace. Implementace chybějících částí by zabrala čas, který byl věnován k vytvoření základní funkčnosti jiných důležitých částí systému.

5.2.1 Vektorové entity

V návrhu aplikace bylo zmíněno, že základními vektorovými entitami jsou úsečky, lomené čáry, kružnice, elipsy, mnohoúhelníky, křivky a textové řetězce. Výsledkem této práce je systém pracující pouze s body, úsečkami, lomenými čarami, kružnicemi a obdélníky. Systém by dále měl umožňovat především práci s křivkami a řetězci, které byly plánovány do systému implementovat. Dále v systému chybí zadávání entit různými způsoby. Například kružnici můžeme zadat pouze pomocí středu a bodu nebo středu a poloměru. Chybí tak například zadávání pomocí tří bodů ležících na kružnici nebo pomocí různých kombinací využívajících tečných přímek.

Výsledný systém rovněž nabízí základní editaci již vytvořených vektorových entit. Umožňuje posouvání vrcholů nebo celých objektů. Další možná editace by byla vzájemné ořezávání entit. Plánována byla implementace metody v třídách vektorových objektů. Počítala by průsečíky s entitami a nevyužívalo by se jí jen u ořezávání, ale i v uchopovacím módu. Další možným rozšířením editace entit je aplikování různých transformací na tyto entity. V systému je možnost posunutí, dále by bylo možné implementovat zvětšení entit nebo jejich otočení, zrcadlení a kopírování.

5.2.2 Uchopovací mód

V aplikaci je obsažen funkční uchopovací mód, který nabízí k výběru několik základních typů bodů. Aplikaci je možné rozšířit přidáním dalších typů bodů. Ve zdrojových kódech je připraveno prostředí pro jejich přidávání. Plánované typy bodů jsou shrnuty v tabulce 4.1 – jedná se o dva typy bodů, které se nepodařilo implementovat. První z nich je průsečíkový bod, druhý je pojmenován `ON_ENTIT_POINT` a měl být určen k přichytávání posouvaných objektů k objektům statickým. V tabulce chybí ještě jeden typ bodu uvažovaný v návrhu – tečný bod.

5.2.3 Hladiny

Systém umožňuje organizovat vektorové entity do hladin. Uživatel může vytvářet nové hladiny, definovat jejich barvu a určovat, které hladiny budou zobrazeny. Systém by mohl nabízet rozsáhlejší práci s hladinami. Například by měl umožnit editaci jména hladiny nebo dovolit přesouvání objektů mezi jednotlivými hladinami. V nynějším stavu aplikace hladiny definují stav viditelnosti a barvu. Při změně stavu hladiny se změní i stavy všech objektů obsažených v hladině. Užitečné by mohlo být umožnit uživateli definovat barvu a viditelnost nejen pomocí hladin, ale i pro jednotlivé objekty samostatně.

Systém by také mohl být rozšířen o další způsob organizace dat jako je seskupování objektů. Umožňoval by vybrat několik vektorových entit a spojit je do jednoho objektu. Funkce by pak byly aplikovány na všechny entity v takto spojených entitách.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit CAD systém pro 2D kreslení. Práce splňuje všechny body zadání. Kapitola druhá přináší seznámení s problematikou CAD systémů. Následující kapitola popisuje analýzu dané problematiky a návrh CAD systému pro 2D kreslení. Čtvrtá kapitola se pak zabývá jeho implementací v jazyce C++ a v kapitole páté jsou zhodnoceny dosažené výsledky a možnosti dalšího vývoje systému. Přínosem práce ovšem nebyl pouze výsledný fungující program. Tvorba tohoto systému mi umožnila podívat se na CAD systém z jiného pohledu než okem pouhého uživatele. Přispěla k lepšímu pochopení principů fungujících v těchto systémech a složitosti jejich tvorby. Také jsem měl možnost vyzkoušet si práci s nástroji popsanými v kapitole 4.1.

Vzhledem k podcenění časové náročnosti vývoje systému, je výsledkem práce jednoduchý 2D CAD systém, který nemůže konkurovat dnes běžně používaným profesionálním systémům, avšak obsahuje základní funkčnost prvků, které jsou od takového systému vyžadovány. Systém tedy nabízí možnosti dalšího rozvoje, lze ho doplnit o další funkce již implementovaných částí.

Literatura

[1] *About us — Qt – A cross-platform application and UI framework* [online]. c2008 – 2010, [cit. 2010-05-04].
URL http://qt.nokia.com/about

[2] *Computer-aided design – Wikipedia, the free encyclopedia* [online]. [rev. 20010-04-30], [cit. 2010-05-01].
URL http://en.wikipedia.org/wiki/Computer-aided_design

[3] *Indefero – Code Hosting and Project Management* [online]. c2008 – 2010, [cit. 2010-05-14].
URL http://indefero.net/

[4] *Qt 4.6: Qt Reference Documentation* [online]. c2010, [cit. 2010-05-14].
URL http://doc.qt.nokia.com/4.6/

[5] *tortoisesvn.tigris.org* [online]. c2001 – 2009, [cit. 2010-05-14].
URL http://tortoisesvn.tigris.org/

[6] van HEESC, D.: *Doxygen* [online]. First release 27 oct 1997, [rev. 2010-4-24], [cit. 2010-05-14].
URL http://www.stack.nl/~dimitri/doxygen/index.html

[7] KRŠEK, P.: *Vizualizace a CAD: Reprezentace dat v CAD systémech* [online]. 2008, brno: Fakulta informačních technologií, VUT, 2008, [cit. 2010-04-29].
URL https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/viz_slide_cad_reprezentace_print.pdf

[8] KRŠEK, P.: *Vizualizace a CAD: Sdílení dat v CAD systémech* [online]. 2008, brno: Fakulta informačních technologií, VUT, 2008, [cit. 2010-04-29].
URL https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/viz_slide_sdileni_dat_print.pdf

[9] KRŠEK, P.: *Vizualizace a CAD: Uživatelské rozhraní v CAD systémech* [online]. 2008, brno: Fakulta informačních technologií, VUT, 2008, [cit. 2010-04-29].
URL https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/viz_slide_gui_print.pdf

[10] KRŠEK, P.: *Vizualizace a CAD: Základy CAD systémů* [online]. 2008, brno: Fakulta informačních technologií, VUT, 2008, [cit. 2010-04-29].
URL https://www.fit.vutbr.cz/study/courses/VIZ/private/lecture/viz_slide_zaklady_cad_print.pdf

[11] SHREINER, D.; WOO, M.; NEIDER, J.; aj.: *OpenGL: Průvodce programátora.* Přeložil Jiří Fadrný. Brno: Computer Press, první vydání, 2006, ISBN 80-251-1275-6.

[12] ŽÁRA, J.; BENEŠ, B.; SOCHOR, J.; aj.: *Moderní počítačová grafika.* Brno: Computer Press, první vydání, 2004, ISBN 80-251-0454-0.

Dodatek A

Obsah CD

Jméno adresáře	Obsah adresáře
src	zdrojové kódy aplikace
bin	spustitelná verze programu pod systémem Windows
doc	programová dokumentace
text	textová zpráva k aplikaci ve formátu PDF
tex	zdrojové soubory textové zprávy psané v LaTeXu
manual	nápověda programu ve formátu PDF