

Vektorový grafický 2D editor v MATLABu

2D Vector Graphics Editor in MATLAB

Bc. Jiří Pavlík

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jiří PAVLÍK**
Osobní číslo: **A09484**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Vektorový grafický 2D editor v MATLABu**

Zásady pro vypracování:

1. Vypracujte literární rešerši týkající se daného tématu.
2. V MATLABu vytvořte nadstavbu grafický editor 2D pro vykreslování entit zadaných vedoucím práce.
3. Program musí obsahovat textová menu i panely nástrojů. Entity se budou vykreslovat myší nebo zadáváním souřadnic, musí být umožněno měnit jejich barvu a přemisťovat objekty. Vytvořené obrázky ukládejte ve vlastním formátu.
4. Program umístěte na samostatný CD-ROM jako přílohu diplomové práce.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. PERŮTKA, K. **MATLAB ? Základy pro studenty automatizace a informačních technologií**. 1. vyd. Zlín: UTB ve Zlíně, 2005. 304 s. ISBN 80-7318-355-2.
2. DOŇAR, B.; ZAPLATÍLEK, K. **MATLAB- Tvorba uživatelských aplikací**. 1. vyd. Praha: BEN- technická literatura, 2004. 216 s. ISBN 80-7300-133-0.
3. PROKOP, D. **Průvodce grafickými nadstavbami Octave**. Zlín, 2007. 44 s. Bakalářská práce na Fakultě aplikované informatiky UTB ve Zlíně. Vedoucí bakalářské práce Karel Perůtka.
4. MICHÁLEK, R. **Multimediální učební pomůcka předmětů A5MAS, A4MAS**. Zlín, 2010. 58 s. Bakalářská práce na Fakultě aplikované informatiky UTB ve Zlíně. Vedoucí bakalářské práce Karel Perůtka.
5. HRUBOŠ, P. **Softwarová pomůcka výuky MATLABu**. Zlín, 2009. 73 s. Bakalářská práce na Fakultě aplikované informatiky UTB ve Zlíně. Vedoucí bakalářské práce Karel Perůtka.
6. **MATLAB Homepage**. URL: [<http://www.mathworks.com/>] [cit. 2011-02-07].

Vedoucí diplomové práce:

Ing. Karel Perůtka, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

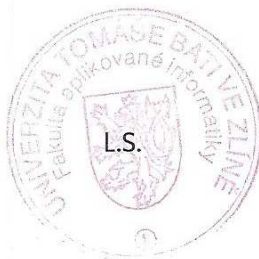
24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Tato práce spočívá v přiblížení oblasti grafiky v MATLABu. Nejdříve je popsána problematika tvorby skriptů a funkcí, které slouží pro naprogramování složitějších a bohatých programů. Dále pak následuje popis tvorby objektově orientovaných programů, které jsou od verze R2008a v MATLABu k dispozici. Poslední oddíl teoretické části spočívá v popisu systému Handle Graphics MATLABu, který detailně popisuje tvorbu oken, menu, tlačítek a různých editovacích prvků, se kterými lze vytvořit bohaté a uživatelsky velmi propracované aplikace.

Praktická část se zabývá tvorbou vektorového grafického 2D editoru, popisem jeho prostředí a funkcí.

Klíčová slova: Matlab, grafika, programování, 2D editor

ABSTRACT

This work describes an area of graphics in MATLAB. It shows the issue of creating scripts and functions that are used for programming more complex and rich programs. Then it describes object-oriented programming, that are available in version R2008a and higher. The final section shows the System Handle Graphics, which describes the creation of windows, menus, buttons and editing boxes, which can be used for creating a rich applications.

The practical part deals with creating 2D vector graphics editor, a description of its environment and functions.

Keywords: Matlab, Graphics, Programming, 2D editor

Rád bych poděkoval vedoucímu práce Ing. Karlu Perůtkovi, Ph.D. za odborný dozor, cenné rady a připomínky, které mi k tvorbě této diplomové práce velice pomohli a byli její nedílnou součástí.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST	11
1 TVORBA SKRIPTŮ A FUNKCÍ.....	12
1.1 TYPY M-SOUBORŮ	12
1.1.1 Skripty	12
1.1.2 Funkce	13
2 OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ.....	15
2.1 POPIS TŘÍDY	15
2.1.1 Definice třídy a instance třídy	15
2.1.2 Atributy třídy	16
2.1.3 Události třídy	18
2.1.4 Metody třídy	19
2.1.4.1 Konstruktor	20
2.1.4.2 Destruktor	21
3 SYSTÉM HANDLE GRAPHICS	22
3.1 OBJEKT FIGURE	24
3.2 OBJEKT AXES	25
3.3 UI OBJEKTY	26
3.3.1 Uimenu	26
3.3.2 Uicontextmenu	27
3.3.3 Uipanel	27
3.3.4 Uicontrol	27
3.3.4.1 Styly uicontrol.....	28
3.3.5 Uitoolbar	31
3.3.6 Uitoggletool.....	31
3.3.7 Uibuttongroup	32
3.3.8 Uipushtool	32
II PRAKTICKÁ ČÁST	33
4 EDITOR	34
4.1 SPUŠTĚNÍ EDITORU	34
4.2 POPIS EDITORU	35
4.2.1 Hlavní lišta	35
4.2.2 Lišta nástrojů	37
4.2.3 Lišta vykreslování entit	37
4.2.4 Panel barev	38
4.2.5 Panel úprav	38
4.2.6 Panel objektů	39
4.3 PRÁCE S EDITOREM	39
4.3.1 Vytvoření úsečky.....	39
4.3.2 Vytvoření obdélníku, čtverce	40

4.3.3	Vytvoření elipsy, kružnice.....	40
4.3.4	Vytvoření bodu.....	40
4.3.5	Vytvoření textu.....	40
4.3.6	Vytvoření šipky a dvojité šipky.....	41
4.3.7	Úprava objektů	41
4.3.7.1	Tloušťka čáry	41
4.3.7.2	Styl čáry	42
4.3.7.3	Průhlednost	42
4.3.7.4	Poměr strany	42
4.3.7.5	Výplň.....	43
4.3.7.6	Obrys.....	43
4.3.8	Přesun objektů	43
4.3.9	Ukládání objektů	44
4.3.10	Otevírání objektů.....	45
5	TECHNIKY POUŽITÉ PŘI PROGRAMOVÁNÍ EDITORU	46
6	POROVNÁNÍ EDITORU S JINÝMI EDITORY	49
	ZÁVĚR	50
	ZÁVĚR V ANGLIČTINĚ	51
	SEZNAM POUŽITÉ LITERATURY.....	52
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	53
	SEZNAM OBRÁZKŮ	54
	SEZNAM TABULEK.....	55
	SEZNAM PŘÍLOH.....	56

ÚVOD

„MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů.“ [7]

Název MATLAB pochází z anglického MATrix LABoratory, což můžeme přeložit jako „laboratoř matic“. Z názvu je patrné, že klíčovou datovou strukturou pro výpočty jsou matice reálných a komplexních čísel. Je to programovací jazyk čtvrté generace, který byl vyvinut a je dále rozšiřován firmou MathWorks. Tento výpočetní systém využijeme v řadě odvětví, jako je zpracování signálů, komunikace, automatické řízení a regulace, vědeckotechnické výpočty, aplikovaná informatika, měření a testování, finančnictví, modelování soustav atd. Jazyk MATLAB je multiplatformní a je implementován pro všechny významné platformy: Windows, Linux, Mac, Solaris.

MATLAB disponuje mimořádně rychlým výpočetním jádrem s optimálními algoritmy a s podporou více jader, vyniká také svou jednoduchostí oproti starším jazykům, nabízí uživateli mnoho výpočetních a grafických nástrojů, dále otevřený a rozšiřitelný systém. Jazyk podporuje objektové programování, paralelní výpočty, práci s 2D a 3D grafikou, je integrován s jazykem Java, animace, zvukový vstup a výstup, velké množství aplikačních knihoven.

MATLAB je možné rozšířit také řadou toolboxů, což jsou knihovny funkcí, díky kterým můžeme využívat mnoho dalších specializovaných funkcí vytvořených v tomto jazyce a rozvíjet tak vědní obory. Nejznámějším toolboxem je Simulink, který slouží pro simulaci a modelování dynamických systémů ve formě blokových schémat či rovnic. Simulink nám jednoduše dovoluje pouhým přesouváním bloků tvořit schémata, jimiž můžeme sledovat modely například lineární, nelineární, diskrétní nebo spojité systémy. Obrovskou výhodou je opět jeho multiplatformnost a snadná přenositelnost modelů, díky čemuž můžeme vytvářet rozsáhlé modely. Simulink i MATLAB dovolují připojovat funkce psané uživatelem v jazyce C. [7]

V této práci a při tvorbě programu v teoretické části jsou používány základní operace s maticemi, podmínky, cykly a různé další základní příkazy, které již nejsou v práci blíže specifikovány a vysvětleny. Předpokládá se tak předešlá základní znalost nad vývojovým prostředím MATLABu, jako je tvorba vektorů a matic, jejich indexace, tvorba podmínek a

cyklů. Ke přiblížení těchto základů bych vás odkázal na seznam použité literatury uvedený v závěru této práce, kde již budou některé vhodné knihy, bakalářské práce či odkazy na internetové stránky k dispozici.

I. TEORETICKÁ ČÁST

1 TVORBA SKRIPTŮ A FUNKCÍ

MATLAB poskytuje kromě konsolové řádky pro zadávání příkazů také editor pro tvorbu skriptů, tzv. m-file souborů. Ten slouží především pro psaní větší porce kódu, což je mnohem efektivnější a přehlednější než jeho zadávání do konzole po jednom příkazu. Tento kód si navíc můžete odladit, opravit případné chyby, odkrokovat jednotlivé úseky kódu atd. Kód vytvořený v tomto editoru je barevně rozlišen, máme tak přehled o použitých funkcích, proměnných atd. Vytváříme jej především pro možnost uložení souboru na disk, sepsání celého kódu jako posloupnosti příkazu a jeho spuštění, vytvoření funkcí a jejich volání mezi sebou, čímž nám poskytuje mnoho prostoru pro rozsáhlejší programování.[2]

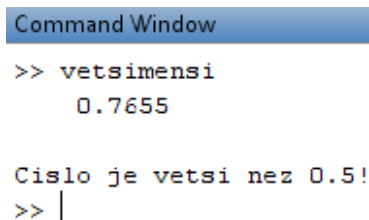
Nový m-soubor vytvoříme kliknutím na položku *File-New-M-file* (případně stlačit Ctrl+N), stisknutím příslušné ikony nebo zadáním příkazu *edit* do příkazového řádku.

1.1 Typy m-souborů

1.1.1 Skripty

Skriptem označujeme jednoduchou posloupnost příkazů, která se jeho spuštěním vykoná. Takto vytvořené m-soubory nepřijímají žádné argumenty a žádné také nevrací. Po uložení tohoto souboru jej můžete spustit buď v editoru stiskem položky Run, která se nachází v menu Debug, případně stiskem tlačítkem F5, anebo stlačením příslušné ikony. Poslední možností je zadání názvu m-souboru do příkazového řádku MATLABu. Při ukládání se musí dbát na to, aby se název skriptu neshodoval s názvem jakékoliv jiné funkce vytvořené MATLABem. Skripty samozřejmě mohou v sobě volat jiné skripty

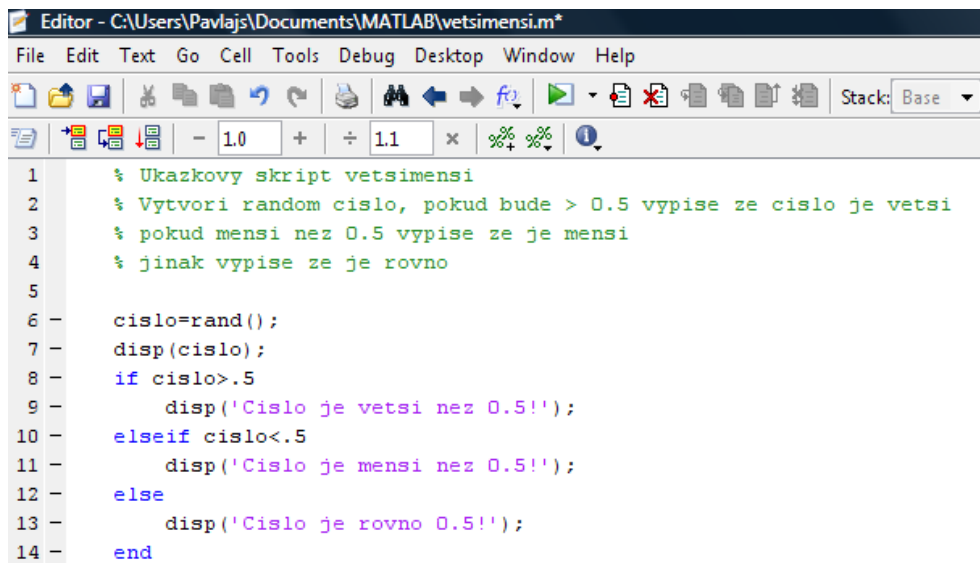
První řádky skriptů a funkcí je vždy dobré okomentovat tak, abychom věděli, co daný m-soubor dělá. Pomocí příkazu *help názevSkriptu* si můžeme zobrazit prvních 5 řádků komentáře a zjistit tak informace o skriptu z příkazového řádku.



```
Command Window
>> vetsimensi
    0.7655

Cislo je vetsi nez 0.5!
>> |
```

Obrázek 1- Volání skriptu



```

1      % Ukazkový skript vetsimensi
2      % Vytvori random cislo, pokud bude > 0.5 vypise ze cislo je vetsi
3      % pokud mensi nez 0.5 vypise ze je mensi
4      % jinak vypise ze je rovno
5
6      cislo=rand();
7      disp(cislo);
8      if cislo>.5
9          disp('Cislo je vetsi nez 0.5!');
10     elseif cislo<.5
11         disp('Cislo je mensi nez 0.5!');
12     else
13         disp('Cislo je rovno 0.5!');
14     end

```

Obrázek 2- Ukázkový skript

1.1.2 Funkce

Funkce jsou naprogramované úseky kódu, které na rozdíl od skriptů mohou přijímat a vracet libovolný počet argumentů, čímž můžeme vytvořit rozsáhlé a bohaté aplikace. Funkce se tvoří do m-souborů, jejichž jména by se měla shodovat. Začíná uvedením klíčového slova *function*, za nímž následuje seznam návratových proměnných (nepovinné), název funkce a definice vstupních parametrů uvedených v závorce (nepovinné).

```
function [vystArg1, vystArg2,...] = nazevFunkce(vstArg1, vstArg2, ...)
```

tělo funkce

end

Funkce se poté volá zadáním jeho názvu, kterému je nutno specifikovat výstupní a vstupní argumenty podle počtu uvedeném při jeho vytváření. V případě nedodržení některého z počtu argumentů dojde k vypsání chybové hlášky.

```
[arg1, arg2, ...]=nazevFunkce(arg1, arg2, ...)
```

MATLAB také poskytuje možnost vytvářet funkce s proměnným typem vstupních i výstupních parametrů. V případě, kdy není předem znám počet vstupních argumentů, je do dobré je nahradit parametrem *varargin*, který pojme všechny argumenty a můžeme tak s ním dále pracovat. Vstupní argumenty v ní budou uloženy ve formě matice, k nimž se přistupuje pomocí indexů, a mohou obsahovat různé datové typy od čísel až po řetězce či

znaky. Proměnný počet výstupních parametrů se poté deklaruje slovíčkem *varargout*. Pro zjištění počtu vstupních a výstupních argumentů slouží funkce *nargin*, resp. *nargout*. Definice funkce s proměnným počtem vstupních a s proměnným počtem výstupních argumentů by vypadala následovně:

```
function varargout = nazevFunkce(varargin)
```

2 OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ

Objektově orientované programování (OOP) je přeloženo z anglického výrazu Object-Oriented Programming. Dřívější metody programování, tzv. strukturované programování, spočívá v rozkladu problémů na menší opakovatelné jednotky, které se nazývají procedury. Programy vytvořené procedurálním programováním se vyznačují obtížným spravováním programu, udržováním, a také se velmi špatně rozšiřují. Na rozdíl od tohoto způsobu procedurálního programování nahlíží objektové programování na věci jako na objekty, např. o autu už nerozmýšlíme jako o nějaké datové položce, se kterou manipulujeme, ale jako o skutečné věci tohoto světa, o objektu, který má určitý počet kol, barvu, značku, může se pohybovat vpřed a vzad atd. OOP je vhodné použít pro větší a složitější programy, které bude možno dále rozšiřovat.

Objektové programování se v programovém prostředí MATLAB objevuje od verze R2008a. Nabízí nám tak možnost definovat třídy, atributy, události a metody.

2.1 Popis třídy

Objekty okolo nás mají dvě charakteristiky: všechny mají nějaký stav a všechny mají nějaké chování. Objekty musí mít proměnné, které současný stav objektu indikují, a také musí mít metody, které nastavují chování objektu.

Pod **třídou** tedy rozumíme soubor proměnných různých typů a sadou funkcí pracujících s těmito proměnnými a ovlivňujícími je. Popisujeme jí obecné vlastnosti, které objekt musí mít, aby do této třídy patřil.

Z hlediska syntaxe je třída obyčejnou deklarací datové struktury, která je rozšířená o metody. Představuje nám objektový typ.

2.1.1 Definice třídy a instance třídy

Objekt je **instance třídy** objektů. Při vytvoření instance třídy se vytvoří objekt a systém alokuje paměť pro proměnné, které jsou ve třídě obsaženy.

Definice třídy v MATLABu se provádí klíčovým slovem *classdef* vyznačujícím blok, za nímž následuje název třídy. Každý blok se ukončuje klíčovým slovem *end*. Třída může být

potomkem supertřídy (superclass) nebo rodičem podtřídy (subclass) a využívat tak dědičnosti.

Existují zde také abstraktní třídy, u kterých nemůžeme vytvořit instanci a fungují jako superclass třídy pro dědění. Patří mezi ně třída *handle*, která je rodičem pro všechny třídy pracující s ukazateli (handles) na jednotlivé instance třídy. Definuje nám metody pro porovnávání, vyhledávání instancí a atributů, pro práci s událostmi a destruktory. Další abstraktní třídou je *hgsetget*, která je potomkem superclass *handle* a definuje nám metody *set* a *get*, které jsou často používané při práci s grafickými objekty. Poslední abstraktní třída (*addprop*) je také potomkem třídy *handle* a umožňuje nám přidávat atributy k objektům.

Třída se v MATLABu vytvoří kliknutím na položky *File- New- Class*. Následující kód definuje třídu, která je potomkem supertřídy *handle*.

```
1 classdef novaTrida < handle
2 end
```

2.1.2 Atributy třídy

Atributy (properties) obsahují členské proměnné třídy a konstanty, jež jednotlivé instance budou nabývat a s nimiž budou pracovat. Proměnné se definují blokem obsaženým ve třídě začínající syntaxí *properties* a končící jako každý blok výrazem *end*. Atributům můžeme nadefinovat spoustu vlastností pomocí následující tabulky.

Tabulka 1- Vlastnosti atributů

Vlastnost	Hodnoty	Popis
AbortSet	false, true	Pokud je true, MATLAB nenastaví hodnotu proměnné, pokud je nová hodnota stejná jako současná.
Abstract	false, true	Pokud je true, vlastnost není implementována, ale konkrétní podtřída musí předefinovat tuto vlastnost bez nastavení Abstract na true.
Access	public, private, protected	Definice přístupu k jednotlivým proměnným v bloku
Constant	false, true	Definice konstantní proměnné

Dependent	false, true	Pokud je false, proměnná je uložená v objektu. Pokud je true, není uložena v objektu, funkce set a get k ní nemají přístup.
GetAccess	public, private, protected	Funkce pro získání přístupu. MATLAB nezobrazuje v příkazovém okně proměnné, které jsou private nebo protected, nebo mají Hidden=true.
GetObservable	false, true	Pokud je true, můžeme vytvořit listenery pro přístup k proměnným, kteří jsou voláni kdykoliv je proměnná dotazována.
Hidden	false, true	„Skrytost“ objektu. Pokud je true, v příkazovém okně se nezobrazí proměnná a její hodnota, stejně jako proměnné private a protected.
SetAccess	public, private, protected, immutable	Určuje podmínky pro možnost úpravy hodnoty proměnných.
SetObservable	false, true	Pokud je true, můžeme vytvořit listenery pro přístup k proměnným, kteří jsou voláni kdykoliv je proměnná upravena.
Transient	false, true	Pokud je true, proměnné se neuloží když je objekt ukládán do souboru.

Pozn.: Defaultní hodnoty vlastností jsou vždy uvedeny jako první, tudíž jimi jsou hodnoty *false* nebo *public*.

Přístupy k proměnným:

- Veřejné (public)- k proměnným může přistupovat libovolný objekt tříd, manipulovat s nimi.
- Chráněné (protected)- k proměnným lze přistupovat a pracovat pouze pomocí metod dané třídy nebo potomků.
- Soukromé (private)- k proměnným lze přistupovat pouze pomocí metod dané třídy

Pro vytvoření atributů s různými vlastnostmi musíme vytvořit samostatný blok pro každé takové proměnné, skládající se z klíčového slova `properties`, definicí vlastností proměnných nacházejících se v bloku, definicemi samotných proměnných a výrazem `end` označujícím konec bloku.

```

3      properties (SetAccess = private, Hidden = true)
4          skrytaPromenna;
5      end
6      properties (Constant)
7          promennaPi=3.14;
8      end

```

2.1.3 Události třídy

Události třídy (events) jsou oznámení, které nám říká, že nastala nějaká událost. Tímto oznámením může být například změna hodnoty proměnné, na kterou čekají další instance třídy, které budou na tuto událost reagovat. Sledování událostí probíhá nejen v rámci třídy, ale na událost může čekat instance z jiné třídy. Události a posluchače (listeners) mohou být definovány pouze u tříd, které jsou potomkem superclass `handle`. V tabulce č.2 jsou zobrazeny atributy pro nastavení událostí. Tyto vlastnosti se zapisují do kulatých závorek za klíčové slovo `events` vyznačují začínající blok pro události. Blok je ukončen koncovým slovem `end`. [8]

Tabulka 2- Vlastnosti událostí

Vlastnost	Hodnoty	Popis
Hidden	false, true	Pokud je true, událost se neobjeví v seznamu událostí
ListenAccess	public, protected, private	Určuje práva pro vytvoření posluchačů pro událost.
NotifyAccess	public, protected, private	Určuje práva kde kód může vyvolat událost.

Hodnoty `public`, `private` a `protected` mají identický význam jako hodnoty popsané pod tabulkou č.1.

Pokud chceme vytvořit události, které se liší vlastnostmi, jež budou nabývat, je nutno pro každé rozdílné události vytvořit nový samostatný blok začínající klíčovým slovem *events* a končící *end*.

Události se vyvolávají v metodách třídy pomocí metody notifiy, které se pouze předá název události. Sledování události se provádí pomocí aktivace metody *addlistener*, která vytvoří novou instanci třídy *event.listener*.

2.1.4 Metody třídy

Metody jsou funkce, které implementují operace provedené na objektech třídy. Metody společně s dalšími členy třídy podporují koncept zapouzdření. Mají přístup k *private* členům ve třídě, mohou měnit atributy, vyvolávat události, spouštět jiné funkce a metody. Metody se vytvářejí definicí bloku pomocí klíčového slova *methods* (ukončen *end*), za nímž můžeme specifikovat vlastnosti daného bloku. Následují definice jednotlivých funkcí pracujících s proměnnými atd. V tabulce č. 3 je zobrazen výčet vlastností metod a jejich hodnot.

Tabulka 3- Vlastnosti metod

Vlastnost	Hodnoty	Popis
Abstract	false, true	Pokud je true, vlastnost není implementována, ale konkrétní podtřída musí předefinovat tuto vlastnost bez nastavení Abstract na true.
Access	public, protected, private	Definuje práva na volání těchto metod.
Hidden	false, true	Ukrytí metody, pokud je nastavena na true, název metody není uveden v seznamu metod.
Sealed	false, true	Je-li nastavena na true, metoda nemůže být předefinována v podtřídě.
Static	false, true	Nastavení hodnoty static na true definujeme metodu, která není závislá na objektu třídy.

Statické metody jsou spojeny s třídou, ale nikoliv s instancemi této třídy. Tyto metody neprovádějí operace na jednotlivých objektech třídy, a proto nevyžadují instanci třídy jako vstupní argument. Jsou užitečné, pokud nechceme nejdříve vytvořit instanci třídy před

provedením nějaké části kódu. Statické metody se volají uvedením názvu třídy následujícím operátorem tečka (.) a uvedením názvu funkce se vstupními argumenty.

Význam hodnot u vlastnosti access:

- `public` (veřejná)- každý kód, který má přístup k objektu třídy může přistupovat k této metodě.
- `protected` (chráněná)- omezení metody přístupu k definování třídy a podtřídy odvozené z definice třídy. Podtřídy dědí tuto metodu.
- `private` (soukromá)- omezení metody přístupu k definování třídy kromě podtřídy. Podtřídy nedědí tuto metodu. Následující úsek kódu popisuje definici metody pro výpočet povrchu válce přijatého čísla.

```
9      methods
10      function x = obsahKruhu(obj,polomer)
11          x = obj.promennaPi*polomer^2;
12      end
13  end
```

2.1.4.1 Konstruktor

Konstruktor je speciální funkce, která se spouští při vytvoření instance třídy. Tato funkce nese název stejný jako je název třídy a přijímá vstupní argumenty k nastavení proměnných v sekci *properties*. Konstruktor má jedinou návratovou hodnotu, a to je ukazatel na novou instanci. Vždy musí vrátit platnou instanci třídy a nesmí vrátit prázdný objekt. Pokud je třída vytvořena jako potomek nějaké třídy, MATLAB k inicializaci objektu zavolá konstruktor každé rodičovské třídy. Třída nepotřebuje definovat konstruktor, není-li potomkem supertřídy, jejíž konstruktor vyžaduje argumenty. V tomto případě se musí explicitně zavolat konstruktor supertřídy s požadovanými argumenty. Konstruktor rodičovské třídy nevrací ukazatel na objekt třídy potomka. Volání po konstruktorech supertřídy nesmí být umístěno do smyček, podmínek, přepínačů, try-catch funkcí a vnořených funkcí. [8]

```
14 classdef auto < handle
15     methods
16         function hObject=auto(hKlic)
17         end
18         function delete(hObject)
```

```
19     end
20 end
21 end
```

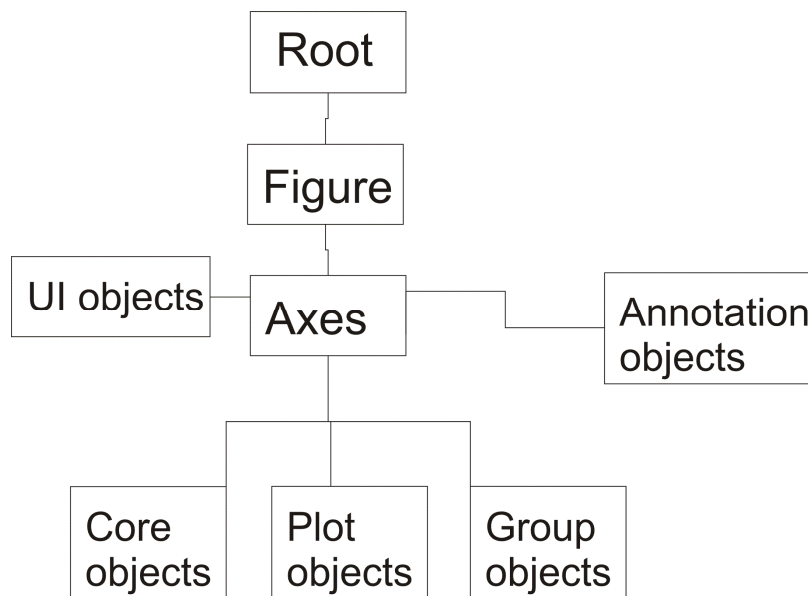
2.1.4.2 Destruktor

Destruktor je opakem konstrukturu. Je to speciální funkce, která se spustí při likvidaci instance třídy (objektu). Tato funkce se deklaruje v bloku *methods* jako funkce *delete(object)*. Zavoláním destrukturu objektu třídy potomka se zavolají destruktory rodičovských tříd. Pokud definujete destruktory s jiným než jedním vstupním argumentem nebo nějakými výstupními argumenty, MATLAB ji vezme jako běžnou metodu a objekt nezničí. Jestliže se funkce destrukturu umístí jako metoda *private*, nemůže žádná explicitní metoda zavolat tento destruktory, ale instanci může zničit pouze metoda volaná ze stejné třídy. Aby mohl potomek třídy zavolat destruktory rodičovské třídy, musí být tento destruktory umístěn v metodě s nastavením přístupu jako *public*, případně *protected*. Je-li ukazatel na objekt (handle) uložen v proměnné a zavoláme-li na tuto proměnnou destruktory, objekt bude odstraněn, ale proměnná zůstane zachovaná. V takovém případě je lepší dokončit odstranění instance pomocí funkce *clear(proměnná)*, která nám smaže proměnnou s ukazatelem na vymazanou instanci.

3 SYSTÉM HANDLE GRAPHICS

Systém MATLAB obsahuje nástroj pro tvorbu grafických objektů. Tento grafický nástroj se nazývá Handle Graphics a poskytuje uživateli mnoho funkcí pro tvorbu vlastních aplikací, nikoliv už jen v textové podobě, ale i podobě grafické. Nabízí se nám tak možnost velmi efektivně pracovat s grafickými objekty, jako jsou okna, osy, grafy, tlačítka, posuvníky, menu a submenu, nástrojové lišty, textová pole a další.

Grafickým objektem budeme rozumět každý nově vytvořený prvek, počínaje samotným oknem, do kterého budeme přidávat další elementy, až po všechny tlačítka, menu, posuvníky, čáry, grafy atd. Každý z těchto nových objektů bude mít svůj jedinečný identifikátor, tzv. *handle* (odtud název Handle Graphics). Tento identifikátor ve formě čísla nebude mít žádný jiný nově vytvořený objekt. Díky těmto ukazatelům tak můžeme jednoduše přistupovat k těmto objektům, jeho vlastnostem, a pracovat tak s nimi. Tyto identifikátory nám tak poskytují přehled o objektech a jsou seříděny do jednoduché hierarchie, která tak pomáhá k vzájemné organizaci objektů.

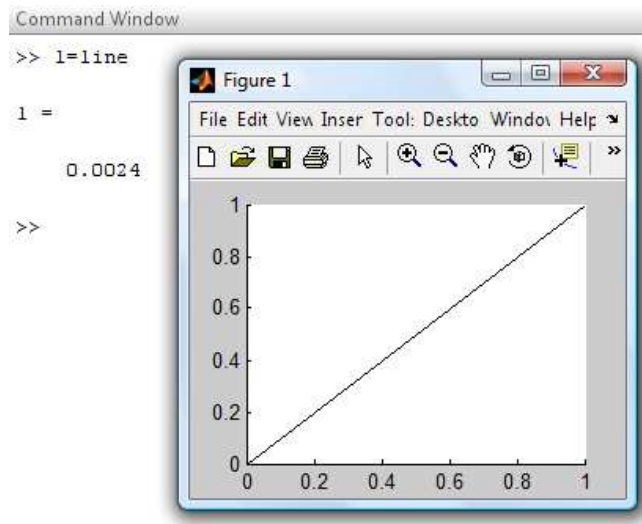


Obrázek 3- Hierarchie systému Handle Graphics

Nejvýše v této vzájemné hierarchii a provázanosti objektů je kořen *Root*. Tento kořen odpovídá obrazovce počítače a jeho *handle* je roven 0. Od tohoto kořene se vytváří všechny objekty, kterým se postupně přidává další *handle*. Vytvořené objekty jsou vždy podřízeny vyšším objektům v tomto zobrazeném stromu. Pokud například chceme vytvořit čáru, která je v předchozí hierarchii poskytnuta pod objekty *Core Objects*, čára pro své vykreslení

potřebuje vykreslení osy (objekt *Axes*), které zase pro své vykreslení potřebují okno (objekt *Figure*). [6]

Pro ukázkou tedy zkusíme do příkazového řádku MATLABu zadat příkaz pro vykreslení čáry (*l=line*).



Obrázek 4- Vykreslení čáry pomocí příkazu „line“

Z obrázku můžeme vyčíst, že uživatel nemusel pro vykreslení čáry nejdříve vytvořit okno *Figure*, osy *Axes*, a teprve potom vytvořit čáru, jak je to již v hierarchii definováno, ale mohl rovnou vytvořit čáru (bez definice pozice je defaultní vykreslení čáry dvou bodů o souřadnicích $X[0,0]$ a $Y[1,1]$) a MATLAB tyto objekty nutné pro vytvoření čáry vytvořil sám. Na příkazovém řádku se taktéž zobrazil *handle* objektu uložen do proměnné *l*, tedy naší čáry (číslo 0.0024). Čára se tak stala potomkem souřadnicových os, které jsou zase potomkem okna, to je zase potomkem kořene *Root*, a naopak *Root* je rodičem pro okno *Figure* atd.

Příkazy *set*, *get*, *findobj*:

Pro zjištění vlastností nebo nastavení různých vlastností grafických objektů existují dva příkazy. První z nich, *get*, což bychom mohli přeložit jako „získej“ nebo „poskytni“, nám slouží pro získání vlastností o objektech. Druhým z nich, příkaz *set* („nastav“), slouží pro nastavení objektů, které se identifikují právě pomocí *handle* objektu. Pro získání seznamu vlastností objektu stačí zapsat do příkazového řádku (nebo M-souboru) příkaz *get(handleObjektu)*. Pro zjištění nastavení konkrétního vlastnosti objektu stačí připsat název vlastnosti do jednoduchých uvozovek (*get(handleObjektu, 'Parent')*). Tato vlastnost

Parent zobrazuje rodiče objektu, kterým by v předchozím příkladu vykreslení čáry byl *handle* na objekt *Axes*. Mezi vlastnostmi se vyskytuje také *Children*, která obsahuje naopak identifikátory na potomky objektu. Příkaz *set* se může využít i jinak, než jen pro nastavení vlastnosti objektu, ale také pro zjištění možností, které má daná vlastnost k dispozici. Pokud například potřebujeme nastavit jednotky zobrazení objektu *Figure*, ale nevíme, jaké prvky máme k dispozici, postačí se zeptat MATLABu příkazem *set(handleObjektu,'Units')*. Ten nám zobrazí všechny možnosti, včetně aktuálně nastavené, která je uvedena ve složených závorkách, viz.Obrázek 4.

```
>> f=figure('Tag','Figure')
f =
     1

>> set(findobj('Tag','Figure'),'Units')
[ inches | centimeters | normalized | points | {pixels} | characters ]
>>
```

Obrázek 5- Využití příkazu „set“ pro zjištění možností určité vlastnosti

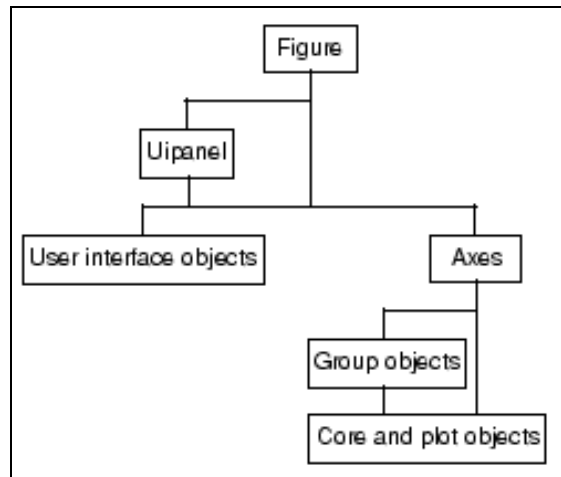
V tomto příkladu jsem pro zapsání *handle* objektu nevyužil proměnné, ve které jsem měl identifikátor okna uložen, ale našel jsem si jej pomocí funkce *findobj*. Této funkci jsem specifikoval vlastnost a hodnotu vlastnosti, jež má u objektu hledat a výsledkem bylo přiřazení ukazatele tohoto objektu. U vytvoření okna byla definována vlastnost *Tag*, což se dá využít jako jmenovka pro daný objekt. Výsledným výpisem příkazu *set* byl výčet možností nastavení jednotek pro okno s defaultním nastavením na *pixels*. [2]

3.1 Objekt Figure

Objekt *Figure* je samotné okno aplikace, do kterého MATLAB zobrazuje grafické prvky. Má dvě základní role:

- zobrazovat grafy
- zobrazovat grafické uživatelské rozhraní, mezi které patří menu, lišty, kontextové menu, osy atd.

Diagram na obrázku č.6 popisuje typy objektů, které může obsahovat okno *Figure*.



Obrázek 6- Typy objektů pro grafický objekt
Figure [6]

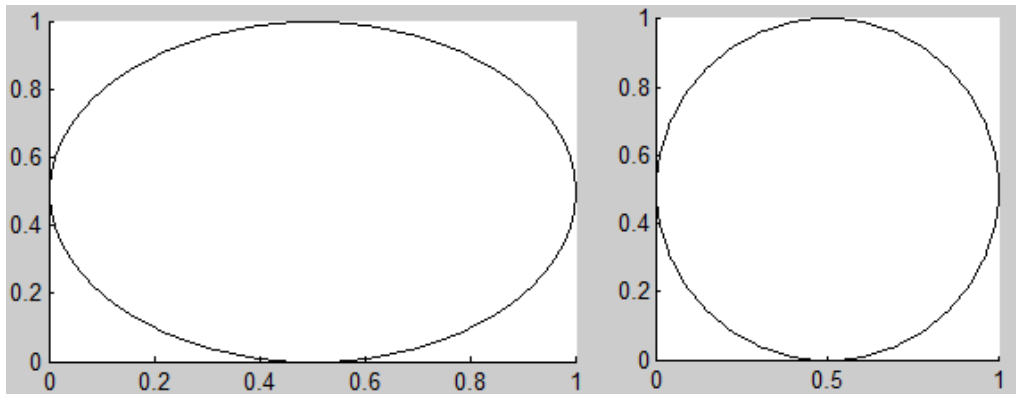
Pro zjištění ukazatele *handle* na objekt *Figure* lze využít příkazu *gcf* (z anglického *Get Current Figure*, tedy získej aktuální okno). V případě kdy existuje více otevřených oken tato funkce vrátí *handle* aktuálního okna, který je také uložen ve vlastnosti *CurrentFigure* kořene *Root*. Pomocí vlastností lze nastavovat a specifikovat mnoho aspektů, událostí či zobrazení okna, mezi které patří například zobrazení názvu, lišty, barevné provedení plochy, jaká událost má nastat při kliknutí do okna, při pohybu po okně atd. Kompletní seznam vlastností lze zobrazit pomocí příkazu *get(gcf)*.

3.2 Objekt Axes

Grafický objekt *Axes* slouží pro vytvoření souřadnicového systému využívaný především pro vykreslování dat, grafů a vykreslovacích grafických objektů. Do okna *figure* lze přidat libovolný počet objektů *os*, pouze jeden je však aktuální a jeho ukazatel je uchovaný ve vlastnosti *CurrentAxes* objektu *Figure*. Aktuální osy lze též identifikovat pomocí příkazu *gca*.

Osy se vytvářejí pomocí příkazu *axes*, specifikovat její vlastnosti lze při vytváření uvedením do závorky (*axes('vlastnost', 'hodnotaVlastnosti')*). Při vytváření os MATLAB vezme poslední dvě položky v definici vlastnosti *Position* (ta se skládá z vektoru dvou souřadnic udávající levý dolní roh, šířky a výšky [levá, dolní, šířka, výška]) a roztáhne osy do maximální využitelné pozice. To má za následek zkruslení především kulatých křivek, jako je třeba kružnice nebo koule, které se poté jeví jako zploštělé. Řešením je změna

některé z položek *DataAspectRatioMode*, *PlotBoxAspectRatioMode* nebo *CameraViewAngleMode* z defaultně nastaveného *auto* na uživatelem definované *manual*.



Obrázek 7- Zobrazení zkreslení objektu kružnice (vlevo) a jeho napravení úpravou jedné z vlastností

3.3 UI Objekty

V MATLABu se dá vytvořit mnoho grafických uživatelských komponent, které jsou nepochybnou součástí každého obrázku nebo framu. Mezi ně patří hlavně menu, přidání panelu do obrázku, vytvoření tlačítek, výběrových tlačítek, tabulek apod.

3.3.1 Uimenu

Funkce *uimenu* vytváří menu a položky menu pro okno obrázku, které se zobrazují na liště obrázku. Příkaz `handle = uimenu('Label', 'nazevMenu')` vytvoří vlastní menu do lišty aktuálního obrázku. Při vytváření lze samozřejmě specifikovat mnoho dalších vlastností, jako je vložení zkratky pro aktivaci této položky, vložení akcelerátoru, položky Callback, která vytváří odezvu na stisknutí tohoto menu. Pro přidání položky do menu se jako první údaj v závorce uvádí *handle* objektu. Nemusí jím být pouze handle menu, ale pokud by jím byl ukazatel na obrázek, MATLAB vytvoří menu do tohoto obrázku. Menu se zobrazují pouze v oknech, která mají vlastnost *WindowStyle* nastavenou na *normal*. Pokud by se tato vlastnost změnila, menu by sice stále existovalo a bylo uvedeno jako *children* obrázku, ale nebylo by již zobrazeno, dokud by se vlastnost nezměnila nazpět.

3.3.2 Uicontextmenu

Uicontextmenu vytváří kontextové menu. Jedná se o menu, které se zobrazí, když uživatel stlačí pravé tlačítko myši na nějakém grafickém objektu. Definuje se stejně jako předešlé menu, $handle = uicontextmenu('Label', 'nazevMenu')$. Jednotlivé položky se vytvářejí pomocí *uimenu*, do kterého se uvede *handle* na námi vytvořené kontextové menu a definicí jejich *Callback* událostí. Menu musí být přiřazeno k nějakému grafickému objektu, které se provede nastavením vlastnosti *UIContextMenu* na ukazatel kontextového menu.

3.3.3 Uipanel

Příkaz *uipanel* vytvoří panel, který seskupuje komponenty typu *uicontrol* (viz dále), se kterými uživatel pracuje přímo, dále může obsahovat osy, další panely nebo skupiny objektů *uicontrol*. Příkaz $uipanel(handle, 'nazevVlastnosti', 'nastaveniVlastnosti')$ vytvoří panel do objektu specifikovaného ukazatelem *handle*. (rodičem může být obrázek, jiný panel nebo skupina *uibuttongroup*) Pokud by nebyl uveden tento ukazatel uveden, panel by byl vytvořen do aktuálního obrázku, v případě kdy by neexistoval žádný obrázek, by jej MATLAB vytvořil. Pokud nastavíte hodnotu vlastnosti *Visible* (viditelnost) na *off*, neviditelnými se stanou všechny objekty obsažené v tomto panelu dokud se viditelnost nenastaví zpět na *on*.

3.3.4 Uicontrol

Tyto objekty mohou vytvářet tlačítka, editovací textová pole, zaškrtačovací pole, seznamy, výběrové menu, posuvníky, vkládání obyčejného textu atd. Vytváří se příkazem $handleNovehoObjektu = uicontrol(handle, 'vlastnost', 'hodnotaVlastnosti')$.

Všechny nedefinované vlastnosti při vytváření objektu jsou nastaveny na defaultní hodnoty. Pokud není definovaný *handle*, rodičem se stává aktuální objekt *Figure*, případně jej MATLAB vytvoří, jak již bylo několikrát zmíněno. Rodičem může být obrázek, panel nebo skupina *uibuttongroup*. Typ vytvořeného objektu se definuje ve vlastnosti *Style*, jejíž nabídku si lze zobrazit využitím příkazu $set(uicontrol, 'Style')$. Při nedefinování této vlastnosti je jako defaultní položka nastavena tlačítko.

3.3.4.1 Styly *uicontrol*

Z ukázkových kódů v následujících odstavcích byla kvůli úspoře místa smazána vlastnost *'Units', 'normalized'* a definice *Position*. Vytvořené objekty jsou zobrazeny a popsány na obr. 7.

```
22 f=figure('Menubar','none');
```

3.3.4.1.1 Pushbutton

Defaultně nastavený styl, vytvoří tlačítko, které lze specifikovat pomocí mnoha vlastností. Událost na kliknutí se definuje položkou *Callback*, která tuto akci vykoná.

```
23 f1=uicontrol(f,'Style','PushButton','String','PushButton');
```

3.3.4.1.2 Togglebutton

Jedná se o přepínací tlačítko, které je vzhledově stejné jako klasické tlačítko, ale vizuálně vyjadřují svůj stav, a to buď jsou stisknuté (*on*) anebo nestisknuté (*off*). Klikáním na tlačítko se mění jeho stav.

```
24 f2=uicontrol(f,'Style','Togglebutton','String','Togglebutton');
```

3.3.4.1.3 Radiobutton

Jedná se o tlačítka, které se mohou seskupovat do bloku a poskytují uživateli zaškrtačací pole, ze kterých si uživatel může vybrat pouze jednu položku. Nejjednodušší způsob implementace vzájemně se vylučujícího chování těchto přepínačů je jejich umístění do skupiny *uibuttongroup*.

```
25 bg = uibuttongroup('Title','ButtonGroup Radiobutton');
26 f31=uicontrol('Style','Radiobutton','String','Polozka 1',...
27 'parent',bg,'HandleVisibility','off');
28 f32=uicontrol('Style','Radiobutton','String','Polozka 2',...
29 'parent',bg,'HandleVisibility','off');
30 f33=uicontrol('Style','Radiobutton','String','Polozka 3',...
31 'parent',bg,'HandleVisibility','off');
```

3.3.4.1.4 Checkbox

Stejně jako v předešlém odstavci se jedná o zaškrtačací políčka, rozdílem mezi těmito objekty je možnost zaškrtnutí počtu pole. U *checkbox* stylu ovládacího prvku lze vybrat více než jednu položku.

```
32 bg2=uibuttongroup('Title','ButtonGroup Checkbox');
33 f312=uicontrol('Style','Checkbox','String','Polozka 1',...
34 'parent',bg2,'HandleVisibility','off');
35 f322=uicontrol('Style','Checkbox','String','Polozka 2',...
36 'parent',bg2,'HandleVisibility','off');
37 f332=uicontrol('Style','Checkbox','String','Polozka 3',...
38 'parent',bg2,'HandleVisibility','off');
```

3.3.4.1.5 Edit

Tento prvek vytvoří editovací textové pole, ve kterém můžeme zadávat nebo upravovat vstupní textové hodnoty, které použijeme například jako vstup nebo nastavení některé z vlastností objektu. V základním nastavení je editovací pole jednořádkové, pro nastavení víceřádkového pole je nutno nastavit rozdíl mezi vlastností *Max* a *Min* na větší jak 1. Vyplněný text je uložen ve vlastnosti *String*.

```
39 f5=uicontrol(f,'Style','Edit','String','Edit','Max',2);
```

3.3.4.1.6 Text

Tento styl zobrazuje text a slouží především k popisu dalších prvků, které uživateli poskytují informace, nebo mohou zobrazovat informace z posuvníků či různých zaškrťávacích tlačítek. Uživatelé nemohou měnit interaktivně text ani aktivovat jakoukoliv zpětnou rutinu pro vykonání události.

```
40 uicontrol(f,'Style','Text','String','Slider',...
41 'BackgroundColor',get(f0,'BackgroundColor'));
```

3.3.4.1.7 Slider

Prvek slider vytvoří posuvník, který poskytuje uživateli číselný vstup daný nastaveným rozmezím. Pohybem jezdce se mění hodnoty daného rozmezí. Rozmezí se definuje nastavením vlastností *Max* a *Min*, aktuální hodnotu hodnotou *Value*, krok pomocí vlastnosti *SliderStep*.

```
42 f7=uicontrol(f,'Style','Slider','Min',1,'Max',100,'Value',32);
```

3.3.4.1.8 Frame

Tento styl vytváří rámec definovaný svou velikostí. Uživatel jej využije pro seskupení ovládacích prvků a dosáhne tak lepší přehlednosti v zobrazení objektů. Na rámy se

nevážou žádné *Callback* rutiny, mohou obsahovat pouze ovládací prvky *uicontrol*. Nejsou průhledné, tudíž je potřeba je definovat před objekty, které bude rám ohraničovat, jinak bychom objekty překryli rámem.

```
43 f0=uicontrol(f'Style','Frame','String','Frame');
```

3.3.4.1.9 Listbox

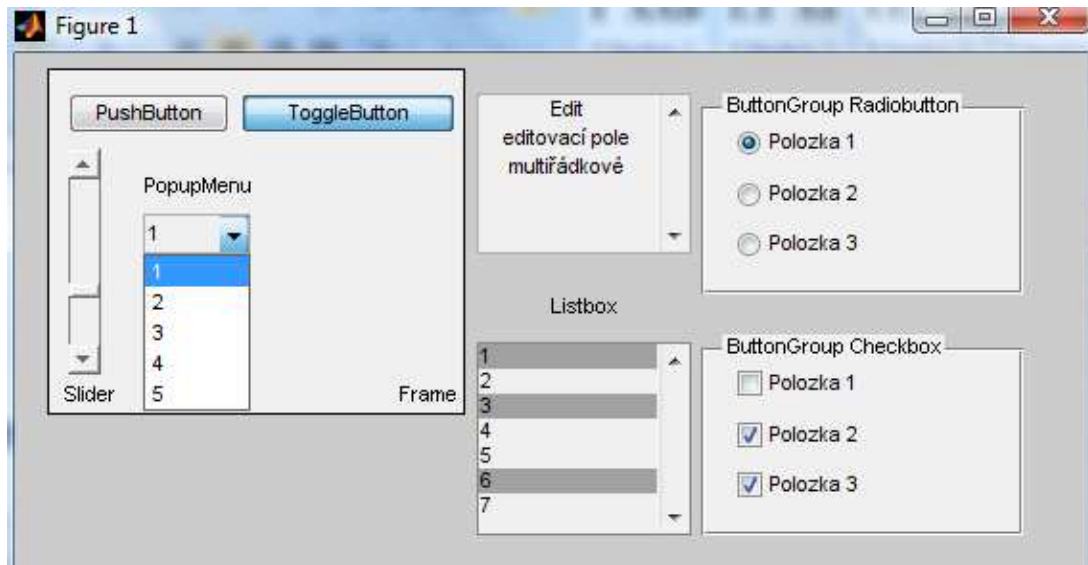
Listbox popisuje seznam položek, ze kterých si uživatel může vybrat jednu nebo více hodnot. Pomocí vlastností *Min* a *Max* definujeme možnost výběru více položek. Pokud je jejich rozdíl větší jak 1, *Listbox* umožňuje vícenásobný výběr. Pokud menší nebo roven 1, uživateli je umožněn výběr pouze jedné položky. Při výběru položky je uložen index řádku do vlastnosti *Value*, v případě vícenásobného výběru vektor indexů. Každý tento výběr řádku vyvolá obslužnou rutinu. Abychom odložili spouštění události po každém výběru, je lepší připojit k tomuto seznamu tlačítko, které provede *Callback* událost až kliknutí na něj. K vícenásobnému výběru lze využít tlačítka Ctrl a Shift, jak již bývá zvykem.

```
44 f9=uicontrol(f,'Style','Listbox','String','1|2|3|4|5|6|7','Max',2);
```

3.3.4.1.10 Popumenu

Popumenu, známé také jako *Combobox*, je menu, jehož položky se zobrazí po kliknutí na menu. Událost se spouští po výběru položky, výběr je uložen v proměnné *Value*. Svým významem je stejný jako *Radiobutton*, tedy výběr jedné z položek, nezabírá však tolik místa.

```
45 f10=uicontrol(f,'Style','Popumenu','String','1|2|3|4|5');
```



Obrázek 8- Ukázka všech stylů pro objekty „uicontrol“

3.3.5 Uitoolbar

Prvek vytvoří prázdný panel nástrojů aktivního obrázku, případně vytvoří obrázek s tímto prázdným panelem.

```
handleUitoolbar = uitoolbar(handleRodiče, 'Vlastnost', 'hodnotaVlastnosti')
```

Při vytvoření se definují vlastnosti párově, tj. uvedením vlastnosti a její hodnoty za sebou oddělené čárkou. Vlastnostem, které se při vytvoření nedefinují, nastaví MATLAB defaultní hodnoty. Panel se může zobrazit pouze v obrázku, jehož *WindowStyle* vlastnost je nastavena *normal* nebo *docked*. V případě nastavení na možnost *modal* panel stále existuje a je uveden jako potomek obrázku, není však zobrazen, dokud se nastavení této položky nezmění.

3.3.6 Uitoggletool

Tímto příkazem se vytváří spínací tlačítko na panelu nástrojů v horní části aktuálního okna.

```
handleUitoggletool = uitoggletool (handleRodiče, 'Vlastnost', 'hodnotaVlastnosti')
```

V případě, kdy neexistuje žádný *toolbar*, MATLAB jej vytvoří a stane se tak rodičem objektu. Tlačítko nemá žádnou ikonu, ta lze přidat pomocí vlastnosti *CData*. Ve vlastnostech se také definují události, které se mají vykonat při aktivaci tlačítka, události při vypnutí nebo kliknutí na tlačítko. Pro nastavení vlastnosti *WindowStyle* obrázku platí stejná pravidla jako v předešlém odstavci.

3.3.7 Uibuttongroup

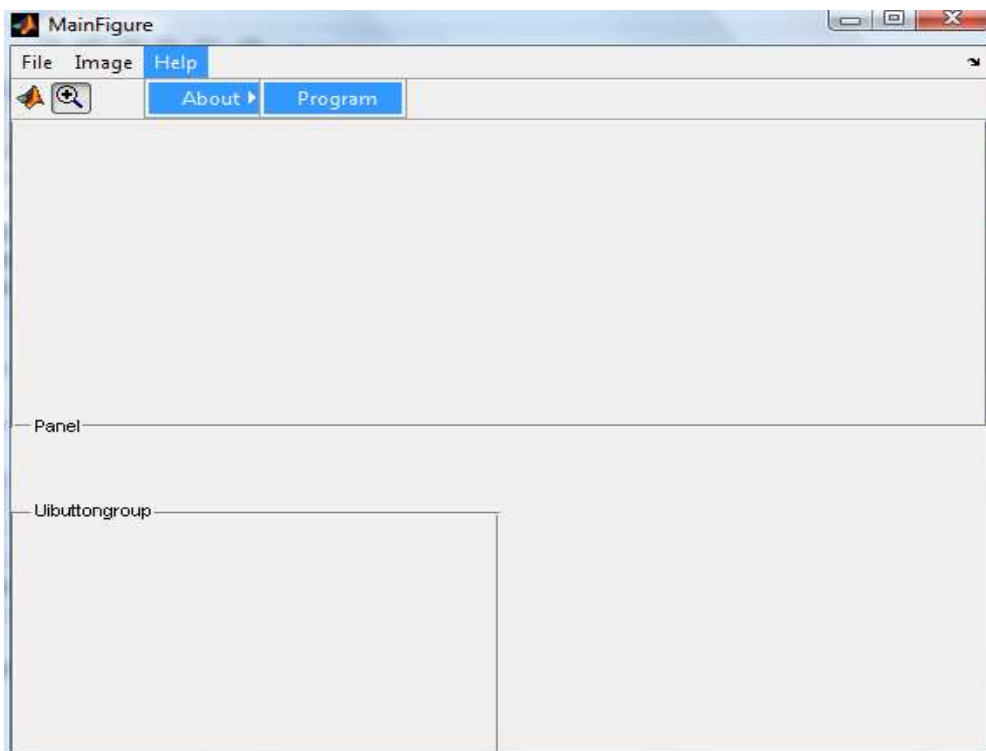
handleUibuttongroup = uibuttongroup(handleRodiče, 'Vlastnost', 'hodnotaVlastnosti')

Zadáním tohoto příkazu do konzole MATLABu se vytvoří kontejner, který řídí hodnoty objektů, jež obsahuje. Těmi mohou být všechny ovládací prvky uživatelského rozhraní, osy, panely nebo další kontejnery. Nedefinuje se zde *Callback* událost, změna hodnoty prvků v této skupině objektů je spravována položkou *SelectionChangeFcn*. Vypnutím položky pro viditelnost tohoto objektu se zneviditelní všechny prvky, které skupiny obsahuje, dokud se vlastnost nezmění. [6]

3.3.8 Uipushtool

handleUipushtool = uipushtool(handleRodiče, 'Vlastnost', 'hodnotaVlastnosti')

Uipushtool vytvoří tlačítko na panelu nástrojů. Nemůže být na něj vázáno kontextové menu, není zobrazen při *WindowState = modal*, událost se provádí položkou *ClickedCallback*, ikona pro tlačítko se nastavuje do vlastnosti *CData*.



Obrázek 9- Zobrazení vytvořených objektů uživatelského rozhraní

II. PRAKTICKÁ ČÁST

4 EDITOR

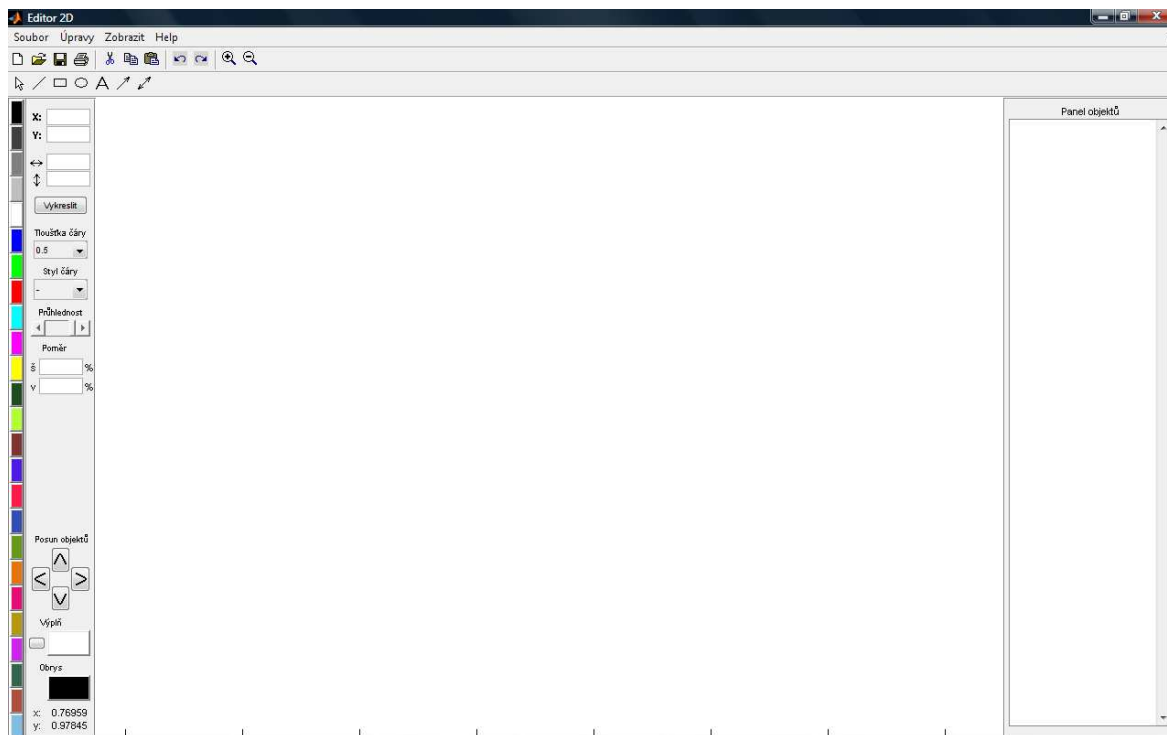
Při tvorbě editoru jsem kladl důraz na jednoduchost práce s editorem. Snažil jsem se nabídnout prvky, jež jsou v dalších podobných programech již zvykem a poskytují uživateli snadné ovládání objektů. Kreslicí okno je vyjádřeno rozměry velikosti 0 až 1 jak pro šířku, tak pro výšku. Editor je vytvořen jako jedna funkce umístěna v jednom souboru napsaném v textovém editoru MATLABu ve verzi 7.5.0 (R2007b).

4.1 Spuštění editoru

V MATLABu verze 7.5.0 (R2007b) otevřete skript nazvaný editor2d.m umístěný na příloženém CD. Pozor na verzi MATLABu, neboť v jiných verzích by mohlo dojít k některým chybám, které by neumožnili spuštění aplikace. Soubor můžete spustit dvojklikem na daný soubor, případně jej v prostředí MATLABu otevřít pomocí *File-Open*, kde již soubor vyhledáte.

Při otevření souboru je možno jej spustit třemi způsoby. Prvním je stisk tlačítka F5, druhým je stisk spouštěcí ikony zobrazené jako zelená šipka, třetím způsobem je kliknutím na položky *Debug- Run editor2d*.

4.2 Popis editoru



Obrázek 10- Zobrazení okna editoru

Editor je rozdělen do několika částí. Skládá se z hlavní lišty, která obsahuje jednotlivá menu s dalšími položkami, lišty s ikonami pro ovládání programu a další lišty s ikonami pro vykreslování a ovládání objektů. Dále je v programu umístěna barevná paleta ležící podél levého okraje okna, vedle níž se nachází panel s umístěnými editovacími boxy pro úpravu vykreslených objektů, dále tlačítka, pop-up menu a slider, s nimiž se ovládají vlastnosti vybraných objektů. Podél pravého okraje je umístěn panel objektů, který obsahuje všechny vytvořené objekty a slouží pro přehled nad všemi vybranými objekty, a také jako výběrové pole vytvořených entit. Zbytek okna je plocha pro vytváření dostupných entit.

4.2.1 Hlavní lišta

Hlavní lišta se skládá z menu a připojených podmenu. Program obsahuje čtyři základní menu, z nichž každé obsahuje další položky. Tyto čtyři menu včetně výčtu obsažených položek jsou následující:

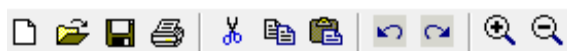
- Soubor

- Nový (Ctrl+N)- vytvoří se čistá plocha pro vykreslování nového obrázku s objekty. V případě, že je již na plátně něco vykresleno, je uživateli zobrazen dialog pro uložení rozdělaného obrázku.
 - Otevřít (Ctrl+O)- zobrazí se dialog pro otevření dříve uloženého obrázku. V případě, kdy je opět rozdělaný nějaký obrázek je nejprve zobrazen dialog pro uložení, aby nedošlo ke smazání rozdělané práce.
 - Uložit (Ctrl+S)- při kliknutí na tuto položku jsou do souboru uloženy vytvořené objekty. Pokud nebyla cesta pro uložení dříve definována, je nejprve nutno tuto cestu a soubor pro uložení vybrat pomocí zobrazeného dialogu.
 - Uložit jako...- otevření dialogu pro nastavení souboru, do kterého je třeba uložit vykreslené objekty.
 - Tisk (Ctrl+P)- zobrazení dialogu pro tisk vytvořeného snímku.
 - Konec- ukončení programu, jemuž předchází volba uživatele na uložení dosavadní práce.
- Úpravy
 - Zpět (Ctrl+Z)- tato volba poskytuje zpětné úpravy, které byly provedené. Pokud se již nově vytvořený objekt nebo jeho úprava uživateli nelíbí, tímto krokem se vrátí o krok zpět do stavu než byl objekt vytvořen nebo upraven.
 - Vpřed (Ctrl+Y)- poskytuje funkci, kdy uživatel provede krok zpět, ale chtěl by jej znovu vrátit, tedy krok vpřed.
 - Vyjmout (Ctrl+X)- tato položka slouží pro vyjmutí vybraných objektů. Vybrané objekty budou vyřiznuty (stanou se neviditelnými) a budou navraceny až v případě stisku položky *Vložit*. Pokud však nebude operace vložit zavolána dříve, než se do schránky například zkopírují nové objekty, budou objekty vymazány.
 - Kopírovat (Ctrl+C)- funkce slouží pro okopírování vybraných objektů, které jsou po zavolání funkce pro vložení zkopírovány a vytvořeny nové, identické objekty na stejných pozicích, jako jsou původní. Jejich pozice lze ihned měnit pomocí šipek nebo tlačítek.

- Vložit (Ctrl+X)- tato operace vkládá vyjmuté nebo okopírované objekty ze schránky.
- Zobrazit
 - Mřížka- poskytuje uživateli možnost zobrazení mřížky
 - Panel objektů- v případě, kdy je položka zaškrtnutá je zobrazen panel objektů, jinak není.
 - Panel barev- možnost zobrazení panelu barev na levé straně okna.
 - Panel úprav- zobrazení panelu úprav pro vytvořené entity.
- Help
 - O programu- Základní údaje o programu.

4.2.2 Lišta nástrojů

Na této liště jsou vytvořena jednotlivá tlačítka, která obsahují funkce popsané v menu *Soubor* a *Úpravy*. Tlačítka jsou zobrazena pomocí ikon vyjadřující význam dané operace. Jsou vytvořena v tomto pořadí: Nový soubor, Otevřít soubor, Uložit soubor, Tisk souboru, Vyjmout, Kopírovat, Vložit, Zpět, Vpřed, Zoom in, Zoom out. Poslední dvě položky slouží pro „zoomování“ vytvořených objektů. Tlačítko *Zoom-in* přiblíží všechny vytvořené objekty, tlačítko *Zoom-out* je naopak vzdálí. Možnost přiblížení a oddálení objektů je také možno provést pomocí rolování kolečka myši. V případě rolování směrem vzhůru se budou objekty přibližovat, v opačném případě oddalovat.



Obrázek 11- Lišta nástrojů

4.2.3 Lišta vykreslování entit

Tato lišta slouží pro samotné přidávání vykreslovacích entit.



Obrázek 12- Lišta vykreslovacích entit

- Výběr- jako první je na této liště zobrazeno tlačítko pro výběr grafických objektů. Při aktivaci tohoto tlačítka lze možno vybírat vytvořené objekty v panelu objektů a dále s nimi pracovat.
- Čára- další ikonou na liště je čára, která zobrazí do kreslicího menu čáru definovanou souřadnicemi počátečního a koncového bodu.
- Obdélník- vykreslí obdélník na pozici danou levým dolním rohem, šířkou a výškou. V případě definice stejného rozměru šířky a výšky je vytvořena entita čtverec.
- Elipsa- pomocí tohoto tlačítka se vykreslí elipsa daná opět levým dolním rohem, šířkou a výškou. Vytvořená elipsa o stejných rozměrech šířky a výšky je kružnice.
- Text- funkce poskytuje přidání textového editovacího pole do obrázku.
- Šipka (arrow)- vykreslení šipky dané souřadnicemi počátečního a koncového bodu šipky.
- Dvojitá šipka (doublearrow)- provádí možnost vykreslení dvojité šipky do prostředí pro vykreslování objektů. Šipka je definována souřadnicemi dvou bodů.

4.2.4 Panel barev

Při aktivaci výběru panelu barev v menu *Zobrazit* se zobrazí postranní panel s předem definovanými barvami. Tímto panelem se nastavuje výplň vybraných objektů. V okamžiku, kdy není vybrán žádný objekt a uživatel vybere barvu výplně, je nastavena tato barva jako výplň pro nově vytvořené objekty, načež jsou zobrazeny informace o tomto nastavení.

4.2.5 Panel úprav

Panel poskytuje možnost úpravy vybraných objektů, vykreslení objektů pomocí souřadnic, úpravy poměru stran objektů, posouvání objektů, definici barvy výplně, obrysu, stylu a velikosti čáry pro vykreslení.

Ve vrchní části panelu jsou vytvořeny čtyři editovací boxy, které na jedné straně slouží pro nové vytváření objektu, jež se definují souřadnicemi levého dolního rohu, šířkou a výškou (v případě šipky a čáry definicí souřadnic obou bodů) a vytváří stiskem tlačítka *Vykreslit*, na druhé straně jako boxy pro zobrazení informací o vybraném objektu a možnosti jeho editace. Další možností je výběr tloušťky čáry (velikosti 0,5-15) vybraných objektů (může



jich být i více) a stylu čáry těchto objektů (normální, čárkovaný, tečkovaný, čerchovaný, žádný). Dalším prvkem v tomto panelu je slider, který ovlivňuje průhlednost vybraných objektů (průhlednost se však vztahuje pouze k objektům obdélník a textbox) danou velikostí 0-1 (velikost 1 označuje objekt za zcela neprůhledný). Následující dva editovací boxy slouží pro zmenšení nebo zvětšení poměru šířky nebo výšky danou procentuálním vyjádřením. Posun objektů je možno provádět pomocí vytvořených čtyř tlačítek, které provedou posun daným směrem. Poslední barevná tlačítka vyznačují barvu výplně a obrysu pro vybrané objekty, v případě neoznačení jediného objektu se jimi definuje nastavení pro nově vytvořené objekty, jak již bylo dříve zmíněno. Vedle tlačítka pro výplň barvy se nachází malé tlačítko pro nastavení barvy výplně na průhlednou. Posledními objekty v tomto panelu jsou textové boxy poskytující informace o aktuálních souřadnicích kurzoru myši v rámci okna definovanou pozicí x a y .

4.2.6 Panel objektů

Panel objektů obsahuje všechny uživatelem vytvořené objekty. Tyto objekty jsou v seznamu zobrazeny pomocí svého jména (např. obdélník, elipsa, textbox atd.) a slouží především pro výběr objektů. S těmito vybranými

Obrázek 13- Panel úprav

objekty pak můžeme dále pracovat. Nad tímto panelem i nad vykreslovací plochou lze aktivovat kontextové menu pomocí stisku pravého tlačítka myši, které nám zobrazí možnosti pro vyjmutí vybraných objektů, kopírování, vložení a odstranění objektů.

4.3 Práce s editorem

Vykreslování pomocí souřadnic je možné, pouze pokud nebude označen žádný objekt v panelu objektů.

4.3.1 Vytvoření úsečky

V liště pro vykreslování entit je potřeba zvolit tlačítko pro vytvoření úsečky. Nyní se nabízí dvě možnosti. Vykreslit úsečku lze definováním souřadnic prvního bodu do kolonek X a Y v panelu úprav a druhého bodu do kolonek vyznačující šířku a výšku objektu. Kliknutím na

tlačítko *Vykreslit* se vykreslí úsečka na daných souřadnicích. Další možností je kliknutí do vykreslovací plochy, čímž se vytvoří úsečka v daném bodě, podržením tlačítka a tažením myši na další pozici se průběžně vykresluje úsečka, která se dokončí upuštěním stisknutého tlačítka myši. Tento nově vytvořený objekt se stane aktuálním a je možné jej dále různě specifikovat. Zároveň s jeho vytvořením se v kolonkách pro pozice objeví jeho informace o poloze, které je možno změnit.

4.3.2 Vytvoření obdélníku, čtverce

Zvolením tlačítka pro vytvoření obdélníka se startuje jeho vykreslení. Vykreslit jej můžeme opět pomocí specifikace souřadnic levého dolního rohu, šířky a výšky v příslušných editovacích polích, nebo kliknutím, podržením a tažením myši po kreslicím plátně. Čtverec se vytvoří definováním stejného rozměru pro výšku a šířku v případě parametrického vytváření, při kreslení myší lze čtverec vytvořit stiskem tlačítka Ctrl a tažením myši jedním ze směrů, čímž je délka stran definována stejným rozměrem vypočteným jako přepona vzdálenosti od počátečního bodu.

4.3.3 Vytvoření elipsy, kružnice

Pro vytvoření elipsy platí stejné funkce jako u vytvoření obdélníku v předešlém odstavci. Kliknutím se vytvoří počáteční bod, pohybem myši je průběžně vykreslována elipsa, jejíž dokončení je provedeno po upuštění tlačítka. Stiskem tlačítka Ctrl při vykreslování dojde k vykreslení kružnice dané stejným rozměrem poloměru. Nutno však podotknout, že dochází ke zkreslení kružnice a čtverce, které se naoko jeví jako elipsa a obdélník, jež je dáno poměrem stran který MATLAB zobrazuje.

4.3.4 Vytvoření bodu

Bod je možné vytvořit jako obdélník velmi malých rozměrů, čímž dostaneme hranatý bod, případně použitím vykreslení elipsy malých rozměrů, kde bod bude mít oválný charakter.

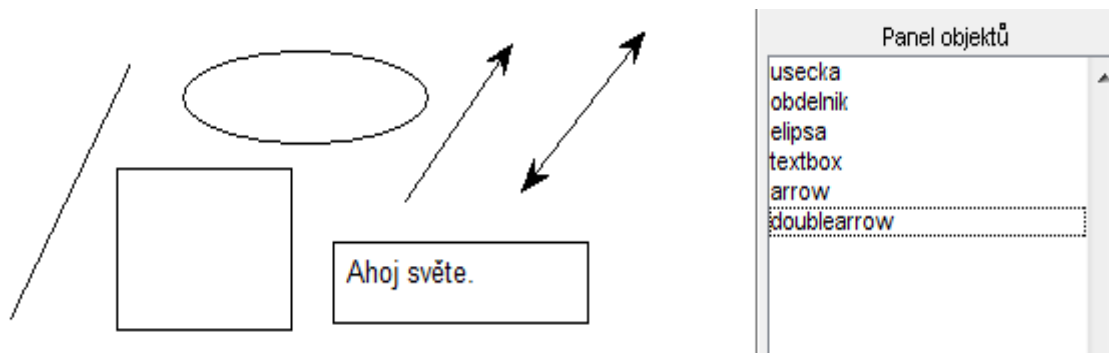
4.3.5 Vytvoření textu

Vytvoření je opět možno provést pomocí definice souřadnic do příslušných boxů (nutno stisk tlačítka *textbox*), nebo vykreslením pomocí stisku tlačítka myši a jeho tažením. Úpravu fontu a stylu textu lze upravit kliknutím na daný objekt v panelu objektu. Barva

písmu se upravuje pomocí barevného tlačítka obrys umístěné ve spodní části panelu úprav. Pozadí za textem se upravuje tlačítkem pro výplň objektu.

4.3.6 Vytvoření šipky a dvojité šipky

K vykreslení těchto šipek je nejprve nutno opět vybrat příslušné tlačítko, vykreslit tažením myši, případně definovat souřadnice počátečního a koncového bodu do editovacích polí pro počáteční bod (x, y) a koncový bod (kolonky šířka a výška).



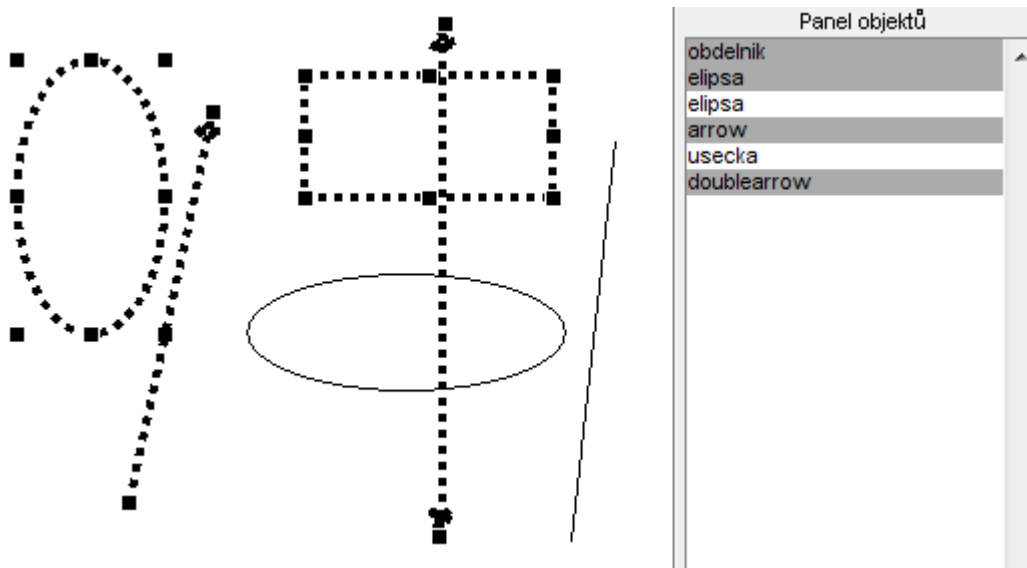
Obrázek 14- Ukázka vytvořených objektů a zobrazení panelu objektů s vytvořenými objekty

4.3.7 Úprava objektů

Objekty, které chce uživatel upravit, nejprve musí vybrat v panelu nástrojů. V tomto panelu lze vybrat více objektů stiskem tlačítka Ctrl, případně Shift. V případě výběru jednoho objektu jsou jeho údaje o pozici, výšce a šířce automaticky zobrazeny v příslušných kolonkách, kde jsou připraveny pro jejich změnu. Větší výběr objektů do těchto kolonek nezobrazí nic a zamítne jejich editaci.

4.3.7.1 Tloušťka čáry

Tloušťka čáry se nastaví v pop-up menu popsaném v panelu úprav. Nabízí hodnoty od 0.5, což je univerzálně nastavená hodnota, až po velikost čáry 15 bodů. Při změně této hodnoty se nastaví tloušťka čáry všech vybraných objektů v panelu objektů. Když nebude vybrán žádný objekt, nastaví se nová tloušťka jako defaultní pro nově vytvořené entity.



Obrázek 15- Ukázka změny tloušťky a stylu čáry pro vybrané objekty z panelu objektů

4.3.7.2 Styl čáry

Styl čáry se mění v pop-up menu definovaném v panelu úprav, jehož možnosti nabízí vytvořit styl čáry plný, čárkovaný, tečkovaný, čerchovaný nebo žádný. Tato možnost změni styl zobrazení čáry pro všechny vybrané objekty v panelu objektů, případně nastaví styl čáry jako defaultní pro nově vytvořené objekty.

4.3.7.3 Průhlednost

Průhlednost objektů se nastavuje pro objekty typu obdélník a text. Pro ostatní objekty již není tato vlastnost definovaná. Při výběru jediného objektu z těchto dvou typů objektů je hodnota průhlednosti nastavena právě na jeho aktuální průhlednost. U vícenásobného výběru objektů se viditelnost změni pouze pro objekty, které průhlednost obsahují.

4.3.7.4 Poměr strany

Pokud chce uživatel změnit délku šířky (\dot{s}) nebo délku výšky (v) objektu, je možné ji zkrátit případně zvětšit uvedením počtu procent do příslušné editovací kolonky pro šířku nebo výšku. Lze jej především využít pro zkrácení nebo zvětšení většího počtu objektů. Např. uvedením počtu 150 do kolonky pro šířku se šířka všech označených objektů zvětší 1,5krát. Číslo mezi 1-99 by danou šířku objektů snížilo (číslo/100)krát, tedy pro číslo 57 by

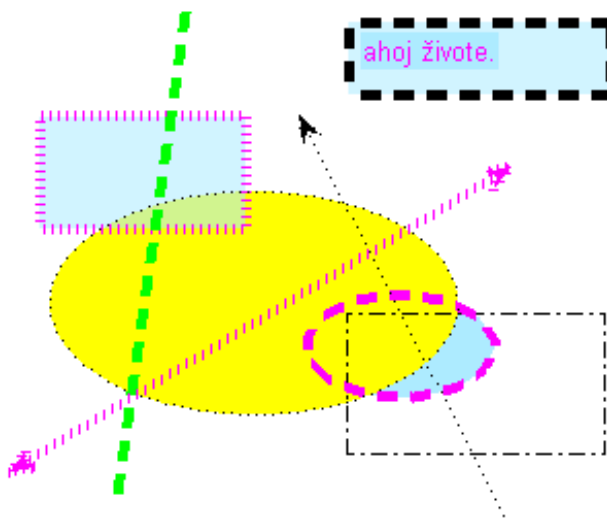
se zmenšila šířka 0.57krát, tedy skoro na polovinu. Pokud uživatel nebude mít vybrán žádný objekt, nebude moci upravovat délku a šířku žádných objektů.

4.3.7.5 Výplň

Výplň objektů se nastavuje pomocí barevného tlačítka umístěného ve spodní části. Podle počtu vybraných objektů se nastaví pro všechny objekty, pro které nastavit lze. Týká se to objektů typu obdélník, elipsa, text, naopak pro objekty čára, šipka, dvojitá šipka výplň nastavit nelze.

4.3.7.6 Obrys

Obrys objektů se nastavuje stejným způsobem jako výplň. Když nebude vybrán žádný objekt, změna barvy obrisu bude nastavena jako výchozí pro nově vytvořené objekty, jinak se změní obrys všech vybraných objektů. Změna obrisu pro objekt text změní barvu písma obsaženého v tomto textboxu.



Obrázek 16- Ukázka vytvoření entit a změny různých parametrů

4.3.8 Přesun objektů

Přesouvání objektů je možné dvěma způsoby. Nejprve je nutné si vyznačit objekty, které chceme přesunout, pomocí panelu objektů. Jakmile budou vyznačené, přesouvat lze pomocí stisku tlačítek vyznačujících směr posunu umístěných ve spodní části panelu úprav.

Další možností je stisk tlačítek pro pohyb na klávesnici. To je však vázáno nutností nemít aktivně zobrazený panel objektů, neboť v případě jeho aktivního zobrazení by pohyb pomocí tlačítek ovlivňoval výběr řádků v tomto panelu, nikoliv posun objektů. Uživatel tedy musí po výběru entit kliknout do vykreslovacího pole (pozor, aby nebylo aktivní vykreslování nějaké entity, ale aby byl aktivní výběr značený ikonou šipky), načež bude moci použít šipky na klávesnici pro pohyb všech označených objektů, který lze používat do doby, než se některá z entit nedostane na hranu pro vykreslení.

4.3.9 Ukládání objektů

Vytvořené objekty si je možno uložit a načíst do vlastního formátu. Tomuto formátu jsem nastavil koncovku *jméno.m2d*, což znamená zkratku MATLAB 2D. Při ukládání se uživateli zobrazí dialog pro místo uložení a název souboru pro uložení. Po zadání těchto údajů se do daného souboru uloží vytvořené objekty, které jsou zde umístěny jako struktura vytvořená v MATLABu. Tato struktura nese nejdůležitější údaje o daném objektu. První položkou v této struktuře je *Tag*, což je taková přezdívka každého objektu. V programu jsem u vytvoření objektu definoval tuto hodnotu vždy na typ objektu, tedy úsečka, elipsa, obdélník, textbox, arrow, nebo doublearrow. Další sekci ve struktuře byla *Pozice1*, do které se v případě objektů mající pozici definovanou jedním vektorem velikostí čtyř čísel uložila tato pozice, u objektů definovaných dvěma body (úsečka, arrow, doublearrow) se zde uložili souřadnice prvního bodu. Dalším prvkem byla *Pozice2*, do které se v případě obdélníku uložila průhlednost objektu, v případě textboxu vyplněný text, při entitě elipsy se uložila stejná pozice jako do *pozice1* (i když nemusela), u čar a šipek se uložili souřadnice druhého bodu. Dalšími kolonkami ve struktuře byli výplň, obrys, tloušťka čar a styl čar. Vytvoření struktury vypadalo následovně:

```
46 vektorUlozit=struct('Tag',{},'Pozice1',{},'Pozice2',{},'FaceColor',...
47 {},'EdgeColor',{},'LineWidth',{},'LineStyle',{});
```

Přidávání prvků pomocí indexů pak vypadá takto:

```
48 vektorUlozit(i)=struct('Tag',get(handleobj,'Tag'),'Pozice1','atd...')
```

K uložení prvku do souboru bylo použito příkazu *save(File, vektorUlozit)*.

4.3.10 Otevírání objektů

Otevření objektu probíhá pomocí dialogu pro výběr souboru, jež byl omezen pouze na soubory s koncovkou *.m2d*. Po výběru souboru jsou data importována do proměnné, k jejímž objektům se přistupuje pomocí indexu. K prvkům struktury pak pomocí metody *tečky*.

```
49 vektorU=importdata(file);  
50 vektorU(i).Pozice1
```

5 TECHNIKY POUŽITÉ PŘI PROGRAMOVÁNÍ EDITORU

Obrázek Figure vytvořený příkazem *figure* obsahuje mnoho funkcí, které byly pro tvorbu této práce vysoce využity. Seznam funkcí a vlastností objektu obrázek si můžeme vypsat pomocí příkazu *get(handleObrázku)*. Pomocí vlastnosti *WindowButtonMotionFcn*, která vykoná nastavenou událost při každém malém posunutí myši po obrázku, jsem si definoval funkci, která se tedy vykoná při každém pohybu myši. Tato vlastnost je tedy například využita pro zobrazení informací o aktuální pozici myši v obrázku, jež je v programu zobrazeno v levém dolním rohu, dále byla nedílnou součástí průběhu vykreslení každé zvolené entity. Jelikož bylo nutné nějakým způsobem zobrazovat průběh vykreslování objektu, právě tato funkce jej v programu nejvíce obsluhovala. Další užitečnou vlastností, která byla nutně využita, bylo vykonání nějaké události při kliknutí do obrázku. To nám obsluhovala vlastnost *WindowButtonDownFcn*, které se nastavil handle funkce pro vykonání. Definice této události, která má nastat při každém kliknutí do obrázku by vypadala následovně:

```
51 set(gcf, 'WindowButtonDownFcn', @FigureKeyButtonDownCallback)
```

Uvedením operátoru @ před názvem funkce dostaneme ukazatel na danou funkci (handle funkce). V programu tak bylo při stisku tlačítka myši zjištěno, které tlačítko je aktivní a podle něj začalo vykreslovat danou entitu, pokud bylo vybráno tlačítko pro vykreslování a nikoliv pro výběr. Zjištění aktuální pozice je umístěno ve vlastnosti *CurrentPoint* objektu *figure*.

Opakem pro definici události, která má nastat při stisku tlačítka je událost, která má nastat při upuštění tlačítka. Ta je již definována vlastností *WindowButtonUpFcn*.

```
52 set(gcf, 'WindowButtonUpFcn', @UpusteniMysi);
```

Pro točení kolečkem myši slouží vlastnost *WindowScrollWheelFcn*, která je v programu využita pro zoom pracovní plochy.

```
53 Souradnice=get(findobj('Tag','Figure'),'CurrentPoint');
54     if evnt.VerticalScrollCount > 0
55         TlZoomIn(src, evnt)
56     elseif evnt.VerticalScrollCount < 0
57         TlZoomOut(src, evnt)
58     end
```

Popis předchozího úseku kódu nacházející se ve funkci, která je volána při každém otočení kolečkem v rámci obrázku, lze vysvětlit tak, že pokud je počet otočení kolečkem větší než

nula, zavolej funkci *TlZoomIn*, jinak pokud je menší než nula, zavolej funkci *TlZoomout*. Kladné hodnoty udávají počet otočení kolečkem směrem dolů, záporné hodnoty směrem nahoru.

Mezi další použitou vlastnost objektu *figure* patřila *SelectionType*, která obsahuje typ informace o posledním kliknutí myši. Může nabývat těchto hodnot:

- Normal- Klasické kliknutí levým tlačítkem myši.
- Extend- Značí kliknutí levého tlačítka myši se stisknutým tlačítkem Shift, nebo stlačení kolečka myši.
- Alternate- Kliknutí levého tlačítka myši se stisknutým tlačítkem Ctrl, nebo stisk pravého tlačítka myši.
- Open- Dvojité kliknutí levého tlačítka myši.

```
59 sel_typ = get(gcf,'SelectionType')
60     switch sel_typ
61         case 'normal'
62             disp('Kliknutí levého tlačítka myši')
63         case 'extend'
64             disp('Kliknutí se Shift')
65         case 'alt'
66             disp('Kliknutí s Ctrl')
67     end
```

Tato operace byla při budování programu použita pro zobrazení kontextového menu vybraných objektů nabízející možnosti vyjmout, kopírovat, vložit a odstranit.

Další použitou funkcí objektu *figure* byla detekce stisku libovolného tlačítka na klávesnici. Tato vlastnost se jmenuje *KeyPressFcn*, která vykoná definovanou funkci při každém stisku tlačítka. Struktura této vlastnosti obsahuje následující položky:

- Character- znak zobrazený jako výsledek stisknuté klávesy.
- Modifier- tato položka obsahuje jména jednoho nebo více modifikátorů, které uživatel stiskl (control, alt, shift).
- Key- stisknuté tlačítko

```
68 if length(evnt.Modifier) == 1 & strcmp(evnt.Modifier{:},'control')
69     stiskCtrl='true';
```

```
70         else
71             stiskCtrl='false';
72         end
```

Opačný stav uvolnění klávesy je definován vlastností *KeyReleaseFcn*, tedy událost při uvolnění stisknuté klávesy. Obsahuje stejné položky jako předchozí vlastnost.

6 POROVNÁNÍ EDITORU S JINÝMI EDITORY

Tento grafický editor je jen jednoduchým vykreslovacím programem pro vykreslování entit, jako je čára, čtverec, obdélník, elipsa, kružnice, šipka. Nabízí několik málo funkcí pro úpravu těchto objektů, od barvy objektů, stylu zobrazení objektů, až po přesouvání objektů. Nemůžeme se rovnat s profesionálními grafickými editory, jako je Corel DRAW, nebo Gimp, které nabízejí větší možnosti funkcí a úpravy. Nicméně bych vyzvedl jeho jednoduchost, kreslení a nastavení vlastností se provádí velmi rychle a pro vykreslení základních útvarů postačí.

ZÁVĚR

Hlavní zásadou pro vypracování této práce bylo vytvoření 2D grafického editoru v Matlabu, který bude poskytovat uživateli vytvářet různé grafické objekty pomocí kliknutí myši, nebo pomocí zadání parametrů. Editor byl vytvořen jako jeden m-soubor ve verzi MATLABu 7.5.0 (R2007b). Obsahuje různé panely nástrojů, jež byly další zásadou pro vypracování, a ukládá vytvořené objekty do vlastního formátu, ze kterého je lze poté opět načíst.

V teoretické části je jednoduše popsána tvorba skriptů a funkcí, které byly využity při tvorbě praktické části, dále o objektově orientovaném programování, které je v MATLABu dostupné od verze R2008a a o systému Handle Graphics, který pokrývá tvorbu oken, menu, grafických ovládacích prvků uživatelského rozhraní.

V praktické části je detailně popsáno prostředí vytvořeného programu, popis použití funkcí programu, a také informace o programování tohoto editoru. Jsou zde zobrazeny i ukázky tvorby a popis využitých funkcí.

Vytvořený grafický editor je jednoduchý pro používání, obsahuje některé funkce, které jsou pro grafické editory typické a poskytuje uživateli jednoduchou tvorbu 2D objektů. Soupeřit s profesionálními editory by určitě nemohl, ale jako taková lehká nadstavba pro MATLAB může docela dobře posloužit a studenti by si díky němu mohli osahat i další sekci tohoto jazyka a vytvořit si zajímavé obrázky.

ZÁVĚR V ANGLIČTINĚ

The main principle for the elaboration of this work was to create a 2D graphics editor in Matlab, which will provide creations of various graphic objects by clicking the mouse or by entering the parameters. Editor was created as one m-file in MATLAB version 7.5.0 (R2007b). It contains a variety of toolbars, which was the other principle for the development, saves created objects into own format, from which it can be read again.

In theoretical part it is simply described creation of scripts and functions that were used in developing the practical part, the object-oriented programming that is available in MATLAB R2008a version, about System Handle Graphics, which covers creations figures, menus, and graphical user interface controls.

In practical part is described the environment of created program in detail, description of use features and information about programming of this editor. In this part it is shown an examples of work and description of the functions which was used.

Created graphics editor is easy to use, it contains some functions, that are typical for graphics editors and provides to users simply creations of 2D objects. Program can't match with other professional editors, but it should be enough as lightweight shell for MATLAB and students would be able to touch him trough the next section of this language and create an interesting images.

SEZNAM POUŽITÉ LITERATURY

- [1] PERUTKA, K. MATLAB - Základy pro studenty automatizace a informačních technologií, 1. vyd. Zlín: UTB ve Zlíně, 2005. 304 s. ISBN 80-7318-355-2.
- [2] DOŇAR, B.; ZAPLATÍLEK, K. MATLAB- Tvorba uživatelských aplikací. 1. vyd. Praha: BEN- technická literatura, 2004. 216 s. ISBN 80-7300-133-0.
- [3] PROKOP, D. Průvodce grafickými nadstavbami Octave. Zlín, 2007. 44 s. Bakalářská práce na Fakultě aplikované informatiky UTB ve Zlíně. Vedoucí bakalářské práce Karel Perůtka.
- [4] MICHÁLEK, R. Multimediální učební pomůcka předmětů A5MAS, A4MAS. Zlín, 2010. 58 s. Bakalářská práce na Fakultě aplikované informatiky UTB ve Zlíně. Vedoucí bakalářské práce Karel Perůtka.
- [5] MATLAB Homepage. URL: [<http://www.mathworks.com/>] [cit. 2011-05-15].
- [6] MATLAB Homepage- Documentation. URL: [www.mathworks.com/help/techdoc/] [cit. 2011-05-15].
- [7] Humusoft Homepage. URL: [<http://humusoft.cz/produkty/matlab/matlab/>] [cit. 2011-05/15].
- [8] Odborné časopisy online- Objektově orientované programování v prostředí Matlab. URL: [<http://www.odbornecasopisy.cz/res/pdf/42116.pdf>] [cit. 2011-05/15].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CD Compact disc.

Např. Například.

Atd. A tak dále.

SEZNAM OBRÁZKŮ

<i>Obrázek 1- Volání skriptu</i>	12
<i>Obrázek 2- Ukázkový skript</i>	13
<i>Obrázek 3- Hierarchie systému Handle Graphics</i>	22
<i>Obrázek 4- Vykreslení čáry pomocí příkazu „line“</i>	23
<i>Obrázek 5- Využití příkazu „set“ pro zjištění možností určité vlastnosti</i>	24
<i>Obrázek 6- Typy objektů pro grafický objekt Figure [6]</i>	25
<i>Obrázek 7- Zobrazení zkrácení objektu kružnice (vlevo) a jeho napravení úpravou jedné z vlastností</i>	26
<i>Obrázek 8- Ukázka všech stylů pro objekty „uicontrol“</i>	31
<i>Obrázek 9- Zobrazení vytvořených objektů uživatelského rozhraní</i>	32
<i>Obrázek 10- Zobrazení okna editoru</i>	35
<i>Obrázek 11- Lišta nástrojů</i>	37
<i>Obrázek 12- Lišta vykreslovacích entit</i>	37
<i>Obrázek 13- Panel úprav</i>	39
<i>Obrázek 14- Ukázka vytvořených objektů a zobrazení panelu objektů s vytvořenými objekty</i>	41
<i>Obrázek 15- Ukázka změny tloušťky a stylu čáry pro vybrané objekty z panelu objektů</i>	42
<i>Obrázek 16- Ukázka vytvoření entit a změny různých parametrů</i>	43

SEZNAM TABULEK

<i>Tabulka 1- Vlastnosti atributů</i>	16
<i>Tabulka 2- Vlastnosti událostí</i>	18
<i>Tabulka 3- Vlastnosti metod</i>	19

SEZNAM PŘÍLOH

P I Disk CD-ROM

PŘÍLOHA P I: DISK CD-ROM

Na disku je uložen kompletní zdrojový kód, který je vytvořen v jediném m-souboru. Disk také obsahuje tuto práci