České vysoké učení technické v Praze

Fakulta elektrotechnická

Bakalářská práce

Grafový editor pro Scheduling toolbox TORSCHE

Editor of graphs for Scheduling toolbox TORSCHE

Vypracoval: Vojtěch Navrátil

Vedoucí práce: Ing. Michal Kutil

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky Školní rok: 2005/2006

Zadání bakalářské práce

Student: Vojtěch Navrátil

Obor: Kybernetika a měření

Název tématu: Grafový editor pro Scheduling toolbox TORSCHE

Zásady pro vypracování:

1. Seznamte se se základy teorie grafů. Prostudujte základní příkazy a možnosti Scheduling toolboxu pro Matlab.
2. Vytvořte pomocí Matlab GUI uživatelsky přístupný edito grafů a jejich vlastností.
3. Editor grafů rozšiřte o systém zásuvných modulů, poskytující jednotné rozhraní pro další rozšiřování vlastností editoru. Připravte ukázkový modul využívající funkce Scheduling toolboxu TORSCHE.

Seznam odborné literatury: Dodá vedoucí práce

Vedoucí bakalářské práce: Ing. Michal Kutil
Datum zadání bakalářské práce: zimní semestr 2005/06

Termín odevzdání bakalářské práce: 30.6.2006

L.S.

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

Prof. Ing. Zbyněk Škvor, DrSc.
děkan

V Praze, dne 9. 3. 2006

Prohlášení

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

V Praze dne ……. …….
podpis

Poděkování

Na tomto místě děkuji vedoucímu své bakalářské práce Ing. Michalu Kutilovi za užitečné rady, pomoc při odstraňování chyb a náměty pro zdokonalení práce. Rovněž děkuji celé své rodině za podporu při studiu a psaní této práce.

Téma

Grafový editor pro Scheduling toolbox TORSCHE

Abstrakt

V rámci tohoto projektu se zabývám tvorbou GUI s využitím systému Handle Graphics implementovaného v Matlabu.

Cílem je vytvořit uživatelsky snadno přístupný editor grafů a jejich vlastností Graphedit, který bude pracovat jako součást Scheduling toolboxu TORSCHE.

V úvodu práce popisuji základy práce s grafickými objekty Matlabu, poté návrh GUI a jeho realizaci. Následuje část, v níž jsou popsány funkce Grapheditu jako import a export z/do workspace Matlabu, načítání a ukládání z/do binárního souboru *.mat, kreslení vrcholů, hran a jejich mazání a další. Závěrečnou část věnuji systému zásuvných modulů umožňujícímu připojení funkcí, které jsou součástí Scheduling toolboxu nebo které uživatel implementuje sám.

Theme

Editor of graphs for Scheduling toolbox TORSCHE

Abstract

In order to this project I inquire into creation of GUI with utilization of system Handle Graphics implemented in Matlab.

The purpose of my work is to create user-friendly editor of graphs and their properties Graphedit, which will work as a part of Scheduling toolbox TORSCHE.

An introduction of the work is focused on basic operations with Matlab's graphical objects and on design of GUI and its realization. It is followed by description of Graphedit's functions import, export from/to Matlab's workspace, loading, saving from/to binary mat-file, drawing of nodes, edges, deleting mode etc. Another part attends to implementation of plugins. Plugins give to user possibility of adding functions, which belong to graph parts of Scheduling toolbox or which user implements by self.

Obsah

1 ÚVOD ... 1

2 MATLAB GUI – SYSTÉM HANDLE GRAPHICS ... 2

2.1 GRAFICKÉ OBJEKTY A JEJICH HIERARCHIE ... 2
2.1.1 Grafický objekt figure ... 3
2.1.2 Grafický objekt uicontrol ... 5
2.1.3 Grafický objekt uimenu ... 5
2.1.4 Grafický objekt uicontextmenu ... 6
2.1.5 Grafický objekt uitoolbar ... 6
2.1.6 Grafický objekt axes ... 6
2.1.7 Grafické objekty rectangle, line a text ... 7
2.2 CALLBACK RUTINY GRAFICKÝCH OBJEKTŮ ... 7
2.3 PROSTŘEDÍ GUIDE ... 9
2.4 PŘEDDEFINOVANÉ DIALOGY MATLABU ... 9

3 GRAF ... 11

3.1 GRAF – OBJEKT GRAPH, OBJEKT AXES ... 12
3.2 VRCHOL – OBJEKT NODE, OBJEKT RECTANGLE ... 13
3.3 HRANA – OBJEKT EDGE, OBJEKT LINE ... 14

4 GRAPHEDIT ... 15

4.1 HLAVNÍ MENU ... 15
4.2 LIŠTA NÁSTROJŮ ... 15
4.3 KRESLICÍ PLOCHA ... 16
4.3.1 List kreslicí plochy ... 16
4.3.2 Posuvníky ... 17
4.4 ZÁLOŽKY ... 18
4.5 PROPERTY EDITOR – EDITOR VLASTNOSTÍ ... 19

5 DIALOGY ... 23

5.1 GRAPHEDITLISTDLG ... 23
5.2 GRAPHEDITADDPLUGIN ... 25
5.3 GRAPHEDITREMOVEPLUGIN ... 26
5.4 GRAPHEDITDOPLUGIN ... 27

6 FUNKCE ... 28

6.1 SPUŠTĚNÍ GRAPHEDITU ... 28
6.2 FILE ... 28
6.2.1 New ... 28
6.2.2 Open ... 28
6.2.3 Save ... 30
6.2.4 Export to picture ... 31
6.2.5 Import from workspace, Export to workspace ... 31
6.2.6 Close, close all, quit ... 31
6.3 EDIT ... 32
6.3.1 Cut, copy ... 32
6.3.2 Paste ... 32
6.4 VIEW ... 32
6.4.1 Property Editor ... 32
6.4.2 Names, UserParams ... 33
6.5 DRAW ... 34
6.5.1 Draw node ... 34
6.5.2 Draw edge ... 34
6.5.3 Delete mode ... 35
6.5.4 Layout ... 36
6.6 OPTIONS ... 36
6.6.1 Unique names ... 36

6.7 PLUGINS ... 37
6.7.1 Add, remove plugin ... 37

7 PLUGINS – SYSTÉM ZÁSUVNÝCH MODULŮ ... 38

7.1 XML (EXTENDED MARKUP LANGUAGE) ... 38
7.2 UKLÁDÁNÍ ... 39
7.3 NAČÍTÁNÍ ... 39
7.4 PŘÍKLADY ... 40
7.4.1 Hamiltonovská kružnice ... 40
7.4.2 Obarvení vrcholů (jednou barvou) ... 41
7.4.3 Export jmen vrcholů a uživatelských parametrů hran ... 42
7.4.4 Vytvoření vrcholů grafu ... 43

8 ZÁVĚR ... 44

9 SEZNAM LITERATURY ... 45

10 POUŽITÝ SOFTWARE ... 45

11 PŘÍLOHA A ... 46

Seznam diagramů

DIAGRAM 5.1 – *FUNKCE DRAWOBJECTGRAPH* 29
DIAGRAM 5.2 – *FUNKCE CREATEOBJECTGRAPH* 29
DIAGRAM 5.3 – *FUNKCE OPENGRAPHFROMFILE* 29
DIAGRAM 5.4 – *FUNKCE SAVEGRAPHTOFILE* 29
DIAGRAM 5.5 – *FUNKCE CREATECANVAS* 30
DIAGRAM 5.6 – *FUNKCE COPYNODEFCN* 30
DIAGRAM 5.7 – *FUNKCE PASTENODEFCN* 30
DIAGRAM 5.8 – *FUNKCE VIEWTEXT* 34
DIAGRAM 5.9 – *FUNKCE DRAWNODE* 35
DIAGRAM 5.10 – *FUNKCE DRAWEDGE* 35

Seznam kódů Matlabu

KÓD MATLABU 2.1 – *PŘÍKLAD NASTAVENÍ PARAMETRŮ GRAFICKÉHO OBJEKTU.* 2
KÓD MATLABU 2.2 – *UMÍSTĚNÍ TLAČÍTKA NA NOVĚ VYTVOŘENÉ OKNO FIGURE.* 5
KÓD MATLABU 2.3 – *PŘÍKLAD VYTVOŘENÍ VLASTNÍHO SUBMENU V HLAVNÍM MENU.* 5
KÓD MATLABU 2.4 – *PŘÍKLAD ZPŮSOBU VYTVOŘENÍ KONTEXTOVÉHO MENU.* 6
KÓD MATLABU 2.5 – *PŘÍKLAD VYTVOŘENÍ VLASTNÍ NÁSTROJOVÉ LIŠTY S TLAČÍTKEM S IKONOU PICTURE.PNG.* 6
KÓD MATLABU 2.6 – *VYTVOŘENÍ KOLEČKA NA POZICI [X,Y] JAKO GRAFICKÉHO OBJEKTU RECTANGLE.* 7
KÓD MATLABU 2.7 – *VYTVOŘENÍ GRAFICKÉHO OBJEKTU LINE – ÚSEČKY MEZI BODY [X1,Y1] A [X2,Y2].* 7
KÓD MATLABU 2.8 – *VYTVOŘENÍ GRAFICKÉHO OBJEKTU TEXT NA POZICI [X,Y] SE ZAROVNÁNÍM NA STŘED.* 7
KÓD MATLABU 2.9 – *PŘÍKLAD REGISTRACE CALLBACK FUNKCÍ PRO OBJEKTY AXES A UIMENU.* 8
KÓD MATLABU 3.1 – *VYTVOŘENÍ PROMĚNNÉ TYPU GRAPH SE TŘEMI VRCHOLY A DVĚMA HRANAMI.* 12
KÓD MATLABU 3.2 – *VÝPIS PARAMETRŮ PROMĚNNÉ TYPU GRAPH.* 12
KÓD MATLABU 3.3 – *STRUKTURA ULOŽENÁ V GRAFICKÉM OBJEKTU AXES.* 13
KÓD MATLABU 3.4 – *PARAMETRY OBJEKTU NODE A NASTAVENÍ JEHO GRAFICKÝCH PARAMETRŮ.* 13
KÓD MATLABU 3.5 – *STRUKTURA ULOŽENÁ V GRAFICKÉM OBJEKTU RECTANGLE.* 14
KÓD MATLABU 3.6 – *PARAMETRY OBJEKTU EDGE A NASTAVENÍ GRAFICKÝCH PARAMETRŮ.* 14
KÓD MATLABU 3.7 – *STRUKTURA ULOŽENÁ V GRAFICKÉM OBJEKTU LINE.* 14
KÓD MATLABU 7.1 – *KÓD FUNKCE* `NODESTOCOLOR` *– MODULU PRO OBRAVENÍ VRCHOLŮ.* 41
KÓD MATLABU 7.2 – *KÓD FUNKCE* `GETNAMESANDUSERPARAMS`. 42
KÓD MATLABU 7.3 – *KÓD FUNKCE* `CREATENODES` *– VYTVOŘENÍ ZADANÉHO POČTU VRCHOLŮ.* 43

Seznam obrázků

OBRÁZEK 1.1 – *HIERARCHIE ZDE POUŽITÝCH ZÁKLADNÍCH GRAFICKÝCH OBJEKTŮ MATLABU.* 2
OBRÁZEK 1.2 – *ODEZVA NA VYTVOŘENÍ NOVÉHO GRAFICKÉHO OBJEKTU TYPU FIGURE.* 4
OBRÁZEK 1.3 – *CHYBA, KTERÁ VZNIKÁ PŘI ŠPATNÉM NASTAVENÍ HANDLEVISIBILITY.* 4
OBRÁZEK 1.4 – *ROZBALENÁ NABÍDKA NOVÉHO GRAFICKÉHO OBJEKTU UIMENU.* 5
OBRÁZEK 1.5 – *PŘÍKLADY PŘEDDEFINOVANÝCH DIALOGŮ V SYSTÉMU MATLAB.* 10
OBRÁZEK 2.1 – *ORIENTOVANÝ JEDNODUCHÝ OHODNOCENÝ GRAF V GRAPHEDITU.* 11
OBRÁZEK 3.1 – *LIŠTA NÁSTROJŮ GRAPHEDITU.* 15
OBRÁZEK 3.2 – *GRAF JE PŘÍLIŠ VELKÝ PRO JEHO ZOBRAZENÍ – POUŽITÍ OBJEKTU SLIDER.* 17
OBRÁZEK 3.3 – *VYUŽITÍ ZÁLOŽEK, ZOBRAZENÍ OBJEKTU SLIDERU PRO JEJICH POSUV.* 18
OBRÁZEK 3.4 – *ZOBRAZENÝ PROPERTY EDITOR I S POUŽITÝM POSUVNÍKEM SLIDER.* 20
OBRÁZEK 3.5 – *ZNÁZORNĚNÍ STRUKTURY S PŘÍKLADY PARAMETRŮ.* 20
OBRÁZEK 4.2 – *ZOBRAZENÍ DIALOGU GRAPHEDITADDPLUGIN.* 25
OBRÁZEK 4.3 – *ZOBRAZENÍ DIALOGU GRAPHEDITREMOVEPLUGIN.* 26
OBRÁZEK 4.4 – *ZOBRAZENÍ DIALOGU GRAPHEDITDOPLUGIN.* 27
OBRÁZEK 5.1 – *VSTUP A VÝSTUP FUNKCE NODESLAYOUT.* 36
OBRÁZEK 6.1 – *PŘÍKLAD POUŽITÍ MODULU HAMILTON_CIRCUIT.* 40
OBRÁZEK 6.2 – *PŘÍKLAD POUŽITÍ MODULU NODESTOCOLOR.* 41
OBRÁZEK 6.3 – *PŘÍKLAD POUŽITÍ MODULU CREATENODES.* 43

Seznam xml kódů

XML KÓD 6.1 – *UKÁZKA KÓDU SOUBORU GRAPEDITPLUGIN.XML.* 39

Seznam tabulek

TABULKA 1.1 – *ZÁKLADNÍ GRAFICKÉ OBJEKTY VE FORMĚ TABULKY S UVEDENÍM VÝZNAMU.* 3
TABULKA 1.2 – *VYBRANÉ CALLBACK ROUTINY GRAFICKÝCH OBJEKTŮ.* 8
TABULKA 1.3 – *VYBRANÉ PŘEDDEFINOVANÉ DIALOGY POUŽITÉ V GRAPHEDITU.* 9
TABULKA 3.1 – *PARAMETRY OBJEKTU NODE.* 19
TABULKA 3.2 – *PARAMETRY OBJEKTU EDGE.* 19
TABULKA 3.3 – *PARAMETRY OBJEKTU GRAPH.* 19
TABULKA 3.4 – *TABULKA TYPŮ PARAMETRŮ ZOBRAZOVANÝCH V PROPERTY EDITORU.* 21
TABULKA 4.1 – *VÝPIS PARAMETRŮ FUNKCE GRAPHEDITLISTDLG.* 24

1 Úvod

Cílem mé práce bude vytvořit grafový editor pro Scheduling toolbox TORSCHE s využitím Matlab systému Handle Graphics. Tento grafový editor (Graphedit) musí uživateli poskytnout prostředí pro snadnou editaci grafů a jejich vlastností s dostatečně intuitivním ovládáním. Graphedit rovněž musí obsahovat systém zásuvných modulů – pluginů pro umožnění rozšiřování svých vlastností a funkcí podle konkrétních potřeb uživatele.

V první fázi se musím věnovat nastudování možností systému Handle Graphics, abych byl schopen vhodně použít jednotlivé grafické objekty a dokázal s nimi podle potřeby manipulovat. Po vysvětlení základních pojmů a objektů systému Handle Graphics a seznámení s prvky grafové části Scheduling toolboxu zvolím představitele těchto prvků v grafické oblasti.

Protože parametry objektů typu edge Scheduling toolboxu zatím neumožňují uchování informací o tvaru grafických objektů, které je budou reprezentovat, zaměřím se pouze na možnost kreslení a zobrazování jednoduchých grafů.

Návrh vzhledu Grapheditu musím provést především s ohledem na požadavek intuitivního ovládání. Vhodnou inspirací mi bude zkušenost s prací s jinými, hojně používanými aplikacemi.

Jako poslední část své práce přidám do Grapheditu, který již bude schopný jednoduchého kreslení a zobrazování grafů, systém zásuvných modulů pro rozšíření nabídky funkcí jak o funkce implementované v Scheduling toolboxu, tak o funkce za nějakým účelem připravené přímo uživatelem.

2 Matlab GUI – Systém Handle Graphics

Vše, co v této kapitole uvádím, je mnohem podrobněji vysvětleno v literatuře [2], z níž jsem čerpal, a každý z grafických objektů je přehledně popsán v nápovědě Matlabu.

Systém Handle Graphics je grafický systém implementovaný v Matlabu, s jehož pomocí lze efektivně pracovat s grafickými objekty. Zahrnuje příkazy pro 2D a 3D vizualizaci dat, zpracování signálů, animací a grafiku obecně. Z hlediska uživatele přináší vyšší efektivnost a mnoho možností při práci s grafikou.

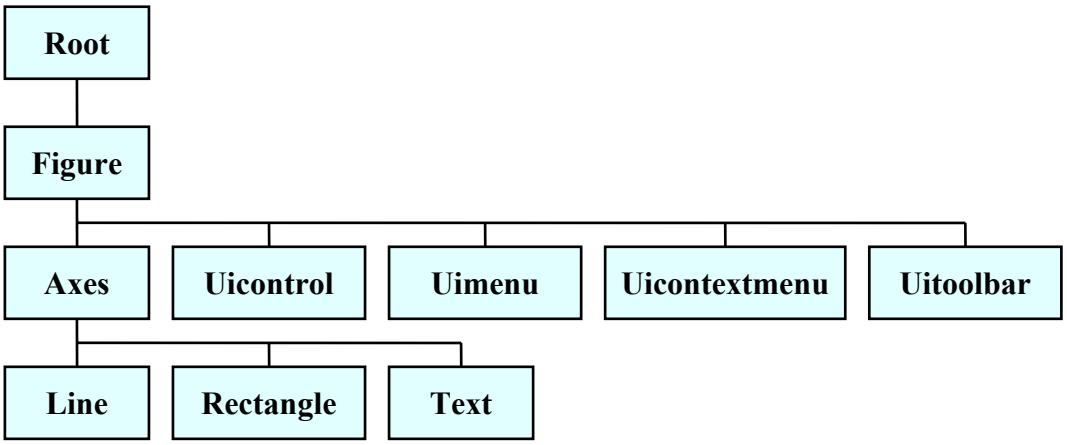
Handle je proměnná, která je při vytvoření grafického objektu naplněna číslem, jež jednoznačně identifikuje grafický objekt. Umožňuje ovládání grafických objektů funkcemi **`get`** a **`set`**. Pomocí **`get`** se ptáme na určité parametry a pomocí **`set`** je naopak měníme. Máme-li k dispozici handle **`hObject`** grafického objektu, získáme zadáním **`get(hObject)`** seznam všech parametrů daného objektu i s jejich aktuální hodnotou. Změnit libovolnou položku tohoto seznamu lze příkazem **`set`** tak, jak je například uvedeno v kódu Matlabu 2.1.

```
>> set = (hObject, 'color', [1 0 0], 'position', [10 10 100 100]);
```

Kód Matlabu 2.1 – *Příklad nastavení parametrů grafického objektu.*

2.1 Grafické objekty a jejich hierarchie

Hierarchie grafických objektů je znázorněna na obrázku 2.1 a v tabulce 2.1. Uvedeny jsou zde pouze ty objekty, které ve své práci využívám. Výpis ostatních základních grafických objektů lze nalézt v literatuře [2] společně s popisem jejich využití.



Obrázek 2.1 – *Hierarchie zde použitých základních grafických objektů Matlabu.*

Tabulka 2.1 – *Základní grafické objekty ve formě tabulky s uvedením významu.*

Jméno grafického objektu	Význam grafického objektu
Root	Vrchol hierarchie grafických objektů, odpovídá obrazovce počítače.
Figure	Okno, ve kterém se zobrazuje grafika a uživatelský interface.
Axes	Souřadný systém pro zobrazení grafů v okně **figure**.
Uicontrol	Uživatelské grafické rozhraní, jež vykoná nějakou funkci jako reakci na zásah uživatele (například stisknutí tlačítka s následnou akcí).
Uimenu	Uživatelem definované menu v horní části okna **figure**.
Uicontextmenu	Klasické (pop-up) menu, využitelné kdekoliv v Root.
Uitoolbar	Uživatelem definovaná nástrojová lišta v horní části okna **figure**.
Line	Objekt čáry, používaný při kreslení grafů.
Rectangle	2D tvar s možností změny od čtvercového po oválný.
Text	Psaní textových řetězců v rámci definované grafiky.

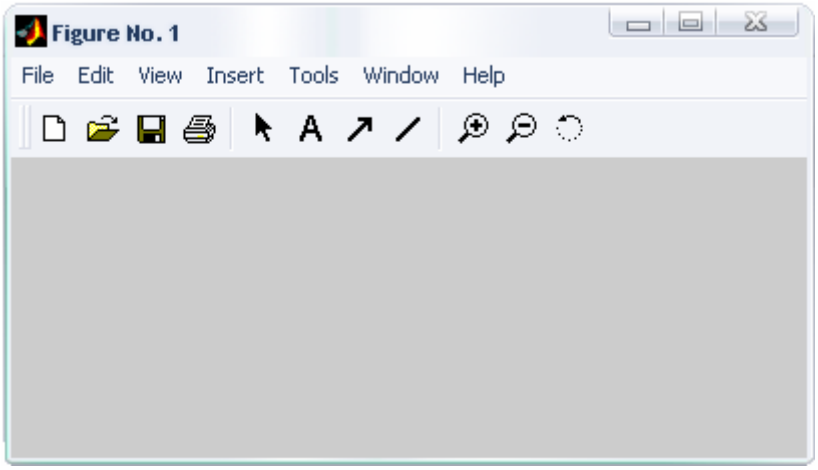
Všechny grafické objekty mají několik shodných parametrů, o kterých by bylo vhodné se zde zmínit, protože jsou v Grapheditu hojně využívány:

- **`Visible`**, **`['on','off']`** – Parametr, který nastavuje viditelnost daného objektu.
- **`Tag`**, **`[objekt typu string]`** – Označení grafického objektu, podle nějž jej lze identifikovat podobně jako pomocí handle. Toto označení nemusí být jednoznačné, což někdy může být a je žádoucí.
- **`UserData`**, **`[proměnná libovolného typu]`** – Prostor, který má uživatel k dispozici pro zachování dat, která se mohou (ale nemusí) týkat daného grafického objektu.

2.1.1 Grafický objekt figure

Grafický objekt **figure** si lze představit jako nové okénko s určitým názvem, standardním menu, nástrojovou lištou a šedou plochou, jež je připravená k definici podřízených grafických objektů. Pomocí **`set`** a **`get`** můžeme nastavovat nebo zjišťovat jeho parametry, například pozici, barvu, jméno, menu, stav lišty nástrojů a mnoho dalších, které ovlivňují nejen jeho vzhled, ale i jiné vlastnosti stejně jako případné potomky tohoto objektu.

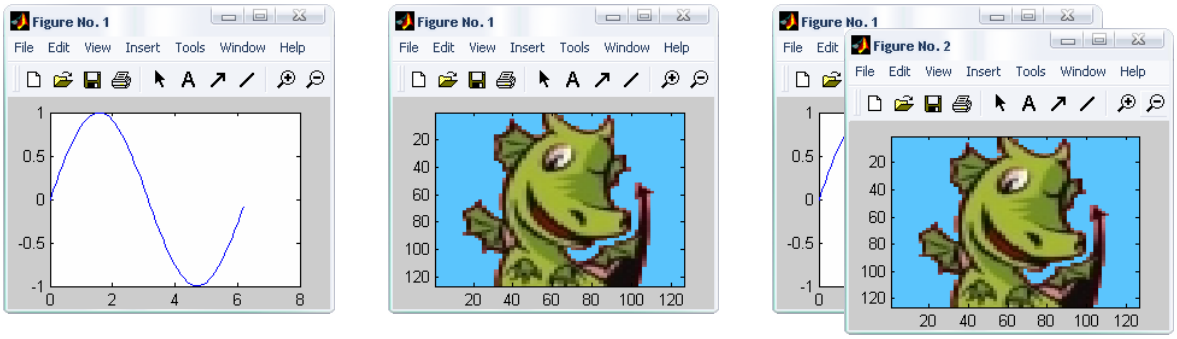
Pro práci s grafikou je velice užitečná funkce **`gcf`** (get current figure), která vrací handle aktuálního objektu **figure**. Není tedy nutné neustále udržovat handle uložený. Podobnou funkci má **`findobj`** (find object), která vrací handle objektu (nebo seznam proměnných handle několika objektů) s vlastnostmi zadanými jako parametry funkce **`findobj`** (například **`findobj('Type','figure');`**).



Obrázek 2.2 – *Odezva na vytvoření nového grafického objektu typu* ***figure***.

Jedním z parametrů objektu **figure** je ***HandleVisibility*** (viditelnost proměnné handle), který může nabývat hodnot **`on`** (výchozí), **`callback`** a **`off`**. Pokud je nastaven na **`on`**, lze handle zjistit pomocí **`gcf`** nebo **`findobj`** z celého systému Matlab, je-li nastaven na **`off`**, nelze jej funkcí **`gcf`** (a jí podobnými) zjistit, takže je „neviditelný“. V Grapheditu je nastaven na hodnotu **`callback`**. To znamená, že lze **`gcf`** použít pouze při vykonávání některé callback funkce (kapitola 2.2).

Důležitost tohoto parametru ukáži na příkladu (obrázek 2.3). Pomocí funkce **`plot`** vykreslíme sinusovku. Necháme-li ***HandleVisibility*** nezměněný a nevytvoříme-li implicitně nové okno **figure**, tak je výsledek další funkce používající **axes** (a potažmo **figure**), například **`image`**, vykreslen do již existujícího objektu **axes.** Sinusovka bude tedy překreslena novým obrázkem (obr. 2.3 b$_{)}$). Pokud však nastavíme ***HandleVisibility*** na **`off`**, vykreslí se obrázek do nového okna a graf sinusovky zůstane zachován (obr. 2.3 c$_{)}$). Lze říci, že ***HandelVisibility*** vlastně izoluje grafický objekt (**figure** či **axes**) od dalších prováděných operací, což právě u Grapheditu potřebujeme.



a$_{)}$ b$_{)}$ c$_{)}$

Obrázek 2.3 – ***a$_{)}$*** *Vykreslená sinusovka;* ***b$_{)}$*** *Obrázek byl vykreslen do* **`figure`** *1;* ***c$_{)}$*** *Pro vykreslení obrázku se automaticky vytvořil nový objekt* **`figure`**.

2.1.2 Grafický objekt uicontrol

Grafické objekty **uicontrol** jsou podřízeny objektu **figure**, na nějž jsou umisťovány. Jedná se o tlačítka (**`pushbutton`**, **`togglebutton`**), výběrová pole (**`radiobutton`**, **`checkbox`**), textová pole (**`text`**) a pole pro editovatelné texty (**`edit`**), posuvníky (**`slider`**), rámy (**`frame`**), seznamy (**`listbox`**) a pop-up menu (**`popupmenu`**). Typ určuje parametr ***Style*** zadaný při vytvoření objektu.

Protože každý grafický objekt by měl mít proměnnou handle, platí to plně také pro objekty **uicontrol**. Tento objekt vytvoříme příkazem **`uicontrol`** a pokud parametrem neurčíme jinak, bude umístěn na aktuální objekt **figure**. Podobně jako u objektů **figure**, ovládáme vlastnosti **uicontrol** pomocí **`get`** a **`set`**.

```
>> hFigure = figure('Name', 'My Figure');
>> uicontrol('Style', 'pushbutton', 'String', 'Button');
```

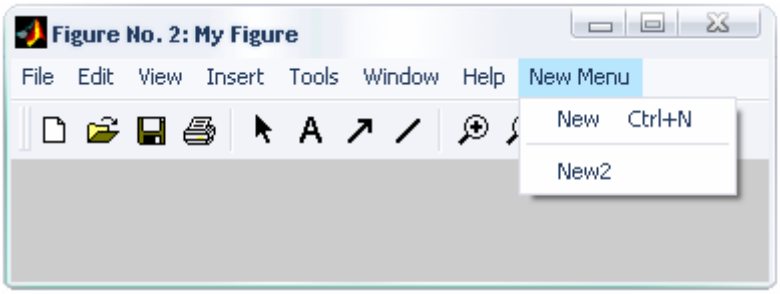
Kód Matlabu 2.2 – *Umístění tlačítka na nově vytvořené okno **figure**.*

2.1.3 Grafický objekt uimenu

Objekt **uimenu** používáme k tvorbě vlastního menu a podmenu (submenu) v oblasti řádkového menu objektu **figure**. Existuje zde i možnost použití klávesových zkratek pro vybrané položky. Pokud nezrušíme standardní menu **figure** nastavením parametru **`'Mainmenu'`** na **`'none'`**, jsou námi nově vytvořené objekty **uimenu** řazeny za něj, jak můžeme vidět na obrázku 2.4.

```
>> hFigure = figure('Name', 'My Figure');
>> hUimenu = uimenu('Label', 'New Menu');
>> uimenu(hUimenu, 'Label', 'New', 'Accelerator', 'N');
>> uimenu(hUimenu, 'Label', 'New2', 'Separator', 'on');
```

Kód Matlabu 2.3 – *Příklad vytvoření vlastního menu v hlavním menu.*



Obrázek 2.4 – *Rozbalená nabídka nového grafického objektu **uimenu**.*

2.1.4 Grafický objekt uicontextmenu

Způsob vytváření **uicontextmenu** a **uimenu** se příliš neliší. **uicontextmenu** lze pomocí funkce **`set`** přiřadit jakémukoliv jinému grafického objektu. Kliknutím pravým tlačítkem myši na tomto objektu pak kontextové menu zobrazíme.

```
>> hFigure = figure('Name', 'My Figure');
>> hContextMenu = uicontextmenu;
>> uimenu(hContextMenu, 'Label', 'New');
>> set(hFigure, 'Uicontextmenu', hContextMenu);
```

Kód Matlabu 2.4 – *Příklad způsobu vytvoření kontextového menu.*

2.1.5 Grafický objekt uitoolbar

Pokud chceme používat v objektu **figure** vlastní nástrojovou lištu, musíme standardní lištu zrušit a poté vytvořit novou. Vypnutí standardní lišty je ukázáno v kódu Matlabu 2.5. Jak můžeme vidět, provádí se obdobně jako zrušení standardního menu. Na nástrojovou lištu lze umístit dva typy tlačítek – **uipushtool** (klasické tlačítko) a **uitoggletool** (tlačítko s aretací). Na každé tlačítko můžeme umístit ikonu. Obrázek, který musí být ve tvaru matice o rozměrech **`m`**x**`n`**x**`3`**, kde **`m`** a **`n`** jsou rozměry obrázku a **`3`** je pro barvy RGB, pak tlačítku přiřadíme pomocí parametru ***CData***.

```
>> hFigure = figure('Name', 'My Figure', 'Toolbar', 'none');
>> hToolbar = uitoolbar(hFigure);
>> im = imread('picture.png');
>> uipushtool(hToolbar, 'CData', im);
```

Kód Matlabu 2.5 – *Příklad vytvoření vlastní nástrojové lišty s tlačítkem s ikonou picture.png.*

2.1.6 Grafický objekt axes

Grafický objekt **axes** hraje důležitou roli při kreslení grafů. Poskytuje nám prostor ke kreslení 2D či 3D grafů různých typů a dává nám k dispozici soubor funkcí, pomocí kterých lze grafy řídit. Stejně jako všechny ostatní grafické objekty lze jeho vlastnosti upravovat funkcemi **`set`** a **`get`**. Určení neznámého handle aktivního grafického objektu **axes** je umožněno funkcí **`gca`** (get current axes) - analogickou **`gcf`** (kapitola 2.1.1).

Stejně jako objekt **figure** má i **axes** k dispozici parametr ***HandleVisibility***, který jeho handle i handle jeho potomků ovlivňuje.

2.1.7 Grafické objekty rectangle, line a text

Příkaz **`rectangle`** vytváří 2D obdélník umístěný na aktuálním objektu **axes**. Vhodným nastavením parametru ***Curvature*** jsme schopni změnit jeho tvar na oválný, což nám dává možnost znázornit vrcholy grafů (kapitola 3.2) jako kolečka.

```
>> pos = [x y diameter diameter];
>> hRect = rectangle(hAxes, 'Position', pos, 'Curvature', [1 1]);
```

Kód Matlabu 2.6 – *Vytvoření kolečka na pozici [x,y] jako grafického objektu rectangle.*

Příkaz **`line`** vytvoří úsečku určenou dvěma body zadanými jako parametry funkce. Tyto body mohou ležet v 3D prostoru, ale pomocí Grapheditu lze kreslit pouze 2D obrázky.

```
>> hLine = line([x1 x2], [y1 y2]);
```

Kód Matlabu 2.7 – *Vytvoření grafického objektu line – úsečky mezi body [x1,y1] a [x2,y2].*

Příkaz **`text`** vytvoří objekt **text** a vykreslí jej do aktuálního objektu **axes** na pozici danou jedním bodem a horizontálním zarovnáním. Výbornou vlastností tohoto objektu je, že pro zobrazení textu je možné použít interpret LaTEXu.

```
>> hText = text(x, y, 'Text');
>> set(hText, 'HorizontalAlignment', 'center');
```

Kód Matlabu 2.8 – *Vytvoření grafického objektu text na pozici [x,y] se zarovnáním na střed.*

Parametry všech těchto objektů lze samozřejmě nastavovat a zjišťovat pomocí **`set`** a **`get`** naprosto stejně jako u všech ostatních grafických objektů postavených v hierarchii výše.

2.2 Callback rutiny grafických objektů

Každý grafický objekt má několik callback funkcí (rutin) obsluhujících určitou událost (vytvoření objektu, stisknutí tlačítka myši, apod.). Tyto callback funkce umožňují veškerou interakci GUI. V tabulce 2.2 je uveden výpis těch, které jsou v Grapheditu využívány.

Příklad uvedený v kódu Matlabu 2.9 vytvoří kontextové menu s položkami Red a Blue a zaregistruje je novému objektu **axes**. Výraz **`'set(gca,''Color'',''red'');'`** nastaví jeho barvu na červenou. Na místě uvedeného výrazu by se mohl nacházet název funkce, která by příslušnou operaci prováděla.

```
>> hContextMenu = uicontextmenu;
>> uimenu(hContextMenu, 'Label', 'Red',...
          'Callback', 'set(gca,''Color'',''red'');');
>> uimenu(hContextMenu, 'Label', 'Blue', 'Separator', 'on',...
          'Callback', 'set(gca,''Color'',''blue'');');
>> hAxes = axes('Color', 'yellow',...
                'UiContextMenu', hContextMenu,...
                'ButtonDownFcn', 'set(gca,''Color'',''yellow'');');
```

Kód Matlabu 2.9 – *Příklad registrace callback funkcí pro objekty axes a uimenu.*

Tabulka 2.2 – *Vybrané callback rutiny grafických objektů.*

Jméno parametru	Vysvětlení významu
ButtonDownFcn	Spuštěno, když na daném objektu dojde ke stisku libovolného tlačítka myši. V případě objektu **figure** musí být místo kliknutí nezakryté. *(pro všechny grafické objekty kromě **uicontrol frame**)*
CloseRequestFcn	Definuje funkci, která je spuštěna při zavolání příkazu **`close`**. *(pouze pro objekt **figure**)*
CreateFcn	Funkce, která je spuštěna při vytvoření grafického objektu. *(všechny grafické objekty)*
DeleteFcn	Funkce volaná při smazání objektu příkazem **`delete`**. *(všechny grafické objekty)*
KeyPressFcn	Rutina vykonaná při stisku klávesy nad aktivním oknem **figure**. *(pouze pro objekt **figure**)*
ResizeFcn	Funkce volaná při změně velikosti okna **figure**. *(pouze pro objekt **figure**)*
WindowButtonDownFcn	Routina vykonaná při stisknutí tlačítka myši kdekoliv na okně **figure**. A to včetně míst zakrytých jiným objektem. *(pouze pro objekt **figure**)*
WindowButtonMotionFcn	Funkce, která je prováděna při pohybu myši se stisknutým tlačítkem. *(pouze pro objekt **figure**)*
WindowButtonUpFcn	Routina vykonaná při stisknutí tlačítka myši kdekoliv na okně **figure**. *(pouze pro objekt **figure**)*
Callback	Funkce volaná při provedení akce, ke které je objekt určen. *(tlačítka, položky menu, výběrová pole,...)*
UIContextMenu	Proměnná handle odkazující na vytvořené kontextové menu. Toto menu je pak zobrazeno při stisknutí pravého tlačítka myši. *(všechny grafické objekty)*

2.3 Prostředí GUIDE

GUIDE (Graphical User Interface Development Environment) je nástroj, který má za úkol usnadnit uživateli tvorbu grafických prvků. Jedná se vlastně o automatický generátor GUI, který je v systému Matlab integrován. Po spuštění příkazem **`guide`** vede uživatele interaktivním způsobem a umožňuje mu vybírat grafické objekty, myší je umisťovat a zadávat jejich parametry. Jde o univerzální a časově nenáročné řešení.

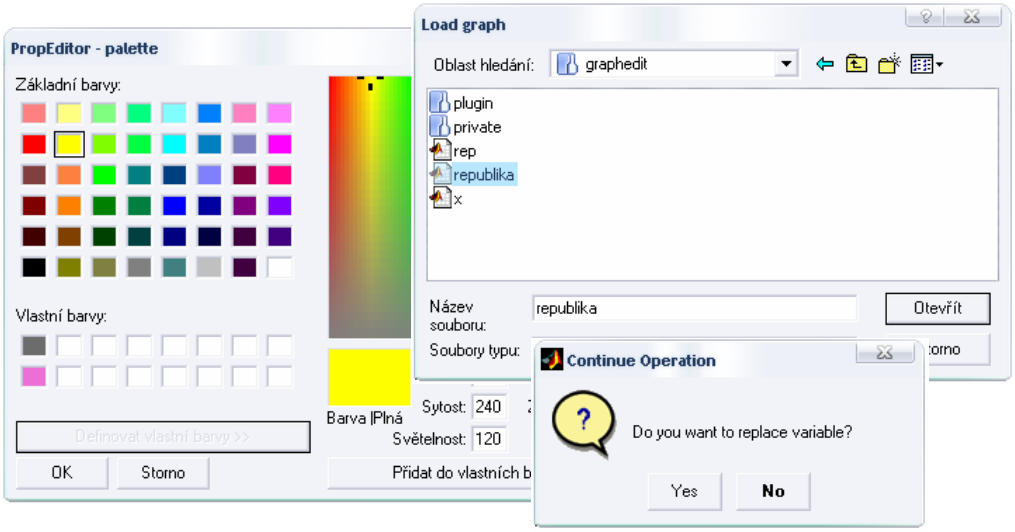
Při použití GUIDE však nalezneme jisté nevhodné vlastnosti, jak je rovněž uvedeno v literatuře [2]. Výsledný kód často z důvodu univerzálnosti není optimální, GUIDE produkuje delší zdrojové texty s odlišnou strukturou kódu, na níž si musí uživatel zvyknout. Modifikovatelnost GUI vytvořeného tímto nástrojem je podle mých zkušeností značně omezená. S výhodou jej lze použít pro vytváření jednodušších GUI, ale pro potřebu tohoto projektu je nevyhovující. Všechny grafické objekty jsem si tedy vytvářel sám přímo v editoru zdrojových textů.

2.4 Předdefinované dialogy Matlabu

Systém Matlab obsahuje řadu předdefinovaných dialogů, které velmi usnadňují implementaci mnoha funkcí. V tabulce 2.3 jsou uvedeny dialogy v Grapheditu využité.

Tabulka 2.3 – *Vybrané předdefinované dialogy použité v Grapheditu.*

Volání dialogu	Vysvětlení jeho použití
dialog	Vytvoří dialogové okno.
errordlg	Vytvoří dialogové okno s chybovou ikonou a zadanou větou.
helpdlg	Vytvoří dialogové okno s ikonou nápovědy a zadanou větou.
listdlg	Dialog pro výběr jedné nebo více položek zadaného seznamu.
questdlg	Vytvoří dialog se zadanou otázkou a tlačítky, obrázek 1.5.
uigetfile	Klasický dialog pro výběr souboru k otevření, obrázek 1.5.
uiputfile	Klasický dialog pro výběr souboru k uložení.
uisetcolor	Dialog pro výběr barvy, obrázek 1.5.
waitbar	Dialog s lištou, používaný při čekání během plnění nějaké operace.



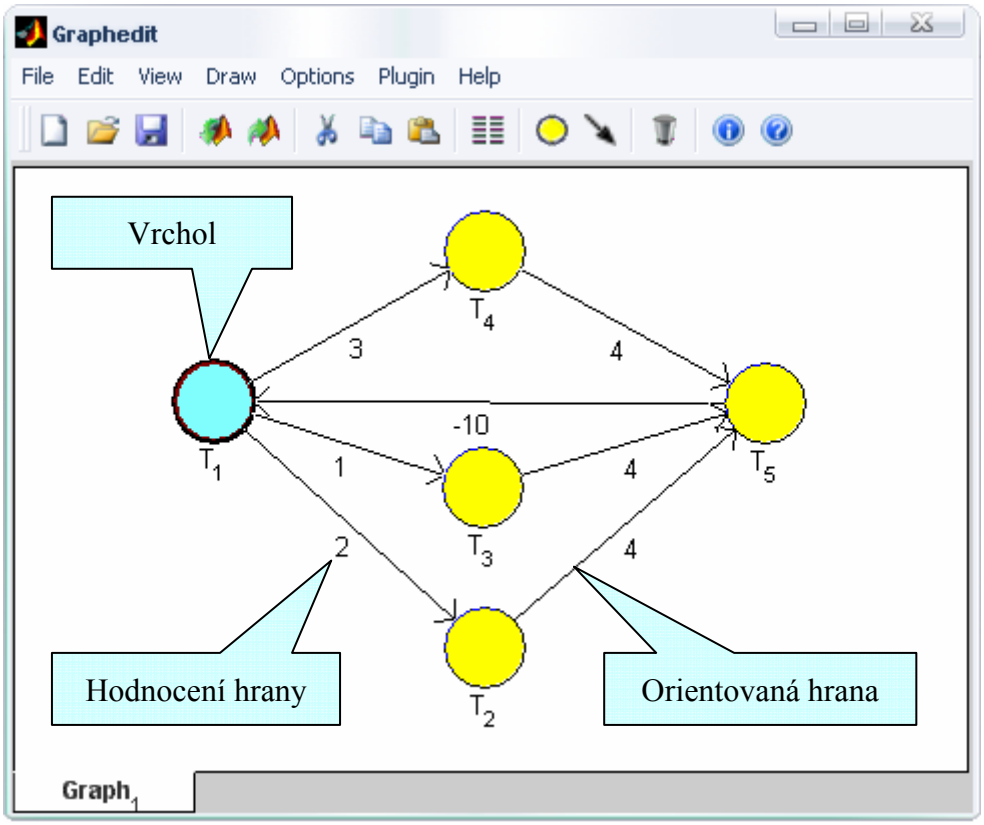
Obrázek 2.5 – *Příklady předdefinovaných dialogů v systému Matlab.*

3 Graf

Graf je základním objektem teorie grafů (literatura [1]). Je to uspořádaná dvojice (V, E), kde V je nějaká neprázdná množina a E množina některých dvojic prvků z V. Prvky množiny V se nazývají vrcholy grafu, prvky množiny E hrany grafu a mohou to být buď uspořádané nebo neuspořádané dvojice vrcholů.

Grafy lze dělit podle mnoha různých hledisek. Například podle četnosti hran na jednoduché grafy, kdy mezi libovolnými dvěma vrcholy vede nejvýše jedna hrana a multigrafy, kdy mezi dvěma vrcholy může vést libovolný počet hran. Další možné dělení je podle orientace hran a to na orientované (hrany jsou uspořádané dvojice vrcholů) a neorientované (hrany jsou dvouprvkové množiny vrcholů), podle ohodnocení na ohodnocené a neohodnocené grafy. Existují ještě další způsoby dělení, např. podle souvislosti. Jejich uvedení však nyní není nutné.

Graphedit umožňuje vytváření a práci s orientovanými jednoduchými ohodnocenými grafy, jak je ukázáno na obrázku 3.1. TORSCHE Scheduling toolbox mimo jiné zavádí objekty typu **graph**, **node** a **edge**, které Graphedit ke své funkci využívá.



Obrázek 3.1 – *Orientovaný jednoduchý ohodnocený graf v Grapheditu.*

3.1 Graf – objekt graph, objekt axes

Objekt **graph** je obecně dán maticí sousednosti (adjacency matrix). Prostřednictvím Scheduling toolboxu jej lze vytvořit příkazem **`graph`**.

```
>> adj = [0 1 0; 0 1 0; 0 0 0];
>> G = graph(adj)

  adjacency matrix:
     0     1     0
     0     1     0
     0     0     0
```

Kód Matlabu 3.1 – *Vytvoření proměnné typu* ***graph*** *se třemi vrcholy a dvěma hranami.*

Příkaz **`get`** vrací hodnotu zadaného parametru nebo hodnoty všech parametrů. Příkaz **`set`** hodnotu zadaného parametru nastaví. Přístup přes **`.`** (tečku) je rovněž možný.

```
>> set(G)
        Name: Name of the GRAPHS
           N: Array of nodes
           E: Array of edges
   UserParam: User parameters
       Notes: An arbitrary string
```

Kód Matlabu 3.2 – *Výpis parametrů proměnné typu* ***graph***.

Parametr ***Name*** je jméno grafu. Parametr ***N*** je pole všech vrcholů – objektů **node**, které graf obsahuje. Podobně ***E*** je pole hran – objektů **edge**. Objekty **node** a **edge** obsahují všechny informace a vrcholu nebo hraně. Pozice konkrétní hrany v poli je určena příkazem **`between`**. ***UserParam*** může obsahovat jakékoliv uživatelské parametry, ***Notes*** je poznámka typu string.

Kromě těchto základních parametrů byly dodatečně přidány parametry pro grafiku, které využívá Graphedit. Jedná se o ***X***, ***Y*** a ***Color***. Lze k nim prozatím přistupovat pouze přes funkce **`set_graphic_param`** a **`get_graphic_param`**.

Reprezentace objektu graph v Grapheditu

Objekt **graph** je v Grapheditu tvořen grafickým objektem **axes**, který umožňuje kreslení prvků představujících vrcholy a hrany. Kromě parametrů určujících funkci a vzhled je potřeba využít také parametr ***UserData***, který je k dispozici u každého grafického objektu a do kterého může uživatel uložit proměnnou obsahující libovolná data. V případě Grapheditu je sem ukládána struktura znázorněná kódem Matlabu 3.3.

```
>> saveStructure = struct('node', [],...
                          'edge', [],...
                          'graph', [],...
                          'flag', [],...
                          'sliders', [],...
                          'toresize', 0);
```

Kód Matlabu 3.3 – *Struktura uložená v grafickém objektu* ***axes****.*

Oddíl **`node`** je jednorozměrné pole jehož prvky obsahují proměnné handle grafického objektu představujícího vrchol – objekt **node**. V oddílu **`edge`** se nachází matice o rozměrech **`m`**x**`3`**, kde **`m`** je počet hran v grafu a **`3`** znamená počet sloupců. První sloupec obsahuje handle objektu **line**, který znázorňuje hranu, v druhém sloupci je uložen handle počátečního a v třetím koncového vrcholu hrany. Do oddílu **`graph`** patří objekt typu **graph**, který **axes** představuje, do **`flag`** proměnná handle představující příslušnou záložku (systém záložek bude vysvětlen v kapitole 4.4) a **`toresize`** slouží jako příznak pro uchování informace o změně velikosti okna Grapheditu.

3.2 Vrchol – objekt node, objekt rectangle

Objekt **node** lze vytvořit příkazem **`node`**. Podobně jako u objektu **graph** lze s jeho základními parametry (které mají stejný význam jako u objektu **graph**) pracovat pomocí funkcí **`set`** a **`get`**, případně k nim přistupovat pomocí **`.`** (tečky).

Kromě těchto základních parametrů obsahuje objekt **node** stejně jako objekt **graph** parametry pro grafiku ***X***, ***Y*** a ***Color***. ***X*** a ***Y*** určují polohu vrcholu v nakresleném grafu a mohou obsahovat pouze numerickou hodnotu, ***Color*** definuje jeho barvu zadanou číselnou hodnotou, vektorem nebo proměnnou typu string.

```
>> n = node;
>> set(n)
        Name: name of the NODE
   UserParam: user parameters
       Notes: An arbitrary string

>> set_graphic_param(n, 'X', 1, 'Y', 2, 'Color', [1 0 0]);
```

Kód Matlabu 3.4 – *Parametry objektu* ***node*** *a nastavení jeho grafických parametrů.*

Reprezentace objektu node v Grapheditu

Objekt **node** v Graphedit představuje grafický objekt **rectangle** s příslušnými parametry (výška a šířka shodné, zaoblení maximální). Tento **rectangle** má nastaveny určité vlastnosti a jako ***UserData*** uloženou strukturu s informacemi nezbytnými k funkci Grapheditu (kód Matlabu 3.5).

```
>> rectangleStructure = struct('node', {[]},...
                               'name', {[]},...
                               'userparam', {[]});
```

Kód Matlabu 3.5 – *Struktura uložená v grafickém objektu* ***rectangle****.*

Do oddílu **`node`** patří objekt **node**, který má **rectangle** představovat, do oddílu **`name`** a **`userparam`** proměnné handle, které odkazují na grafické objekty typu **text**, pomocí nichž je zobrazeno jméno vrcholu ***Name*** a uživatelské parametry ***Userparam***.

3.3 Hrana – objekt edge, objekt line

Hranu reprezentovanou objektem **edge** vytvoříme příkazem **`edge`**. Její parametry (základní i grafické) a jejich nastavování je zcela stejné jako u objektu **node**. Parametry ***X*** a ***Y*** zde nemají význam, ***Color*** určuje barvu čáry, která představuje danou hranu.

```
>> e = edge;
>> set(e)
        Name: name of the NODE
   UserParam: user parameters
       Notes: An arbitrary string

>> set_graphic_param(e, 'Color', [0 0 1]);
```

Kód Matlabu 3.6 – *Parametry objektu* ***edge*** *a nastavení grafických parametrů.*

Reprezentace objektu edge v Grapheditu

Objekt **edge** v Grapheditu představuje grafický objekt **line**. Stejně jako **rectangle** má **line** při vytvoření nastaveny určité vlastnosti a jako ***UserData*** uloženou strukturu (kód Matlabu 3.7).

```
>> lineStructure = struct('edge', {[]},...
                          'tip', {[]},...
                          'name', {[]},...
                          'userparam', {[]});
```

Kód Matlabu 3.7 – *Struktura uložená v grafickém objektu* ***line****.*

Do oddílu **`edge`** patří analogicky jako v předchozím případě objekt **edge** – hrana, do oddílu **`name`** a **`userparam`** proměnné handle, které odkazují na grafické objekty typu **text**, pomocí nichž je zobrazeno jméno hrany ***Name*** a uživatelské parametry ***Userparam***. Oddíl **`tip`** obsahuje proměnné handle odkazující na další dva objekty **line**, pomocí nichž je každá hrana znázorněna jako šipka.

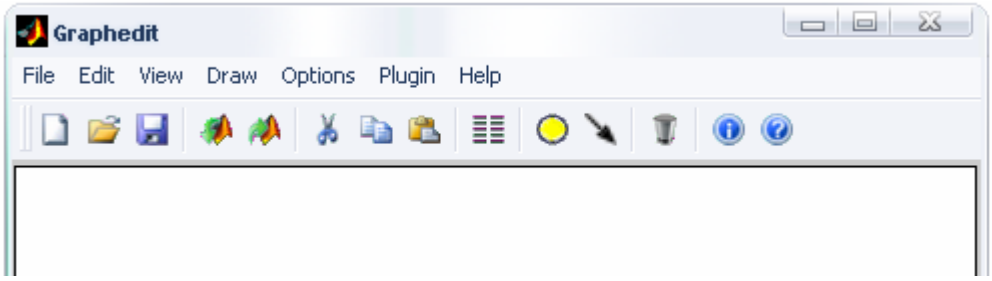
4 Graphedit

4.1 Hlavní menu

Hlavní menu zprostředkuje přístup ke všem funkcím Grapheditu. Standardní menu vytvořené společně s každým novým oknem voláním **figure** jsem nahradil vlastním, definovaným ve funkci ***`createmenu`***. Ta je volána hned při spuštění Grapheditu.

4.2 Lišta nástrojů

Některé (nejpoužívanější) funkce Grapheditu jsem kromě hlavního menu umístil také na lištu nástrojů. Lze na ní nalézt ikony pro vytvoření nového listu, otevření souboru, uložení grafu do souboru, import grafu z workspace Matlabu, export nakresleného grafu do workspace, ikony pro vyjmutí, kopírování a vložení vrcholu, zobrazení či skrytí property editoru, pro přepínání kreslicího módu (kreslení vrcholů, hran, mód mazání) a nakonec typické ikony pro zobrazení nápovědy a dialogu „O aplikaci“.



Obrázek 4.1 – *Lišta nástrojů Grapheditu.*

Při realizaci panelu nástrojů spočíval největší problém v zobrazování ikon. A to především problém s průhledností nepokreslených ploch. Formát zvolený pro uchování obrázků PNG je sice schopen nést informaci o průhlednosti, ale samotná tlačítka lišty zobrazují data zadaná jako parametr ***Cdata***. Sem jsme schopni uložit pouze matici – v případě obrázku `m`x`n`x`3`, kde `m` a `n` jsou jeho rozměry a `3` představuje tři základní barvy. To znamená, že pro použití průhlednosti zde není prostor. Řešení, jež jsem použil, je nahradit místo v obrázku ikony, které by mělo být průhledné, barvou pozadí lišty. Abych mohl průhledná místa snadno identifikovat, použil jsem pro ně při kreslení obrázků růžovou barvu. Při načítání ikon ji nahradím barvou lišty určenou nastavením operačního systému. Tu získám

příkazem `get(0,'factoryUicontrolBackgroundColor')`. Tímto způsobem sice získám barvu pozadí objektů **uicontrol**, ale v operačních systémech Windows by tato barva měla být shodná s barvou pozadí **uitoolbar**. Celá tato operace poněkud zpomaluje spuštění Grapheditu, ale poskytuje mu komfortnější vzhled.

Vytvoření lišty nástrojů společně s načtením ikon je provedeno funkcí ***`createtoolbar`***, vytvoření iluze transparentnosti pak funkcí ***`getcdata`***.

Obrázky ikon se stejně jako všechny ostatní oddělené součásti Grapheditu nacházejí v adresáři **private**.

4.3 Kreslicí plocha

Základ celého Grapheditu tvoří kreslicí plocha představována grafickým objektem **axes**. Na něj je možné umístit vrcholy a hrany ve formě objektů **rectangle** a **line**.

4.3.1 List kreslicí plochy

Každý list kreslicí plochy je vytvořen funkcí ***`createcanvas`***. V ní se nastaví všechny potřebné parametry včetně callback funkce ***`buttondownoveraxes`***, která se provede po stisknutí tlačítka myši. Rovněž je mu přiřazeno kontextové menu nadefinované již při spuštění Grapheditu prostřednictvím funkce ***`createcontextmenu`***.

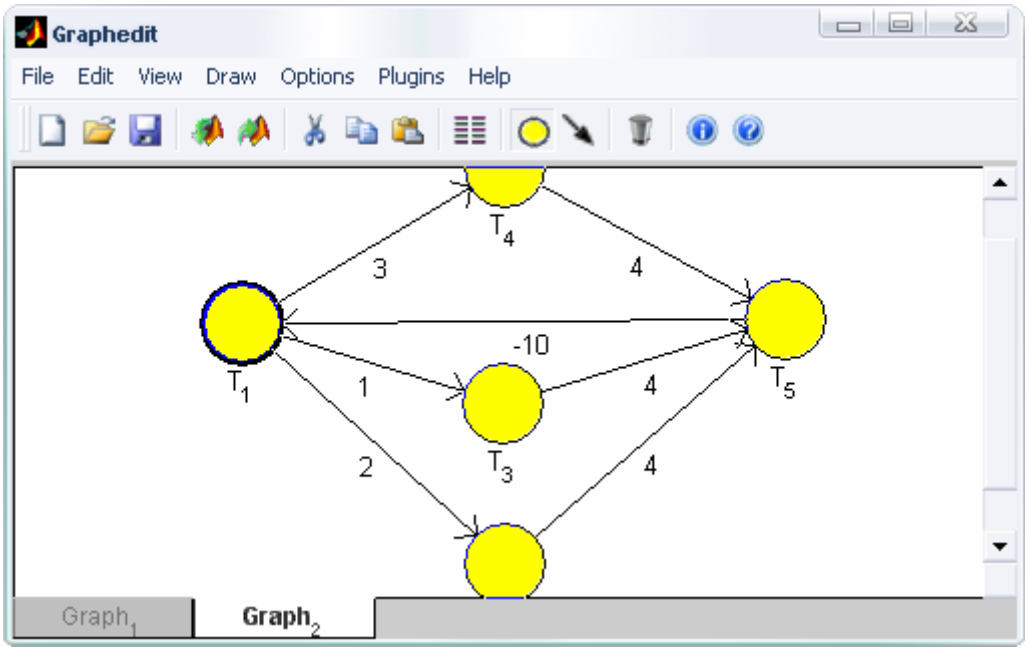
Poté je vytvořena struktura (objekt **struct**), v níž jsou uloženy potřebné parametry. Skládá se z oddílů **`node`**, **`edge`**, kam se ukládají kreslené vrcholy a hrany, oddílu **`graph`**, v němž je uložen kvůli jednotlivým parametrům objekt typu **graph**, oddílu **`flag`**, který uchovává handle na příslušnou záložku, oddílu **`sliders`**, kam se ukládají proměnné handle posuvníků určených k ovládání kreslicí plochy v případě potřeby a oddílu **`toresize`**, jenž slouží jako příznak pro uchování informace o změně velikosti okna Grapheditu.

Zavoláním funkce ***`addflag`*** s příslušnými parametry je vytvořena i záložka určená tomuto novému listu. Systém záložek je podrobně vysvětlen níže.

V případě, že je nový list vytvářen pro již existující graf, například při importu z workspace, je volána funkce ***`drawobjectgraph`*** (vývojový diagram 6.1), která má jako jeden z parametrů požadovaný graf a vykreslí jej.

4.3.2 Posuvníky

Nakreslený graf může zabírat větší plochu, než je velikost okna Grapheditu. Proto je nezbytná možnost posuvu zobrazované části **axes**. To lze provést s použitím posuvníku (**slider**) patřícího mezi prvky **uicontrol** v Matlabu.



Obrázek 4.2 – *Graf je příliš velký pro jeho zobrazení – použití objektu* ***slider***.

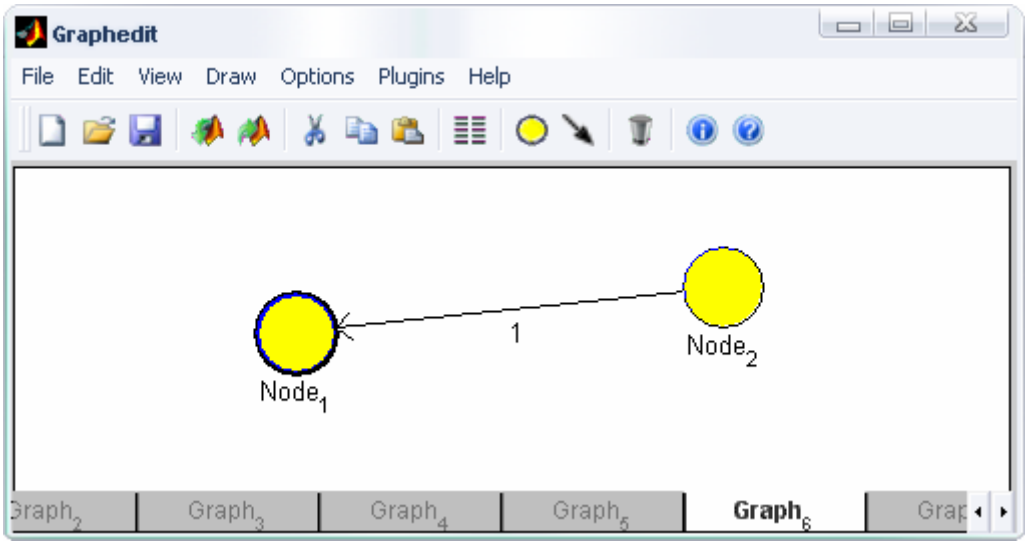
V situaci, kdy celkové rozměry zobrazeného grafu nepřesahují velikost kreslicí plochy, nejsou posuvníky potřeba, a jsou tudíž skryty. Pokud však vertikálně či horizontálně graf přesahuje, zobrazí se a umožní tak posuv v příslušném rozměru. V pravém dolním rohu kreslicí plochy, kde se oba posuvníky sbíhají, vzniká malý čtverec nevyužitého místa. Do něj jsem umístil tlačítko, které má funkci vycentrování grafu na střed (nastavuje aktuální hodnotu objektů **slider** na polovinu).

Použití posuvníků poněkud komplikuje text zobrazený na kreslicí ploše. Při jeho posuvu i mimo plochu **axes** je totiž narozdíl od ostatních použitých grafických objektů stále viditelný. To rozhodně není žádoucí vlastnost, proto je potřeba zajistit zneviditelnění a zviditelnění textu samostatně. To je ošetřeno ve funkci ***setpositiontext***.

4.4 Záložky

Protože jedna jediná kreslicí plocha neposkytuje příliš mnoho prostoru k práci, je možné využívat větší počet listů najednou. Tím se objevuje nutnost výběru každého z nich, vytvořil jsem proto systém záložek pro přepínání mezi jednotlivými plochami.

Realizoval jsem jej dalším objektem **axes** vytvořeným funkcí ***`createaxesbar`*** a umístěným v dolní části okna Grapheditu. Barva jeho výplně i obrysu odpovídá barvě vnitřku okna a není tedy vizuálně rozeznatelný. Je nezbytné počítat s možností tak vysokého počtu záložek, že pro jejich zobrazení nebude šířka Grapheditu stačit. Z toho důvodu jsem do pravého dolního rohu okna umístil objekt **slider**, pomocí nějž je snadné měnit zobrazenou část **axes** (funkce ***`slideraxesbarchanged`***) a umožňuje tedy poměrně efektní posuv záložek.



Obrázek 4.3 – *Využití záložek, zobrazení objektu* ***slider*** *pro jejich posuv.*

Jednotlivé záložky jsou tvořeny objekty **rectangle** a **text**. **Rectangle** zde představuje obdélník – tvar záložky a **text** zobrazuje název příslušného kreslicího listu – jméno grafu. Každá z nich je úzce svázána s příslušným listem pomocí číselného údaje handle uloženého jako ***UserData***.

Při vytvoření záložky funkcí ***`addflag`*** (ta je volána při vytváření nového listu) je obdélníku (mimo jiné) přiřazena callback funkce, která se vykoná při kliknutí a která umožňuje přepínání mezi jednotlivými listy, případně zobrazení kontextového menu záložky. A protože text se jménem grafu překrývá část záložky, je pro něj nadefinována stejná callback funkce. Její jméno je ***`buttondownoverflag`***.

Každé záložce je přiřazeno kontextové menu, které se zobrazí, tak jak je obvyklé, po kliknutí pravým tlačítkem myši. Jeho jedinou položkou **Close** lze zavřít příslušný list naprosto stejně jako pomocí hlavního menu.

Pro správné zobrazování záložek také při odstranění některé z nich je nutné srovnání zbývajících. To zajišťuje funkce ***setpositionflags*** volaná při odstranění záložky (funkce ***removeflag***).

4.5 Property editor – Editor vlastností

Property editor je nástroj umožňující editování parametrů grafových objektů – vrcholů, hran a grafů. Výčet těchto parametrů je uveden v tabulkách 4.1, 4.2, 4.3.

Tabulka 4.1 – *Parametry objektu* ***node***

Vrchol – node	
Name	Jméno vrcholu
UserParam	Uživatelské parametry
Notes	Poznámka
Color	Barva
X	Pozice v ose x
Y	Pozice v ose y

Tabulka 4.2 – *Parametry objektu* ***edge***

Hrana – edge	
Name	Jméno hrany
UserParam	Uživatelské parametry
Notes	Poznámka
Color	Barva
X	Nemá význam
Y	Nemá význam

Tabulka 4.3 – *Parametry objektu* ***graph***

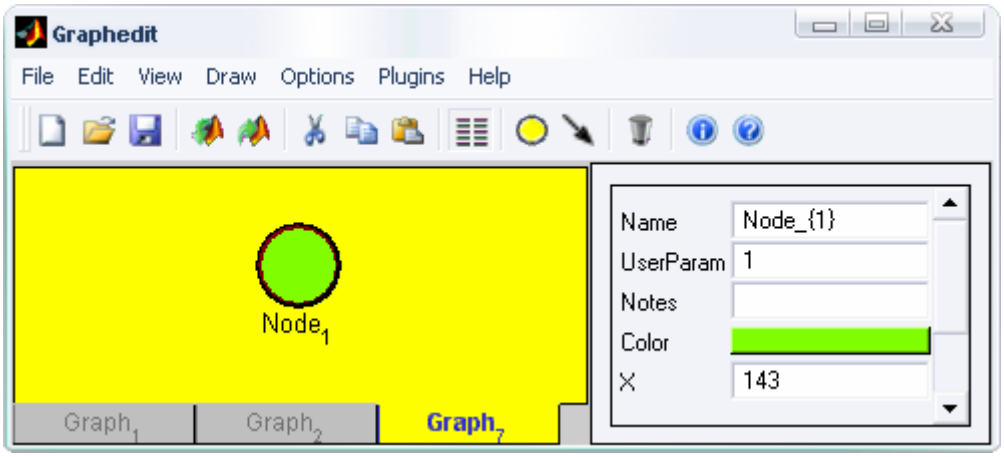
Graf - graph			
Name	Jméno grafu	***Notes***	Poznámka
N	seznam vrcholů (cell array)	***Color***	Barva
E	seznam hran (cell array)	***X***	Využívám pro uložení rozměrů grafu
UserParam	Uživatelské parametry	***Y***	

U všech tří objektů je možné editovat ***Name***, ***Notes*** (což jsou objekty typu string) a ***UserParam*** (může obsahovat libovolná data). ***Color*** je zadána formou vektoru barev. U vrcholu lze pozici danou souřadnicemi ***X***, ***Y*** nastavit zadáním číselných hodnot. Pro hrany ani grafy tato možnost zpřístupněna není. Stejně tak nelze prostřednictvím property editoru pracovat s parametry grafů ***N*** a ***E***.

Realizace property editoru prošla poměrně složitým a zdlouhavým vývojem. Zkoušel jsem několik variant jeho umístění a podoby. Velkou inspirací mi byl Property inspector, kterým lze editovat vlastnosti grafických objektů při použití nástroje Matlabu – GUIDE.

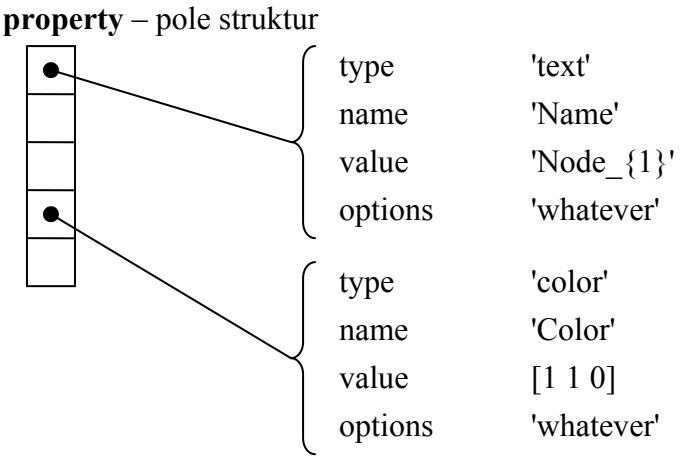
Jak je vidět na obrázku 4.4, rozhodl jsem se pro pozici v levé části základního okna Grapheditu. Jeho základ tvoří dva grafické objekty **frame**, které patří mezi prvky **uicontrol**.

V novějších verzích Matlabu existuje objekt **panel**, který by byl k tomuto účelu mnohem vhodnější, protože je nadřízený všem prvkům **uicontrol** a umísťování editačních i textových polí a především možnost posuvu pomocí objektů **slider** by bylo umožněno jednodušším a pravděpodobně méně systémově náročným způsobem. Z důvodu požadované kompatibility Grapheditu s verzí Matlabu 6.5 R13 však není možné vlastností objektu **panel** využít.



Obrázek 4.4 – *Zobrazený property editor i s použitým posuvníkem* ***slider***.

Pro vypsání jednotlivých parametrů je nejdříve pomocí funkce ***`grapheditdictionary`*** umístěné v samostatném souboru v adresáři **private** získáno pole struktur, z nichž každá obsahuje informace o jednom z parametrů vybraného objektu (vrcholu, hrany nebo grafu).



Obrázek 4.5 – *Znázornění struktur s příklady parametrů.*

Oddíl struktury **`type`** určuje, o jaký typ parametru se jedná. Možnosti, kterých může nabývat, včetně jejich vysvětlení, jsou zobrazeny v tabulce 4.4. Oddíl **`name`** určuje pojmenování parametru, které bude zobrazeno jako statický text vedle editačního pole. **`Value`**

je hodnota konkrétního parametru u vybraného objektu. **`Options`** je v současnosti nevyužitý údaj určený pro uchování možností například pro pop-up menu, které by bylo možné v property editoru rovněž využít. Stávající parametry grafů, vrcholů ani hran však jeho použití nevyžadují. Uvedený string **`'whatever'`** tedy znamená, že pro zobrazení v editoru nebude použito pop-up menu.

Tabulka 4.4 – *Tabulka typů parametrů zobrazovaných v property editoru.*

Typ parametru	Vysvětlení významu	Použitý objekt Uicontrol
coordinates	souřadnice	editovatelný text
coordinateslocked	needitovatelné souřadnice	editovatelný text (nepřístupný)
text	text	editovatelný text
textlocked	needitovatelný text	editovatelný text (nepřístupný)
color	barva ve složkách RGB	obarvené tlačítko
other	jiný údaj	editovatelný text
otherlocked	needitovatelný jiný údaj	editovatelný text (nepřístupný)

Při pohledu na zdrojový kód funkce ***`grapheditdictionary`*** stejně jako na kód zajišťující nastavení nově zadaných hodnot parametrů zpět vybranému objektu (v souboru grapheditpropertyeditor.m) se zřejmě nabízí otázka: „Proč je v podstatě každý parametr ošetřen zvlášť?“

Toto poměrně neuniverzální řešení vyžaduje stav, ve kterém se nachází grafová část Scheduling toolboxu. Parametry grafových objektů uvedené v tabulkách 4.1, 4.2, 4.3 označené okrovou barvou jsou parametry základní. Lze s nimi pracovat pomocí funkcí **`set`** a **`get`** a je mimo jiné možné snadno získat jejich výpis. Zde by bylo možné jednotné zpracování. Problém je s parametry označenými zelenou barvou. Jedná se o parametry grafické, dodatečně přidané, nicméně funkční. Práce s nimi je prozatím prováděna funkcemi **`set_graphic_param`** a **`get_graphic_param`** a je poněkud obtížnější než u základních parametrů.

Pro případ zmenšení základního okna na velikost, kdy není možné zobrazit všechny řádky property editoru, musí existovat možnost jejich rolování. Pokud bych měl k dispozici jako mateřský prvek editoru **panel**, byla by situace jednodušší, ale je řešitelná i bez něj. Jak je na obrázku vidět, umístil jsem do pravé části vnitřního rámu **frame slider**.

Ten je zobrazen pouze v případě, kdy má jeho přítomnost smysl. Posuv parametrů nahoru či dolů, a tedy jejich zviditelnění a zneviditelnění, musí být prováděno po celých řádcích. Při

změně hodnoty nastavené posuvníkem se zjistí nová pozice text-boxů i edit-boxů (v případě barvy tlačítka) a podle ní je rozhodnuto o nastavení parametru ***Visible***, poté je nová pozice nastavena. Toto pořadí je nutné kvůli estetickému zobrazení GUI. Při pořadí opačném totiž grafické objekty, které již nemají být vidět, „probliknou“.

Samotná editace je prováděna jednoduše zapsáním nové hodnoty do textového pole. V případě tlačítka pro nastavení barvy se po jeho stisknutí zobrazí dialog **`uisetcolor`** (jeho vzhled je závislý na konkrétním operačním systému), který vrací zvolenou barvu v požadovaném tvaru.

Parametry ***Name*** a ***Notes*** mohou nabývat hodnoty znakové, číselné, či jiné – v důsledku jde vždy o typ string. ***UserParam*** může být v podstatě libovolná hodnota. Při jejím nastavování se pomocí funkce **`eval`** zadaný řetězec vykoná. V budoucnu však bude potřeba nahradit tento způsob způsobem vhodnějším. Parametr ***Color*** je nastavován pomocí výše zmíněného dialogu. Při zadávání souřadnic ***X***, ***Y*** v případě vrcholu musí být zadána číselná hodnota. Zadávání parametrů je ošetřeno s využitím konstrukce **`try-catch`**.

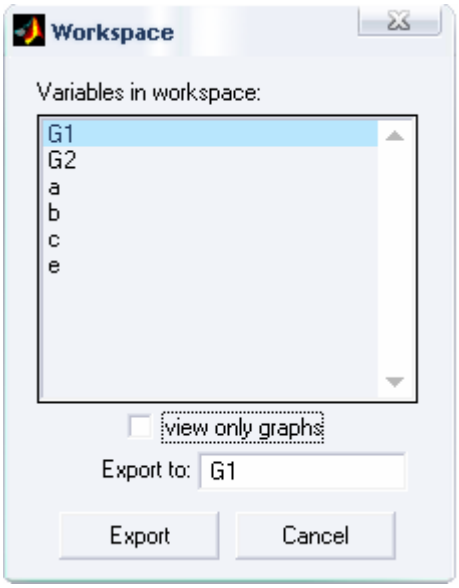
5 Dialogy

5.1 Grapheditlistdlg

Funkce ***graphedit listdlg*** vytvoří dialogové okno, které bylo původně určené pro zobrazení existujících proměnných ve workspace Matlabu při importu nebo exportu grafu. Její využití jsem však v průběhu vývoje rozšířil a nyní je používána také pro načítání a ukládání objektu typu **graph** z/do binárního souboru *.mat a pro umožnění přejmenování vrcholu nebo hrany nakresleného grafu (tuto funkci umožňuje také property editor – kapitola 4.5). Rovněž jsem ji zakomponoval do systému zásuvných modulů (kapitola 7). Zde je zavolána v případě, kdy uživatelem definovaný plugin nevrací objekt **graph**, jenž by se pouze vykreslil, ale jiný typ proměnné určené k uložení do workspace (použití je tedy podobné jako při exportu nakresleného grafu).

Jako vzor mi posloužila funkce **`listdlg`**, která je součástí Matlabu. Ta je sama o sobě pro mé účely nevyhovující, ale některé zkušenosti získané při pečlivém nastudování jejího zdrojového kódu jsem využil u všech dialogů, které jsem později vytvářel.

Nejdůležitější částí dialogu je **listbox** sloužící pro zobrazení seznamu jmen (proměnných, vrcholů, hran). K němu patří **checkbox**, jehož stav určuje zobrazený seznam. Jako vstup funkce lze totiž zadat dvě odlišná pole **`cell`**, která obsahují jména. Z důvodu vhodné volby nového jména je důležité umožnit uživateli nejen zobrazení jmen například objektů **graph**, ale rovněž ostatních proměnných ve workspace či v souboru. Dalším prvkem je **uicontrol edit** pro zadání požadovaného jména.



Obrázek 5.1 – *Dialog* ***graphedit listdlg***

Důležité je, že všechny texty v dialogu včetně popisu tlačítek lze měnit zadáním příslušných parametrů při volání funkce.

Grapheditlistdlg vrací proměnnou **`name`**, v níž je jméno ve formě řetězce, a proměnou **`isNewName`**, která určuje, zda se jedná o jméno nové (**`isNewName`** = 1) nebo zda jde pouze o položku vybranou ze seznamu (**`isNewName`** = 0).

Tabulka 5.1 – *Výpis parametrů funkce* ***graphedit listdlg***.

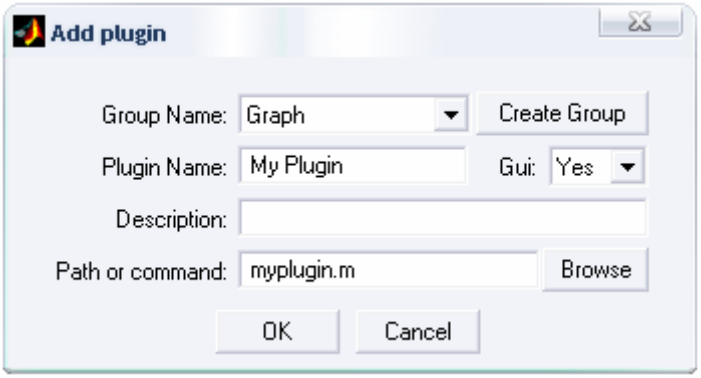
Název	Výchozí	Stručné vysvětlení významu
parentFigure	`[]`	**figure** handle
currentAxes	`[]`	**axes** handle
windowStyle	`'modal'`	styl dialogu `['normal','modal']`
position	`'center'`	pozice dialogu vzhledem k rodiči `['center','rightup','rightdown','leftup', 'leftdown']`
list1	`{}`	cell array se jmény
list2	`{}`	cell array se jmény
listHandles1	`[]`	vektor handlů korespondujících s list1
listHandles2	`[]`	vektor handlů korespondujících s list2
initialValue	`1`	vybraná pozice v seznamu
initialName	`'g'`	zvolené výchozí jméno `[string]`
okString	`'Ok'`	popis tlačítka Ok `[string]`
cancelString	`'Cancel'`	popis tlačítka Cancel `[string]`
fileName	`''`	jméno dialogu (například jméno souboru) `[string]`
editEnable	`'on'`	zpřístupnění edit-boxu `['on','off']`
editName	`''`	text návěští edit-boxu `[string]`
listEnable	`'on'`	zpřístupnění list-boxu `['on','off']`
listName	`''`	text návěští list-boxu `[string]`
checkEnable	`'on'`	zpřístupnění přepínání mezi seznamy jmen `['on','off']`
checkText	`''`	text návěští checkboxu `[string]`
replaceable	`'on'`	možnost nahrazení již existujícího jména `['on','off']`
uniqueName	`'on'`	jsou dovolena pouze unikátní jména `['on','off']`
questString	`'Do you want to replace name?'`	otázka při nesrovnalosti po stisku Ok tlačítka `[string]`

Z uvedených parametrů není možná význam ***listHandles1*** a ***listHandles2*** zcela jasný. Jsou využity v případě použití dialogu pro přejmenování vrcholů a hran. K jejich zavedení mne vedl požadavek na zvýraznění objektu na kreslicí ploše při výběru jeho jména. Pořadí handlů v ***listHandels1***,***2*** koresponduje s pořadím jmen v ***list1***,***2***. Takže při výběru určitého jména (na známé pozici v seznamu) lze pomocí příslušného handle přesně adresovat odpovídající grafický objekt a dále s ním pracovat.

Domnívám se, že funkci ***graphedit listdlg*** činí způsob jakým funguje poměrně univerzální a využitelnou možná i pro jiné aplikace než pouze pro Graphedit.

5.2 Grapheditaddplugin

Tento dialog slouží k přidání nového zásuvného modulu (pluginu), který bude pracovat s nakresleným grafem. Jako jediný vstupní parametr funkce je zadána struktura (objekt typu **struct**) obsahující informace o současných pluginech. Ty jsou při spuštění Grapheditu načteny z příslušného **xml** souboru. Po nadefinování všech příslušných údajů je pak vrácena vstupní struktura doplněná o nový plugin a určená k uložení.

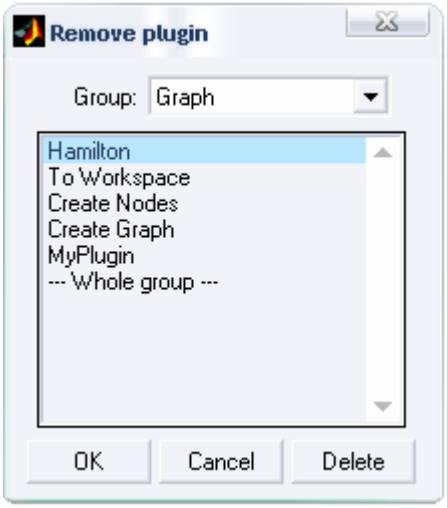


Obrázek 5.2 – *Zobrazení dialogu* ***graphedit addplugin***.

Prostřednictvím pop-up menu označeném "Group Name" uživatel zvolí jméno skupiny pluginů, do které chce nový přidat. Pokud mu žádná existující skupina nevyhovuje, lze vytvořit novou po kliknutí na tlačítko "Create Group". Nezbytné je zadání jména nového pluginu, pod kterým jej bude možné nalézt v hlavním menu Grapheditu. Zda bude plugin ke svému spuštění využívat dialogové okno ***graphedit doplugin*** nebo ne, je určeno volbou v pop-up menu označeném "Gui". Edit-box "Description" slouží k zadání nějakého bližšího vysvětlení tohoto nového pluginu. To není povinné a záleží pouze na uživateli, zda jej uvede. Vysvětlení se zobrazí v dialogu ***graphedit doplugin*** (kapitola 5.4) nebo si je lze přečíst alespoň po zobrazení xml souboru **grapheditplugin.xml**. Velice důležitý údaj je označen "Path or Command". Sem je potřeba zapsat absolutní cestu k dané funkci, kterou lze získat pomocí tlačítka "Browse", nebo název této funkce. Druhou možnost lze využít jedině v případě, kdy se funkce nachází v definovaných cestách Matlabu nebo v adresáři **plugin**, který Grapheditu přísluší (blíže popsáno v kapitole 7).

5.3 Grapheditremoveplugin

Pomocí tohoto dialogu lze vybrat a odstranit plugin nebo celou skupinu. Vstupním parametrem funkce ***grapheditremoveplugin*** je stejně jako u ***grapheditaddplugin*** struktura s informacemi o pluginech. Návratovou hodnotu pak tvoří struktura upravená.

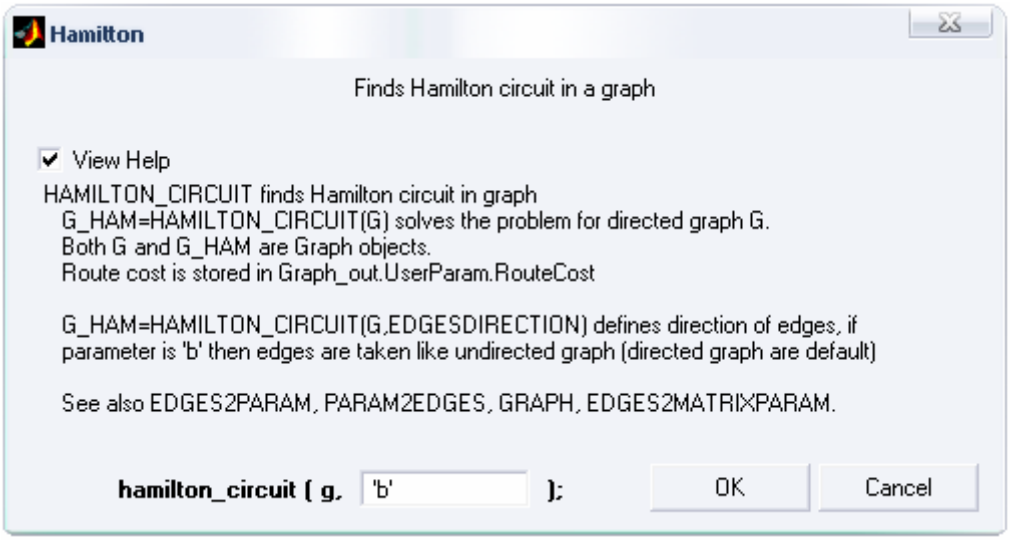


Obrázek 5.3 – *Zobrazení dialogu* ***grapheditremoveplugin***.

Uživatel v pop-up menu "Group" vybere určitou skupinu, po té v list-boxu označí plugin, který chce odstranit. Stisknutím tlačítka "Delete" nebo double-klikem na pluginu jej vymaže. Pokud vybere "--- Whole group ---", smaže celou vybranou skupinu. Změny se provedou až po stisknutí tlačítka "OK". Uživatel tedy může najednou vymazat několik položek.

5.4 Grapheditdoplugin

Grapheditdoplugin má za úkol sloužit jako univerzální rozhraní pro spuštění libovolné funkce zařazené mezi pluginy. Prvním vstupem je struktura **plugin** obsahující název pluginu, absolutní cestu k funkci, která jej představuje, nalezenou při spuštění Grapheditu nebo po přidání pluginu, stručný popis jeho činnosti, pokud byl při definování pluginu zadán. Druhým vstupním parametrem je pak graf zobrazený na aktuální kreslicí ploše – na objektu typu **axes**. Pokud je kreslicí plocha prázdná, je tento parametr prázdný vektor. Návratovou hodnotou funkce ***grapheditdoplugin*** je nová proměnná vytvořená pluginem (a to i v případě, kdy plugin nic nevrací – pak je touto hodnotou prázdný vektor).



Obrázek 5.4 – *Zobrazení dialogu* ***grapheditdoplugin***.

Na obrázku je vidět použití ***grapheditdoplugin*** s pluginem **Hamilton**, který volá funkci **hamilton_circuit**. V horní části dialogu je zobrazen popis pluginu. Pokud má funkce pluginu nápovědu, je umožněno její zobrazení. V dolní části je kromě tlačítek "OK" a "Cancel" zobrazen text, jenž znázorňuje zadání funkce podobně jako v příkazové řádce Matlabu. Jako první parametr je předán graf **g** a pro volitelné zadání dalších parametrů je určen edit-box.

Protože může existovat mnoho typů pluginů, nejsme schopni vytvořit jeden univerzální dialog tak, aby umožňoval komfortnější ovládání. ***Grapheditdoplugin*** je prosté, ale funkční řešení. Je předpokládáno, že pokud bude uživatel pro svůj plugin potřebovat lepší prostředí, sám si je vytvoří (kapitola 7).

6 Funkce

Vzhledem k tomu, že všechny funkce, které jsou v Grapheditu k dispozici, jsou přístupné z hlavního menu, popíši je právě tak, jak jsou v menu seřazeny. Nejdříve ze všeho však uvedu způsob vytvoření hlavního okna Grapheditu při jeho spuštění.

6.1 Spuštění Grapheditu

Spuštění je provedeno zadáním příkazu ***graphedit*** do příkazového řádku v Matlabu. Jako parametr může mít tato funkce zadanou proměnnou obsahující objekt typu **graph**. V takovém případě jej po spuštění okamžitě vykreslí. Dalším parametrem lze rozhodnout, zda bude okamžitě po spuštění zobrazen property editor či ne parametrem **on**, **off**.

Primární funkce ***graphedit*** pouze rozhoduje o volání následující funkce podle podmínek, které splňují zadané parametry. Při spuštění je volána funkce ***createfigure*** s parametry určujícími vzhled nového okna. Zadáním **figure** je vytvořeno nové okno, které má jako callback rutinu ***CreatFcn*** funkci ***figurecreatefcncallback***. Ta nejdříve ověří, zda již není Graphedit spuštěn (je dovolena jedna instance) a poté postupně spustí ***createmenu*** (nadefinuje hlavní menu), ***createtoolbar*** (nadefinuje lištu nástrojů), ***createcontextmenu*** (nadefinuje všechna kontextová menu používaná v Grapheditu), ***createaxesbar*** (vytvoří lištu pro záložky tvořenou objektem **axes**), ***createcanvas*** (vytvoří list kreslicí plochy) a pokud je to vyžadováno, zobrazí property editor.

6.2 File

6.2.1 New

Kliknutí na položku "New" vytvoří nový kreslicí list pomocí funkce ***createcanvas*** (diagram 6.5). Ta může mít jako jeden za svých parametrů objekt graph (nyní však prázdný).

6.2.2 Open

Volá funkci ***opengraphfromfile*** (diagram 6.3), která umožní vykreslit objekt **graph** uložený v binárním souboru *.mat. Protože v souboru může být uloženo více proměnných tohoto typu, je zde využito dialogu ***grapheditlistdlg*** pro výběr konkrétního grafu.

Ve vývojovém diagramu funkce ***opengraphfromfile*** je zvýrazněný blok pro vykreslení grafu pomocí funkce ***drawobjectgraf***. Algoritmus je v diagramu 6.1.

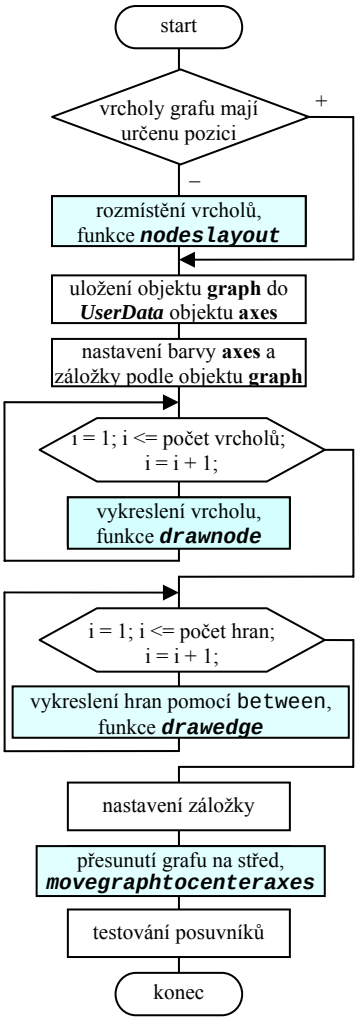


Diagram 6.1 – *Funkce* ***drawobjectgraph***

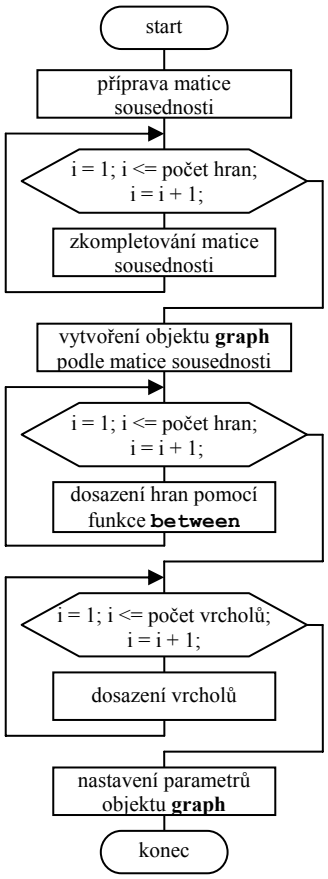


Diagram 6.2 – *Funkce* ***createobjectgraph***

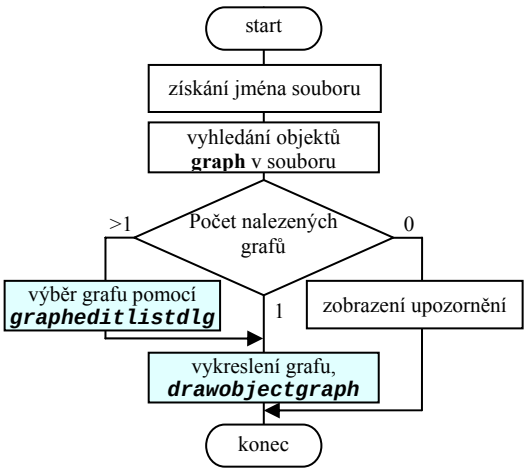


Diagram 6.3 – *Funkce* ***opengraphfromfile***

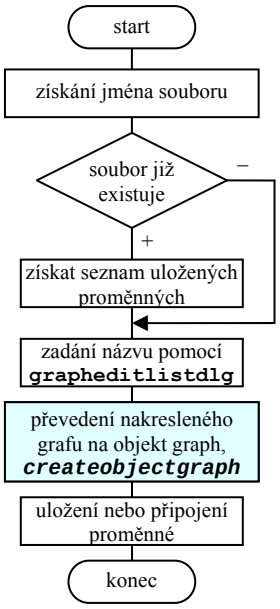


Diagram 6.4 – *Funkce* ***savegraphtofile***

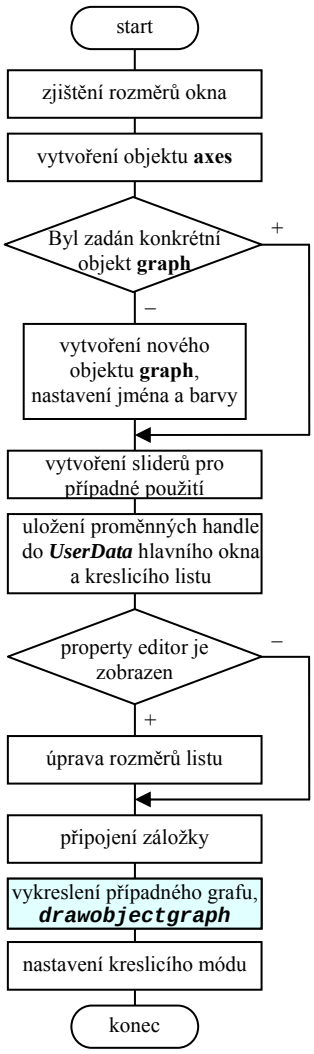


Diagram 6.5 – *Funkce* **`createcanvas`**

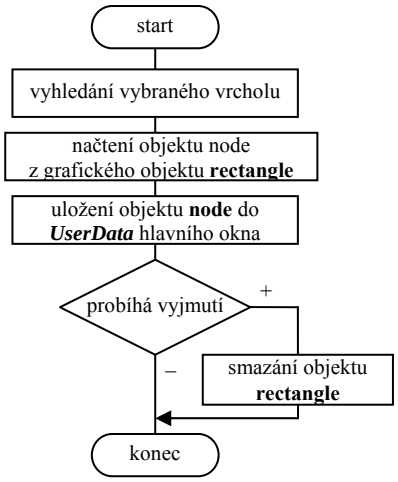


Diagram 6.6 – *Funkce* **`copynodefcn`**

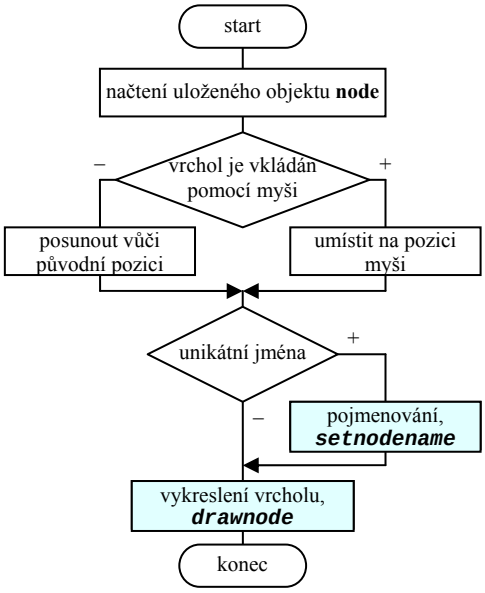


Diagram 6.7 – *Funkce* **`pastenodefcn`**

6.2.3 Save

Zavoláním ***`savegraphtofile`*** (diagram 6.4) je umožněno uložení grafu nakresleného na aktuálním kreslicím listu do binárního souboru *.mat. V případě, že soubor neexistuje, je vytvořen, v případě, že již existuje, je do něj proměnná obsahující tento graf přidána. I zde je použit dialog ***`graphedit listdlg`***, tentokrát pro zadání či výběr jména proměnné, do níž bude graf uložen.

V uvedeném vývojovém diagramu je zvýrazněný blok převedení nakresleného grafu na objekt typu **graph**. To se zde nachází kvůli tomu, že při kreslení není upravován přímo objekt **graph** uložený ve struktuře uchované v ***UserData*** daného grafického objektu **axes**, ale pouze obrázek, který jej vizuálně představuje. K vytvoření grafu jako objektu **graph** Scheduling toolboxu dochází v případech, jako je ukládání a export do workspace, případně při použití některého pluginu. Vytváření grafu nějakou dobu trvá, takže zpomaluje proces ukládání a jiné úlohy, ale jsem si jist, že je to lepší řešení, než každou změnu aplikovat přímo na objekt **graph**. V takovém případě, například při každé změně barvy vrcholu, by bylo nezbytné načíst objekt **graph** s ***UserData*** objektu **axes**, z něj příslušný vrchol vybrat, provést změnu a poté opět vše uložit zpět.

Vytvoření grafu provádí funkce ***`createobjectgraph`*** nastíněná v diagramu 6.2.

6.2.4 Export to picture

Funkce ***`exportcanvastopicture`*** poskytuje možnost uložit aktuální kreslicí list jako bitmapový obrázek. Dává k dispozici obvyklé typy jako bmp, png, jpg a další. Vývojový diagram funkce není nutné uvádět, protože její složení je velmi jednoduché – výběr jména a typu obrázku pomocí **`uiputfile`**, získání obrázku pomocí funkce **`getframe`** a jeho uložení **`imwrite`**. V případě příliš velkého zobrazení grafu je do obrázku exportována pouze viditelná část (v budoucnu by se mohla tato funkce upravit tak, aby se exportoval graf celý včetně zakryté části). Viditelný grafický objekt **uicontrol slider** (+ tlačítko v rohu) by byl rovněž zahrnut do obrázku, proto je potřeba zajistit jeho skrytí a opětovné zobrazení.

6.2.5 Import from workspace, Export to workspace

Obě tyto funkce (***`importobjectgraph`*** a ***`exportobjectgraph`***) jsou velice podobné otevírání a ukládání ze/do souboru. Odlišnost je v tom, že v tomto případě není potřeba provádět žádné operace se soubory, ale přenosy dat se uskutečňují pouze z/do workspace reprezentovaným klíčovým slovem **`base`** prostřednictvím funkcí **`evalin`** a **`assignin`**.

6.2.6 Close, close all, quit

Funkce ***`closecanvas`*** volaná po kliknutí na položku menu close otestuje, zda se nachází v aktuálním kreslicím listě alespoň jeden vrchol grafu, a pokud ano, zeptá se uživatele pomocí **`qestdlg`**, zda chce tento graf uložit. Pokud chce, je zavolána funkce ***`savegraphtofile`***. Posledním příkazem funkce ***`closecanvas`*** je **`delete(gca);`** který smaže aktuální objekt **axes**.

Protože při vytvoření má každý objekt **axes** přiřazenu callback funkci ***DeleteFcn***, je před samotným smazáním vykonána ona. Jmenuje se ***deletefcncallbackaxes*** a existuje proto, aby bylo zajištěno, že při smazání kreslicího listu bude smazána také příslušná záložka a nastaven jiný kreslicí list jako aktuální. Zvláštní situace nastane, když zavíráme jediný kreslicí list, který se v Grapheditu nachází. V takovém případě tato callback funkce zajistí otevření nového listu.

Význam funkce ***closeallcanvas*** volané při kliknutí na **close all**, je jistě zřejmý.

Při stisku "Quit" dojde k zavolání funkce ***quitgraphedit***, která se dotáže, zda chce uživatel skutečně program ukončit. Při kladné odpovědi hlavní okno Grapheditu zavře. Neuložené nakreslené grafy v tom případě budou ztraceny.

6.3 Edit

6.3.1 Cut, copy

"Cut", "copy" a "paste" jsou prozatím jediné funkce, které se v tomto oddíle hlavního menu nacházejí. Lze je aplikovat pouze na vrcholy, protože mohou existovat v grafu zcela samostatně. Hrana musí vždy spojovat dva vrcholy a její kopírování by tedy bylo obtížné.

Funkce ***copynodefcn*** je vykonána jak pro vyjmutí, tak pro kopírování. V případě, kdy jde o vyjmutí, je však původní vrchol smazán. Tato funkce vloží do oddílu **Box** ve struktuře uložené jako ***UserData*** hlavního okna Grapheditu vybraný vrchol ve formě objekt **node**. Zde se pak nachází do ukončení Grapheditu. Jen v případě kopírování či vyjmutí jiného vrcholu je přepsán dalším. Funkce je zobrazena na vývojovém diagramu 6.6.

6.3.2 Paste

Funkce ***pastenodefcn*** (vývojový diagram 6.7) z již uvedeného místa uložený vrchol vezme a pomocí funkce ***drawnode*** (vývojový diagram 6.9) jej vykreslí na pozici určenou algoritmem, který je v ní implementován.

6.4 View

6.4.1 Property Editor

Kliknutím na tuto položku v menu "View" dojde ke zobrazení property editoru, který již byl popsán v kapitole 4.5.

Stejně jako v primární funkci ***graphedit*** i zde, při práci s property editorem (funkcí ***grapheditpropertyeditor*** umístěnou v adresáři **private**), využívám příkazy **Switch** a **Case**. Je to z toho důvodu, že v případě, kdy je funkce umístěna v jiném souboru, jsem schopen volat pouze funkci primární. Pokud tedy potřebuji spustit operaci, kterou vykonává funkce, jíž kvůli tomuto problému nevidím, mohu její jméno v podobě proměnné typu string zadat jako parametr primární funkce a pomocí **switch** a **case** ji zavolat. Tento způsob je použit pro komunikaci mezi Grapheditem a property editorem.

Jak je ve zdrojovém kódu vidět, kliknutím v menu na "Property Editor" se uvedeným způsobem spustí funkce ***menupropertyeditor*** (stejně tak při dvojkliku někde v kreslicí ploše či zmáčknutí ikony v nástrojové liště), která buď zobrazí nebo skryje property editor v pravé části hlavního okna. Pokud jej právě zobrazuje, volá klíčovou funkci ***createpropertyeditor***. Ta vytvoří grafické objekty, které property editor představují a zavolá funkci ***grapheditpropertyeditor*** (primární funkce) bez parametrů. Tímto způsobem jsou vytvářeny jednotlivé řádky určené vybraným grafickým objektem (**graph** – **axes**, **node** – **rectangle**, **edge** – **line**) pomocí funkce ***grapheditdictionary*** zmíněné již v kapitole 4.5. Příkaz ***grapheditpropertyeditor*** (bez parametrů) je rovněž volán vždy, při změně vybraného objektu.

Ostatní funkce, které se v souboru **grapheditpropertyeditor.m** nacházejí, jsou pomocné a slouží k úpravě vzhledu při změně velikosti (celého okna nebo pouze property editoru), k vracením a nastavováním změn v parametrech objektů **graph**, **node**, **edge**.

6.4.2 Names, UserParams

Nastavením jednoho z těchto parametrů ovlivňujeme zobrazení grafického objektu **text,** v němž je uvedeno jméno ***Name*** vrcholu či hrany grafu nebo textová forma uživatelského parametru ***UserParam***. Změnit zobrazení těchto informací je možné i pomocí kontextového menu zobrazovaného po kliknutí pravým tlačítkem na příslušném objektu. Uživatel má možnost ovládat oba typy **text**ů pro vrcholy (**node**s) a hrany (**edge**s) zvlášť.

Změny zobrazení provádí funkce ***viewtext*** (vývojový diagram 6.8), která se dělí na části pro ***Name*** a pro ***UserParam***. Ty jsou obě velmi podobné, ale liší se v nastavování pozice ***UserParam*** v závislosti na zobrazeném textu ***Name***. ***UserParam*** jsou vždy zobrazeny pod texty ***Name***, pokud jsou viditelné, v opačném případě se ***UserParam*** z estetických důvodů přesouvají na jejich místo. Přesun je zajištěn funkcí ***repairypositionuserparam***.

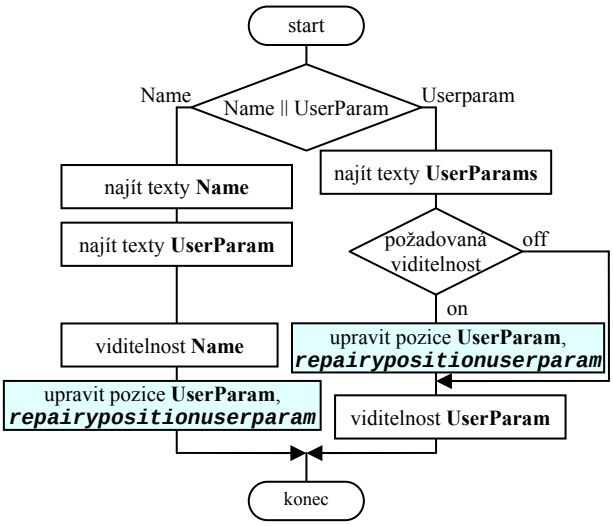


Diagram 6.8 – *Funkce* ***viewtext***

6.5 Draw

6.5.1 Draw node

Nastavit kreslicí mód lze jak v hlavním menu, tak na liště nástrojů. Kliknutím je zavolána funkce ***selectdrawtype***, která podle volby uživatele nastaví jako callback rutinu ***ButtonDownFcn*** grafických objektů **axes** (změna má vliv na všechny kreslicí listy) funkci ***drawnode*** (diagram 6.9), je-li kreslení vrcholů zapínáno, případně ***buttondownoveraxes***, je-li vypínáno. Po zapnutí uživatel kliknutím na kreslicí plochu umístí vrchol představovaný grafickým objektem **rectangle**.

Funkce ***buttondownoveraxes*** slouží pro umožnění výběru kreslicího listu a zapínání/vypínání property editoru.

6.5.2 Draw edge

Zde může uživatel nastavit kreslicí mód naprosto stejně jako u "Draw Node" s tím rozdílem, že jako callback funkce ***ButtonDownFcn*** je nastavena ***drawedge*** (diagram 6.10).

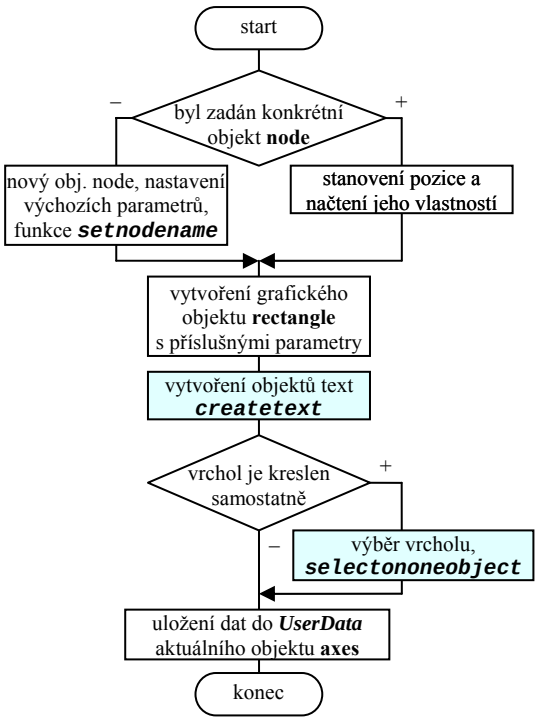


Diagram 6.9 – *Funkce* ***drawnode***

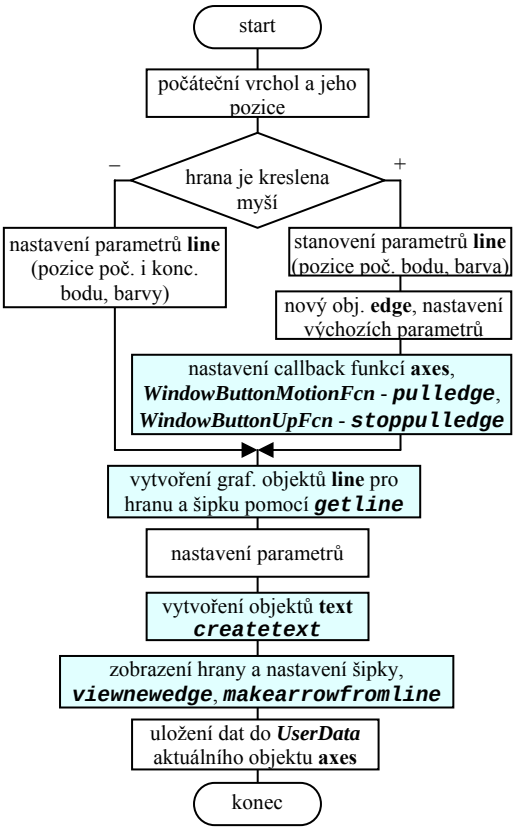


Diagram 6.10 – *Funkce* ***drawedge***

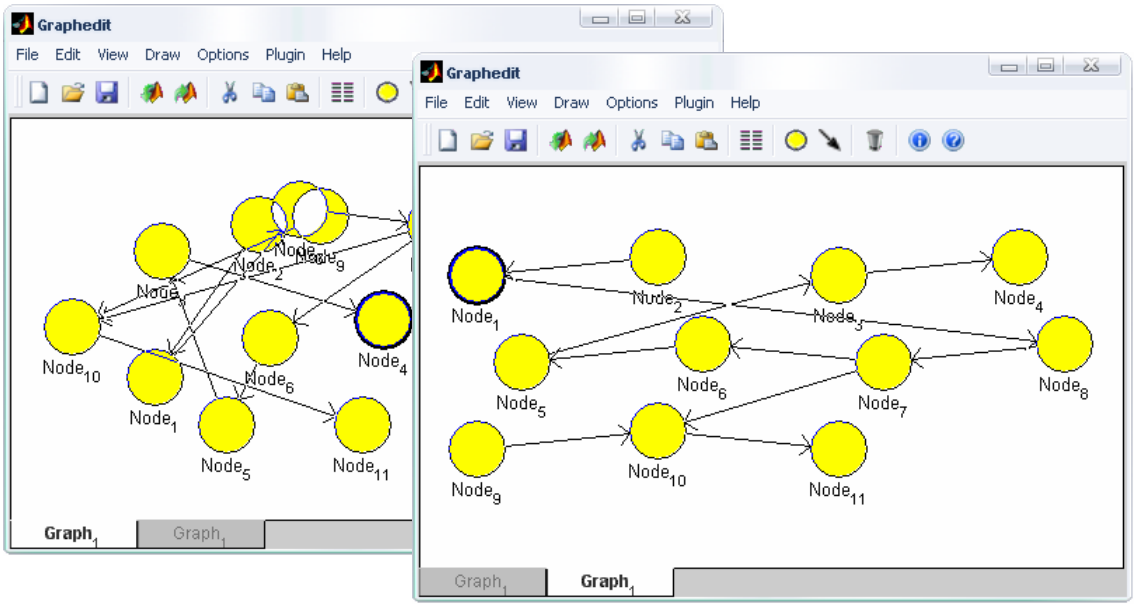
6.5.3 Delete mode

"Delete mode" se od obou předchozích funkcí způsobem své činnosti liší tím, že callback rutinu ***ButtonDownFcn*** pro objekt **axes** úplně zruší a tu samou rutinu pro grafické objekty reprezentující vrcholy a hrany nastaví na ***deleteselectedobject*** (při vypnutí režimu na ***buttondownovernode*** a ***buttondownoveredge***). Jak grafickému objektu **rectangle**, tak **line** je nastavena jako callback routina ***DeleteFcn*** funkce, která při smazání automaticky odebere také příslušné objekty **text** a všechny informace ze struktury v ***UserData*** kreslicího listu.

Mazání nakreslených objektů umožňuje rovněž klávesa Delete. Vybráním požadovaného objektu myší a jejím stisknutím dojde k smazání. Funkce ***deleteselectedobject*** funguje stejně, ale po výběru již není potřeba Delete stisknout. Automaticky se totiž nastaví na znak představující Delete ***CurrentCharacter***, což je parametr okna **figure**, který uchovává informaci o posledním zadaném znaku.

6.5.4 Layout

Tato položka menu rozmístí vrcholy nakresleného grafu na pozice určené algoritmem funkce ***nodeslayout***. Tato funkce původně vznikla kvůli potřebě vhodně rozložit importovaný nebo otevíraný graf, jehož vrcholy zatím nemají určenu pozici. Jejím úkolem není nalézt inteligentní rozložení takové, aby se v grafu nekřížily hrany nebo splňoval jiné podmínky např. určitý tvar atd. Bez jakéhokoliv ohledu na hrany umístí vrcholy podle jejich pořadí.



Obrázek 6.1 – *Vstup a výstup funkce* ***nodeslayout***.

6.6 Options

Options je nabídka nastavení připravená spíše do budoucna, kdy bude do programu přidáno větší množství parametrů ovlivňujících funkce Grapheditu. Pak bude vytvořen dialog umožňující pohodlné nastavování parametrů stejný, jako známe z jiných aplikací. Prozatím zde lze nalézt pouze možnost unikátních jmen vrcholů a hran.

6.6.1 Unique names

Klinutím na položku menu “Unique Names” není spuštěna žádná zvláštní funkce. Stav tohoto nastavení je testován při přejmenování a vkládání vrcholů a hran a podle výsledku je pak nové jméno změněno či ponecháno.

6.7 Plugins

6.7.1 Add, remove plugin

Kliknutím na “add plugin” a “remove plugin” v menu “Plugins” jsou spouštěny funkce ***`grapheditaddplugin`*** a ***`grapheditremoveplugin`***, které vytvoří dialogy, pomocí nichž může uživatel celý systém pluginů ovládat. Oba tyto dialogy stejně jako univerzální dialog pro spouštění pluginů jsou popsány v kapitole 5 a vysvětlení systému pluginů je věnována poslední část práce.

7 Plugins – Systém zásuvných modulů

Plugins je systém zásuvných modulů, poskytující jednoduché rozhraní pro další rozšiřování vlastností editoru grafů. Umožňuje tedy uživateli přidat libovolnou funkci, která nějakým způsobem pracuje s grafy. Požadavek na vytvoření systému je posledním bodem zadání bakalářské práce.

7.1 Xml (eXtended Markup Language)

Protože je nezbytné, aby si Graphedit připojené moduly pamatoval také při dalším spuštění a ne jen po dobu existence jedné jeho instance, musí být informace o systému pluginů někam ukládány. K tomuto účelu slouží xml soubor **grapheditplugin.xml**, který se nachází v adresáři **private** společně s dalšími soubory využívanými Grapheditem.

Pro ukládání informací o celé soustavě vložených modulů jsem použil jazyk xml, protože je to jazyk umožňující pomocí značek (tzv. tagů) popis strukturovaných dat. K jeho využití mě vedlo především to, že v Matlabu jsou k dispozici funkce **`xmlread`** a **`xmlwrite`**, které výrazně usnadňují čtení i zápis souborů xml se zachováním struktury.

Xml kód 7.1 ukazuje xml soubor s daty, která definují několik zásuvných modulů. Je na něm vidět, že bodem nejvyšší úrovně je seznam pluginů (tag **<pluginlist>**), který se dělí na několik uživatelem vytvořených skupin (**<group>**) a každá tato skupina obsahuje žádný nebo několik zásuvných modulů (**<plugin>**). Ke každému modulu musí být k dispozici konkrétní informace. Parametr tagu **<plugin> gui** rozhoduje, zda pro spuštění pluginu bude využit dialog ***`grapheditdoplugin`*** popsaný v kapitole 5.4 nebo zda jej ke své funkci nepotřebuje. Dalším parametrem je **name** – jméno, které se zobrazí v hlavním menu. Tagem **<description>** je určeno stručné vysvětlení modulu, **<command>** je značka uvozující příkaz, kterým je volána funkce představující modul.

Při porovnání bloků **<command>** u jednotlivých modulů v xml kódu 7.1 zjistíme, že je možno uchovávat a po té použít pouze název funkce nebo celou absolutní cestu v adresářové struktuře. V Grapheditu jsou implementovány čtyři stupně hledání funkce zadaného modulu. Nejdříve je použita absolutní cesta, pokud podle tohoto údaje není funkce nalezena, prohledá se adresář **plugin**, který náleží ke Grapheditu, následuje .../**toolbox/scheduling/@graph**, který slouží jako mateřský adresář grafové části Scheduling toolboxu TORSCHE. Jako poslední jsou prohledány cesty Matlabu (**`path`**).

```xml
<?xml version="1.0" encoding="utf-8" ?>
<pluginlist application="Graphedit" ver="1.0">
  <group name="Graph">
    <plugin gui="1" name="Hamilton">
      <description>Finds Hamilton circuit in a graph</description>
      <command>C:\Matlab65\toolbox\scheduling\@graph\hamilton_circuit.m</command>
    </plugin>
  </group>
  <group name="Graphics">
    <description>Graphics plugins</description>
    <plugin gui="0" name="Nodes to color">
      <command>C:\Matlab65\work\graphedit\plugin\nodestocolor.m</command>
    </plugin>
  </group>
  <group name="Creating">
    <description />
    <plugin gui="1" name="Create Nodes">
      <command>createnodes</command>
    </plugin>
  </group>
  <group name="Exporting">
    <description />
    <plugin gui="0" name="Export Names and UserParams">
      <description>Exports names of nodes and userparams of edges to workspace.</description>
      <command>getnamesanduserparams</command>
    </plugin>
  </group>
</pluginlist>
```

XML kód 7.1 – *Ukázka kódu souboru* ***grapeditplugin.xml***.

7.2 Ukládání

Ukládání do xml souboru se provádí vždy po přidání nového pluginu. Řeší je funkce ***grapheditplugstructure2xml*** (umístěna v samostatném souboru), které je jako parametr předána struktura obsahující všechny dostupné informace o zásuvných modulech. Jejím prostřednictvím je pomocí příkazů uvedených v nápovědě k funkci **xmlwrite** převedena zmíněná struktura seznamu pluginů na datový objekt, který je pak pomocí **xmlwrite** uložen jako soubor **grapheditplugin.xml**

7.3 Načítání

Načtení pluginů je provedeno okamžitě při spuštění Grapheditu. Zajišťuje je funkce ***loadplugin***, která je nastavena jako callback routina ***CreateFcn*** objektu **uimenu** jménem Plugins. V ní je zavolána funkce ***grapheditxml2struct*** umístěná i s dalšími uvedenými funkcemi v samostatném souboru adresáře **private**. Ta načte pomocí **xmlread** obsah **grapheditplugin.xml.** Funkce ***xmlplugins2struct***, která se sama rekurentně volá, pak vytvoří poměrně složitou strukturu. Tu je potřeba zjednodušit do použitelného tvaru, což vykonají funkce ***getpluginlist***, ***getgroup***, ***getplugin*** a ***getparam***.

7.4 Příklady

Jako příklady zde uvádím zásuvné moduly nadefinované v xml souboru, který lze spatřit v xml kódu 7.1.

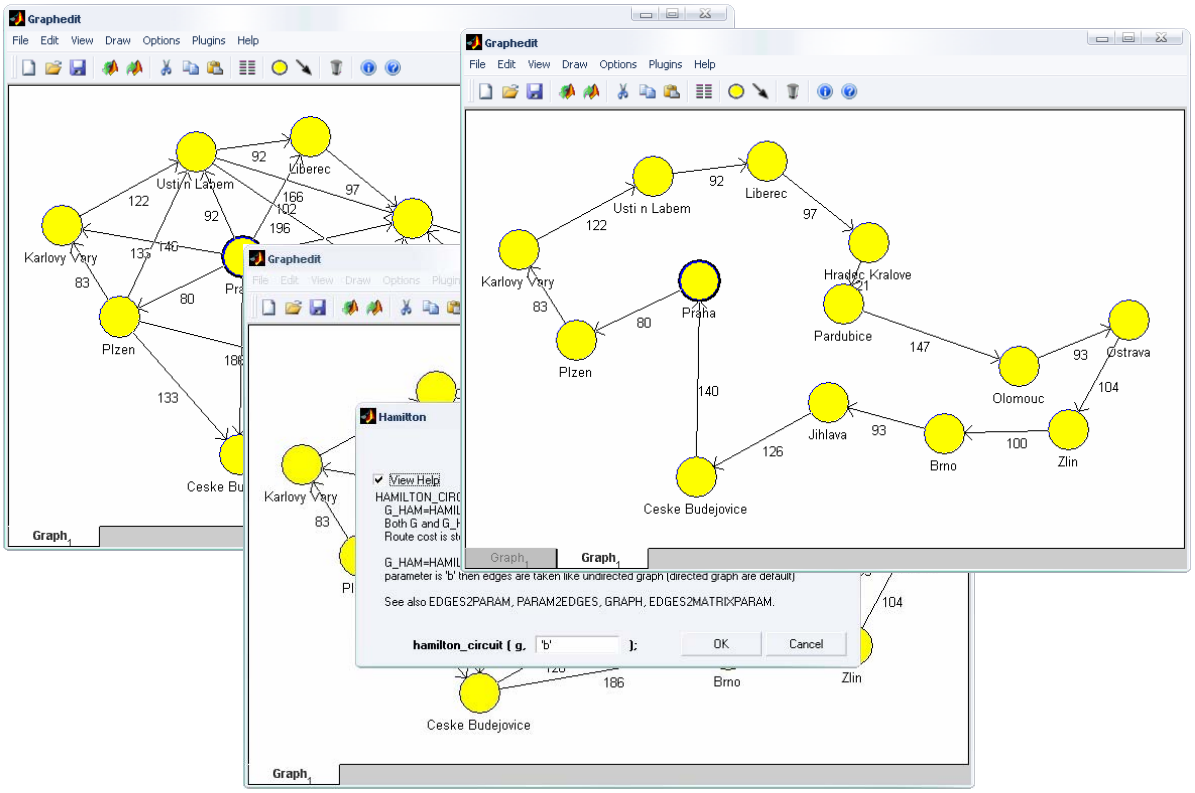
7.4.1 Hamiltonovská kružnice

Hamiltonovská kružnice grafu je taková cesta, která projde právě jednou všechny vrcholy grafu.

Zde je použita funkce **`hamilton_circuit`** implementovaná v části Scheduling toolboxu TORSCHE věnující se grafům. Zařazení do systému zásuvných modulů mi umožní používat ji i bez potřeby příkazového řádku Matlabu.

Postup použití:

1. Nakreslení nebo načtení grafu (splňujícího podmínky existence Hamiltonovské kružnice). V tomto případě je tímto grafem model České republiky.
2. Spuštění modulu pomocí hlavního menu.
3. Z nápovědy funkce **`hamilton_circuit`** se dozvím, že je nutné zadat parametr `'b'`.
4. Stisknutí "OK" funkci spustí, ta vrátí upravený graf, který je pak vykreslen na nový kreslicí list.



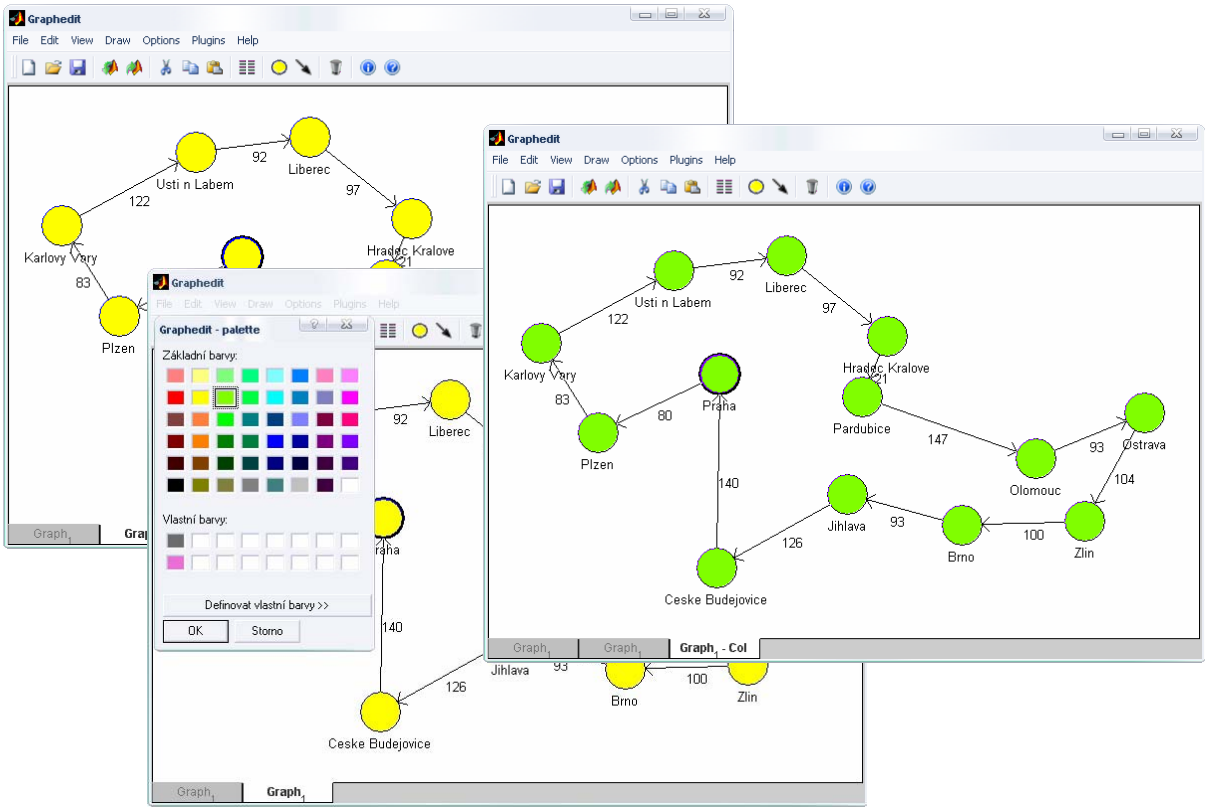
Obrázek 7.1 – *Příklad použití modulu hamilton_circuit.*

7.4.2 Obarvení vrcholů (jednou barvou)

Nejedná se o barvení grafu tak, jak je známe z teorie grafů (literatura [1]). Funkce ***nodestocolor*** uvedená v kódu Matlabu 7.1 má za úkol ukázat použití pluginu, který nevyužívá dialog ***grapheditdoplugin***, ale použije svůj vlastní – v tomto případě **uisetcolor**. Postup jejího použití je patrný na obrázku 7.2.

```
function g = nodestocolor(g)
    if isa(g, 'graph')
        color = uisetcolor('Graphedit - palette');
        if (length(color) ~= 1)
            for i = 1:length(g.N)
                node = g.N(i);
                set_graphic_param(node, 'Color', color);
                g.N(i) = node;
            end
            g.Name = [g.Name ' - Color'];
        else
            g = [];
        end
    else
        g = [];
    end
```

Kód Matlabu 7.1 – *Kód funkce* ***nodestocolor*** *– modulu pro obarvení vrcholů.*



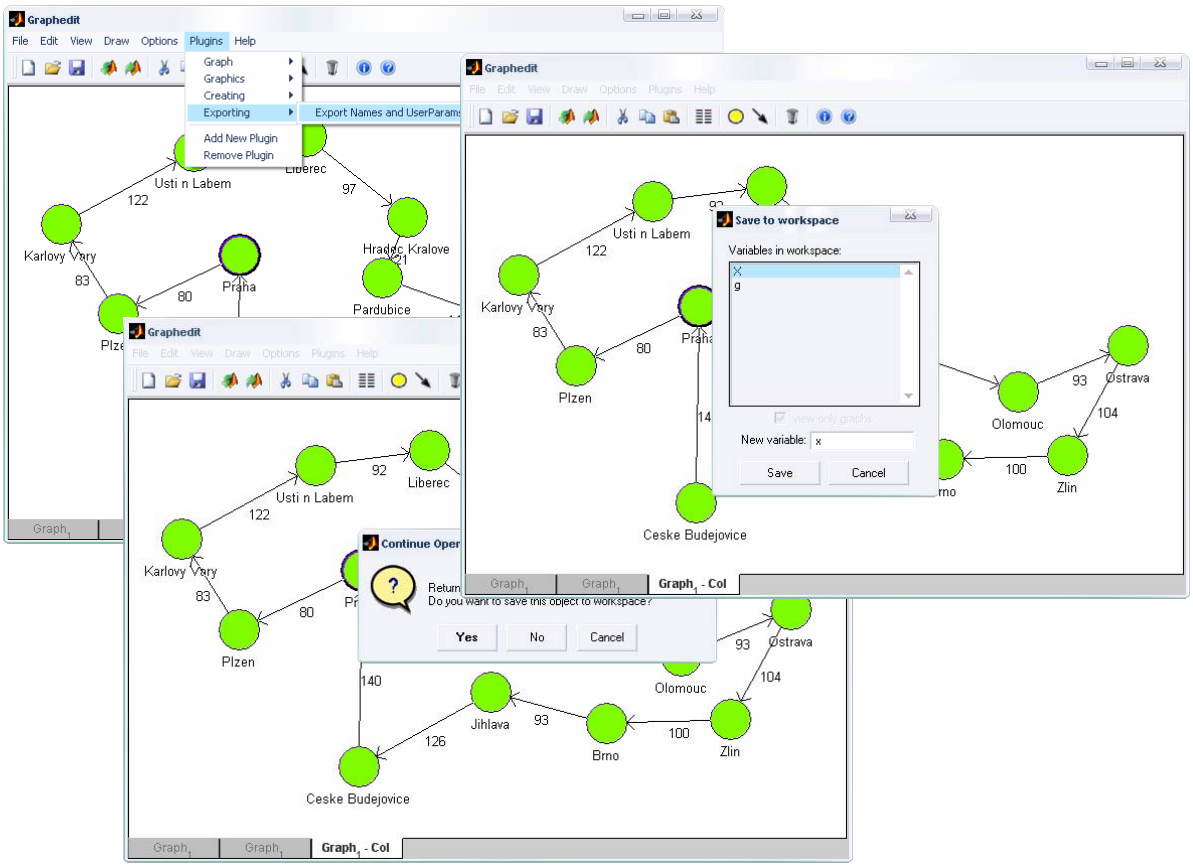
Obrázek 7.2 – *Příklad použití modulu* ***nodestocolor***.

7.4.3 Export jmen vrcholů a uživatelských parametrů hran

Funkce ***getnamesanduserparams*** uvedená v kódu Matlabu 7.2 zjistí jména vrcholů nakresleného grafu a hodnoty ***UserParams*** hran. Všechny tyto údaje vloží do struktury, kterou vrátí. Jak je vidět na obrázku 7.3, Graphedit pozná, že vracená proměnná není typu graph a nabídne uživateli možnost exportovat ji do workspace Matlabu. Pomocí dialogu ***grapheditlistdlg*** je pak zvolen název proměnné, pod nímž bude struktura uložena ve workspace Matlabu.

```
function out = getnamesanduserparams(g)
    nodes = []; edges = [];
    for i = 1:length(g.N)
        nodes{i} = g.N{i}.Name;
    end
    for i = 1:length(g.E)
        edges{i} = g.E{i}.UserParam;
    end
    out = struct('Nodes',nodes,'Edges',edges);
```

Kód Matlabu 7.2 – *Kód funkce* ***getnamesanduserparams***.



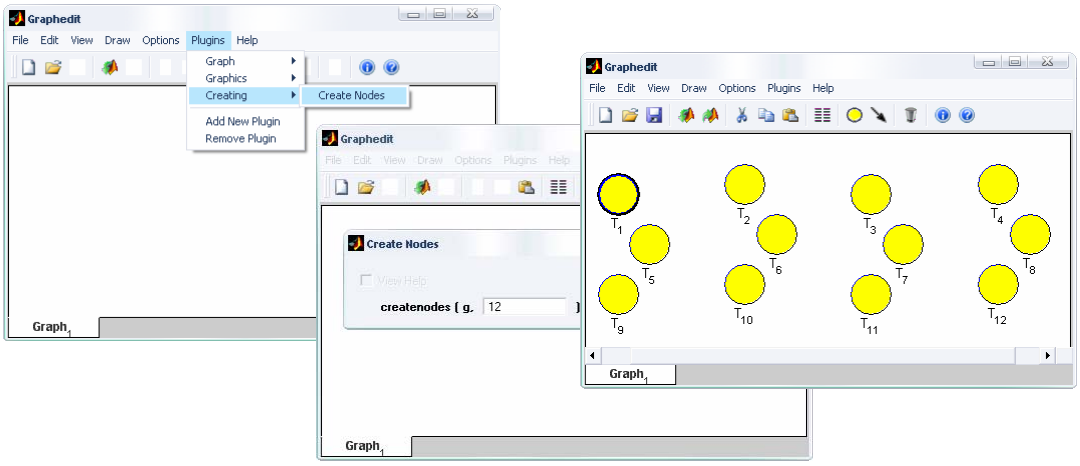
Obrázek 7.3 – *Příklad použití modulu* ***getnamesanduserparams***.

7.4.4 Vytvoření vrcholů grafu

Funkce z kódu Matlabu 7.3 vytvoří graf s takovým počtem vrcholů, kolik je zadáno jako její parametr prostřednictím dialogu ***graphedit*doplugin**. Na obrázku 7.4 je zobrazen postup spuštění modulu a jeho výsledek.

```
function g = createnodes(g,varargin)
    g = graph;
    for i = 1:round(varargin{1})
        n = node;
        n.Name = ['T_{' num2str(i) '}'];
        g.N(i) = n;
    end
```

Kód Matlabu 7.3 – *Kód funkce* ***createnodes*** *– vytvoření zadaného počtu vrcholů.*



Obrázek 7.4 – *Příklad použití modulu* ***createnodes***.

8 Závěr

Tato práce popisuje návrh, implementaci i způsob ovládání editoru grafů vytvořeného jako součást Scheduling toolboxu TORSCHE pro Matlab. Úkolem Grapheditu je zpřístupnit uživateli práci s grafovou částí tohoto toolboxu a umožnit mu interaktivní vytváření grafů, editaci jejich parametrů i parametrů každého vrcholu či hrany a práci s nimi.

K vytvoření GUI jsem použil systém Handle Graphics, který je implementován v Matlabu. Z důvodu požadované kompatibility nebyly využity grafické objekty nabízené novějšími verzemi systému Matlab. Přizpůsobil jsem se podmínkám verze 6.5, R13.

V kapitole Matlab GUI – Systém Handle Graphics jsou popsány grafické prvky využité při implementaci Grapheditu i jejich důležité vlastnosti a zjednodušené příklady vytvoření každého z nich.

Kapitola Graf je věnována popisu základního objektu – grafu a prvků z nichž je složen – vrcholů a hran. Graphedit umožňuje vytvoření grafu a manipulaci s ním i s jeho částmi.

Realizace vzhledu editoru, která prošla poměrně dlouhým vývojem v několika fázích zdokonalování, je popsána a vyobrazena v kapitole Graphedit. Dialogům, které bylo potřeba vytvořit pro snazší umožnění některých funkcí, je věnována kapitola Dialogy.

Předposlední kapitola Funkce přináší přehled všech funkcí editoru společně s jejich slovním popisem. Pro stěžejní funkce jsem také uvedl vývojové digramy, jimiž je popsán způsob provádění jednotlivých operací.

V Grapheditu jako takovém nejsou implementovány žádné zvláštní funkce pro práci s grafy. Jim je věnována grafová část TORSCHE Scheduling toolboxu. Pro umožnění pohodlného využití takových funkcí prostřednictvím Grapheditu vznikl požadavek na vytvoření systému zásuvných modulů označovaných jako Plugins. Ten dělá z Grapheditu skutečně využitelný program a ne pouze nástroj „pro kreslení koleček a čar“.

Graphedit je v této fázi funkční, ale náročnějšímu uživateli zřejmě budou chybět některé možnosti nastavení a funkce, jež by práci s ním jistě usnadnily. Jedná se například o undo a redo, možnost nastavení a uchování jeho stavu pro další spuštění, možnost spouštět Graphedit z příkazové řádky s parametry ovlivňujícími jeho vlastnosti. Z hlediska teorie grafů pak prozatím chybí možnost vytváření multigrafů a práce s nimi.

Nelze tady mluvit o uzavřeném projektu a do budoucna se počítá s dalším vývojem a rozšířením jeho možností.

9 Seznam literatury

[1] DEMEL, J. *Grafy a jejich aplikace*. 1. vyd. Praha: Academia, 2002. ISBN 80-200-0990-6.

[2] ZAPLATÍLEK, K., DOŇAR, B. *Matlab tvorba uživatelských aplikací*. 1. vyd. Praha: BEN, 2004. ISBN 80-7300-133-0.

[3] *TORSCHE Scheduling Toolbox for Matlab* [online], [cit. 20.června 2006]. Dostupné z <http://rtime.felk.cvut.cz/scheduling-toolbox>.

[4] The MathWorks. *Matlab and Simulink for Technical Computing* [online], [cit. 19.června 2006]. Dostupné z <http://www.matlab.com>.

[5] MOUTOVIC, V. Tvorba algoritmů. *Programování ve VB.NET* [online]. 2004, č.2 [cit. 16.června 2006]. Dostupné z <http://www.pcsvet.cz>.

[6] C*omp.soft-sys.matlab* [online]. [cit. 19.června 2006] Dostupné z <http://groups.google.com/group/comp.soft-sys.matlab>.

[7] JOHNSON, R. *MATLAB Programming Style Guidelines* [online], Ver. 1.5 October 2002. Dostupné z <http://www.datatool.com>.

10 Použitý software

The MatWorks, Matlab 6.5, R13

The MatWorks, Matlab 7.0.4, R14

Microsoft Windows XP Professional Verze 2002, SP2

Microsoft Office Word 2003

11 Příloha A

Obsah CD:

- elektronická podoba této práce ve formátu .pdf.
- zdrojový kód programu a jeho další součásti – adresář **graphedit**.
- TORSCHE Scheduling toolbox pro Matlab – adresář **scheduling** (aktuální verzi lze nalézt na <http://rtime.felk.cvut.cz/scheduling-toolbox>).
- m-file **setgrapheditpath.m**, jehož spuštěním v systému Matlab dojde k připojení adresářů **graphedit** a **scheduling**.
- binární soubor **czech_rep.mat** obsahující grafový model České republiky.