

# Počítačová hra ve 2D v MATLAB

David Fiala

---

Bakalářská práce  
2014



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2013/2014

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: David Fiala  
Osobní číslo: A11092  
Studijní program: B3902 Inženýrská informatika  
Studijní obor: Informační a řídicí technologie  
Forma studia: prezenční

Téma práce: Počítačová hra ve 2D v MATLAB

Zásady pro vypracování:

1. Provedte literární rešerši o MATLAB.
2. Seznamte se s počítačovými hrami dostupnými na serveru MATLAB Central a s bakalářskými pracemi na téma počítačové hry v MATLAB.
3. Provedte specifikaci hry a popište její pravidla.
4. Vytvořte variantu počítačové hry Člověče nezlob se ve 2D pro MATLAB.
5. Uveďte popis tvorby hry a použité funkce.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB pro začátečníky. 1. vydání. Praha: BEN – technická literatura, 2003. 144 s. ISBN: 80-7300-095-4.
2. DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB – tvorba uživatelských aplikací. 1. vydání. Praha: BEN – technická literatura, 2004. 216 s. ISBN: 80-7300-133-0.
3. DUŠEK, František. MATLAB a Simulink – Úvod do používání. 1.vydání. Pardubice: Univerzita Pardubice, 2000.
4. HECZKO Michal. Výukový a zkušební program pro předmět PPAŘ. Bakalářská práce. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2006.
5. PERŮTKA, Karel. MATLAB – Základy pro studenty automatizace a informačních technologií. 1. vyd. Zlín: UTB ve Zlíně, 2005. 304 s. ISBN 80-7318-355-2.
6. MATLAB GUI Tutorial. URL: <http://www.matlabgui.com> Icit. 2013-02-041.

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**

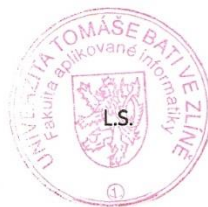
Ústav řízení procesů

Datum zadání bakalářské práce: **28. února 2014**

Termín odevzdání bakalářské práce: **13. června 2014**

Ve Zlíně dne 28. února 2014

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

### **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/bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## **ABSTRAKT**

Práce obsahuje teoretickou část věnující se programu Matlab. Uvádí informace o programu, jeho použití, popisuje prostředí a funkce programu. Popisuje tvorbu GUI objektů. Jsou uvedeny příklady kódu.

Teoretická část také popisuje hry jiných autorů. Hry jsou stažené ze serveru Matlab Central nebo se jedná o jiné bakalářské práce.

V praktické části je uvedena ukázka a popis vytvořené hry „Člověče, nezlob se“ a je popsán zdrojový kód hry.

Klíčová slova: Matlab, GUI, hra, programování

## **ABSTRACT**

Theoretical part of the bachelor thesis deals with program Matlab. It contains informations about program, its use, describe enviroment and functions. It shows creation of GUI ob-ject. In the work are examples of codes.

Theoretical part also describe games from other autors. Games are downloaded from Matlab Central server or they are the others bachelor thesis.

In practical part is stated illustration and description of created game „Člověče, nezlob se“. Source codes contains description.

Keywords: Matlab, GUI, game, programming

Tímto bych chtěl poděkovat vedoucímu práce panu Ing. Karlu Perůtkovi, Ph.D. za podněty, rady a kritiku v průběhu psaní práce a realizace programu.

“Důkazem vysokého vzdělání je schopnost mluvit o největších věcech nejjednodušším způsobem.“

Emerson Ralph Waldo (1803 - 1882)

# OBSAH

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 MATLAB</b> .....	<b>11</b>
1.1 O PROGRAMU .....	11
1.2 POPIS PROSTŘEDÍ.....	11
1.2.1 Menu toolbar .....	12
1.2.2 Průzkumník .....	13
1.2.3 Příkazové okno .....	13
1.2.4 Okno aktuální složky.....	13
1.2.5 Náhledové okno .....	13
1.2.6 Workspace.....	13
1.2.7 Historie .....	14
1.3 UŽITEČNÉ PŘÍKAZY A FUNKCE.....	14
1.4 DATOVÉ TYPY A PROMĚNNÉ.....	15
1.4.1 Numerický datový typ.....	16
1.4.2 Datový typ char .....	18
1.4.3 Datový typ cell .....	19
1.4.4 Datový typ struktura.....	19
1.4.5 Datový typ handle funkce .....	20
1.5 OPERÁTORY .....	21
1.5.1 Relační operátory .....	21
1.5.2 Logické operátory .....	22
1.6 PODMÍNKY A CYKLY.....	22
1.6.1 Podmínky if, elseif, else .....	22
1.6.2 Příkaz switch .....	23
1.6.3 Cyklus while.....	23
1.6.4 Cyklus for.....	24
1.7 PRÁCE S MATICEMI.....	24
1.7.1 Definice a přístup .....	24
1.7.2 Operace s maticemi .....	26
1.8 PRÁCE SE SOUBORY .....	27
1.8.1 Binární ukládání .....	27
1.8.2 Textové ukládání .....	28
1.9 M - SOUBORY .....	29
1.9.1 Matlab Editor .....	29
1.9.2 Skript .....	31
1.9.3 Funkce .....	32
1.10 GUI.....	33
1.10.1 GUIDE editor rozložení .....	33
1.10.2 Manuální programování .....	34
1.10.3 Figure .....	34
1.10.4 Uicontrol .....	35
1.10.5 Set, get.....	36

<b>2</b>	<b>POČÍTAČOVÉ HRY V MATLABU</b>	<b>37</b>
2.1	STELLARIA	37
2.2	FLAPPY BIRD	38
2.3	FIELDS CRAZY	39
2.4	OPERATION EIGENFAUST 3D	39
<b>II</b>	<b>PRAKTICKÁ ČÁST</b>	<b>41</b>
<b>3</b>	<b>POUŽITÉ PROGRAMY</b>	<b>42</b>
3.1	COREL DRAW	42
3.2	AUDACITY	43
3.3	MATLAB	43
<b>4</b>	<b>POPIS HRY</b>	<b>44</b>
4.1	PRAVIDLA HRY	44
4.2	REALIZACE HRY	44
4.2.1	Hlavní menu	45
4.2.2	Nová hra	45
4.2.3	Načíst hru	46
4.2.4	Pravidla hry	46
4.2.5	Ukončit hru	47
4.2.6	Hlavní hra	47
4.2.7	Uložit hru	49
4.2.8	Pravidla	49
4.2.9	Konec	49
4.2.10	Okno Vítěz	50
4.3	ADRESÁŘOVÁ STRUKTURA	51
4.3.1	Adresář clovece	51
4.3.2	Adresář multimédia	52
4.3.3	Adresář uložení	52
4.3.4	Adresář menu	52
4.3.5	Adresář hra	54
	<b>ZÁVĚR</b>	<b>64</b>
	<b>SEZNAM POUŽITÉ LITERATURY</b>	<b>65</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK</b>	<b>67</b>
	<b>SEZNAM OBRÁZKŮ</b>	<b>68</b>
	<b>SEZNAM PŘÍLOH</b>	<b>69</b>

## ÚVOD

Cílem této bakalářské práce je vytvoření deskové hry „Člověče, nezlob se“ ve vývojovém prostředí programu Matlab. Způsob zpracování práce by měl být na takové úrovni, aby práce mohla být použita jako výuková pomůcka.

V první části práce je vysvětleno, co program Matlab je a v jaké oblasti dochází k jeho uplatnění. Je popsáno jeho prostředí, jeho vlastnosti, funkce, práce se soubory a editor GUIDE pro tvorbu GUI objektů. Ke všem těmto bodům jsou uvedeny příklady kódu pro názornost jejich použití.

Pomocí webu Matlab Central jsou uvedeny ukázky zajímavých her vyvíjených v Matlabu. Samotné hry slouží jako prezentace možností a bohaté rozmanitosti funkcí Matlabu.

V praktické části jsou uvedeny další využití programy pro tvorbu grafické a zvukové stránky hry. Samotná hra je nejdříve obecně popsána z hlediska obyčejného uživatele. Spolu s popisem jsou uvedeny obrázky všech oken vyskytujících se ve hře. V poslední části je popsána adresářová struktura. Všechny vytvořené skripty jsou popsány z hlediska své funkčnosti.

Okruhy zpracovaných témat:

- Program Matlab, jeho vývojové prostředí a užitečné funkce
- Datové typy, operátory a práce s maticemi
- Práce se soubory, cykly a podmínkami
- Tvorba grafického uživatelského prostředí
- Web Matlab Central, ukázka jiných her
- Popis vývoje hry a zdrojových souborů, její ukázka

## **I. TEORETICKÁ ČÁST**

## 1 MATLAB

### 1.1 O programu

Matlab je vysoce výkonný skriptovací jazyk 4. generace pro řešení technických výpočtů. Program Matlab je vyvíjen společností MathWorks, tato společnost byla založena roku 1984 a hlavní sídlo má v americkém městě Natick, stát Massachusetts. Program je vyvíjen pro platformy Microsoft Windows, Linux a Mac OS.

Matlab je využitelný v širokém spektru oblastí, jednotlivé rozšíření pro specifické prostředí se nazývají toolbox.

Příklady oblastí:

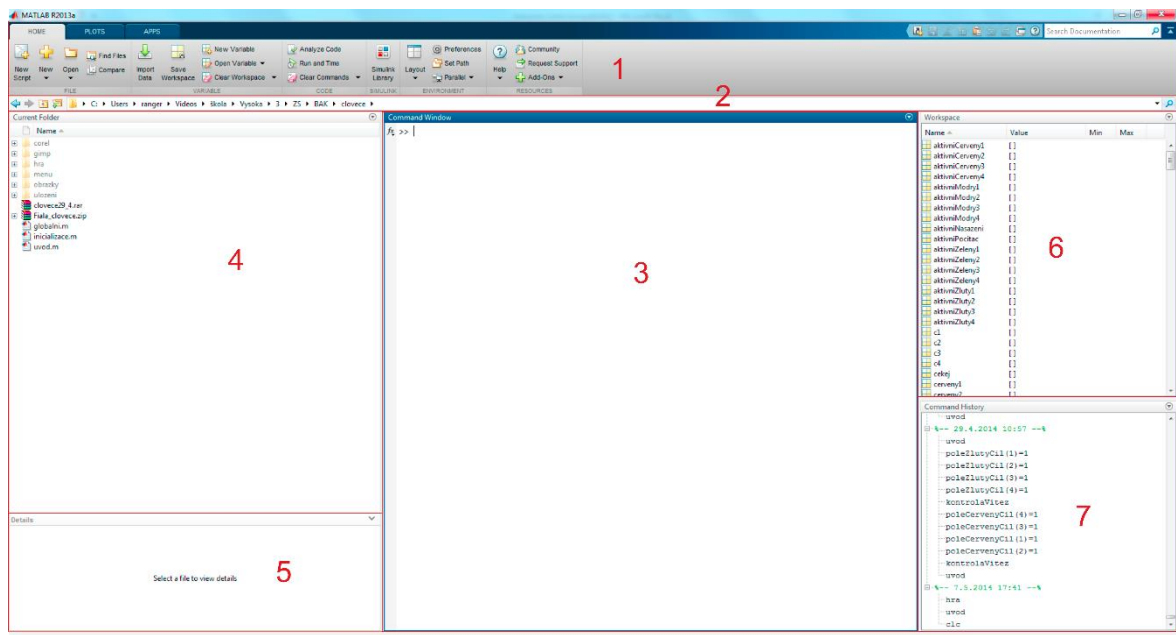
- Matematika, Statistika, Optimalizace
- Návrh řídicích systémů a analýza
- Zpracování signálu a komunikace
- Testování a měření
- Zpracování obrazu a počítačové vidění
- Vícevláknové programování
- Výpočetní finance a biologie
- Bioinformatika
- A další [9]

### 1.2 Popis prostředí

Po zapnutí Matlabu se objeví úvodní obrazovka, která má následující prvky:

1. Menu toolbar
2. Průzkumník
3. Příkazové okno
4. Okno aktuální složky
5. Náhledové okno
6. Workspace

## 7. Historie



Obrázek 1. Náhled úvodní obrazovky Matlabu

### 1.2.1 Menu toolbar

Jedná se o Menu celého programu, je rozděleno do tří karet. Každá karta je poté rozdělena do jednotlivých sektorů podle funkcionalit. Lze si vytvořit i svou vlastní kartu se svými zkratkami.

#### Home

Karta pro základní ovládání a nastavení Matlabu.

**FILE** – Umožňuje vytvořit nové objekty Př.: skript, funkce, GUI, okna. Dál lze otevírat už existující soubory, vyhledávat soubory nebo porovnávat obsah dvou souborů.

**VARIABLE** – Zde lze vytvářet, otevírat a mazat proměnné ve *workspace*. Samotné *workspace* lze také ukládat na disk nebo je otevírat.

**CODE** – Kontrola syntaxe kódu, jeho spuštění, mazání příkazového okna nebo okna historie.

**SIMULINK** – Otevírá knihovnu Simulink.

**ENVIROMENT** – Nastavení rozložení výchozí obrazovky, nastavení celého Matlabu, nastavení výchozí cesty pro vyhledávání a nastavení pro paralelní programování.

**RESOURCES** – Obsahuje nápovědu Matlabu, přístup ke komunitě a technické pomoci a spravování přídatných doplňků.

## Plots

Karta nabízí možnosti zobrazení dané proměnné.

*SELECTION* – Slouží pro výběr proměnné.

*PLOTS* – Pro vybranou proměnnou nabízí možné funkce zobrazení do nového okna. Pro pole například sestavení různých grafů. Pro obrázek jeho zobrazení.

*OPTIONS* – Možnost zobrazení do nového nebo do stávajícího okna.

## Apps

Karta pro správu aplikací, označených jako Apps.

*FILE* – Možnost získat novou Apps, instalovat staženou Apps nebo ji vytvořit.

*APPS* – Přehled nainstalovaných Apps a přístup k nim.

### 1.2.2 Průzkumník

Okno zobrazuje cestu k aktuálnímu adresáři. Lze adresáře procházet přes průzkumníka Windows nebo pomocí zkratk měnit úroveň adresářů, či v nich vyhledávat.

### 1.2.3 Příkazové okno

Jedná se výchozí okno, kde se na řádek zapisuje příkaz. Výsledky se mohou také v okně zobrazovat. Okno uchovává historii příkazů a výsledků, lze jej mazat.

### 1.2.4 Okno aktuální složky

Zde se zobrazují soubory obsažené v aktuálním adresáři, soubory zde lze měnit, přesunovat, kopírovat, procházet. Lze procházet i archivované soubory s koncovkou *.zip*.

### 1.2.5 Náhledové okno

Pokud to jde, zobrazuje náhled souboru označeného v okně aktuální složky. V případě obrázku zobrazuje obrázek, v případě souboru typu *.m* zobrazuje popis skriptu/funkce. V případě souboru *.mat* zobrazuje obsah celého archívu, tedy proměnné i jejich data.

### 1.2.6 Workspace

Velice důležité okno, které zobrazuje proměnné v paměti. Zobrazuje se jejich název, struktura, či rozměr a uložená data.

### 1.2.7 Historie

Zde se oproti příkazovému oknu zobrazuje pouze historie příkazů, ne tedy jejich výsledek. I po smazání příkazového okna historie zůstává.

## 1.3 Užitečné příkazy a funkce

Pro práci s Matlabem je dobré znát příkazy, které uživatel nepoužívá k řešení výpočetního problému, ale pro řešení problémů se samotným programem. Tedy uvedu uživatelsky nápomocné příkazy:

- *help* – Vypíše seznam všech primárních témat, ke kterým existuje nápověda. Při doplnění 2. argumentu příkazové řádky, se vypíše, pokud existuje, nápověda k danému argumentu, například příkaz *help help* vypíše nápovědu k používání nápovědy, *helpwin* zobrazí nápovědu v novém okně.
- *who* – Vypíše seznam všech proměnných v paměti, *whos* vypíše i jejich vlastnosti.
- *clc* – Vymaže celé příkazové okno.
- *clear* – Při vložení 2. argumentu vymaže danou proměnnou, při příkazu *clear all* vymaže celý *workspace*.
- *iskeyword* – Vypíše seznam rezervovaných slov používaných systémem.

V příkazovém okně lze šipkami nahoru a dolů procházet historii příkazů, což se využije například při napsání dlouhého chybného zápisu vzorce, stačí se tedy vrátit na tento příkaz a opravit chybu.

Do jednoho řádku také lze psát více příkazů a je více způsobů, jak je oddělovat. Při oddělování čárkou se příkazy provedou s výpisem výsledku. Při oddělení středníkem dojde k zpracování příkazu, ale ne k vypsání výsledku.

Příkaz je možné psát i na více řádků, stačí pouze na konec řádky napsat tři tečky. Komentář se píše pomocí znaku *%* na začátek řádku. Další důležitou funkcí je zkratka *CTRL + C*, která ukončí aktuálně běžící úlohu. [4,8]

Příklad psaní více příkazů s výpisem:

```
>> a=5,b=10
```

```
a =
```

```
b =  
    10
```

Příklad psaní na více řádcích:

```
>> a...  
+...  
10  
ans =  
    15
```

## 1.4 Datové typy a proměnné

Matlab při deklaraci proměnné nemusí znát její typovou hodnotu, zjistí ji až při přiřazení. Jedná se tedy o dynamické typování. Nedeklarovaná proměnná je prázdná matice. Například proměnnou typu *integer* tedy lze přepsat typem *string*. Z jednoho pohledu je tento způsob jednodušší pro práci uživatele, z druhého pohledu se takto může dojít jednoduše k chybě, když přetypování nevyvolá výjimku. Ošetření tedy musí spravit sám programátor. Jazyk Matlab je *Case sensitive*, tedy rozlišuje velká a malá znaménka. Proměnná musí začínat písmenem. [2,4]

Pro deklaraci proměnných jsou zde ještě pravidla rezervovaných názvů, které Matlab používá a nejdou přepsat, jsou to: *break, case, catch, classdef, continue, else, elseif, end, for, function, global, if, otherwise, parfor, persistent, return, spmd, switch, try, while*. [8]

Další názvy proměnných, které jsou používány systémem, ale jdou přepsat, jsou:

*ans* – výsledek předešlého výpočtu

*i, j* – symboly komplexních proměnných

*pi* – Ludolfovo číslo

*inf* - nekonečno

*NaN* – neplatná numerická hodnota

Tyto hodnoty tedy lze přepsat, ale při jejich vymazání z paměti nebo znovu definování je Matlabem mají zase svou původní hodnotu. [1]

Jako základní typ proměnných vytvořených v příkazovém řádku nebo ve skriptech jsou lokální proměnné, jsou uloženy ve *workspace*. Funkce mají také lokální proměnné, ale uložené v lokálním prostoru funkce, nedostupné z *workspace*. V případě, když je více funkcí, tak každá funkce má své vlastní lokální proměnné. Znamená to tedy, že ve dvou funkcích je proměnná se stejným jménem, ale jinou hodnotou. Jestliže chceme, aby funkce pracovali se stejnou proměnnou, musíme ji deklarovat jako globální pomocí funkce *global*. Stejně jako u lokálních proměnných i při deklaraci globální proměnné bez inicializace se jedná o prázdnou matici. Jestliže už existuje ve *workspace* lokální proměnná se stejným jménem jako globální, Matlab na tohle upozorní a změní hodnotu lokální proměnné na globální. Podobným typem jsou perzistentní proměnné, deklarace pomocí příkazu *persistent*. Stejně jako globální mají stálé uložení a nezaniknou při ukončení funkce jako proměnné lokální. Viditelné jsou pouze ve své funkci, nemůžou být tedy změněny z jiných funkcí. Při mazání funkce se smažou i její perzistentní proměnné. Perzistentní proměnná také nesmí být deklarována jako vstupní proměnná funkce. [ 1, 4, 3]

#### 1.4.1 Numerický datový typ

Matlab všechny číselné typy ukládá jako proměnnou *double*.

##### Double

Jedná se o datový typ s plovoucí řadovou čárkou. Jedná se o způsob reprezentace čísel moc velkých nebo moc malých pro zobrazení v pevné řadové čárce. Tedy typ s dvojitou přesností a velikostí 8B.

##### Single

Také se jedná o typ s plovoucí řadovou čárkou, ale s jednoduchou přesností. Jeho velikost je dvakrát menší, než datový typ *double*, tedy 4B. Menší velikost tedy zaručuje i rychlejší řešení algoritmu.

##### int8, int16, int32, int64

Jsou to celočíselné hodnoty s rozsahem, podle svého označení. Například *int16* má rozsah 65535 čísel, ale jelikož se může jednat o kladné i záporné čísla, včetně nuly, rozsah se vypočítá nesledujícím způsobem:

$$-2^{15} = -32768$$

$$2^{15} - 1 = 32767$$

Rozsah je od -32768 do 32767. Datový typ zabírá 16 bitů.

### **uint8, uint16, uint32, uint64**

Opět se jedná o celočíselné hodnoty s rozsahem podle konkrétního typu. Ale tento typ je pro bezznaménková čísla, včetně nuly. Jako v předešlém případě provedu výpočet rozsahu u 16 bitové proměnné *uint16*:

$$2^{16} - 1 = 65535$$

Rozsah je od 0 do 65535. Datový typ zabírá 16 bitů. [8]

### **Užitečné příkazy**

Numerické datové typy lze mezi sebou jednoduše převádět pomocí funkcí datových typů.

Příklady převedení datového typu *double* na *int16*:

Příklad1:

```
>> a=12.8
a =
    12.8000
>> b=int16(a)
b =
    13
```

Příklad2:

```
>>a =1.1611e+14
>> b=int16(a)
b =
    32767
```

Z příkladu 1 lze vidět, že při převodu z neceločíselného typu na celočíselný, se zaokrouhlí hodnota k nejbližšímu celému číslu.

V příkladu 2 byla hodnota *a* větší, než maximální velikost *int16*, proto byla do proměnné *b* zapsána právě tato hraniční hodnota datového typu *int16*. Převod ostatním numerických typů funguje na stejném principu.

Ověřovací příkazy kontrolují vstupy a testují je danou podmínku, vrací *1*, pokud vstup vyhovuje, *0* pokud nevyhovuje.

*isinteger* – Kontroluje, zda je na vstupu celočíselný datový typ.

*isfloat* – Kontroluje, jestliže je na vstupu datový typ s plovoucí čárkou.

*isnumeric* – Kontroluje, jestli se jedná o číselnou hodnotu.

*isreal* – Funkce zjišťuje, jedná-li se o reálné číslo, neobsahuje imaginární část.

Další příkazy:

*format* – Přesnost zobrazení proměnné při výpisu na obrazovku. Jedná se pouze o reprezentaci dat, ne o jejich skutečnou přesnost, ta zůstává stejná.

*ceil* – Zaokrouhluje na nejbližší celočíselné číslo.

*floor* – Zaokrouhluje na nejbližší nejnižší celočíselné číslo.

*round* – Zaokrouhluje na nejbližší celočíselné číslo.

#### 1.4.2 Datový typ char

Řetězec, neboli v Matlabu *string*, se zapisuje v apostrofech a jsou složeny právě z datových typů *char*.

```
>> retezec = 'ahoj'
```

```
retezec =
```

```
ahoj
```

Velikost řetězce odpovídá počtu jeho prvků. Jedná se vlastně o vektor obsahující jednotlivá písmena jako prvky. Pro uživatele se zobrazují dané znaky, ale v paměti jsou znaky převedeny na číselné znaky ASCII tabulky. [5]

Funkce pro práci s řetězci:

*num2str* – Slouží ke konverzi numerického typu na řetězec.

*strjoin* – Spojuje více řetězců do jednoho.

*strcmp* – Porovnává řetězce mezi sebou. Může porovnávat i řetězec a vektor, či matici řetězců. Jestliže se řetězce rovnají, výstup je *1*, jestli se nerovnají *0*. V případě porovnávání v matici je výstup matice s *0* a *1*. Jsou i funkce pro porovnávání *case insensitive*.

```
>> a = 'tři';
```

```
>> b={'jedna', 'dva' ; 'tři', 'čtyři'};
>> strcmp(a,b)

ans =

     0     0
     1     0
```

### 1.4.3 Datový typ cell

Tento datový typ obsahuje indexované datové kontejnery, tedy jedná se o vektor, či matici s prvky, které nemusí být stejného datového typu. Každá buňka, tedy *cell*, může obsahovat libovolný datový prvek. [4, 8]

Deklarujeme ji následovně:

```
>> bunka(1,1) = {'jedna'};
>> bunka(2,1) = {1000};
>> bunka(3,1) = {[2, 100]}

bunka =

     'jedna'
     [     1000]
     [1x2 double]
```

Jedná se tedy o datový typ *cell* s velikostí 3x1. První řádek obsahuje *string*, v druhém je číslo *double* a v posledním řádku je pole s dvěma čísly typu *double*.

K jednotlivým prvkům se přistupuje následovně:

```
>> bunka(3)

ans =

     [1x2 double]
```

### 1.4.4 Datový typ struktura

Struktura je datový typ, která obsahuje více různorodých datových typů, takzvaný kontejner. Je to tedy mnohorozměrné pole. Struktura obsahuje název struktury. Pole struktury jsou názvy polí, do kterých se ukládají data. K jednotlivým polím struktury se přistupuje přes tečku.

1. způsob zápisu

```
>> telefon.vyrobce = 'LG';
```

```
>> telefon.nazev = 'Optimus L9';  
>> telefon.cena = 6000;  
>> telefon.rozliseni = '540 x 960';
```

## 2. způsob zápisu

```
>> telefon(2)=struct('vyrobce','Samsung','nazev','Galaxy  
Ace','cena',3000,'rozliseni','320 x 480')
```

K jednotlivým prvkům struktury se přistupuje jako k poli.

```
>> telefon(2)  
  
ans =  
  
    vyrobce: 'Samsung'  
      nazev: 'Galaxy Ace'  
       cena: 3000  
rozliseni: '320 x 480'
```

### 1.4.5 Datový typ handle funkce

Tento datový typ obsahuje informace pro odkazování na funkci obsaženou v Matlabu nebo funkci vytvořenou uživatelem. V *handle* jsou uloženy všechny potřebné informace o funkci a informace potřebné ke spuštění, pomocí příkazu *feval*. Dále umožňuje přistupovat k metodám dané funkce. Tento datový typ lze používat jako ostatní, tedy vkládat do matic, polí, či struktur.

Výhody používání handle funkce:

- umožňuje přístup k funkci jinými funkcemi
- zachycuje všechny metody přetížené funkce
- umožňuje širší přístup k subfunkcím a privátním funkcím
- zvyšuje čitelnost
- snižuje počet souborů
- zvyšuje výkon programu při opakovaných operacích [4]

Příklad vytvoření funkce a následného vytvoření handle funkce:

```
function c=soucet(a,b)  
c=a+b;  
end
```

```
>> h_soucet = @soucet;
```

```
>> feval(h_soucet, 2, 6)
```

```
ans =
```

```
8
```

Pomocí funkce *functions* lze zjistit informace o daném handlu.

```
>> functions(h_soucet)
```

```
ans =
```

```
function: 'jedna'  
type: 'simple'  
file: [1x66 char]
```

## 1.5 Operátory

### 1.5.1 Relační operátory

Relační operátory slouží pro porovnávání dvou hodnot. Lze porovnávat i pole se stejnou velikostí. Výstup tohoto porovnávání je při pravdivosti *true* nebo *1*, a při nepravdivosti *false* nebo *0*. Je logické, že využití je především v podmínkách.

< - menší než

> - větší než

== - rovnost

<= - menší nebo rovno

>= - větší nebo rovno

~= - nerovnost

Porovnání jedné hodnoty a pole:

```
>> a = 5;
```

```
>> b = [ 4 5 6 ];
```

```
>> a == b
```

```
ans =
```

```
0     1     0
```

```
>> a>=b
```

```
ans =
```

```
1     1     0
```

Porovnání dvou polí:

```
>> a = [ 4 4 5 6 ];
>> b = [ 3 4 1 6 ];
>> a<=b

ans =

     0     1     0     1
```

## 1.5.2 Logické operátory

Jedná se o operátory pracující s logickými hodnotami *true* a *false*, popřípadě 1 a 0.

| - logický součet

|| - logický součet podmínkový

& - logický součin

&& - logický součin podmínkový

~ - negace

Příklad:

```
>> a= [1 0 1 0];
>> b= [0 0 1 1];
>> c=a&b

c =

     0     0     1     0

>> ~c

ans =

     1     1     0     1
```

## 1.6 Podmínky a cykly

### 1.6.1 Podmínky if, elseif, else

Podmínky slouží k větvení programu na základě testované proměnné. Základní příkaz pro podmínku je *if*, jako argument následuje logický výraz, který má vždy logický výsledek. V podmínkovém bloku lze mít více podmínek po sobě pomocí příkazu *elseif*. V logických výrazech lze testovat pokaždé jinou proměnnou. Pokud je splněna některá z podmínek, tak

další podmínky už se nekontrolují a program pokračuje až od příkazu *end*, který ukončuje blok podmínky. Jestliže není splněna ani jedna z podmínek, vykonávají se příkazy pod příkazem *else*, který už neobsahuje argument. Příkazy *elseif* a *else* jsou nepovinné.

V příkazovém řádku se inicializuje proměnná *cislo* a je zapsána podmínka *if*:

```
>> cislo = 11;

>> if cislo < 10
    disp('Číslo je menší');
elseif cislo == 10
    disp('Číslo je deset');
else
    disp('Číslo je větší');
end
Je větší
```

### 1.6.2 Příkaz switch

Slouží stejně jako podmínka k větvení programu. Po příkazu *switch* následuje proměnná, která bude testována. Po příkazu *case* se udává hodnota nebo pole hodnot, podle které dochází k větvení, jestliže není splněn ani jeden *case*, provede se část kódu pod příkazem *otherwise*, ten také neobsahuje argument, stejně jako příkaz *else*. Větvící blok je ukončován příkazem *end*. Jako u podmínky *if*, pokud je splněn jeden případ *case*, další už nejsou vyhodnocovány.

V příkazovém řádku se inicializuje proměnná *cislo* a je zapsána větvící příkaz *switch*:

```
>> cislo=11;

>> switch(cislo)
    case 11
        disp('Číslo je 11')
    case {10, 11}
        disp('Číslo je 10 nebo 11')
end
Číslo je 11
```

### 1.6.3 Cyklus while

Pro vícenásobné opakování části kódu programu, při neznalosti počtu opakování, se používá příkaz *while*. Jako argument se zadává podmínka. Cyklus se opakuje, dokud je podmínka pravdivá. Cyklus se ukončuje příkazem *end*. Jako ukončovací podmínku lze zapsat hodnotu *1* a ukončit cyklus pomocí příkazu *break*. Často je tento způsob používán, kdy *break* bývá dostupný až po splnění určité podmínky uvnitř cyklu.

Jednoduchý příklad použití příkazu *while*:

```
>> k=0;

>> while k<2
    disp(k);
    k=k+1;
end
    0
    1
```

### 1.6.4 Cyklus for

Cyklus *for* se používá pro vícenásobné opakování části kódu programu, při znalosti počtu opakování. Za příkaz *for* se zadává výraz, který určuje počet cyklů. Proměnné se přiřazuje vektor hodnot pomocí dvojtečkové syntaxe, která slouží pro generování čísel vektoru, neboli pole. První číslo určuje startovní hodnotu, druhé velikost kroku, třetí konečnou hodnotu, která je konečná a nemůže být překročena.

Příklad pro vypsání aktuálního čísla v cyklu *for*:

```
>> for i=1:2:6
    disp('Číslo je:')
    disp(i)
end
Číslo je:
    1
Číslo je:
    3
Číslo je:
    5
```

## 1.7 Práce s maticemi

### 1.7.1 Definice a přístup

Jelikož matice a vektory jsou základními prvky Matlabu, je níže uvedena základní práce s těmito prvky a užitečné funkce.

Základ je správné definování vektoru a matice. Vektor se definuje v hranatých závorkách, jednotlivé hodnoty jsou od sebe odděleny mezerou nebo čárkou.

```
>> vektor = [1,2,3,4,5]

vektor =

    1     2     3     4     5
```

Zápis prvků matice je také mezi hranatými závorkami, jednotlivé řádky jsou od sebe odděleny středníkem.

```
>> matice = [1 2 3; 4 5 6; 7 8 9]
```

```
matice =
```

```
    1    2    3
    4    5    6
    7    8    9
```

Přístup k jednotlivým prvkům ve vektoru je index hledaného prvku neboli sloupce. V matici jsou to souřadnice prvku, tedy řádek a sloupec. Souřadnice se zadávají v kulatých závorkách.

Přístup k prvku vektoru:

```
>> vektor(3)
```

```
ans =
```

```
    3
```

Přístup k prvku matice:

```
>> matice(2,3)
```

```
ans =
```

```
    6
```

V matici lze vypsát i například všechny řádky v jednom sloupci nebo všechny sloupce v jednom řádku, používá se dvojtečková syntaxe.

```
>> matice(2,:) 
```

```
ans =
```

```
    4    5    6
```

```
>> matice(:,3)
```

```
ans =
```

```
    3
```

```
    6
```

```
    9
```

Vypsát lze i submatice, také pomocí dvojtečky vytvářející aritmetickou posloupnost.

```
>> matice(2:3,1:2)
```

```
ans =
```

```
    4    5  
    7    8
```

Pro zjištění velikosti matice slouží funkce *size*. Návrátová hodnota funkce je pole s dvěma prvky, první udává počet řádků, druhý počet sloupců.

```
>> [pocetRadku, pocetSloupce]=size(matice)
```

```
pocetRadku =
```

```
    3
```

```
pocetSloupce =
```

```
    3
```

Některé z funkcí pro vytvoření matic:

*zeros* – vytvoří matici s nulovými prvky o zadaných rozměrech

*ones* – vytvoří matici naplněnou jedničkami o zadaných rozměrech

*eye* – vytvoří jednotkovou matici o zadaných rozměrech

*rand* – vytvoří matici s náhodnými hodnotami o zadaných rozměrech

### 1.7.2 Operace s maticemi

Matice lze mezi sebou sčítat, násobit, získat inverzní matici. Při aritmetických operacích se samozřejmě musí dodržovat matematická pravidla pro práci s maticemi.

Sčítání a násobení matic:

```
>> A= matice + matice
```

```
A =
```

```
    2    4    6  
    8   10   12  
   14   16   18
```

```
>> A= matice * matice
```

```
A =
```

```
   30   36   42  
   66   81   96  
  102  126  150
```

Zápisem apostrofu se získá transponovaná matice.

```
>> A=matice'
```

```
A =
```

```
     1     4     7
     2     5     8
     3     6     9
```

Některé z funkcí pro práci s maticemi:

*inv* – výpočet inverzní matice

*det* – výpočet determinantu matice

*eig* – výpočet vlastních čísel a vlastních vektorů matice

*rank* – výpočet hodnosti matice

Přehled funkcí pro práci s maticemi lze například nalézt v nápovědě, pomocí příkazu *help matfun*.

## 1.8 Práce se soubory

Při práci s daty v Matlabu je možné si dané data ukládat, popřípadě je načítat. Matlab umožňuje využívat vlastní formát *.mat* nebo pracovat se známými formáty jiných programů. Ukládání lze provádět v binárním, či textovém tvaru. [8]

### 1.8.1 Binární ukládání

Pro ukládání dat slouží příkaz *save*. Při použití příkazu bez parametru, tedy jména souboru se uloží defaultně jako *matlab.mat* se všemi proměnnými v pracovním prostoru, tedy *workspace*. Při zapsání prvního parametru určujeme jméno souboru, další parametry jsou konkrétní proměnné, které se mají do souboru uložit. [8]

Příklad:

V pracovním prostoru jsou proměnné a, b, c.

Metoda pro uložení všech aktuálních proměnných s defaultním názvem souboru:

```
save
```

Metoda pro uložení všech aktuálních proměnných do souboru *soubor.mat*:

```
save soubor
```

Metoda pro uložení vybraných proměnných  $a$ ,  $c$  do souboru *soubor.mat*:

```
save soubor a c
```

Obdobně jako ukládání se provádí i načítání, a to pomocí příkazu *load*. Jako parametr se uvádí název souboru, případně jako další parametry se uvádějí konkrétní proměnné.

Příklad:

Soubor *soubor.mat* obsahuje proměnné  $a, b, c$ .

Metoda pro načtení všech proměnných:

```
load soubor.mat
```

Metoda pro načtení konkrétní proměnné:

```
load soubor.mat a
```

Pomocí příkazu *whos* se dá jednoduše zjistit, jaké proměnné soubor uchovává:

```
>> whos -file soubor.mat
```

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	
b	1x1	8	double	
c	1x4	8	char	

### 1.8.2 Textové ukládání

Textový formát lze číst v libovolném textovém editoru, ale tento formát neobsahuje informace o rozměrech a jménech proměnných, pouze jejich hodnotu. Princip ukládání v textovém formátu je stejný jako u binárního ukládání, je ale potřeba dopsat klíčový parametr – *ascii*. Při načítání se data souboru ukládají do matice se jménem, jako je jméno souboru. Data musí mít stejnou délku, jinak se nepodaří načíst. [8]

Následující příklad demonstruje ukládání a načítání souborů v textovém formátu:

```
save -ascii data.dat a b c
```

Stejným způsobem probíhá i načítání dat:

```
load data.dat -ascii
```

Kontrola obsahu dané matice v paměti:

```
>> whos data
```

Name	Size	Bytes	Class	Attributes
data	3x1	24	double	

Příkazy *save* a *load* mají také funkční zápis:

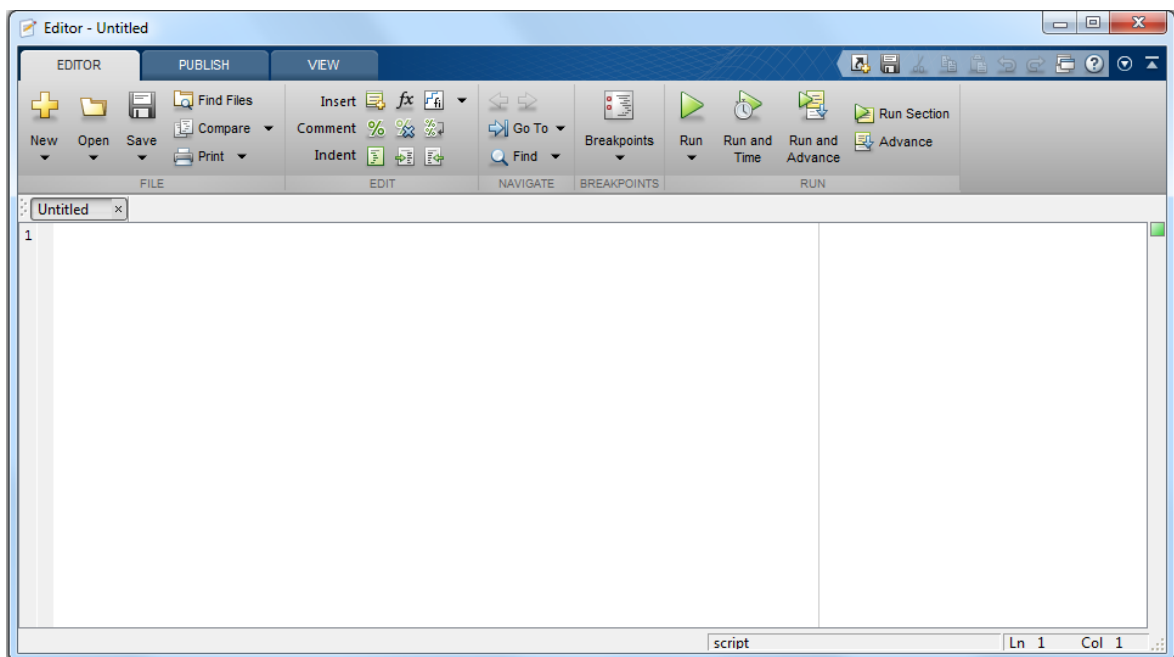
```
save('soubor.mat', 'a', 'b')  
jmeno = 'soubor';  
load(jmeno)
```

## 1.9 M - soubory

Při využívání Matlabu pro jednorázové výpočty, které obsahují pár příkazů je dostačující příkazový řádek. Pokud se ale výpočty budou často opakovat a využívat, je vhodné si je uložit do tzv. M – souborů. Při práci se složitějšími algoritmy je tohle nutností. M – soubor slouží pro ukládání posloupnosti příkazů. Tento soubor je uložen na pevném disku a vyvolává se z příkazového řádku Matlabu. Edituje se pomocí Matlab Editoru. M – soubory se dělí na dva typy - skript a funkce. [6]

### 1.9.1 Matlab Editor

Slouží pro vytváření, editaci, debugingu skriptů, či funkcí. Spustit se dá několika způsoby, v záložce *Home* -> *New*->*Script/Function*, zkratkou *Ctrl + N*, příkazem *edit* nebo po klikáním na již existující soubor v okně aktuální složky.



Obrázek 2. Obrazovka Matlab Editor

Matlab Editor se dělí na 3 karty, každá karta obsahuje několik sektorů s funkcemi. Pod nimi se vyskytuje *bar*, neboli pruh, kde jsou znázorněny otevřené soubory. V pravém dolním rohu se zobrazuje aktuální poloha kurzoru vzhledem k řádkům a sloupcům, nalevo od něj je identifikátor souboru, jestli se jedná o skript nebo funkci.

#### Editor

**FILE** – Slouží pro vytvoření nových dokumentů, otevírání, ukládání, vyhledávání dokumentů, porovnávání dvou dokumentů a tisk.

*EDIT* – Umožňuje rozdělit kód do sekcí pro lepší orientaci, vkládat funkce a datové typy, vkládat komentáře a odstraňovat komentáře, formátovat vzhled kódu.

*NAVIGATE* – Slouží pro pohyb uvnitř kódu na konkrétní řádek, na naposled upravený řádek, dále lze vyhledávat v kódu, vkládat záložky.

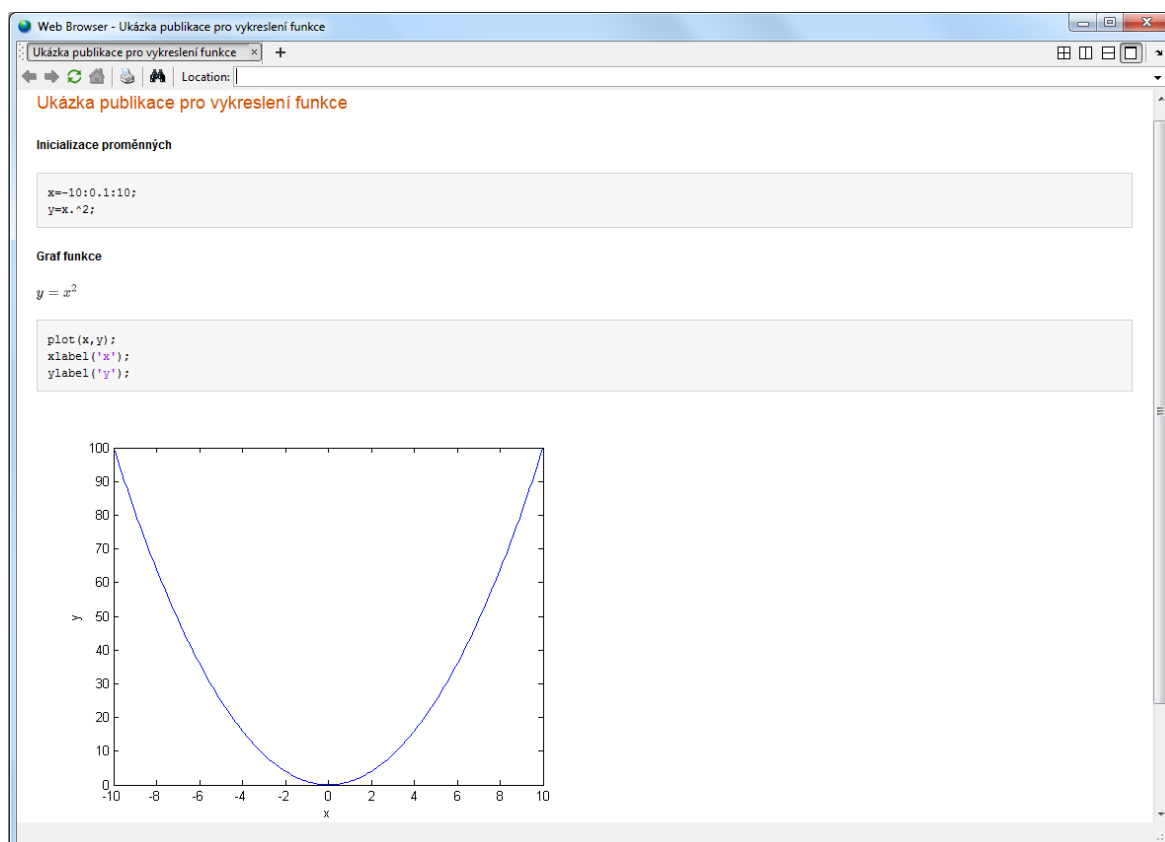
*BREAKPOINTS* – Zarážky slouží k ladění kódu, nejčastěji pro hledání chyb, tato sekce umožňuje zarážky vytvářet, mazat, nastavovat.

*RUN* – Spouští celý kód, spouští celý kód a měří dobu spuštění, spouští pouze vybrané sekce a prochází mezi nimi.

## Publish

Matlab umožňuje napsaný kód publikovat jako tzv. publikaci. Do kódu se musí umístit speciální značky pro výslednou reprezentaci. Lze umístit nadpisy, odrážky, rozdělovat kód do bloků, oddělovat kód od psaného textu, přikládat obrázky. Výsledný soubor je v *html* formátu. Takto je psána i nápověda v Matlabu.

Příklad výstupní publikace:



Obrázek 3. Ukázka Matlab publikace

*FILE* – Obsahuje funkci pro ukládání souborů.

*INSERT SECTION* – Rozděluje kód na sekce s možností názvu sekce.

*INSERT INLINE MARKUP* – Vkládá značky do kódu, pro reprezentaci textu tučným písmem, se zkosením, bezpatkovým písmem. Slouží pro vkládání hyper odkazů, pro převod matematických zápisů psaných programovacím jazykem do obecné podoby.

*INSERT BLOCK MARKUP* – Vkládá do publikace odrážky, obrázky, kód.

*PUBLISH* – Generuje a nastavuje výstupní publikaci.

## **VIEW**

*TITLES* – Při otevření více souborů lze rozdělit okno Editoru na poloviny, či čtvrtiny, kde každá oblast obsahuje jiný otevřený soubor.

*DOCUMENT BAR* – Nastavuje viditelnost pruhu a jeho seřazení podle abecedy nebo podle času otevření.

*SPLIT DOCUMENT* – Umožňuje rozdělit otevřený soubor na dva bloky.

*CODE FOLDING* – Expanduje nebo smršťuje napsaný kód. Provádět to jde pouze u určitých funkcí, například při definici funkce, cyklu *for* a *while*.

*DISPLAY* – Možnost zvýraznění aktuálního řádku, číslování řádků a při najetí myši na proměnnou zobrazení jejího typu a aktuálního obsahu.

### **1.9.2 Skript**

Slouží pro vkládání posloupnosti příkazů, příkazy se dají snadno editovat a přesouvat, není zde limitování jako u příkazového řádku. Napsané příkazy se ve stejném pořadí, jak jsou napsány i volají. Skript se musí ukládat na pevný disk a až poté ho spustit. Spouštění se provádí přímo v Editoru, v příkazovém řádku, pomocí jména skriptu bez koncovky nebo voláním v jiném skriptu, či funkci. Proměnné obsažené ve skriptu jsou lokální a uloženy do pracovního prostoru, po ukončení skriptu nezanikají. Při vložení komentáře na začátek skriptu se daný komentář zobrazuje v nápovědě k danému skriptu, dokud není komentář přerušen kódem nebo prázdným řádkem. Lze tak snadno zjistit k čemu soubor je, či co obsahuje bez jeho otevírání. [4, 6]

Příklad kódu pro publikaci:

```
%% Ukázka publikace pro vykreslení funkce
% *Inicializace proměnných*
```

```
x=-10:0.1:10;
y=x.^2;

%%
% *Graf funkce*
%%
% $y=x^2$
plot(x,y);
xlabel('x');
ylabel('y');
```

### 1.9.3 Funkce

Funkce vždy začíná klíčovým slovem *function*, dále následuje výstupní proměnná, jestli je více výstupních proměnných, jsou uvnitř hranatých závorek odděleny čárkou. Po výstupních proměnných následuje symbol přiřazení s názvem funkce, který obsahuje v kulatých závorkách vstupní proměnné. Na další řádek lze napsat komentář jako nápovědu, stejně jako u skriptu. Dále následuje sled příkazů, konec funkce je označen příkazem *end*. Název souboru je vždy stejný, jako název funkce. Se vstupními funkcemi lze pracovat pomocí buněčného pole *varargin*, obsahující všechny vstupní proměnné. Ekvivalentem pro výstupní proměnné je buněčné pole *varargout*. Velikost vstupních a výstupních prvků lze zjistit příkazy *nargin* a *nargout*. Proměnné funkce jsou lokální, ale mají vlastní pracovní prostor. Nelze je tedy měnit ze základního pracovního prostoru a ani je vidět, protože zanikají po ukončení funkce. Funkce je při volání napřed celá zkompileována, poté až zpracovávána. [4, 6]

Příklad funkce:

```
function [x1,x2]= podminka(a,b,c)
%Funkce pro výpočet kořenů kvadratické rovnice
if nargin == 3
    D = b.^2-4*a*c;

    if D == 0
        x1 = -b/(2*a);
        x2=x1;
    else
        x1 = -b+sqrt(D)/(2*a);
        x2 = -b-sqrt(D)/(2*a);
    end
else
    Error('Chyba ve vstupních parametrech');
end
end
```

## 1.10 GUI

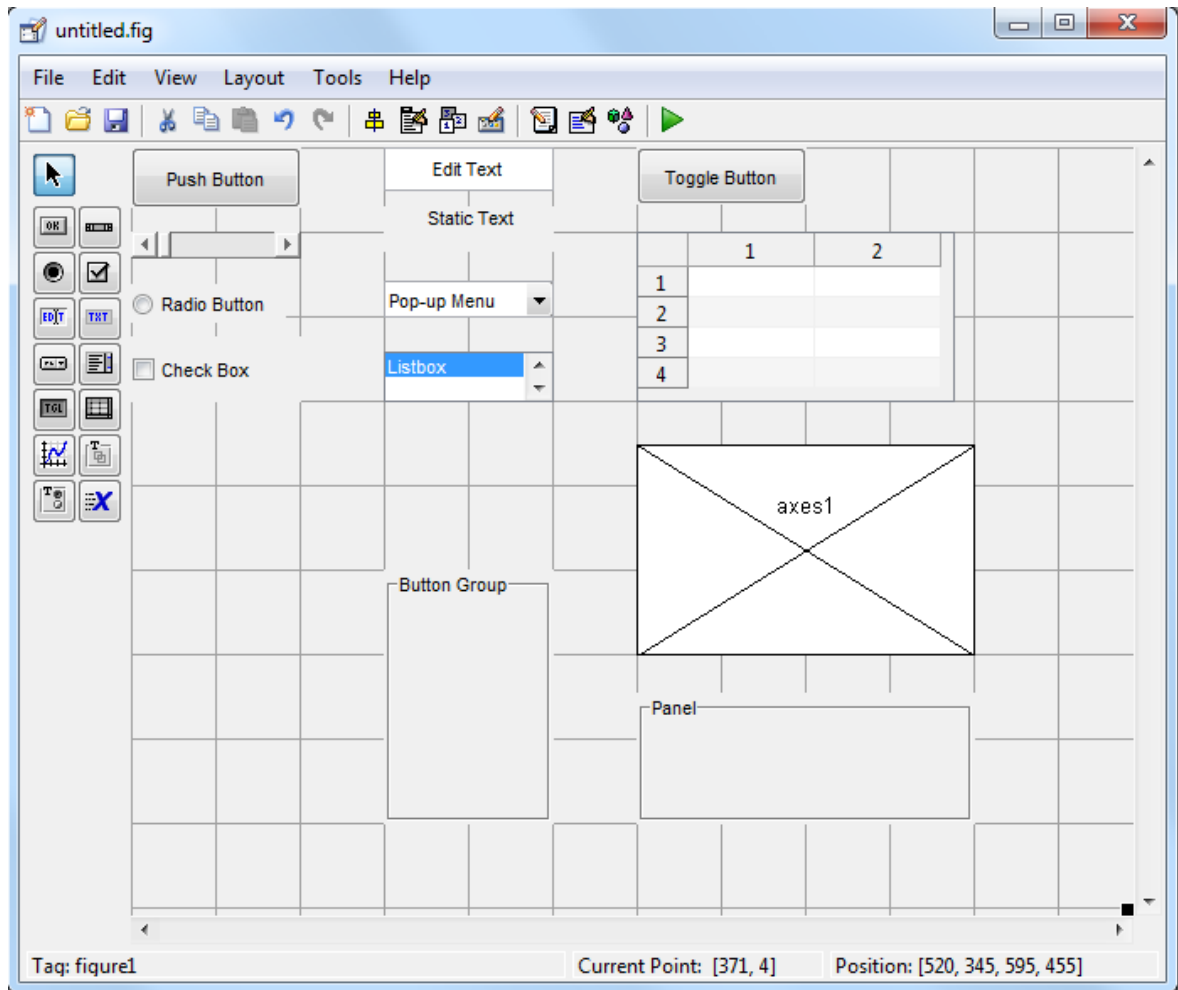
### 1.10.1 GUIDE editor rozložení

Pro tvorbu dialogových oken slouží v Matlabu program GUIDE, tedy GUI editor rozložení. Spuštění programu se provede příkazem *guide* nebo spuštěním z menu Home->New->Graphical User Interface. Lze vytvořit nový prázdný objekt nebo s přednastavenými vzorovými příklady prvky nebo objekt už existující. Okno v Matlabu se nazývá *figure*. Tyto soubory mají koncovku *.fig*.

Po výběru souboru se otevře okno pro tvorbu a editaci okna s ovládacími prvky. Všechny prvky obsahují *callback*, což je volání daného prvku při změně, která nastane, jako je například vytvoření, smazání kliknutí tlačítkem, stisk klávesy. Tyto volání spustí funkci, v které se provede daná obsluha. Každé okno je uloženo v jednom souboru se jménem okna, kód je v souboru generován automaticky podle rozestavení prvků. Tento generovaný kód lze upravovat a doplňovat o vlastní příkazy, jakou jsou například výše zmíněné *callback* funkce.

Seznam prvků:

- Push Button
- Slider
- Radio Button
- Check Box
- Edit Text
- Static Text
- Pop-up Menu
- Listbox
- Toggle Button
- Table
- Axes
- Panel
- Button Group
- ActiveX Control



Obrázek 4. Náhled grafických prvků GUIDE editoru

### 1.10.2 Manuální programování

Tvořit *GUI* objekty je možné i bez pomoci Matlab GUIDE, a to psaním rovnou funkcí daných objektů a jejich nastavení. Výhoda je kontrola nad prvkem, který si může programátor podle sebe nastavit. Nevýhoda je časová náročnost psaní tohoto kódu a znalost všech parametrů, které je potřeba nastavit. Parametry, které nejsou nastaveny uživatelem, jsou nastaveny na defaultní nastavení v daném Matlabu.

### 1.10.3 Figure

Základním grafickým prvkem je okno, ve kterém se umísťují jiné ovládací nebo zobrazovací prvky. Pro vytvoření okna se používá funkce *figure*.

Příklad funkce *figure*:

```
okno=figure('Color',[0.82 0.82 0.82],...
           'MenuBar','None',...
           'Resize','off',...)
```

```
'NumberTitle', 'off', ...
'WindowStyle', 'normal', ...
'Units', 'pixels', ...
'Position', [412 384 400 200]);
```

Ukázka několika parametrů funkce *figure*:

*MenuBar* – Nastavuje zobrazení lišty obsahující menu.

*NumberTitle* – Nastavuje zobrazení jména okna.

*Position* – Určuje pozici a velikost okna.

*Pointer* – Nastavuje vzhled ukazatele myši.

*Resize* – Povoluje změnu velikosti okna.

*Units* – Nastavuje jednotky pro práci s oknem.

*Visible* – Nastavuje viditelnost okna.

*WindowStyle* – Určuje typ okna.

*DeleteFcn* – Volá danou funkci při vymazání okna.

*KeyPressFcn* – Volá danou funkci při stisku tlačítka klávesy.

*WindowButtonDownFcn* – Volá danou funkci při stisku tlačítka myši.

*WindowButtonUpFcn* – Volá danou funkci při puštění tlačítka myši.

*WindowButtonMotionFcn* – Volá danou funkci při pohybu myši v okně.

#### 1.10.4 Uicontrol

Pro vytvoření ovládacích prvků uživatelského rozhraní se používá funkce *uicontrol*. Dle parametrů této funkce se vytváří daný prvek. Konkrétně se jedná o parametr *style*, který může nabývat těchto hodnot: *checkbox*, *edit*, *listbox*, *popupmenu*, *pushbutton*, *radiobutton*, *slider*, *text*, *togglebutton*. Jedná se o ekvivalentní názvy prvků v GUIDE editoru.

Příklad použití *uicontrol* jako tlačítka:

```
tlacitko = uicontrol('Style','text', ...
    'BackgroundColor','red', ...
    'ForegroundColor','black', ...
    'FontSize',25, ...
    'FontWeight','bold', ...
    'Enable','inactive', ...
    'ButtonDownFcn','volaniFunkce', ...
    'Units','pixels', ...
    'Position',[402 50 200 40], ...
    'String','Název prvku');
```

Ukázka několika parametrů funkce *uicontrol*:

*BackgroundColor* – Určuje barvu pozadí.

*ButtonDownFcn* – Volá danou funkci při stisku tlačítka myši.

*Callback* – Volá danou funkci při aktivaci prvku.

*Enable* – Nastavuje aktivitu prvku, jak reaguje na kliknutí.

*FontSize* – Nastavuje velikost písma.

*ForegroundColor* – Určuje barvu popředí.

*FontWeight* – Nastavuje váhu písma.

*Parent* – Nastavuje rodiče prvku, zadává se *handle*.

*Position* - Určuje pozici a velikost prvku v okně.

*String* – Nastavuje řetězec prvku.

*Units* - Nastavuje jednotky pro práci s prvkem.

*Value* – Nastavuje hodnotu prvku.

*Visible* – Nastavuje viditelnosti prvku.

Pro vytváření dalších prvků se používají tyto funkce: *axes*, *uitable*, *uipanel*, *uibuttongroup*, *actxcontrol*.

### 1.10.5 Set, get

Pro nastavení parametrů prvků po inicializaci, za běhu programu slouží funkce *set*. Parametry této funkce jsou *handle* objektu, vlastnost, kterou chceme nastavit a hodnota.

Příklad funkce *set* pro vypnutí viditelnosti okna:

```
set(okno, 'Visible', 'off');
```

Pro získávání parametrů objektu slouží funkce *get*. Parametry funkce jsou *handle* objektu a vlastnost, kterou chceme získat.

Příklad pro zjištění pozice okna:

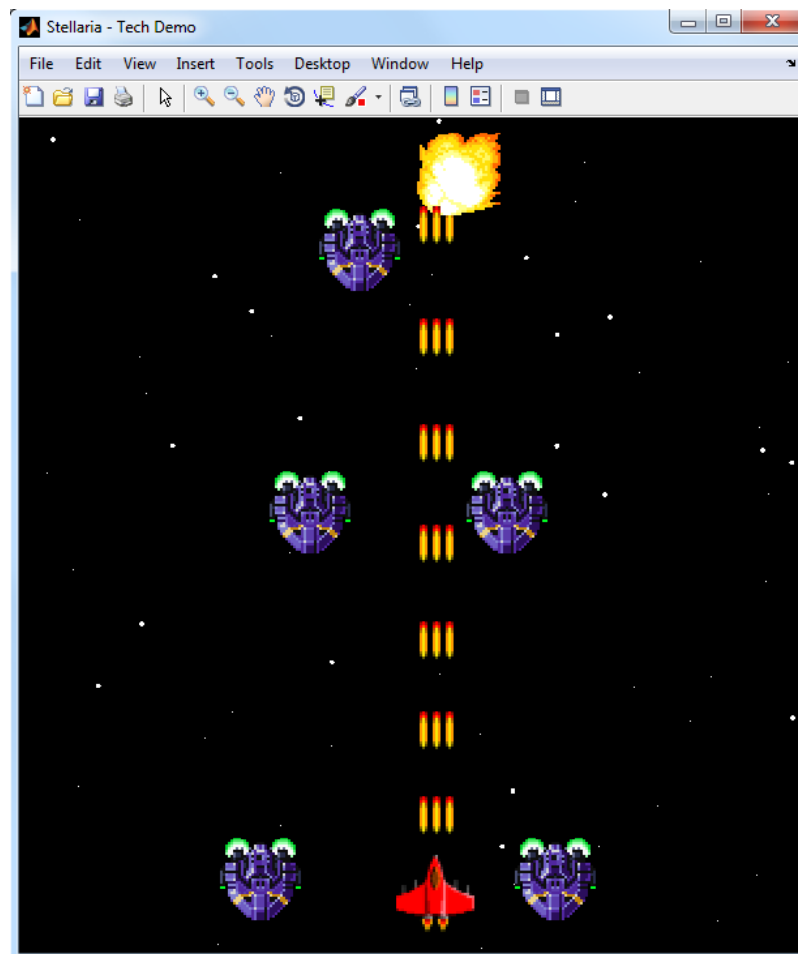
```
get(okno, 'position');
```

## 2 POČÍTAČOVÉ HRY V MATLABU

Uvedené ukázky her jsou staženy ze serveru Matlab Center, kde jsou se svolením jejich tvůrců volně stažitelné nebo se jedná o bakalářské práce. Uvedené hry slouží pro představu, co vše je možné udělat v programovém prostředí Matlab, kromě rutinních výpočtů a grafů. [10]

### 2.1 Stellaria

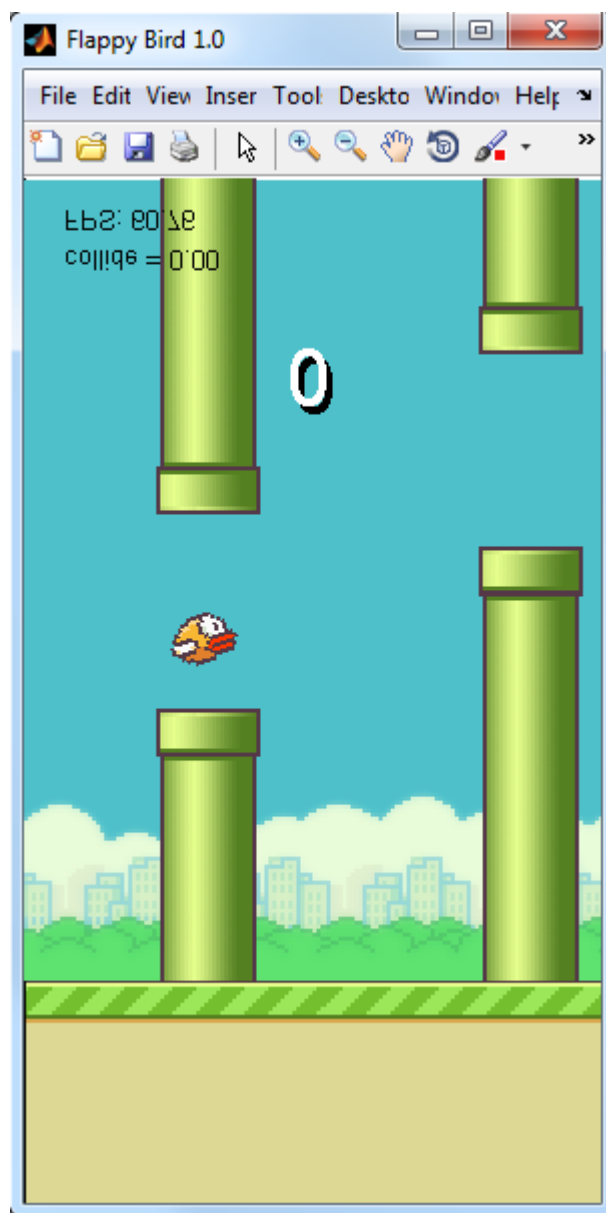
Jedná se o arkádu, klasickou *Shoot-em-up* hru, kde má hráč omezenou kontrolu nad stíhačkou, střílí nepřátele a uhýbá jejich střelám. Hra je pro 1 nebo 2 hráče. Jedná se pouze o demo, obsahující jednu úroveň, nepočítá se skóre a hráč má 30 životů. Jak autor Mingjing Zhang píše, slouží pro demonstraci, čeho je Matlab schopný. Zdrojový kód obsahuje přes 80 souborů včetně obrázků a zvuků. Hra se může na slabších počítačích začít sekát, při zobrazování 50 a více objektů. [13]



Obrázek 5. Hra Stellaria

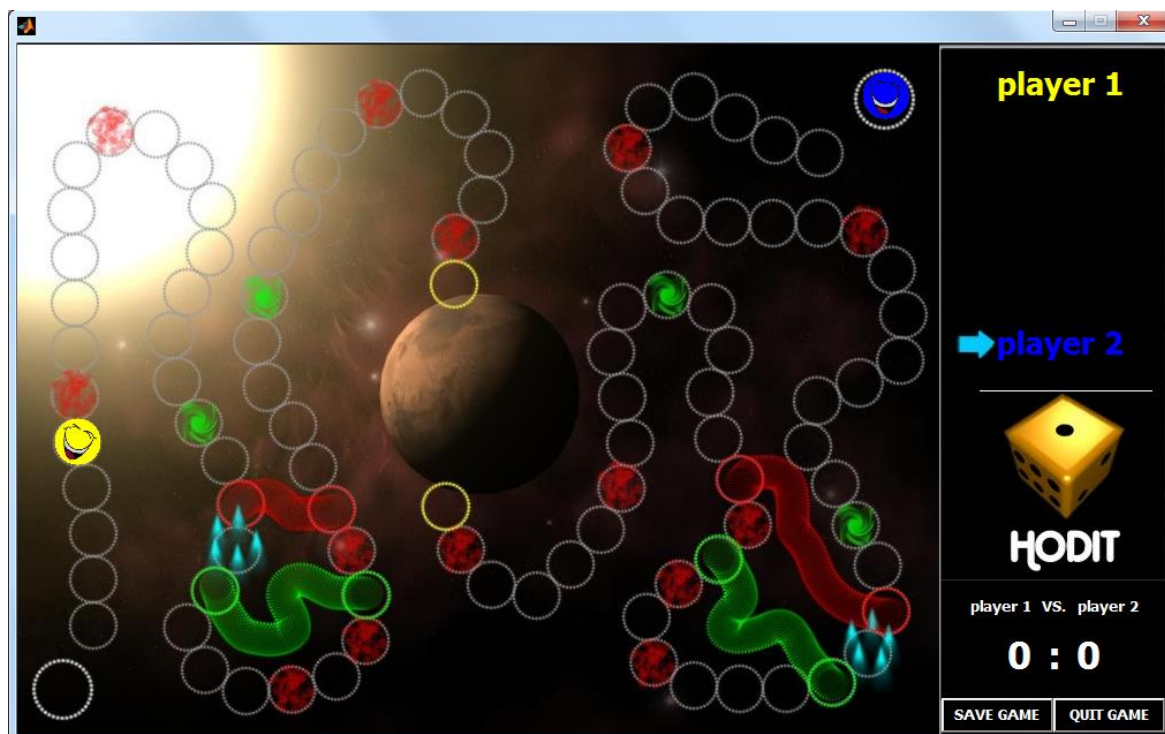
## 2.2 Flappy Bird

Tato hra je v současnosti velmi populární na dotykových přístrojích v operačních systémech iOS a Android. Stiskem klávesy se dávají pulsy letícímu ptákovy, který musí proletět mezi potrubím bez doteku. S jakýmkoliv dotek s okolím hra končí. Mingjing Zhang ji převedl i do prostředí Matlab, a jak sám píše, grafické objekty a jejich fyzika je identická s původní hrou. Například objekt ptáka je reprezentován jako *surface* objekt, což se hlavně používá pro prezentaci 3D grafů, což umožňuje rotaci daného objektu. Celý program je napsán v jedné funkci obsahující 561 řádků. [11]



Obrázek 6. Hra Flappy Bird

## 2.3 Fields Crazy

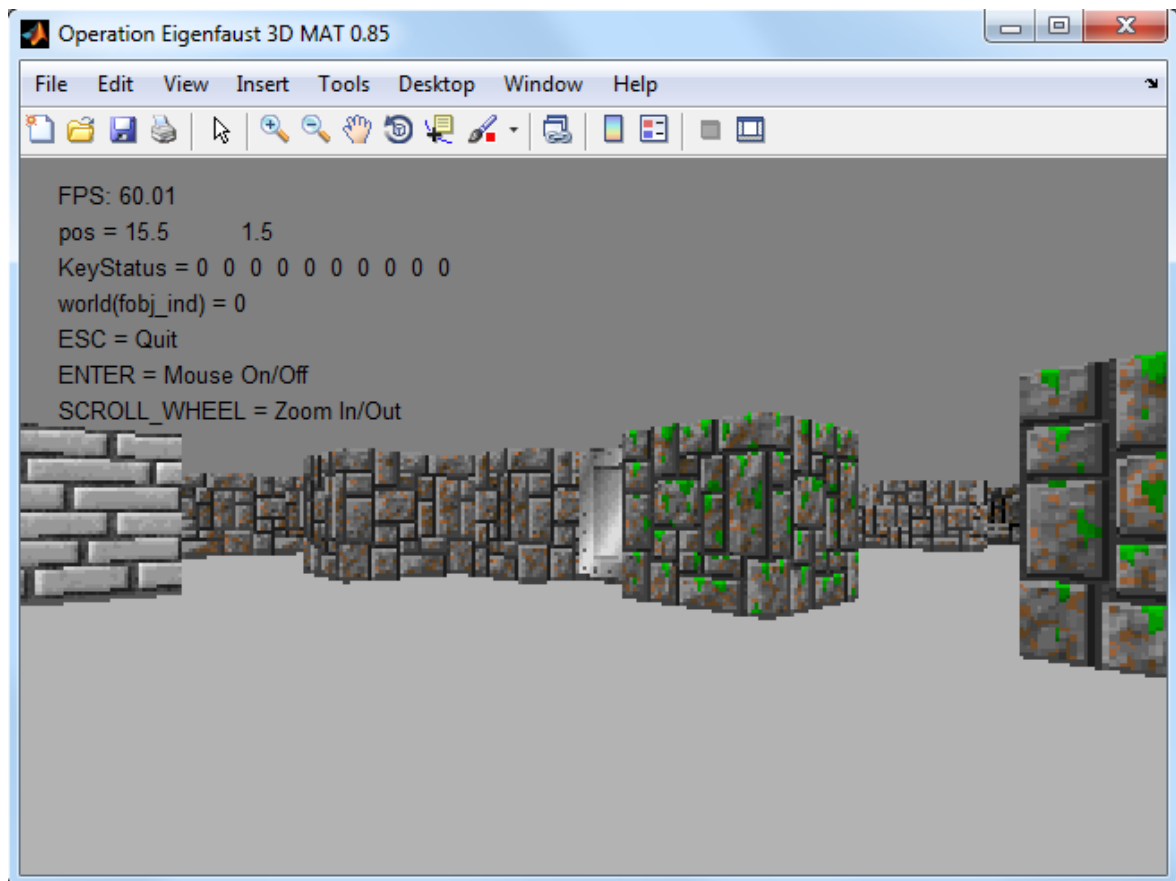


Obrázek 7. Hra Fields Crazy

Fields Crazy je desková hra pro dva hráče, kteří prochází po vyznačené cestě na základě hozené hodnoty kostky. Na cestě jsou teleporty posunující hráče dopředu nebo dozadu. Kdo dojde první do cíle, vyhrává. Hra neobsahuje hrací počítač. Umožňuje ukládat a načítat rozehrané hry. Má pěkné audiovizuální zpracování, zdrojový kód obsahuje 65 M – souborů a další grafické zvukové soubory. Autorem je Petr Hruboš, jehož hra byla součástí bakalářské práce. [3]

## 2.4 Operation Eigenfaust 3D

Tohle demo hry představuje opravdovou 3D hru. Umožňuje volný pohyb v 3D prostředí. Hra využívá algoritmu Ray – casting používaný v 90. letech pro tvorbu her. Autor Mingjing Zhang přepsal tento kód do jazyku Matlab. Celý kód je na 1050 řádcích. Hra umožňuje pouze pohyb a otevírání dveří, vše pouze v jedné úrovni, což slouží také pouze pro prezentaci funkcí Matlabu. [12]



Obrázek 8. Hra Operation Eigenfaust 3D

## **II. PRAKTICKÁ ČÁST**

### 3 POUŽITÉ PROGRAMY

Vytvoření hry je velmi komplexní záležitost, proto je potřeba více programů pro jeho realizaci. Při tvorbě hry „Člověče, nezlob se“ byly využity programy s časově omezenými licencemi nebo programy volně stažitelné.

#### 3.1 Corel Draw

Corel Draw je profesionální grafický editor pracující s vektorovou grafikou. Dostupný je v českém jazyce. Funkčně je velice dostupný začátečníkovi i díky dostupným manuálům, hlavně na webových stránkách.

Při tvorbě hry byl program využit pro tvorbu hrací plochy, pozadí jednotlivých oken, figurek, hrací kostky a tlačítka. Všechny grafické prvky vyjma prvků Matlabu jsou tedy originální.

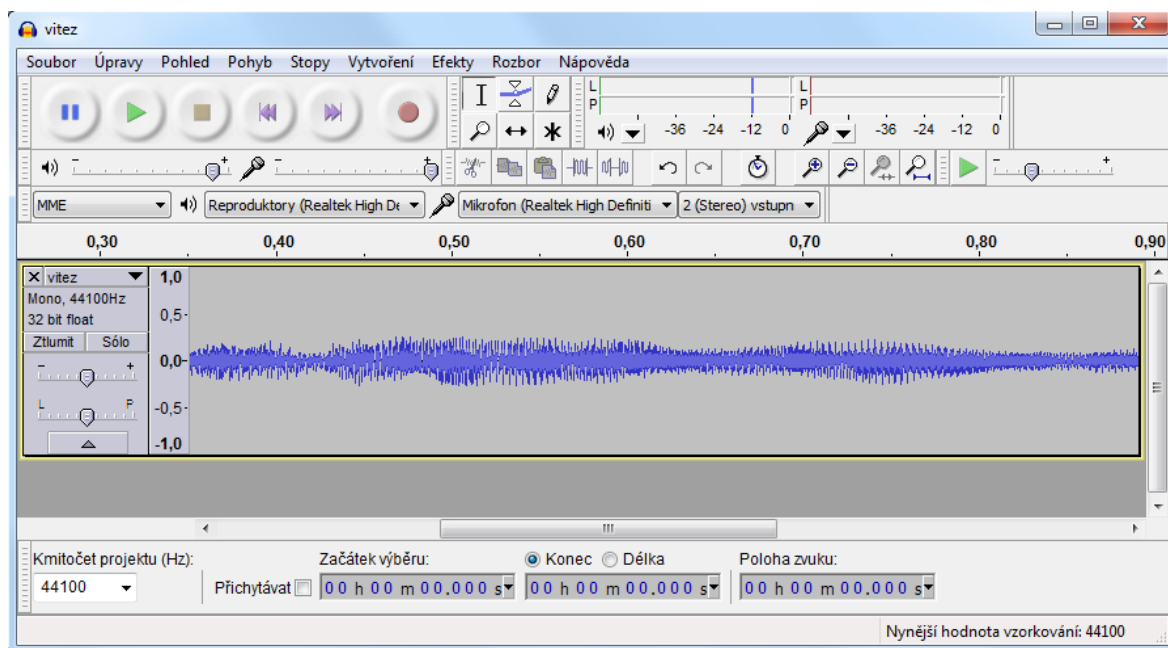


Obrázek 9. Náhled okna programu Corel Draw

## 3.2 Audacity

Audacity je zvukový editor a rekordér. Software je zdarma s otevřeným zdrojovým kódem s českým překladem.

Pro vývoj hry byl použit při zkrácení některých zvuků a při ztišení jejich hlasitosti. Všechny zvuky použité ve hře jsou staženy z uvedených zdrojů a jsou volně šiřitelné. [7, 14, 15, 16]



Obrázek 10. Náhled okna programu Audacity

## 3.3 Matlab

Pro vývoj hry byl použit Matlab ve verzi R2013a pro tvorbu doma. Verze R2012b a R2006b byly použity ve školních prostorách Fakulty aplikované informatiky. Program byl na těchto verzích testován a byla prověřena bezchybná funkčnost.

## 4 POPIS HRY

Hra „Člověče nezlob se“ je určena pro 2 – 4 hráče. Jedná se o deskovou hru, kde každý hráč vlastní čtveřici figurek, každý jiné barvy. Hráč má na začátku seřazeny figurky na pozicích základny. Úkolem je dostat figurky za pomoci házení kostky ze základny na hrací pole a dojít s nimi do cíle. Hráč, který dojde do cíle jako první, vyhrává.

### 4.1 Pravidla hry

1. Na začátku si každý hráč hodí kostkou. Má 3 pokusy k tomu, aby hodil „6“ a mohl si nasadit jednu figurku na nástupní pole se šipkou. Nepodaří-li se mu hodit „6“ pokračuje v házení další hráč.
2. Po hození „6“ a nasazení figurky na nástupní políčko hází hráč ještě jednou a postoupí o tolik polí, kolik bodů mu padlo.
3. Padne-li hráči během hry opět „6“, musí ihned nasadit svou figurku na nástupní políčko a hází ještě jednou. Pak musí posunout o hozený počet touto figurkou i v tom případě, že si vyřadí ze hry svou vlastní figurku.
4. Má-li hráč již všechny figurky ve hře a padne mu „6“, pak hází ještě jednou a o součet bodů posunuje kteroukoli svojí figurkou.
5. Při postupu překračuje hráč v cestě se nacházející figurky protihráčů i své vlastní. Všechna překročená pole se náležitě počítají.
6. Dostane-li se hráč se svou figurkou na obsazené pole, pak dostiženou figurku vyhodí a postaví se na její místo. Vyhozenou figurku si protihráč postaví zpět na zásobní políčko své barvy.
7. Vyhozené figurky se mohou nasadit zpět do hry až po hození „6“.
8. Figurky, které prošly celou dráhu, umisťují hráči na stejnobarevných polích ve středu herního plánu. Cílová barevná pole mohou obsadit teprve tehdy, když hráči padne tolik bodů, kolik k dosažení určitého pole potřebuje. Ten, kterému se nejdříve podaří umístit všechny figurky na cílová pole, vyhrává.

### 4.2 Realizace hry

Pro spuštění hry je potřeba nastavit aktuální adresář Matlabu na složku *clovece*. Startovní soubor je *hra.m*, volá se příkazem *hra*. Minimální rozlišení obrazovky je 1024x768.

#### 4.2.1 Hlavní menu

Po spuštění hry a načtení obrázků a zvuků se zobrazí základní menu hry se čtyřmi tlačítky – *Nová hra*, *Načíst hru*, *Pravidla hry*, *Ukončit hru*.



Obrázek 11. Okno Hlavní menu

#### 4.2.2 Nová hra

Po stisku tlačítka *Nová hra* se objeví nové dialogové okno pro nastavení hráčů. Každý hráč má fixně nastavenou barvu. U hráče lze nastavit jméno, maximální délka je 7 znaků. Poslední nastavení je, zda daný hráč bude člověk, počítač anebo nebude hrát vůbec. Proto, aby byla hra spuštěna, musí být nastaven minimálně jeden člověk a celkově dva hráči. Stiskem tlačítka *OK* se spustí hra, stiskem tlačítka *STORNO* se vrací do základního menu.



Obrázek 12. Okno Nová hra.

#### 4.2.3 Načíst hru

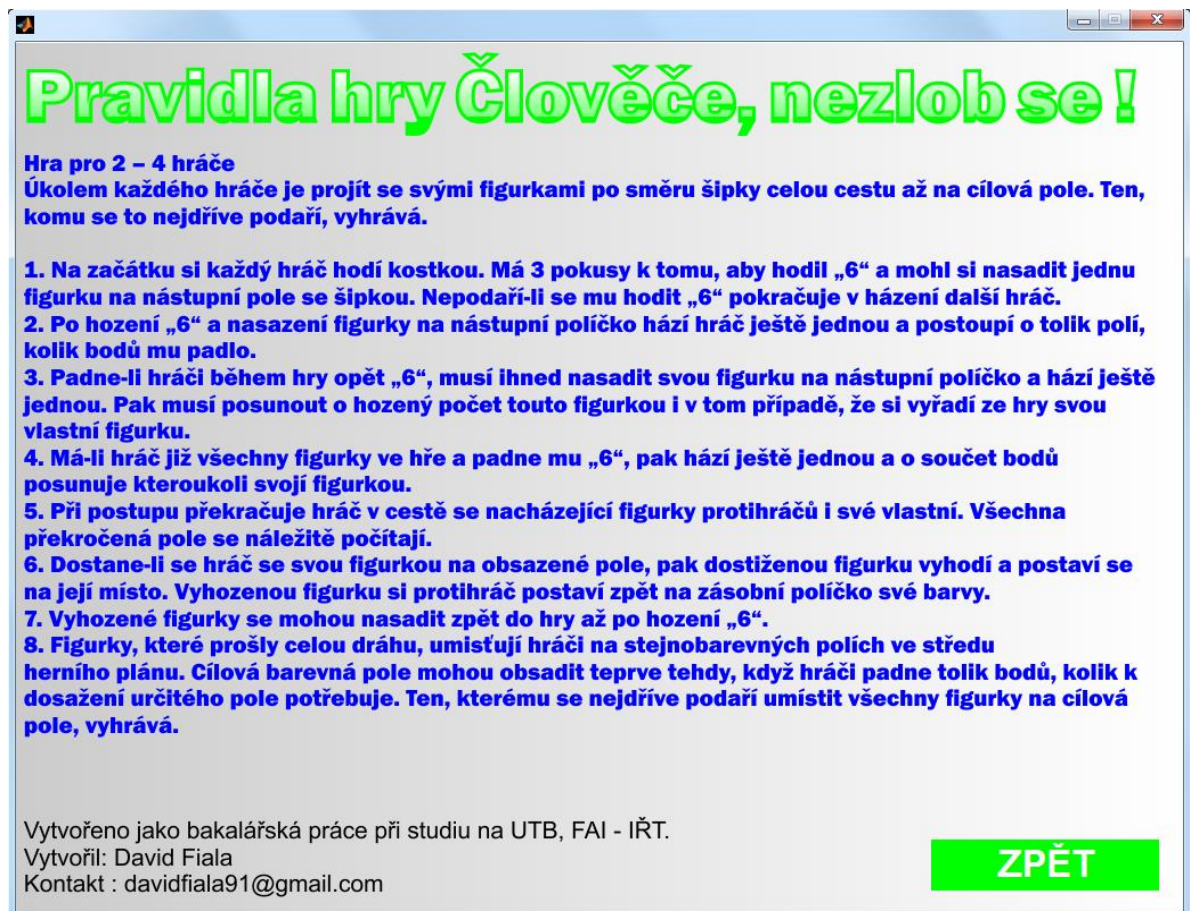
Při stisku tlačítka *Načíst hru* se zobrazí okno pro výběr souborů uložených her, uložených v adresáři *ulozeni*. Po stisku tlačítka *NAČÍST* se provede načtení vybraného souboru a spustí se hra. Stiskem tlačítka *STORNO* se program vrací do základního menu.



Obrázek 13. Okno Načíst hru

#### 4.2.4 Pravidla hry

Výběrem tlačítka *Pravidla hry* program zobrazí okno obsahující popis a pravidla hry, informace o autorovi hry a tlačítko *ZPĚT* pro vrácení se do základního menu.



Obrázek 14. Okno Pravidla hry

#### 4.2.5 Ukončit hru

Tlačítko slouží pro okamžité ukončení celé hry.

#### 4.2.6 Hlavní hra

Okno hlavní hry obsahuje hrací pole, základny a cílové pole hráčů a figurky hráčů.

Hra je ošetřena, aby hráč mohl táhnout pouze svými figurkami. Ať už má hráč v poli jednu nebo více figurek, pokaždé musí kliknout na tu, kterou chce táhnout. Dalším případem je situace, kdy hráč nemůže táhnout ani jednou figurkou, například pokud stojí před cílem. V tomto případě hráč i tak musí kliknout na svou figurku, je to z důvodu zachování autenticity, aby vše ovládal uživatel a ne počítač.

Pokud za některého hráče je nastaven počítač, všechny úkony se samozřejmě vykonávají automaticky a uživatel do nich nemůže zasahovat.

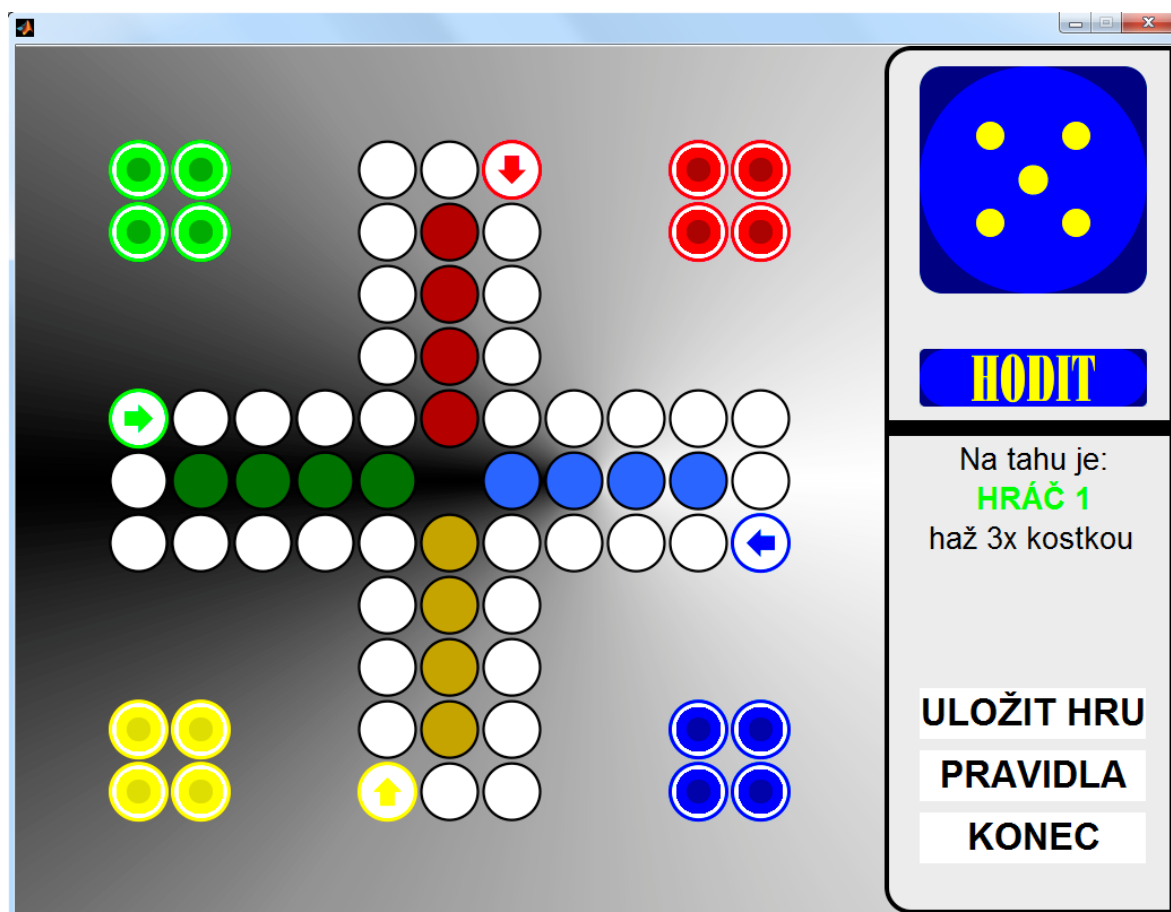
Na pravé straně okna se nachází ovládací panel s kostkou a tlačítkem *HODIT*. Tohle tlačítko má žlutou barvu, pokud je možné házet kostkou pro daného hráče. Jestliže je barva šedá, kostkou házet nelze.

Pod ovládním kostky je informační blok. Tento blok indikuje, který hráč je na řadě a zobrazuje pokyny, co má hráč dělat.

Poslední blok v ovládacím panelu je blok tří tlačítek – *ULOŽIT HRU*, *PRAVIDLA*, *KONEC*.

Všechny tlačítka v ovládacím panelu lze ovládat pouze, je-li na řadě člověk. Pokud je na tahu počítač, tak nejsou dostupná z důvodu případného přerušení běhu programu, kde není možné identifikovat místo zastavení programu pro pozdější návrat.

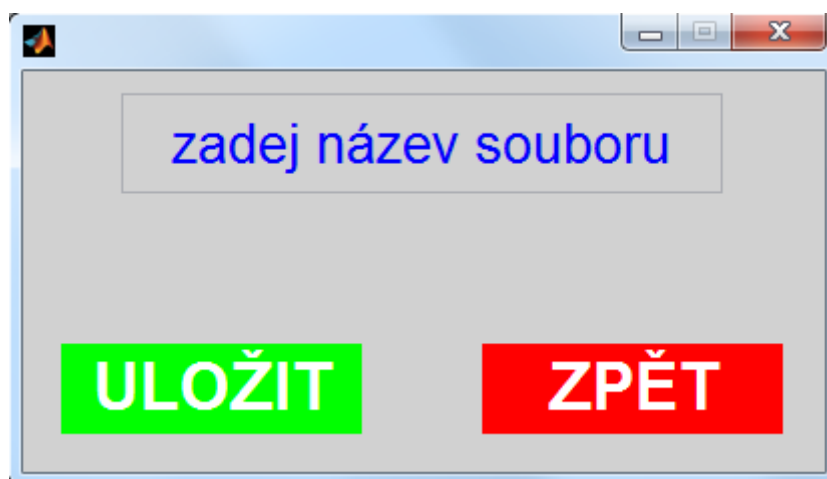
Pokud se podaří hráči dostat všechny figurky do cíle, objeví se okno s označením vítěze a pořadí hry.



Obrázek 15. Okno Hlavní hra

#### 4.2.7 Uložit hru

Výběrem možnosti *ULOŽIT HRU* se aktivuje dialogové okno pro uložení aktuální hry. Okno obsahuje pole pro zadání jména souboru pro uložení a dvě tlačítka. Jméno souboru smí obsahovat pouze alfanumerické znaky a jeho délka musí být 1-20 znaků. Název souboru také nesmí být stejný jako už existující soubor. Tlačítkem *ULOŽIT* se vyhodnocuje, zda je možné hru uložit. Vypíše se tedy případná chyba nebo potvrzení o úspěšném uložení. Tlačítkem *ZPĚT* se vrací zpět do hlavní hry.



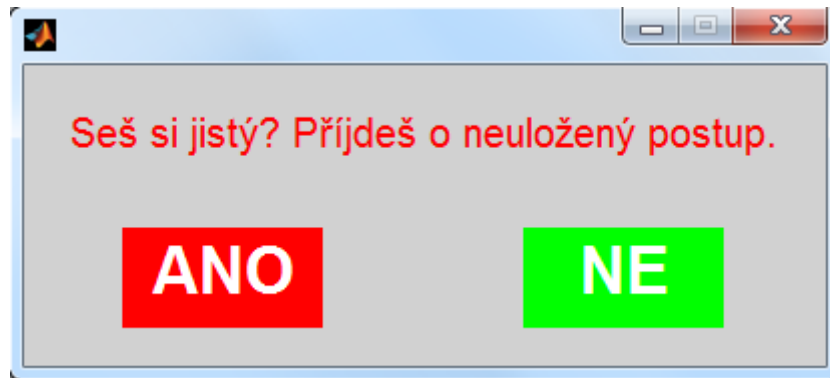
Obrázek 16. Okno Uložit hru

#### 4.2.8 Pravidla

Tlačítko *PRAVIDLA* aktivuje identické okno s pravidly hry, jako je v hlavním menu. Tlačítkem *ZPĚT* se hra vrací zpět do hlavní hry.

#### 4.2.9 Konec

Poslední tlačítkem v ovládacím panelu je *KONEC*, jeho aktivováním se vytvoří okno s ujištěním, zda si je uživatel jistý ukončit hru. Tlačítkem *ANO* se hra vrací do hlavního menu, tlačítkem *NE* hra pokračuje.



Obrázek 17. Okno Konec.

#### 4.2.10 Okno Vítěz

Pokud se hráči podaří dostat všechny své figurky do cíle, zobrazí se okno s vítězem, případně s pořadím hráčů. Okno obsahuje také dvě tlačítka. Tlačítkem *POKRAČOVAT* hra pokračuje dál ve hraní, kde se hraje pouze o lepší umístění. Tlačítkem *UKONČIT* se hlavní hra ukončuje a je proveden návrat do hlavního menu.

Tlačítko *POKRAČOVAT* se zobrazuje pouze, když ve hře zůstává více než jeden hráč, protože pokud zbyde jen jeden hráč, je automaticky poslední a je tedy zařazen do pořadí. Další podmínka je, aby mezi zbývajícím hráčem byl člověk. Jednak je to z důvodu, nemožnosti poté zastavit hru a jednak zbývajícím hráčem jsou počítače, tedy už nezáleží na jejich pořadí.

Při ukončení okna pomocí křížku se ukončí okno a hra se vrací do hlavní hry nebo do základního menu za stejných podmínek, jako při tlačítku *POKRAČOVAT*.



Obrázek 18. Okno Vítěz

### 4.3 Adresářová struktura

#### 4.3.1 Adresář clovece

Jedná se o výchozí složku, která musí být označena jako aktuální složka. Ve složce se vyskytují podadresáře a soubory. Obsah složky je tedy následovný:

Podadresáře: *hra, menu, multimedia, ulozeni*

M - soubory: *globalni.m, hra.m, inicializace.m*

##### **globalni.m**

Obsahuje několik proměnných, které musí být nastaveny jako globální, aby k nim měli přístup funkce. Volá se hned po spuštění hry a ve funkcích.

##### **inicializace.m**

Provádí se načítání obrázků a zvuků a inicializace proměnných potřebných pro spuštění hry. Skript se volá při vytvoření nové hra a při načítání uložené hry.

## **hra.m**

Jedná se o spouštěcí skript celé hry. Jako první se vytváří okno *zakladniMenuFig* s nápisem načítání, poté se volají skripty pro nastavení globálních proměnných a inicializaci (*globalni.m*, *inicializace.m*). Vytváří se okno hlavního menu se čtyřmi ovládacími prvky *novaHra*, *nacistHru*, *pravidlaHry*, *ukonciHru*, tyto prvky obsahující *callback* funkce při stisku tlačítka myši. Tyto prvky tedy slouží jako tlačítka hlavního menu. Poslední prvek je objekt *axes* pro zobrazení obrázku na pozadí.

### **4.3.2 Adresář multimédia**

Tento adresář obsahuje 7 zvukových souborů a 17 grafických souborů. Tyto soubory jsou načítány ve skriptu *inicializace.m*.

Zvukové soubory jsou použity pro zvuk pohybu figurky, hod kostkou, vyhození figurky, změna hráče, vítězství hráče, kliknutí tlačítka, chyby.

Grafické soubory jsou použity pro pozadí hlavního menu, hlavní hry, okna s pravidly, okna pro oznámení vítěze, 6 čísel kostky, 2 stavy tlačítka na hod kostkou, 4 barevné figurky a šablona figurky.

### **4.3.3 Adresář uložení**

V tomto adresáři se ukládají *.mat* soubory pro ukládání rozehrané hry.

### **4.3.4 Adresář menu**

Nachází se zde soubory pro obsluhu v hlavním menu hry.

## **menuNovaHra.m**

Jedná se o *callback* funkci tlačítka *novaHra* v hlavním menu. Vytváří se okno *novaHraFig* se čtyřmi ovládacími editačními prvky pro určení jména hráče. Další čtyři prvky jsou *popupmenu* pro nastavení hráče (počítač, člověk, nikdo).

Informační textový prvek *novaInfo* slouží po vypsání případných chyb. Poslední dva textové prvky reprezentují tlačítka, mají tedy své *callback* funkce. Prvek *tlacitkoOk* volá při stisknutí skript *novaTlacitkoOk*, prvek *tlacitkoStorno* volá *novaTlacitkoStorno*.

## **novaTlacitkoStorno.m**

Zavírá okno *novaHraFig*.

**novaTlacitkoOk.m**

Inicializuje proměnné pro správný běh skriptu. Načítá hodnoty ovládacích prvků pro nastavení hráče. Pokud je proměnná *I* - daný hráč je počítač, *2* - daný hráč je člověk, *3* - hráč nehraje. Do proměnných se načítají i jména hráčů.

Následuje algoritmus pro zjištění počtu hráčů a počtu lidí. Podle toho, jaký hráč hraje první, se nastavují proměnné *infoString* a *infoBarva* pro informační prvky v hlavní hře, určující jméno hráče a jeho barvu.

Následuje ošetření, aby délka jména hráče nebyla delší než 7 znaků, aby musel hrát alespoň jeden člověk a aby hráli alespoň dva hráči. Pokud není splněna některá z podmínek, je vypsána chyba. Pokud je vše v pořádku, vymaže se okno *novaHraFig*, oknu *zakladniMenuFig* se nastaví neviditelnost, volá se skript pro inicializaci *inicializace.m* a skáče se do adresáře *hra*, kde se spouští skript hlavní hry *hlavniHra.m*.

**menuNacistHru.m**

Jedná se o *callback* funkci tlačítka *nacistHru* v hlavním menu. Vytváří nové okno *nacistFig* s možností výběru souboru, tedy uložené hry přes prvek *popupmenu*. Okno obsahuje dva textové ovládací prvky, sloužící jako tlačítka s *callback* funkcemi a jeden textový ovládací prvek pro vypisování chyb.

Tlačítka a jejich *callback* funkce:

*tlacitkoNacist* – *callback* funkce *nacistTlacitkoNacist*

*tlacitkoZpet* – *callback* funkce *nacistTlacitkoStorno*

**nacistTlacitkoNacist.m**

Na základě vybraného souboru se volá skript *nacistKontrola.m*, který kontroluje, zda soubor obsahuje všechny potřebné proměnné. Jestli neobsahuje, vypíše se chyba a čeká se na další akci. Pokud je soubor správný, načítají se proměnné ze souboru, volá se skript *hlavniHra.m* a *pokynyNastaveni.m*, nastavuje se obrázek kostky podle hozené hodnoty, nastavuje se aktivita tlačítka kostky, nastavuje se informační panel podle hráče, jenž je na tahu.

**nacistTlacitkoStorno.m**

Zavírá okno *nacistFig*.

**nacistKontrola.m**

Obsahuje vektor s názvy proměnných, které musí být uloženy v souboru pro načtení hry. Ověřuje se, jestli daný soubor obsahuje tyto proměnné.

#### **menuPravidlaHry.m**

Jedná se o *callback* funkci tlačítka *pravidlaHry* v hlavním menu. Vytváří nové okno *pravidlaFig* s vlastností *DeleteFcn - pravidlaUkonceni*. Okno obsahuje obrázek na pozadí, kde je napsaný celý text. Dále obsahuje ovládací prvek *tlacitkoZpet* s *callback* funkcí *pravidlaZpet*.

#### **pravidlaZpet.m**

Zavírá okno *pravidlaFig*.

#### **pravidlaUkonceni.m**

Rozlišuje, v kterém okně bylo okno *pravidlaFig* zavřeno a podle toho se vrací do složky *hra* nebo *menu*.

#### **menuUkoncitHru.m**

Jedná se o *callback* funkci tlačítka *ukoncitHru* v hlavním menu. Zavírá okno *zakladniMenuFig*.

#### **menuUkonceni.m**

Volá se při ukončení okna *zakladniMenuFig*. Posunuje souborový adresář do úroveň výš, tedy do složky *lovece*.

### **4.3.5 Adresář hra**

V této složce se nachází soubory spojené s hlavní hrou.

Důležité proměnné:

- ❖ *aktivniPocitac* – Jestliže je nastavena na *1*, udává, že aktuální hráč je počítač.
- ❖ *poleCall* – Jedná se o pole reprezentující odezvu kliknutí na políčko hracího pole (bez cíle a startu). Ukládá se hodnota *1*.
- ❖ *hraciPole* – Proměnná typu pole. Reprezentuje hrací pole, do kterého se ukládají unikátní hodnoty figurek na index, dle jejich polohy (*1 - 40*). Hodnoty zelených figurek – *11,12,13,14*. Hodnoty červených figurek – *21,22,23,24*. Hodnoty modrých figurek – *31,32,33,34*. Hodnoty žlutých figurek – *41,42,43,44*.
- ❖ *odezva* – Hodnota nabývajícího stavu *0* a *1*. Jestliže je *0*, hra nereaguje na stisk tlačítka, je-li *1*, hra reaguje na kliknutí.

- ❖ *kostkaHod* – Určuje, zda bylo hozeno kostkou, jestli ano, nabývá hodnoty 1.
- ❖ *stav* – Každý hráč má vlastní proměnnou, určující jeho stav (*stav1*, *stav2*, *stav3*, *stav4*), tyto proměnné se zapisují do pomocné proměnné *stav*. Stav 0 – hráč má figurky na základně a musí hodit 6, aby dostal figurku do pole. Stav 1 – Hráč nasazuje figurku ze základny. Stav 2 – Hráč hraje s figurkami v poli.
- ❖ *kontrola* – Brání přístupu do důležité části kódu ve skriptu *logikaHry.m*, je-li rovna 0. Nastavuje se na hodnotu 1, jestliže byla vybrána platná figurka.
- ❖ *kostkaCislo* – Udává hodnotu hozeného čísla na kostce.
- ❖ *aktivniNasazeni* – Pokud je hodnota 1, určuje, má-li dojít k nasazení nové figurky.
- ❖ *hodDvakrat* – Rovná-li se tato proměnná 1, daný hráč má házet ještě jednou.
- ❖ *poloha* – Do proměnné je uložena poloha v poli aktuálně vybrané figurky.
- ❖ *vzdalenost* – Do proměnné je uložena vzdálenost, kterou aktuální figurka ušla od startu.
- ❖ *aktivniZeleny1* – Každá figurka má svou proměnnou, název se liší dle barvy a čísla. Hodnota se nastavuje na 1, jestliže na ni její vlastník klikne na poli. Jedná se o indikátor aktivity pro práci pouze s touto figurkou.
- ❖ *poleModryCil* – Každý hráč má svoji proměnnou, lišící v názvu barvou svých figurek (uvedená proměnná patří modrému hráči). Proměnná je typu pole, reprezentující obsazenost cílových políček. Pokud se rovná 0, je políčko prázdné, pokud se rovná 1, políčko je obsazené.
- ❖ *kontrolaCil* – Indikuje, zda je možné táhnout některou z figurek aktuálního hráče, nabývá-li hodnoty 1. Hodnota 0 značí nemožnost táhnutí.

### hlavniHra.m

Než se provede načtení všech prvků v okně samotné hry, je zobrazeno okno *nahravaniFig*, signalizující nahrávání. Na konci skriptu je smazáno a zviditelněno je okno *hlavniHraFig*. Okno má nastavené vlastnosti *WindowButtonUpFcn - hlavniHraCall a DeleteFcn - hlavniHraUkonceni*. Okno obsahuje *axes* prvky pro zobrazení pozadí okna (hracího pole), obrázek kostky, obrázek tlačítka kostky a samotných figurek hráčů. Informační panel je složen z textových ovládacích prvků. Menu ve hře jsou také textové prvky, ale *callback* funkcí jsou volány ve skriptu *hlavniHraCall.m*.

Na konci skriptu je smyčka *for* pro zjištění, zda začíná první počítač, v tom případě se nastavuje *aktivniPocitac* na *1* a spouští skript *logikaHry.m*.

### **hlavniHraCall.m**

Skript je volán pokaždé, je-li kliknuto tlačítkem myši v okně *hlavniHraFig*. Skript obsahuje řadu podmínek, které určují místo kliknutí. Je-li kliknuto na políčko hracího pole, zapíše se do daného prvku proměnné *hraciPole* hodnota *1*. Podobné je to u základen, kde každé políčko základny má svou proměnnou (Př.: *zelenyZakladnaCall4*). Je-li kliknuto na tlačítko kostky, je spuštěn skript *kostka.m*. Při kliknutí na ovládací prvky hry, tedy menu, jsou volány skripty pro daný prvek – *hraPravidla.m*, *hraKonec.m*, *hraUlozitHru.m*.

Na konci skriptu se volá skript *logikaHry.m*, pokud tedy není provedena obsluha tlačítek, v tom případě se nevolá, jelikož se ve skriptech mění adresář, došlo by k chybě. Celý algoritmus ve skriptu je dostupný, pokud *odezva* je rovna *1*. Zamezuje to vícenásobnému hodování kostky a volání obsluhy tlačítek nebo zasahování do hry počítače.

### **hraPravidla.m**

Mění se adresář na *menu*, poznamená se, že se funkce volala z okna hlavní hry a volá se skript *menuPravidlaHry.m*.

### **hraUlozitHru.m**

Je vytvořeno nové okno *ulozFig* s editačním ovládacím prvkem pro zadání názvu souboru. Další kontrolní prvek je textový pro vypsání chyby. Poslední jsou dva textové prvky s *callback* funkcemi: *tlacitkoUloz2* – *callback* funkce *ulozitTlacitkoUloz*, *tlacitkoZpet* – *callback* funkce *ulozitTlacitkoZpet*.

### **ulozitTlacitkoUloz.m**

Kontroluje se, zda zadaný název obsahuje jen alfanumerické znaky a zda délka názvu je v rozmezí *1-20* znaků. Poslední kontrola, zda už soubor se stejným názvem neexistuje. Pokud není některá z podmínek splněna, vypíše se chyba a čeká se na další akci. Pokud je vše v pořádku, zapisuje se aktuální pozice *axes* prvků figurek a ukládají se všechny proměnné potřebné pro načtení hry. Vypíše se hláška o potvrzení uložení.

### **ulozitTlacitkoZpet.m**

Zavírá okno *ulozFig*.

### **hraKonec.m**

Je vytvořeno okno *konecFig* s textovým kontrolním prvkem s nastaveným řetězcem, zda si uživatel opravdu přeje ukončit hru.

Okno obsahuje následující ovládací prvky:

*tlacitkoAno* – obsahuje *callback* funkci *konecTlacitkoAno*

*tlacitkoNe* – obsahuje *callback* funkci *konecTlacitkoNe*

### **konecTlacitkoAno.m**

Zavírá okna *hlavniHraFig* a *konecFig*.

### **konecTlacitkoNe.m**

Zavírá okno *konecFig*.

### **kostka.m**

Celý skript je dostupný pouze, pokud proměnná *cekej* se rovná 0. Ihned se volá skript *tlacitkoOff.m*, Proměnná *kostkaHod* je nastavena na 1.

Je vygenerováno číslo kostky, podle hozené hodnoty, se zobrazuje obrázek kostky. Tento blok je v cyklu *While*, což je z důvodu simulace hodu kostky.

Důležité je, že se provádí nulování proměnné *poleCall*, mažou se tím předešlá označená pole. Volá se skript *tlacitkoOn.m*.

### **tlacitkoOn.m**

Zobrazuje obrázek aktivního tlačítka kostky, nastavuje proměnnou *cekej* na 0.

### **tlacitkoOff.m**

Zobrazuje obrázek vypnutého tlačítka kostky, nastavuje proměnnou *cekej* na 1.

### **nacitaniHrace.m**

Do pomocné proměnné *stav* se ukládá aktuální stav aktuálního hráče. Je-li aktuální hráč počítač, spouští se skript *pocitac.m*.

### **ukladaniHrace.m**

Pomocná proměnná *stav* se ukládá do stavu aktuálního hráče.

### **zmenaHrace.m**

Inkrementuje se proměnná *hrac*. Jelikož může dojít k několika variacím hrajících hráčů, testuje, zda hráč o aktuální hodnotě proměnné hraje (neplést s právě aktivním hráčem), když ano, algoritmus končí, jestliže nehraje, proměnná se inkrementuje a znovu testuje. Test se provádí pro různě možné posloupnosti. Proměnná *konec* slouží, aby při nálezu hrajícího hráče algoritmus nepokračoval. Na základě aktuálního hráče se vypisuje jeho jméno a pokyny na informační panel hlavní hry a zjišťuje se, zda je hráč počítač.

#### **aktivniNastaveni.m**

Skript je volán po kliknutí na hrací pole ve skriptu *hlavniHraCall.m*. Na začátku jsou nulovány indikátory aktivity figurek. V *poleCall* se hledá, na které políčko se kliklo, to samé políčko je kontrolováno v proměnné *hraciPole*. Jestliže se na daném políčku objevuje figurka aktuálního hráče, daná figurka má nastaven indikátor aktivity.

#### **aktivniNacitani.m**

Aby hlavní logika ve skriptu *logikaHry.m* probíhala co nejprehledněji, zjišťuje se, která figurka je aktivní a její vlastnosti (poloha, *axes* prvek, vzdálenost) se přiřadí do pomocných proměnných (*poloha*, *figurka*, *vzdalenost*). Dochází-li k nutnosti hraní právě nasazené figurky, je pomocí hodnoty proměnné *aktivniNasazeni* nastaven indikátor aktivity figurky.

#### **aktivniUkladani.m**

Skript je obdobou skriptu *aktivniNacitani.m*, dochází zde ale k ukládání pomocných proměnných do původních proměnných figurky. Skript je volán až po ukončení práce s pomocnými proměnnými.

#### **resetZakladen.m**

Aby se nevyhodnocovali dřívější kliknutí na základny, provádí se nulování jejich „*callbacků*“.

#### **prazdneZakladny.m**

Ve skriptu se kontroluje poloha figurek, jestliže nejsou v bázi, hráč hází dvakrát.

#### **nasazeni.m**

Skript se volá, když má dojít k výběru figurky z báze pro nasazení na startovní políčko. Kontroluje se, na jaké políčko báze aktuálního hráče bylo kliknuto a zda se tam figurka nachází. Po výběru figurky se nastavuje vzdálenost figurky a poloha. Unikátní hodnota figurky

je zapsána do proměnných *aktivniNasazeni* a *hraciPole*. Pokud se na startovním políčku objevuje figurka, je vyhozena. Mění se *stav* na *1* a nastavují se pokyny.

### **vyhozeni.m**

Podle unikátní hodnoty uložené v proměnné *hraciPole* na indexu políčka, kam se aktuální figurka posouvá, se zjistí, která figura se zde nachází. Grafická podoba figurky je přesunuta do báze, její vzdálenost a poloha je vynulována. Kontroluje se, zda danému hráči zůstali ještě nějaké figurky v poli, v opačném případě se mění stav hráče na *0*.

### **pokynyNastaveni.m**

Podle proměnné *pokyn* se nastavuje textový ovládací prvek v informační části panelu.

### **figurkaCisloCall.m**

Na základě aktivní figurky vrací skrze proměnnou *figurkaCislo* její unikátní hodnotu.

### **cil.m**

Jestliže nová vzdálenost figurky na základě hozené hodnotě kostky je větší než 40 (délka hracího pole), figurka má možnost dojít do cíle a volá se tento skript. Proměnná *zbytek* udává, o kolik přesahuje nová vzdálenost jedno kolo hracího pole. Pokud je hodnota *1* až *4*, kontroluje se, zda se na dané pozici políčka v cíli nenachází jiná figurka. Když je políčko prázdné, změní se status na obsazené, mění se poloha figurky na hodnotu *41*, předešlá pozice figurky se nuluje, mění se i pozice grafického zobrazení figurky. Na hodnotu *1* je přepsána proměnná *cilProveden*.

### **cilKontrolaPole.m**

Provádí se kontrola, zda hráč má nějakou figurku v poli, jestli ne, *stav* se mění na *0*.

### **cilKontrolaOstatnich.m**

Pro daného hráče se přepočítává každá figurku, jestli je možné s ní táhnout. Proměnná *kontrolaCil* je nastavena na *1*, pokud vzdálenost po hodu je menší, než délka hracího pole nebo pokud není obsazené políčko v cíli, kam figurka dojde.

### **logikaHry.m**

Jedná se o hlavní skript, obsahující hlavní logiku hry. Uvnitř algoritmu se volá mnoho pomocných skriptů. Pro přehlednost je rozdělen do několika bloků. Celý kód je ve smyčce

*While* pro opětovné spuštění, je-li na řadě počítač. Pokud je na řadě člověk, smyčka je ukončena, čeká se na akci člověka.

1. Blok 0 – Před klíčovým větvením programu pomocí funkce *Switch*, testovanou na proměnnou *stav*, se volá skript *nacitaniHrace.m*. Tento blok kódu realizuje situace, je-li *stav* rovno 0. Pokud je hozeno kostkou, kontroluje se, zda je dané číslo 6. Volají se pomocné skripty a mění se *stav* na 1.
2. Blok 1 – Část kódu, která realizuje nasazení figurky ze základny. Volá se skript *nasazeni.m* a nuluje se proměnná *kontrola*.
3. Blok 2.1 – Poslední *case*, zajišťuje logiku pohybu kostky po poli. Podmínka *If* kontroluje, zda je hozena šestka, není-li již nasazena figurka a jestli se nemá házet dvakrát. Pokud je podmínka pravdivá, kontroluje se, zda je nějaká figurka ještě na základně, vypíná se tlačítko kostky, nastavujíc se pokyny, mění se *stav* na 1, ukládá se stav hráče a čeká se na vybrání figurky ze základny. Jestliže má hráč házet ještě jednou, aktivuje se tlačítko kostky a hráč může házet.
4. Blok 2.2 – Tento blok je přístupný za předpokladu, že je hozena jiná hodnota, než 6 nebo má hráč házet dvakrát. Načítá se aktivní figurka (*aktivniNacitani.m*). Kontroluje se, zda bylo kliknuto na políčko o poloze, jako načtená figurka a *kontrola* je 1. V případě házení dvakrát se přičítá ko hodnotě druhého hodu kostky 6. Vypočítá se index na poli, kam se má figurka posunout a vzdálenost, kterou figurka ušla.
5. Blok 2.3 – Pokud je vzdálenost menší nebo rovna 40, vykoná se kód pro posunutí figurky na dané políčko. Pokud je na políčku jiná figurka, je vyhozena zpět na základnu (*vyhozeni.m*). Zjistí se číslo aktivní figurky, to se zapíše do proměnné *hraciPole* na nové políčko, staré políčko se nuluje. Grafický prvek figurky je nastaven na novou pozici. Ukládá se aktivní figurka, mění se hráč.
6. Blok 2.4 – Blok řeší situaci, kdy figurka při hozené hodnotě může dojít do cíle, vzdálenost figurky je tedy větší než 40. Vypočítá se o kolik je přehozena délka pole, tzv. *zbytek*. Volá se skript *cil.m*, ten nastavuje proměnnou *cilProveden*. Pokud figurka došla do cíle, je znovu aktivní tlačítko, ukládá se aktivní figurka, kontroluje se, zda má hráč ještě figurku v poli, mění se hráč a provádí se kontrola, zda daný hráč nemá všechny figurky v cíli. Pokud figurka nemůže dojít do cíle, kontroluje se, jestli může hráč táhnout jinou figurkou, když ano, čeká se, až hráč bude táhnout správnou figurkou. Pokud hráč nemůže táhnout žádnou figurkou, provádí se změna hráče.

**kontrolaVitez.m**

Skript se volá, pokud dojde nějaká figurka do cíle. Kontroluje se každý hráč, zda má všechny figurky ve hře. Pokud ano, podle toho, o kterého se jedná hráče, je nastavena proměnná (*zelenyKonec*, *cervenyKonec*, *modryKonec*, *zlutyKonec*) na hodnotu 1. Hráči, který má takovou hodnotu je přiřazeno místo v pořadí a proměnná je změněna na hodnotu 2, tedy už se s ní nebude pracovat a nebude vyvolávat události. Jméno hráče je dopsáno do tabulky podle toho, jaké místo obsadil. Pokud zbývá ve hře už jen jeden hráč, opakuje se algoritmus ve smyčce *While*, aby bylo hráči přiřazeno poslední místo. Proměnná *aktivniPocitac* je nastavena na 0, aby nedocházelo k pokračování programu.

Je vytvořeno nové okno *vitezFig* s *DeleteFcn* (*vitezUkonceni*). Okno obsahuje *axes* pro vykreslení obrázku na pozadí. Dalšími prvky jsou ovládací textové prvky pro výpis tabulky pořadí. Poslední dva prvky prezentující tlačítka jsou *tlacitkoPokracovat* s *callback* funkcí - *vitezPokracovat* a *tlacitkoKonec* s *callback* funkcí - *vitezKonec*. Ovládací prvek *tlacitkoPokracovat* je dostupné, pokud zbývá ve hře více, než jeden hráč a aspoň jeden je člověk.

#### **vitezPokracovat.m**

Zavírá okno *vitezFig*, pokud je aktuální hráč počítač, *aktivniPocitac* je nastaveno na 1 a je volán skript *logikaHry.m*.

#### **vitezKonec.m**

Okno *vitezFig* je zavřeno, proměnná *pokracovat* je nastavena na 0, kvůli orientaci ve skriptu *vitezUkonceni.m*.

#### **vitezUkonceni.m**

Pokud je okno *vitezFig* ukončeno, křížkem okna, tak jestliže je možné pokračovat ve hře, pokračuje se ve hře. Naopak není-li možné pokračovat, zavírá se i okno *hlavniHraFig*.

#### **pcVyhodit.m**

Funkce zjišťuje, zda při pohybu kostky dojde k vyhození figurky. Funkce vrací jednu ze tří hodnot: 0 – figurka nikoho nevyhodí, 1 – figurka vyhodí cizí figurku, 2 – figurka vyhodí svoji figurku.

#### **pcStart.m**

Funkce zjišťuje, zda se figurka nachází na cizím startovním políčku. Návrátové hodnoty: 0 – figurka nestojí na cizím startovním políčku, 1 – figurka stojí na cizím startovním políčku.

#### **pcOkoli.m**

Funkce zkoumá okolí za figurkou před hodem a po hodů ve vzdálenosti 5 políček. Návrátové hodnoty funkce:

$p = 1$  – za figurkou nikdo nestojí

$p = 2$  – za figurkou stojí cizí figurka

$z = 1$  – za figurkou po hodů nikdo nestojí

$z = 2$  – za figurkou po hodů stojí cizí figurka

### **pcCil.m**

Funkce zkoumá, je-li možné dojít s figurkou di cíle. Funkce vrací hodnoty: 0 – figurka nedojde do cíle, 1 – figurka se vleze do cíle, 2 – políčko v cíli je obsazeno nebo figurka překračuje rozsah cíle

### **pocitac.m**

Tento skript představuje umělou inteligenci hráče. Je volán ve skriptu *logikaHry.m*, když je na řadě počítač. Úkolem počítače je hlavně vybrat nejlepší možnost tahu, tedy vhodně aktivovat jednu ze svých figurek.

Na začátku skriptu se načítají vlastnosti všech figurek daného hráče, pole indikující obsazenost cíle, hodnotu, od které jsou odvozeny unikátní hodnoty figurek a pozice startovního políčka.

Skript se také orientuje podle proměnné *stav*. Pokud je hodnota 0, hází se kostkou. Je-li hodnota 1, aktivuje se *callback* základěn, kde se nachází figurka. Poslední je možnost, rovná-li se hodnota 2. Zjišťuje se počet figurek ve hře. Má-li hráč v poli jen jednu figurku, je aktivována a v proměnné *poleCall* je prvek o indexu polohy figurky nastaven na 1 (simuluje to označení políčka člověkem).

Pokud se nachází v poli více figurek, je použit algoritmus, kde se zjistí situace všech figurek v poli. To se provede pomocí funkcí zjišťující okolí figurky, dojde-li figurka co cíle, jestli někoho může vyhodit, nachází-li se na cizím startu. Posloupnost podmínek *If* seřazených podle priority určuje, která figurka se aktivuje.

Podmínky, seřazené podle nejvyšší priority:

- 1) Je-li figurka schopna dojít do cíle.
- 2) Může-li figurka vyhodit cizí figurku.
- 3) Je-li figurka od cíle na vzdálenost deset polí a nevyhodí přitom svou figurku.

- 4) Pokud figurka stojí na cizím startovním políčku a nevyhodí svou figurku.
- 5) Má-li za sebou cizí figurku, při postupu za ní cizí figurka nebude a zároveň nevyhodí svou.
- 6) Pokud je za figurkou cizí figurka a při postupu nevyhodí svou.
- 7) Za figurkou se nevyskytuje cizí figurka a to ani při postupu, zároveň nevyhodí vlastní.
- 8) Není-li splněna ani jedna z podmínek, rozhoduje se podle figurek, které jsou nejdál. Neberou se v potaz figurky, které nemohou táhnout (nevlezou se do cíle).

Na konci skriptu se zapisují pomocné proměnné vlastností figurky do svých originálů.

## ZÁVĚR

Vytvořená práce je určena i pro naprosté nováčky, kteří přijdou do styku s programem Matlab. Teoretická část je popsána tak, aby čtenář získal představu rozsáhlosti použití Matlabu, aby se orientoval ve vývojovém prostředí a měl přehled o možnosti použití datových typů. Nutno dodat, že z práce nevyčte postup jak něco vytvořit, ale naučí se používat prostředky pro realizaci vlastní práce. Měl by tedy být schopen psát složitější algoritmy za pomoci cyklů, podmínek s využitím souborů na ukládání dat nebo kódu. Teoretická část také proto obsahuje mnoho příkladů, aby čtenář získal větší představu o použití funkcí a jejich chování.

Z ukázky hry v praktické části by se měl uživatel orientovat v její struktuře a měl by ji být chopen ovládat a hrát. Kód v každém skriptu je řádně okomentován, aby vysvětloval jeho význam a u složitějších algoritmů i podrobněji vysvětleny jeho části a důležité proměnné, podle kterých se algoritmus orientuje. Kód je volně přístupný pro potřeby výuky na fakultě aplikované informatiky.

## SEZNAM POUŽITÉ LITERATURY

### Monografie:

- [1] DUŠEK, František. *MATLAB a SIMULINK: úvod do používání*. Vyd. 1. Pardubice: Univerzita Pardubice, 2000, 146 s. ISBN 80-719-4273-1.
- [2] HECZKO Michal. Výukový a zkušební program pro předmět PPAŘ. Bakalářská práce. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2006.
- [3] HRUBOŠ Petr. Softwarová pomůcka výuky MATLABu. Bakalářská práce. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2009.
- [4] PERŮTKA, Karel. *MATLAB - základy pro studenty automatizace a informačních technologií*. 1. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005, 303 s. ISBN 80-731-8355-2.
- [5] ZAPLATÍLEK, Karel. *MATLAB pro začátečníky*. Vyd. 1. Praha: BEN, 2003, 143 s. ISBN 80-730-0095-4.
- [6] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB: tvorba uživatelských aplikací*. 1. vyd. Praha: BEN, 2004, 215 s. ISBN 80-730-0133-0.

### Internetové zdroje:

- [7] Free Sound Effects. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.pachd.com/sounds.html>
- [8] MathWorks: Documentation Center. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/help/>
- [9] MathWorks: Products and Services. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/products/>
- [10] MATLAB CENTRAL: File Exchange. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/>
- [11] MATLAB CENTRAL: File Exchange - File Information - Flappy Bird for MATLAB. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/45795-flappy-bird-for-matlab>

- [12] MATLAB CENTRAL: File Exchange - File Information - Operation Eigenfaust 3D (Tech Demo). [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/42251-operation-eigenfaust-3d--tech-demo->
- [13] MATLAB CENTRAL: File Exchange - File Information - Stellaria (Tech Demo) - The best MATLAB shooting game ever. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/31449-stellaria--tech-demo--the-best-matlab-shooting-game-ever>
- [14] Simply the best. [online]. [cit. 2014-05-26]. Dostupné z: <http://simplythebest.net/>
- [15] Sound jay. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.soundjay.com/>
- [16] Wav Source. [online]. [cit. 2014-05-26]. Dostupné z: <http://www.wavsource.com/>

## SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GUI      Graphical User Interface – Grafické uživatelské rozhraní

GUIDE   Graphical User Interface Development Enviroment

**SEZNAM OBRÁZKŮ**

Obrázek 1. Náhled úvodní obrazovky Matlabu .....	12
Obrázek 2. Obrazovka Matlab Editor .....	29
Obrázek 3. Ukázka Matlab publikace .....	30
Obrázek 4. Náhled grafických prvků GUIDE editoru .....	34
Obrázek 5. Hra Stellaria.....	37
Obrázek 6. Hra Flappy Bird.....	38
Obrázek 7. Hra Fields Crazy.....	39
Obrázek 8. Hra Operation Eigenfaust 3D.....	40
Obrázek 9. Náhled okna programu Corel Draw .....	42
Obrázek 10. Náhled okna programu Audacity .....	43
Obrázek 11. Okno Hlavní menu .....	45
Obrázek 12. Okno Nová hra. ....	46
Obrázek 13. Okno Načíst hru .....	46
Obrázek 14. Okno Pravidla hry .....	47
Obrázek 15. Okno Hlavní hra .....	48
Obrázek 16. Okno Uložit hru.....	49
Obrázek 17. Okno Konec.....	50
Obrázek 18. Okno Vítěz .....	51

## **SEZNAM PŘÍLOH**

**PŘÍLOHA P I: CD - ROM**