

# **Vývoj prostředí pro počítačovou hru v programu MATLAB**

Lenka Šarmanová



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2015/2016

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Lenka Šarmanová

Osobní číslo: A13838

Studijní program: B3902 Inženýrská informatika

Studijní obor: Informační a řídicí technologie

Forma studia: prezenční

Téma práce: Vývoj prostředí pro počítačovou hru v programu MATLAB

Téma anglicky: The Development of a Computer Game Environment in MATLAB

Zásady pro vypracování:

1. Vypracujte literární rešerši o programu MATLAB.
2. Seznamte se s počítačovými hrami dostupnými na serveru MATLAB Central a s bakalářskými pracemi na téma počítačové hry v programu MATLAB.
3. Vytvořte strukturu programu a programové prostředí pro počítačovou hru.
4. Proveďte specifikaci hry a popište její pravidla.
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, 2001. 146 s. ISBN: 80-7194-273-1
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.

Vedoucí bakalářské práce: Ing. Karel Perůtka, Ph.D.  
Ústav řízení procesů  
Datum zadání bakalářské práce: 19. února 2016  
Termín odevzdání bakalářské práce: 27. května 2016

Ve Zlíně dne 19. února 2016



doc. Mgr. Milan Adámek, Ph.D.  
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 připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- 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/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24. 5. 2016

  
.....  
podpis diplomanta

## **ABSTRAKT**

Bakalářská práce se věnuje vývoji prostředí pro 2D počítačovou hru v programu MATLAB.

Teoretická část popisuje prostředí MATLAB a několik her tvořených v tomto programu, které jsou k dispozici na MATLAB Central nebo jako studentské práce.

Druhá část bakalářské práce je zaměřena prakticky, obsahuje popis vývoje hry, která je přiložena na disku CD-ROM. Popis hry je proveden ve dvou variantách, a to z pohledu uživatele a programátora.

Klíčová slova: MATLAB, GUI, Programování, Hra

## **ABSTRACT**

This Bachelor thesis deals with the development environment for 2D computer game in the MATLAB.

The theoretical part describes the MATLAB environment and several games developed in this program, which are available on the MATLAB Central or as the results of the student works.

The second part is oriented practically, it contains the description of the game design. The game is in the CD-ROM. The game description is made in two versions, in a programmer and user perspective.

Keywords: MATLAB, GUI, Programming, Game

Tímto bych chtěla poděkovat vedoucímu práce, panu Ing. Karlu Perůtkovi, Ph.D., za odborné vedení, připomínky, rady a návrhy během vypracování bakalářské práce.

Taktéž chci poděkovat svým kolegům, spolužákům, kteří si našli čas a stali se testery, vyzkoušeli funkčnost hry a sdělili mi své připomínky z pohledu uživatele.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

# OBSAH

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST.....</b>	<b>11</b>
<b>1 MATLAB .....</b>	<b>12</b>
1.1 O PROGRAMU .....	12
1.2 POPIS PROSTŘEDÍ.....	12
1.2.1 Command Window .....	13
1.2.2 Command History .....	13
1.2.3 Workspace.....	13
1.2.4 Current Folder .....	13
1.2.5 Menu .....	13
1.3 TYPY SOUBORŮ .....	13
1.3.1 Soubory typu FIG.....	14
1.3.2 Soubory typu M.....	14
1.3.3 Soubory typu MAT .....	14
1.4 PROMĚNNÉ .....	14
1.4.1 Lokální proměnné .....	14
1.4.2 Globální proměnné.....	14
1.5 DATOVÉ TYPY .....	14
1.5.1 Číselné datové typy .....	14
1.5.1.1 Double.....	15
1.5.1.2 Single .....	15
1.5.1.3 Datové typy uint.....	15
1.5.1.4 Datové typy int.....	15
1.5.2 Char .....	15
1.5.3 Cell .....	15
1.5.4 Struct .....	15
1.5.5 Handle funkce @ .....	16
1.6 OPERÁTORY .....	16
1.6.1 Relační operátory .....	16
1.6.2 Logické operátory .....	16
1.7 PODMÍNKY A CYKLY.....	16
1.7.1 Podmínka if .....	16
1.7.2 Switch.....	17
1.7.3 Cyklus while.....	17
1.7.4 Cyklus for.....	17
1.8 POLE A MATICE .....	18
1.8.1 Vytváření polí.....	18
1.8.2 Operace s polí.....	18
1.8.3 Indexace polí a matic .....	18

1.9	KOMPLEXNÍ ČÍSLA.....	19
1.10	NÁPOVĚDA.....	19
1.11	GRAFIKA VE 2D.....	19
1.12	GRAFICKÉ UŽIVATELSKÉ PROSTŘEDÍ.....	20
<b>2</b>	<b>POČÍTAČOVÉ HRY V MATLABU.....</b>	<b>24</b>
2.1	MATLAB CENTRAL.....	24
2.1.1	2048.....	24
2.1.2	Sudoku.....	25
2.1.3	Matris.....	26
2.1.4	Matlab Space Invaders.....	27
2.2	STUDENTSKÉ PRÁCE.....	27
2.2.1	Člověče, nezlob se.....	28
2.2.2	Riskuj.....	29
2.2.3	Pexeso.....	30
2.2.4	Pozor na kameny.....	31
<b>II</b>	<b>PRAKTICKÁ ČÁST.....</b>	<b>32</b>
<b>3</b>	<b>SPECIFIKACE HRY.....</b>	<b>33</b>
3.1	POPIS A PARAMETRY HRY.....	33
3.2	PRAVIDLA HRY.....	33
<b>4</b>	<b>REALIZACE HRY.....</b>	<b>35</b>
4.1	POPIS TVORBY HRY.....	35
4.1.1	Databáze otázek a odpovědí.....	38
4.1.2	Databáze nejlepších hráčů.....	39
4.1.3	Databáze rozehraných her.....	40
4.1.4	Testování funkčnosti hry.....	40
4.1.4.1	Otázky.....	41
4.1.4.2	Čas.....	41
4.1.4.3	Zablokovaná cesta.....	41
4.2	POUŽITÉ FUNKCE.....	42
4.3	LABYRINTHOFMATLAB.M.....	44
4.3.1	pravidla.m.....	45
4.3.2	ovladani.m.....	46
4.3.3	nejlepsiHraci.m.....	46
4.3.4	unik.m.....	47
4.4	GAMELABYRINTHOFMATLAB.M.....	48
4.4.1	OpeningFcn.....	49
4.4.2	drawLabyrinth.m.....	49
4.4.3	prekresliZdi.m.....	50
4.4.4	casovac.m.....	51
4.4.5	KeyPressFcn.....	52
4.4.6	CloseRequestFcn.....	53
4.4.7	ulozeniHry.m.....	54
<b>5</b>	<b>HRA Z POHLEDU UŽIVATELE.....</b>	<b>55</b>

5.1	HLAVNÍ OKNO .....	55
5.2	PRAVIDLA HRY .....	55
5.3	OVLÁDÁNÍ HRY .....	56
5.4	NEJLEPŠÍ HRÁČI.....	57
5.5	UKONČENÍ HRY .....	57
5.6	HRA.....	58
<b>ZÁVĚR .....</b>		<b>63</b>
<b>SEZNAM POUŽITÉ LITERATURY .....</b>		<b>64</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>		<b>65</b>
<b>SEZNAM OBRÁZKŮ .....</b>		<b>66</b>
<b>SEZNAM TABULEK.....</b>		<b>68</b>
<b>SEZNAM PŘÍLOH.....</b>		<b>69</b>

## ÚVOD

Cílem této bakalářské práce je vytvoření programového prostředí pro počítačovou hru v programu MATLAB. Výsledkem je počítačová hra, která je typu platformer ve 2D a je členěna tak, že představuje průchod místnostmi. Průchod je realizován přes tzv. brány, které se otevrou po správném zodpovězení otázek týkající se MATLABu.

První část práce popisuje základy programu MATLAB z pohledu uživatele a součástí je popis prostředí programu, dále jsou zde popsány typy souborů, se kterými se v programu setkáváme, datové typy, operátory, podmínky a cykly, pole a matice, komplexní čísla a tvorba grafických uživatelských aplikací pomocí GUIDE.

Součástí teoretické části práce jsou i ukázky her, respektive práce s grafickým uživatelským prostředím v MATLABu. Jsou zde popisy her, které jsou dostupné na MATLAB Central a také hry, které byly výsledkem studentských prací.

Druhá část práce je zaměřena prakticky. V této části popisují vývoj hry od návrhu struktury až po výsledný koncept a testování. Naleznete zde popis hry, její parametry, vývojové diagramy, blokové schéma, popis jednotlivých funkcí a krátké ukázky zdrojových kódů. Jsou zde i detailní popisy jednotlivých databází, se kterými program pracuje.

Součástí textu práce je mimo jiné i popis vytvořeného programu z uživatelského pohledu, což umožňuje vytvořit si jasnou představu o hře ještě před jejím spuštěním.

## **I. TEORETICKÁ ČÁST**

## 1 MATLAB

MATLAB je zkratkou anglických slov *matrix laboratory* a jedná se o programový balík, který poprvé představila v roce 1984 americká společnost *The MathWorks*.

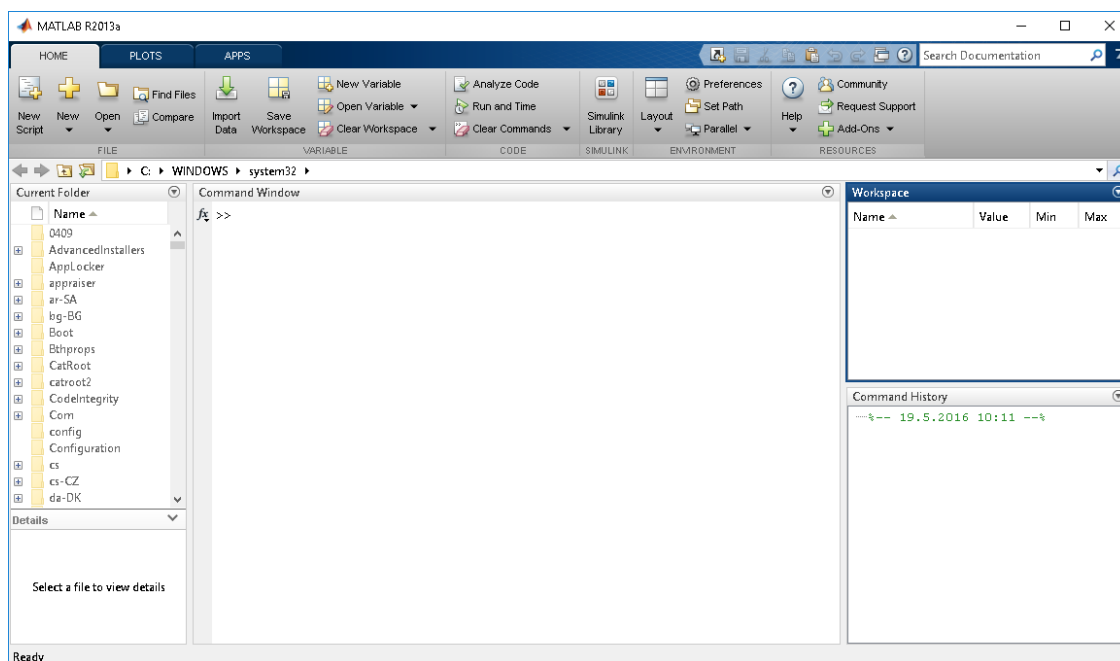
### 1.1 O programu

MATLAB je integrované prostředí, díky němuž lze zpracovávat matematické výpočty, modelovat, analyzovat a vizualizovat data, měřit data a zpracovávat je, vyvíjet algoritmy a navrhovat řídicí a komunikační systémy. [4]

Tento program je maticově orientován a je schopen za nás vypočítat mnohé složitější výpočetní operace. Co za nás program ale neudělá je návrh řešení našeho problému, pouze nám pomůže naše řešení zrealizovat. [5]

### 1.2 Popis prostředí

Po spuštění programu MATLAB se zobrazí prostředí s pracovní plochou.



Obr. 1. Pracovní plocha MATLABu

Pracovní plocha obsahuje několik částí, a to *Command Window*, *Command History*, *Workspace*, *Current Folder* a menu.

### 1.2.1 Command Window

Okno *Command Window* je nedůležitější část pracovní plochy. Slouží pro komunikaci mezi uživatelem a systémem. Uživatel zapisuje příkazy a spouští v tomto okně jednotlivé funkce. Program zde vypisuje výsledky výpočtů a systémová hlášení.

### 1.2.2 Command History

V okně *Command History* jsou uloženy informace o použitých příkazech a spuštěných funkcích. Pokud chce uživatel použít příkaz, který již jednou použil, může si jej vyhledat v tomto okně a odsud spustit. Stejně tak můžeme již použité příkazy a funkce spouštět přímo v *Command Window* použitím kláves šipky nahoru a dolů. Díky těmto možnostem je uživateli nabídnut rychlejší přístup k již použitým příkazům.

### 1.2.3 Workspace

Díky oknu *Workspace* má uživatel přehled o proměnných se kterými aktuálně pracuje. Najde zde důležité informace o proměnných jako je jejich název, rozměr a hodnota.

### 1.2.4 Current Folder

Okno *Current Folder* zobrazuje aktuální adresář a jeho obsah, jako jsou složky a jednotlivé soubory. Aktuální adresář lze jednoduše změnit dle potřeby uživatele.

### 1.2.5 Menu

Menu nad pracovní plochou slouží pro uživatele jako přehled možností práce s programem. Pomocí menu může uživatel importovat a exportovat data, nastavit zobrazení dat i celého pracovního prostředí, vytvářet uživatelské grafické prostředí, nové soubory, funkce a skripty. [4]

Menu je přehledné a usnadňuje práci s programem především uživatelům, kteří jsou méně zdatní na programování.

## 1.3 Typy souborů

V programu MATLAB se můžeme setkat s několika typy souborů, vybrané základní a často používané běžnými uživateli jsou zmíněny níže.

### 1.3.1 Soubory typu FIG

S tímto typem souboru se uživatel setká především při tvorbě grafického uživatelského prostředí, protože nese informaci o obsahu grafického okna.

### 1.3.2 Soubory typu M

Jedná se o textové soubory, do kterých je zapsán kód funkcí a příkazů, které mají být vykonány při spuštění souboru. Tyto soubory jsou vytvářeny uživatelem jako vlastní funkce nebo jednoduché skripty.

### 1.3.3 Soubory typu MAT

Tyto typy souborů slouží pro ukládání dat na disk, převážně řetězců, matic a buněčných polí. Zápis dat na disk se provádí pomocí příkazu *save*, čtení dat příkazem *load*.

## 1.4 Proměnné

V MATLABu není třeba definovat či deklarovat proměnné, protože MATLAB tuto akci udělá sám při prvním výskytu proměnné v *Command Window*, M-souborech či skriptech.

### 1.4.1 Lokální proměnné

Výchozí formát proměnné je lokální proměnná, kterou není třeba deklarovat ani definovat. Pokud se ale hodnota proměnné v průběhu programu mění, například v cyklu nebo podmínce, je třeba ji definovat na začátku skriptu nebo funkce.

### 1.4.2 Globální proměnné

Tento formát proměnných se používá zejména kvůli funkcím. Každá funkce má své lokální proměnné, jejichž hodnoty ale nelze zjistit z ostatních funkcí. Pokud tedy chceme, aby proměnná byla dostupná z více funkcí, musíme ji deklarovat jako globální pomocí příkazu *global*.

## 1.5 Datové typy

Jako v každém programovacím jazyce, i v MATLABu lze využít několika datových typů.

### 1.5.1 Číselné datové typy

Datové číselné typy lze dělit do několika kategorií – datové typy pracují s celými čísly, reálnými čísly, nezápornými celými čísly a zápornými čísly.

#### 1.5.1.1 *Double*

*Double* je jedním z nejběžněji používaných datových typů v MATLABu. Používají se při výpočetních operacích s maticemi a reálnými čísly. Tato čísla se skládají z celé a desetinné části, přičemž je v paměti uloženo tak, aby mohlo dojít ke změně jeho přesnosti, tj. počtu desetinných míst. Díky této vlastnosti lze zajistit vysokou přesnost při výpočtech.

#### 1.5.1.2 *Single*

*Single* je podobný datovému typu *double*, avšak jeho přesnost je nižší. Čísla zabírají menší část paměťového prostoru, mají menší rozsah a tudíž i přesnost. Pro objekty tohoto typu nejsou definované žádné matematické operace.

#### 1.5.1.3 *Datové typy uint*

Existuje několik datových typů *uint* – *uint8*, *uint16*, *uint32* a *uint64*. Společnou vlastností je, že se jedná o nezáporná celá čísla, přičemž rozdílnou vlastností je rozsah těchto datových typů.

Pro tato čísla nejsou definované žádné matematické operace, výjimkou je funkce *sum*, která slouží pro součet čísel těchto datových typů.

#### 1.5.1.4 *Datové typy int*

Stejně jako u datového typu *uint* existuje několik již zmíněných typů, stejné typy existují i zde – *int8*, *int16*, *int32* a *int64*. Tento datový typ slouží pro celá čísla jak nezáporná tak záporná. Opět rozdílnou vlastností jednotlivých typů je rozsah čísel.

### 1.5.2 *Char*

Tento datový typ je určen pro znaky a řetězce. Řetězce typu *char* jsou ukládány jako u datového typu *uint16*, tedy jako celá nezáporná 16ti-bitová čísla.

### 1.5.3 *Cell*

Datový typ *cell*, neboli buňka, má své použití v buněčných polích, o kterých lze říct, že to jsou matice, které místo číslic používají řetězce a znaky.

### 1.5.4 *Struct*

Mluvíme-li o strukturách, mluvíme tedy o mnohorozměrných polích. Struktura se chová podobně jako buněčné pole. Každé pole struktury má své jméno, tzv. textový určovatel

pole, pomocí kterého můžeme k poli přistupovat. Pole struktury může obsahovat novou strukturu, v tomto případě hovoříme o vnořené struktuře.

### 1.5.5 Handle funkce @

Datový typ *handle funkce* obsahuje informace o odkazování se na funkci, umožňuje tedy přístup k funkci jinými funkcemi. Při volání funkce přes *handle* následně použijeme funkci *feval*, kde zadáme jako vstupní parametry *handle* funkci a vstupní argumenty do funkce.

## 1.6 Operátory

Operátory v MATLABu lze rozdělit do dvou kategorií – relační a logické operátory.

### 1.6.1 Relační operátory

Relační operátory používáme pro porovnávání dvou hodnot stejného datového typu, popřípadě dvou polí stejné dimenze a délky. Výsledkem porovnání je logický typ 0 nebo 1, tudíž hodnota *false* nebo *true*.

Mezi relační operátory patří: menší než (<), větší než (>), menší než nebo rovno (<=), větší než nebo rovno (>=), rovná se (==) a nerovná se (~=).

### 1.6.2 Logické operátory

MATLAB má 5 základních logických operátorů, které se aplikují do pole po prvcích. Do této kategorie patří (podle sestupné priority): Negace (~), logický součin pro pole (&), logický součet pro pole (|), logický součin pro podmínku (&&) a logický součet pro podmínku (||). Výsledkem logického porovnání je, stejně jako u relačních operátorů, logická 0 nebo 1.

## 1.7 Podmínky a cykly

Při programování je nezbytné použití podmínek a cyklů.

### 1.7.1 Podmínka if

Je-li vyhodnocen logický výraz za *if* nebo *elseif* jako logická jednička, provede se příkaz v dané větvi. Větev *elseif* může být v podmínce použita v libovolném množství nebo vůbec. Pro vyhodnocení složitějších logických výrazů se používají relační či logické operátory. Základní syntaxe podmínky *if* v programu MATLAB vypadá následovně:

```
1 if logický výraz
```

```
2      příkazy
3 elseif logický výraz
4      příkazy
5 elseif logický výraz
6      příkazy
7 else
8      příkazy
9 end
```

### 1.7.2 Switch

Stejně jako v každém programovacím jazyce, i v MATLABu můžeme použít *switch* pro mnohonásobné větvení. Pokud existuje výraz za slovem *switch*, vezme se jeho hodnota a porovnává se s výrazy ve větvích *case*. Pokud je nalezena shoda, provedou se příkazy v dané větvi. Pokud shoda nebyla nalezena, provede se příkaz ve větvi *otherwise*.

Základní syntaxe větvení pomocí *switch*:

```
1 switch výraz
2     case výraz,
3         příkaz, ..., příkaz
4     case výraz {výraz1, výraz2, výraz3, ...}
5         příkaz, ..., příkaz
6     ...
7     otherwise
8         příkaz, ..., příkaz
9 end
```

### 1.7.3 Cyklus while

Principem cyklu je, že pokud je logický výraz za slovem *while* vyhodnocen jako pravda (logická 1), provádí se příkazy uvnitř cyklu, dokud není výraz vyhodnocen jako nepravda (logická 0). Cyklus lze předčasně ukončit pomocí příkazu *break*. Základní syntaxe cyklu *while* je následující:

```
1 while logický výraz
2     příkazy
3 end
```

### 1.7.4 Cyklus for

Cyklus *for* používáme, pokud známe počet opakování cyklu. Principem je provádění příkazů uvnitř cyklu pro daný počet opakování. Stejně jako u cyklu *while*, i zde je možnost

předčasně cyklus ukončit pomocí příkazu *break*, který je obvykle ukryt v nějaké *if* podmínce. Syntaxe *for* cyklu v MATLABu:

```
1 for proměnná = výraz
2     příkazy
3 end
```

## 1.8 Pole a matice

Pokud se rozhodneme použít MATLAB, tak hlavním důvodem bude použití právě polí a matic pro výpočetní operace. Pole je sada čísel, kde se na jednotlivé elementy odkazujeme pomocí indexů. Počet indexů potřebných pro specifikaci elementu pole se nazývá dimenze. Matice je dvoudimenzionálním polem, kde je jednou dimenzí řádek a druhou sloupec.

### 1.8.1 Vytváření polí

Při tvorbě pole vkládáme elementy do hranatých závorek, přičemž jednotlivé prvky sloupce jsou odděleny mezerou nebo čárkou a řádky středníkem. Vektor lze jednoduše vytvořit pomocí dvojteček, kde definujeme první prvek, za dvojtečkou krok a za další dvojtečkou konečný prvek. Stejného výsledku docílíme použitím funkce *linspace*, kde do vstupních parametrů zadáváme první číslo, poslední číslo a místo kroku zadáváme počet čísel, které chceme v daném rozmezí vygenerovat.

### 1.8.2 Operace s poli

Výhodou MATLABu je, že pro operaci s poli nepotřebujeme používat cykly jako v jiných programovacích jazycích. Operace lze provádět s celými poli, nebo s jednotlivými prvky, kdy před symbol operace umístíme tečku.

Typickým příkladem pro operaci s poli je řešení lineárních algebraických rovnic. Do matice *A* vložíme koeficienty neznámých na levé straně rovnic, pravé strany rovnic vložíme do sloupcového vektoru *b*. Pro řešení soustavy rovnic použijeme operaci levostranného maticového dělení, čímž dostaneme výsledek ve vektoru *x*:

```
1 A = [2 8 1; 4 5 3; 9 7 4];
2 b = [3 8 5];
3 x = A\b
```

### 1.8.3 Indexace polí a matic

Indexace je možná dvojím způsobem, po prvcích nebo podle dimenze. Indexy jsou celá čísla zapsaná do závorek za název matice.

Máme-li matici  $A$  o 3 řádcích a 6 sloupcích, celkem tedy s osmnácti prvky, a chceme zobrazit poslední prvek, tedy prvek v posledním řádku i sloupci, použijeme indexaci po prvcích:

```
1 A(18)
```

Stejného výsledku lze docílit pomocí dimenzí:

```
1 A(3, 6)
```

Pokud chceme vyjmout více prvků, například celý řádek nebo sloupec, používáme dvojtečku. Dvojtečka značí celý rozsah. Tento způsob se používá při indexování pomocí dimenzí.

## 1.9 Komplexní čísla

I práce s komplexními čísly je jednou z možností, které nabízí program MATLAB. Pokud je imaginární část komplexního čísla nulová, pak se s ní nepočítá a číslo je automaticky vyhodnoceno jaké reálné. Pro identifikaci imaginárního čísla se v MATLABu používá znak  $i$  nebo  $j$ . Mezi hodnotou imaginární části komplexního čísla a hodnotou reálné části je dovoleno psát znak  $*$ , ale není podmínkou. [6]

## 1.10 Náповěda

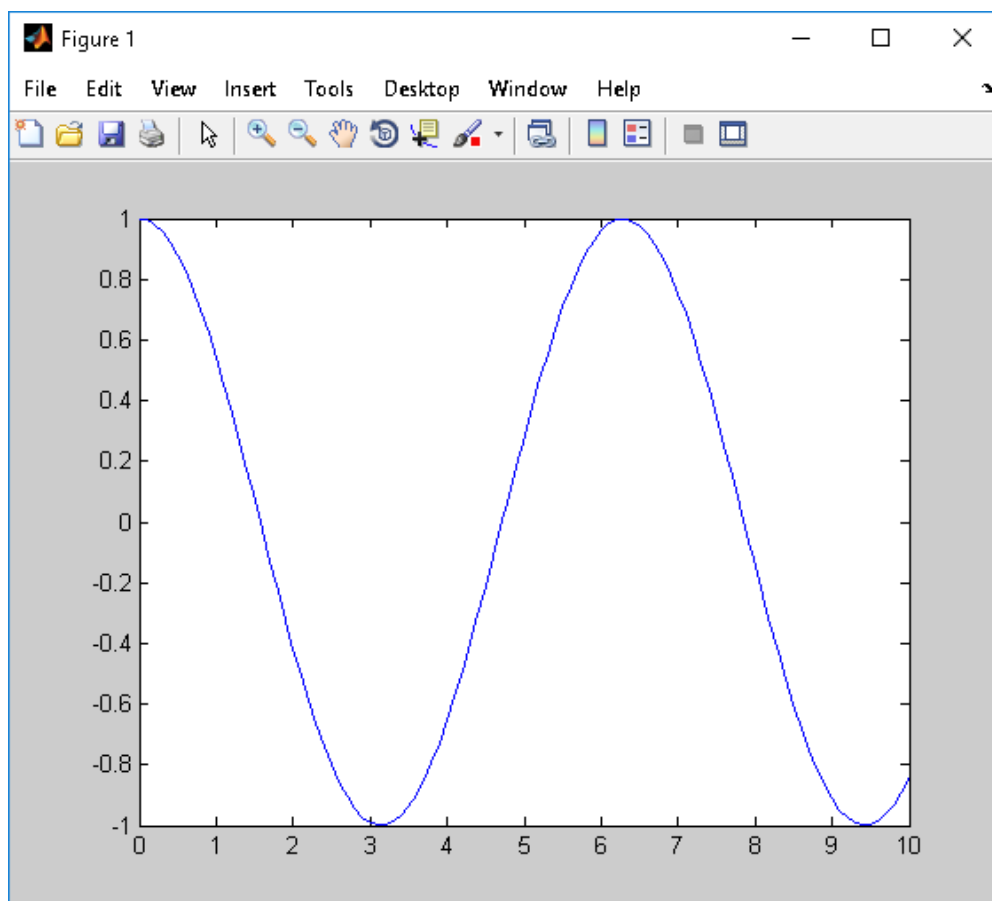
Příkaz *help* je nejpoužívanějším v MATLABu. Pomocí tohoto příkazu zavoláme nápovědu. Náповěda nám nabídne všechny možné funkce MATLABu a jejich funkčnost většinou odpovídá jejich názvu, takže pokud chceme vypočítat například determinant matice, ale neznáme název funkce pro výpočet, otevřeme si nápovědu a podle názvu funkcí hledáme funkci, která by odpovídala danému problému. V tomto konkrétním případě se jedná o funkci s názvem *det*.

Pokud známe název funkce, ale nevíme jak ji použít, můžeme pomocí příkazu *help* s názvem funkce otevřít dokumentaci této funkce, kde zjistíme, jak vypadá její syntaxe a jak se funkce používá. [5]

## 1.11 Grafika ve 2D

Pro grafický výstup nejčastěji používá funkci *plot* pro zobrazení grafu.

```
1 t = 0:0.1:10;  
2 kosinus = cos(t);  
3 plot(t, kosinus)
```



Obr. 2. Grafický výstup funkce *plot*

Vyobrazené grafické okno kromě výsledku funkce *plot* má menu a ikonovou lištu. Díky těmto pomůckám můžeme napsat libovolný text do grafického okna, nakreslit úsečku, zvětšovat a zmenšovat výstup či jím rotovat.

Výstup z grafického okna můžeme pomocí menu uložit jako obrázek. Programováním nebo pomocí menu můžeme nastavit popisy os, nadpis a legendu a také můžeme zvolit typ grafu, jeho barvu a vzhled.

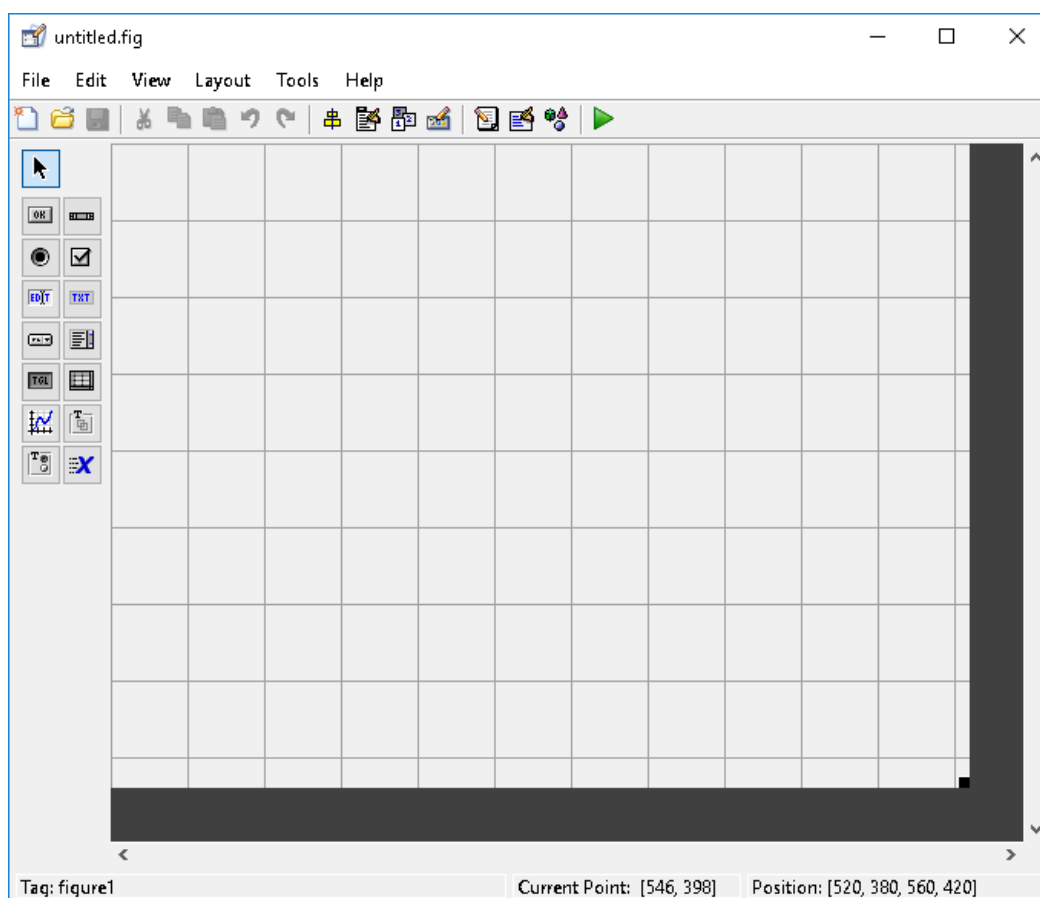
Funkce *plot* nám dovoluje také vložit více grafů s různými osami do jednoho grafického okna. Uživatel má mnoho možností vzhledu grafu, přičemž je uskuteční nastavením různých vlastností grafů, které nalezne pomocí nápovědy. [6]

## 1.12 Grafické uživatelské prostředí

Nástroj pro interaktivní tvorbu grafického rozhraní v MATLABu se nazývá GUIDE. Ať už je uživatel zkušeným programátorem, nebo začátečníkem, vždy dá přednost nástroji, který uživateli pomůže s návrhem dialogového okna. GUIDE nabízí interaktivní způsob, kdy uživatel pomocí myši umisťuje grafické objekty a zadává jejich parametry.

Výhodou GUIDE je univerzální a časově nenáročná tvorba grafického uživatelského prostředí, přičemž není potřeba, aby si uživatel pamatoval jednotlivé položky grafických objektů. Nevýhodou GUIDE je delší zdrojový kód s odlišnou strukturou, která může být pro uživatele nezvyklá.

Průvodce prostředím GUIDE spustíme pomocí hlavního menu MATLABu v záložce *New -> Graphical User Interface -> Default GUI*, přičemž se nám zobrazí prázdné grafické okno, které lze kdykoliv uložit jako soubor s příponou FIG.

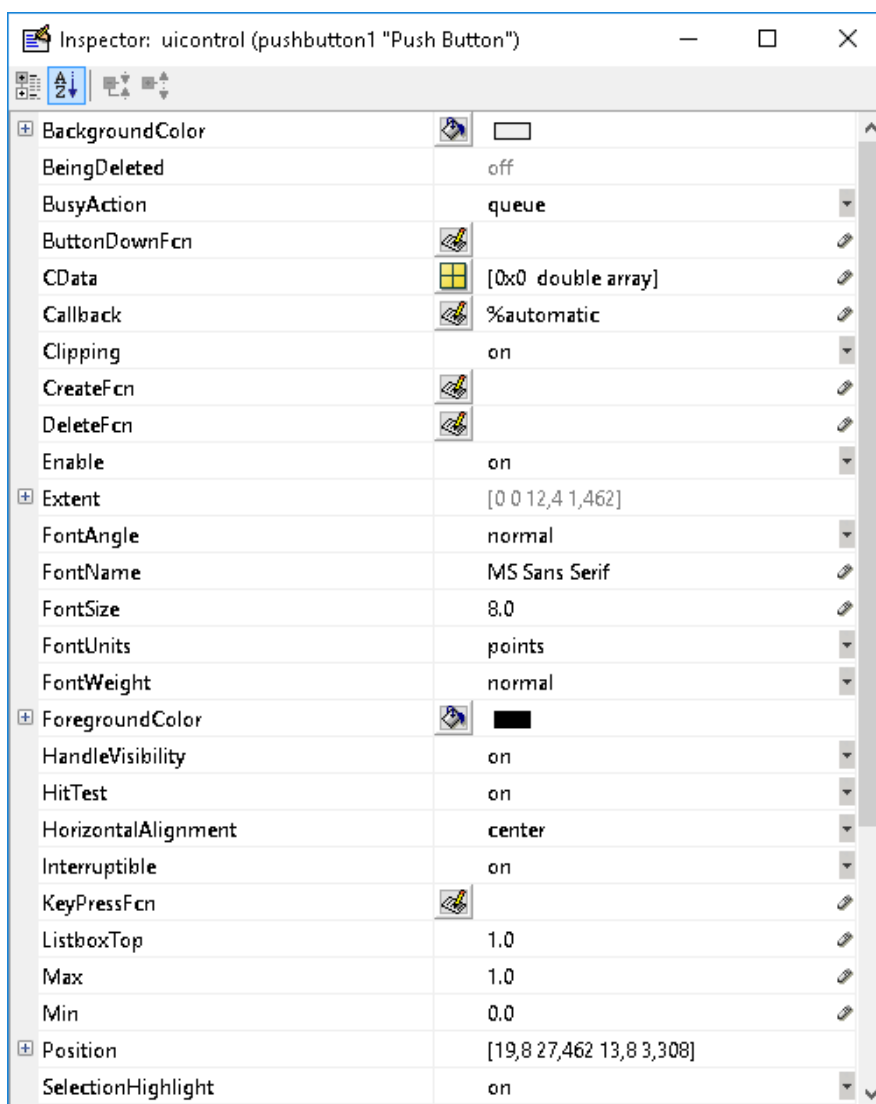


Obr. 3. Prázdné grafické okno při tvorbě GUI

V levé části okna vidíme nabídku grafických objektů, které lze do grafického uživatelského prostředí vložit. Najdeme zde posuvník (*Slider*), přepisovatelné textové pole (*Edit Text*), tlačítko (*Push Button*), zaškrťovací tlačítko (*Radio Button*), další typ zaškrťovacího tlačítka (*Check Box*), text vhodný pro nadpisy (*Static Text*), nabídku v rozbalovacím seznamu (*Pop-up menu*), nabídku v seznamu (*ListBox*), dvoustavové tlačítko (*Toogle Button*), tabulku (*Table*), osu (*Axes*), panel (*Panel*) a skupinu tlačítek (*Button Group*) pro seskupení tlačítek do skupiny.

Pro umístění jakéhokoliv ovládacího prvku ze zmíněných, klikneme na daný objekt v levé části okna a myši přesuneme do grafického okna, kde si jej pomocí myši můžeme umístit na libovolnou pozici, popřípadě změnit rozměry pomocí myši.

Pokud chceme nastavit další vlastnosti grafického objektu, pravým kliknutím myši se dostaneme do *Property Inspector*, čili do nastavení grafického objektu. Zde lze v závislosti na typu nastavovaného objektu nastavit mnoho vlastností, například barvu objektu, barvu pozadí, styl, zarovnání a velikost písma, výchozí stav, rozměry, umístění, viditelnost a tag pro tento grafický objekt, pomocí kterého budeme v kódu na daný objekt volat.



Obr. 4. Ukázka Property Inspector dialogu pro tlačítko

Po uložení všech vložených grafických objektů, jejich nastavení a uložení grafického okna, se automaticky vytvoří M-soubor se stejným názvem jako grafické okno, kde už jsou předepsané jednotlivé funkce, do kterých lze přidávat doplňující kód.

V GUIDE lze také jednotlivým objektům nastavit různé akce pravým kliknutím, respektive funkce zpětného volání. Například, klikneme-li pravým tlačítkem myši na plochu grafického okna a vybere volbu *KeyPressFcn*, automaticky se v kódu vytvoří tato funkce, která bude při běhu aplikace reagovat na stisk tlačítka. Do této funkce napíšeme kód, který bude aplikace vykonávat při reakci na stisk klávesy. [7]

```
% --- Outputs from this function are returned to the command line.
function varargout = testovaci_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --- Executes on key press with focus on figure1 and none of its controls
function figure1_KeyPressFcn(hObject, eventdata, handles)
% hObject handle to figure1 (see GCBO)
% eventdata structure with the following fields (see FIGURE)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles structure with handles and user data (see GUIDATA)
```

*Obr. 5. Ukázka části kódu vygenerovaného MATLABem*

## 2 POČÍTAČOVÉ HRY V MATLABU

Součástí mé práce bylo seznámení se s počítačovými hrami na serveru MATLAB Central a s bakalářskými pracemi na téma počítačové hry v MATLABu.

### 2.1 MATLAB Central

Na serveru MATLAB Central jsou hry se svolením autorů volně stažitelné. Objevují se zde různé kategorie her, jako jsou logické, oddechové, zábavné, stříleční a 3D hry.

#### 2.1.1 2048

Tato hra se v posledních letech stala velmi populární a je rozšířením hry 1024 se stejnými pravidly. Jejím autorem je Ital Gabriele Cirulli. Uživatel @bmtran ji převedl do programu MATLAB a zveřejnil na serveru MATLAB Central.

Hra se ovládá pomocí šipek na klávesnici. Cílem hry je spojit stejná čísla dohromady a dosáhnout konečné hodnoty 2048. Stisknutí šipky na klávesnici určuje směr, kterým se všechny dlaždice vydají v poli 4x4, dokud nejsou zastaveny jinou dlaždicí, či okrajem. Dlaždice stejné hodnoty se posunem spojí dohromady a sečtou. Při každém tahu, respektive pohybu v poli, se objeví nová dlaždice s minimální hodnotou 2 a tak, pokud hráč nezvolí dobrou strategii, může se celé pole zaplnit dlaždicemi s různými hodnotami, které nejdou spojit, a hráč v takové situaci nemá možnost dalšího tahu.

Hra je ke stažení přímo jako aplikace, tudíž není možno vidět její kód. [1]



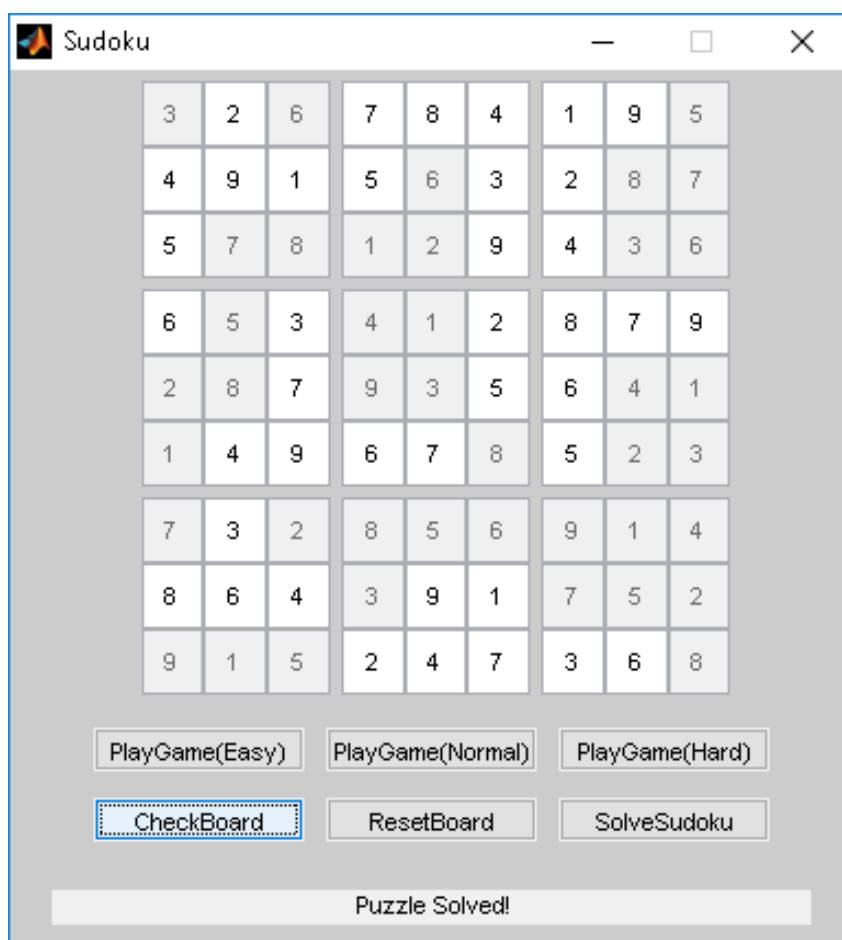
Obr. 6. Hra 2048

### 2.1.2 Sudoku

Autorem této známé hry v MATLABu je Stephen Sahrn. Cílem hry Sudoku je vyplnit prázdná místa v poli 9x9 pomocí čísel 1-9 a to tak, aby v každém řádku, sloupci i čtverci byla čísla 1-9 a neopakovala se.

Při hře je možnost vybrat si ze tří obtížností – lehké, střední nebo těžké sudoku. Je zde také možnost resetovat rozehranou hru – spustit ji od začátku, nebo spustit řešitele, který hru vyřeší za nás. Po dokončení hry máme možnost spustit kontrolu hry, která vyhodnotí, zda je pole vyplněné správně, a pokud ne, zvýrazní políčka s chybami. Čísla do jednotlivých políček zapisujeme pomocí numerické klávesnice. Pro výběr obtížnosti hry nebo resetu, kontroly či řešitele používáme myš.

Program obsahuje dva M-soubory, přičemž jeden je pro hru a druhý je jako řešitel sudoku. Dohromady mají oba dva soubory cca 850 řádků kódu. [2]



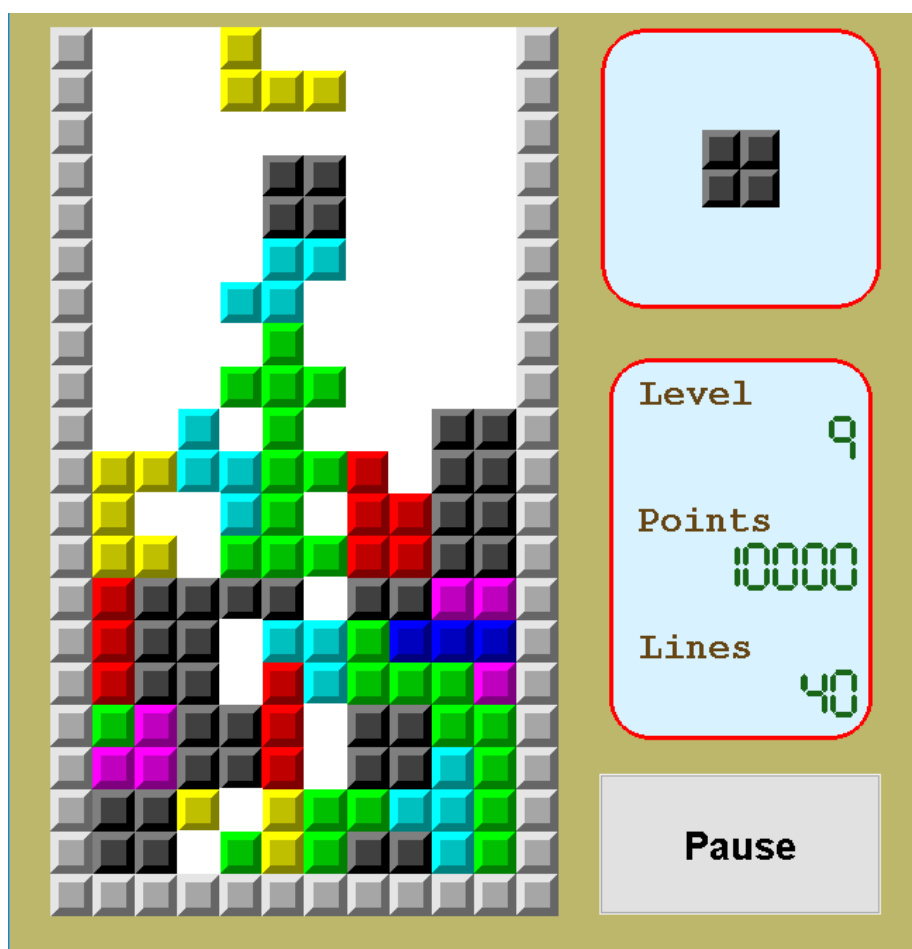
Obr. 7. Hra Sudoku

### 2.1.3 Matris

Hra Matris je variantou známé hry Tetris, vytvořené pomocí programu MATLAB, jejímž autorem je Mingjing Zhang.

Hra Matris se ovládá pomocí myši a kláves. Pro spuštění hry je nutno kliknout na tlačítko Start. Do pole padají náhodně tvary z databáze a hráčovým úkolem je poskládat padající tvary vedle sebe tak, aby zaplnily celou řadu a následně řada zmizela. S tvary lze hýbat doleva a doprava pomocí levé a pravé šipky, pomocí šipky nahoru lze s tvarem otáčet a pomocí šipky dolů lze urychlit jeho spadnutí na zem. V pravém horním rohu je zobrazen tvar, který se do pole spustí jako další, což umožňuje hráči dopředu taktizovat. V pravé části okna vidíme také informace o výsledcích hry, a to naše skóre a počet zmizelých řad. Se vzrůstajícím skóre hry se zvyšuje rychlost padání jednotlivých tvarů, čímž se hra stává obtížnější. Pokud hráč nestihne uspořádat tvary a nějaký z tvarů se dotkne horní osy, hra končí. Hru je možné také kdykoliv pozastavit.

Program obsahuje jeden M-soubor s cca 850 řádky kódu. Hra má velmi pěknou grafiku.[3]



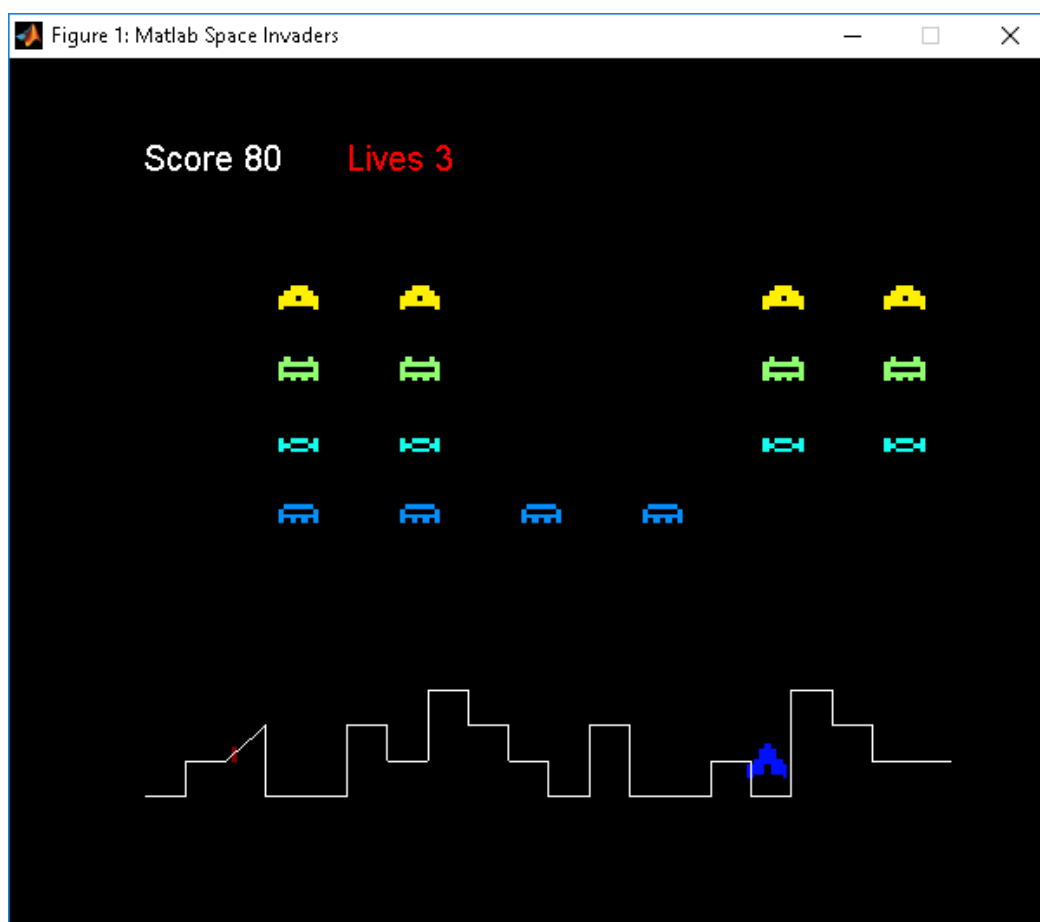
Obr. 8. Hra Matris

### 2.1.4 Matlab Space Invaders

Hra Matlab Space Invaders je dílem autora jménem Héctor Corte. Jedná se o střeleckou hru podobnou klasické videohře.

Hra je pro jednoho hráče. Cílem hry je postřílet všechny kosmické lodě pomocí laseru, přičemž palbu spouštíme mezerníkem a šipkami se pohybujeme doleva a doprava s naší vesmírnou lodí. Zároveň se musíme vyhýbat také laserovým střelbám od ostatních kosmických lodí. Hráč má pouze tři životy. Hra je doplněna o hudbu a zvukové signály střelby.

Program obsahuje tři třídy a dva M-soubory a je opravdu věrnou napodobeninou klasické videohry. [8]



Obr. 9. Hra Matlab Space Invaders

## 2.2 Studentské práce

Několik mých kolegů v minulých letech vytvořilo počítačové hry v MATLABu. Tyto programy jsem získala od vedoucího práce.

### 2.2.1 Člověče, nezlob se

Autorem této klasické stolní hry je David Fiala, kterou vytvořil v roce 2013/2014 jako svou bakalářskou práci na téma Počítačová hra ve 2D v MATLAB.

Hra je pro 2-4 hráče, přičemž hráči mohou být buď reální lidé, nebo počítač. Cílem hry Člověče, nezlob se je projít svými čtyřmi figurkami až do cílového pole určeného stejnou barvou, jakou mají figurky hráče a to dříve, než ostatní hráči. Jednotliví hráči se střídají v hodů kostkou, přičemž výsledek hodu udává počet políček posunu. Pokud ještě není žádná z našich figurek v herním poli, ale je v domečku, máme možnost 3x hodit kostkou, dokud nehodíme číslo 6, díky kterému můžeme vložit figurku z domečku do pole. V případě, že máme v herním poli více figurek, můžeme po hodu kostkou vybrat, která z figurek se posune o počet políček daných hodem. Pokud se naše figurka objeví na políčku, kde už stojí figurka jiného hráče, tak se protihráčova figurka musí vrátit zpátky do domečku.

Hru můžeme kdykoliv ukončit, či si ji uložit a opětovně spustit. Také můžeme během hry nahlédnout do pravidel. Autor si dal práci s vytvářením grafické stránky hry a tak je hra velmi pěkně graficky zpracována. V *CommandWindow* MATLABu se během hry objevují komentáře k průběhu hry a v pravé části herního okna máme přehled o aktuálním stavu hry, konkrétně o tom, který hráč je na tahu.

Program obsahuje cca 50 M-souborů, z nichž většina jsou skripty podle jména jasně definující funkčnost.



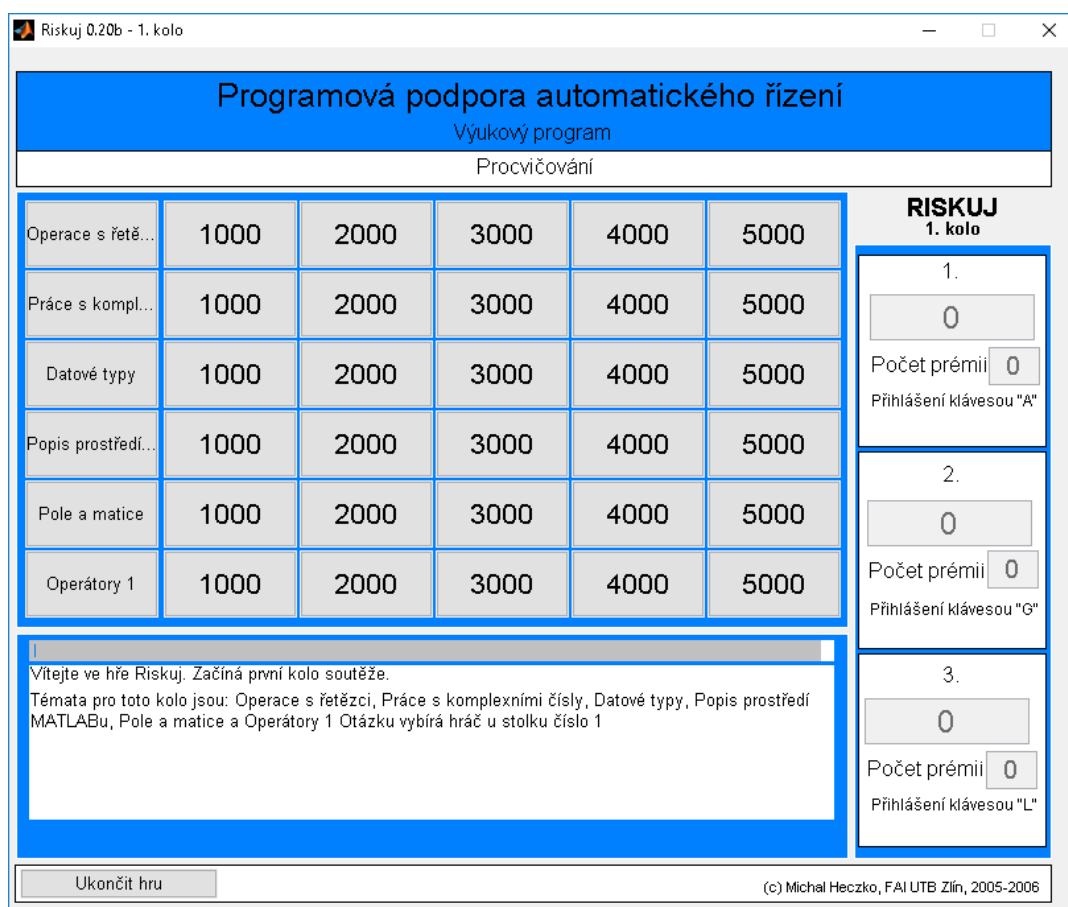
Obr. 10. Hra Člověče, nezlob se

### 2.2.2 Riskuj

V roce 2005/2006 Michal Heczko vytvořil jako součást své práce, Výukového a zkušební-ho programu pro předmět PPAŘ, hru Riskuj, jakožto mód procvičování tematiky MATLABu.

Hra je určena pro 1-3 hráče. Jak už název napovídá, pravidla hry jsou stejná, jako v televizním zábavném soutěžním pořadu Riskuj. Konkrétně v této hře jsou okruhy témat zaměřeny na popis prostředí MATLABu, operátory, operace s řetězci, práce s komplexními čísly, pole a matice. Hra se ovládá pomocí myši i klávesnice. Hráč, který je na tahu, si vybírá téma a hodnotu otázky, na kterou chce odpovídat. Při správném zodpovězení se mu tato hodnota přičte do skóre. Ve hře je zaveden časový limit jak pro odpověď hráče, tak pro délku kola, oba dva jsou reprezentovány indikátorem průběhu (*ProgressBar*). Po skončení časového limitu vyhrává hráč s více body.

Samotná hra obsahuje asi 40 M-souborů a 5 souborů s příponou FIG. Jelikož je celá aplikace výukovým programem je tato hra osvěžením pro uživatele, který si díky ní může ověřit své znalosti týkající se MATLABu zábavnou formou.



Obr. 11. Hra Riskuj

### 2.2.3 Pexeso

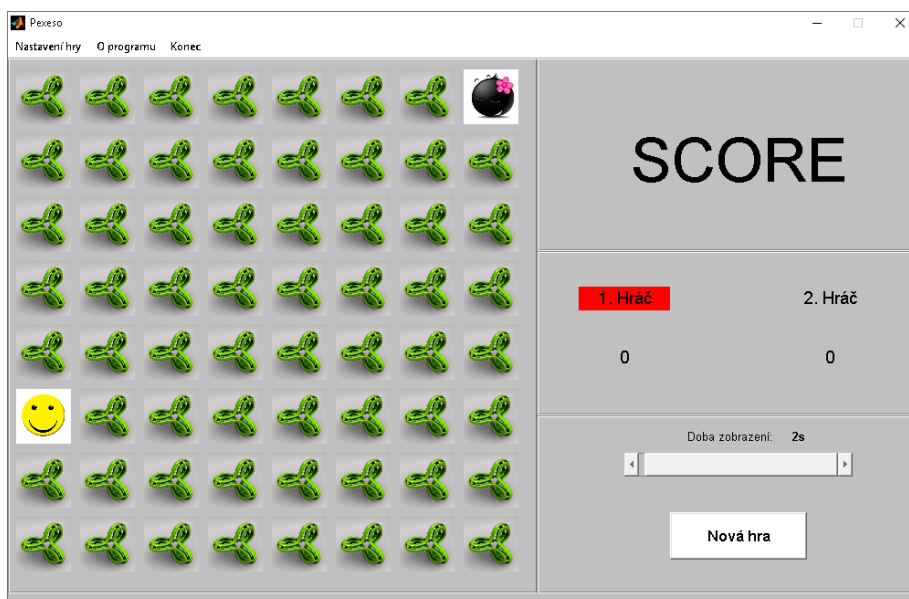
V roce 2008/2009 byla vytvořena počítačová hra Pexeso v MATLABu Tomášem Sedlákem.

Hra je pro 2 hráče. Před spuštěním samotné hry lze navolit velikost pexesa (4x4, 6x6 nebo 8x8), vzhled rubu pexesa (zámek, semafor, smyčka nebo klasický), vzhled líce pexesa (abeceda, smajlíci nebo dopravní značky) a také lze zadat jména hráčů. Po základním nastavení je hra spuštěna a řídí se pravidly klasické hry pexeso.

Pravidla hry pexeso jsou jednoduchá. Každý hráč musí během svého tahu otočit dvě kartičky, o kterých si myslí, že k sobě patří. Pokud tomu tak je, tato dvojice kartiček je smazána z pole a hráči, který společnou dvojici našel, se přičte bod a má nárok na další tah. Pokud hráč otočí kartičky, které k sobě nepatří, karty se otočí nazpět rubem a táhne protihráč. Ve hře lze nastavit čas pro prohlížení karet od 0,5 do 2 sekund, čímž lze nastavit obtížnost hry. Hráči se v tazích jednotlivě střídají a hra končí až je pole prázdné a vyhrává hráč s vyšším počtem bodů.

Program obsahuje asi 20 M-souborů. Zaujala mě možnost vybrat si z více grafických možností vyobrazení rubu karet a také nastavení obtížnosti hry pomocí času pro prohlížení karet v rozmezí po 0,1 sekundě.

Avšak si myslím, že hra by mohla být doplněna o možnost hrát proti počítači a také by karty pexesa neměly obsahovat jednoduché obrázky, ale výukovou tematiku, například v oblasti prostředí MATLAB nebo automatizace.



Obr. 12. Hra Pexeso

### 2.2.4 Pozor na kameny

Protože mě baví programování v MATLABu, tak jsem si pro procvičení jeho možností v roce 2015 vytvořila jednoduchou počítačovou hru, kterou jsem nazvala Pozor na kameny.

Hra je pro jednoho hráče a ovládá se pomocí šipek doleva a doprava. Cílem hry je vyhýbat se padajícím kamenům a zůstat naživu co nejdéle. Při spadnutí jednoho kamene, kterému se hráč vyhne, získá hráč jeden bod. Na hráče padají vždy tři kameny vygenerované na náhodné počáteční pozici a odlišné barvou podle rychlosti. S vyšším skóre hráče se postupně zrychluje padání kamenů a hráčovi reakce na úhyb musí být rychlejší a rychlejší.

Program obsahuje jeden soubor s příponou FIG a M-soubor s cca 300 řádky kódu. Jednoduchý retro design této hry posloužil jako předloha grafického návrhu bakalářské práce.



Obr. 13. Hra Pozor na kameny

## **II. PRAKTICKÁ ČÁST**

### 3 SPECIFIKACE HRY

Kapitola je věnována popisu hry, jejím parametrům a pravidlům.

#### 3.1 Popis a parametry hry

Hra Labyrinth of MATLAB je určena pro jednoho hráče. Hra obsahuje celkem 10 místností, kterými musí hráč v daném časovém limitu projít. V každé místnosti je labyrint, ve kterém jsou překážky ve formě tzv. bran. Bránou hráč projde, pokud správně odpoví na otázku týkající se MATLABu.

Pozice hráče v labyrintu se indikuje zeleným čtvercem, kterým lze pohybovat klávesami šipek. Na otázky odpovídáme tak, že zadáme písmeno nabízené volby na klávesnici, tj. např. a, b, c nebo d.

Labyrinth of MATLAB si může zahrát kdokoliv, ke správnému zodpovězení otázek je nutné mít základní zkušenosti s programem MATLAB. Hra je založena na více činnostech najednou, kdy hráč musí hledat správnou cestu pro průchod labyrintem, hlídat si ubíhající čas, posbírat co nejvíc bodů a také vykazovat své znalosti v oblasti MATLABu.

Hru lze spustit v *Command Window* programu MATLAB pomocí funkce *LabyrinthOfMatlab.m*, tedy zadání příkazu *LabyrinthOfMatlab*.

Tab. 1. Parametry hry

<i>Autor</i>	Lenka Šarmanová
<i>Určeno pro</i>	lidé se znalostí programu MATLAB
<i>Počet hráčů</i>	1
<i>Délka hry</i>	maximálně 20 min

#### 3.2 Pravidla hry

Cílem hry je projít deseti místnostmi. V každé místnosti je labyrint, kterým musí hráč projít a dostat se k cíli, respektive do další místnosti, v časovém limitu dvou minut.

V labyrintu jsou umístěny během cesty do cíle překážky ve formě tzv. bran, které hráč odstraní pouze správnými odpověďmi na otázky týkající se MATLABu. Pokud hráč odpoví špatně, nebo vůbec, tak se ze základní (žluté) brány stane varovná (oranžová) brána. V této situaci má hráč už jen jeden pokus na správnou odpověď a zároveň poslední možnost od-

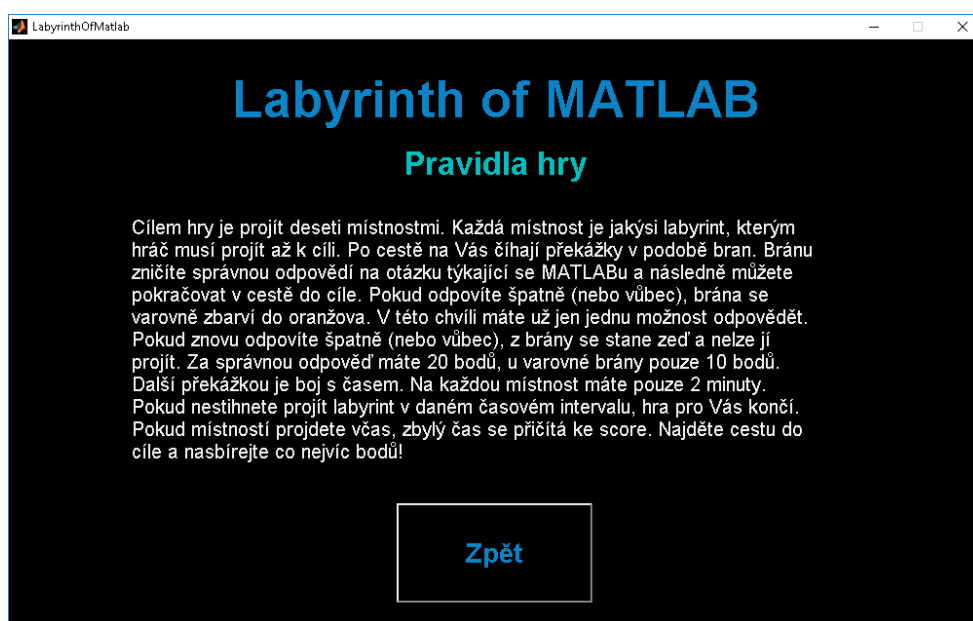
stranit překážku. Pokud hráč znovu odpoví špatně, nebo vůbec, z varovné brány se stane neprůchodná zeď a hráč si tak zablokuje jednu z možných cest k cíli.

Pokud hráč odpoví na otázku správně napoprvé, nejen že odstraní bránu, ale získá 20 bodů za správnou odpověď, při správné odpovědi až na druhý pokus získá hráč 10 bodů. Jestliže hráč projde místností dříve, jak za daný časový limit, za každou zbývajících sekundu získá 1 bod.

V databázi je celkem 200 otázek. Každá místnost obsahuje 20 otázek a labyrinty jsou navrženy tak, aby hráč měl více možností průchodu k cíli a nemusel přitom odpovídat na úplně všechny otázky, které místnost obsahuje, alespoň u prvních, jednodušších levelů. U složitějších místností si hráč může zablokovat jednou nesprávně zodpovězenou otázkou cestu do cíle a prohrát. Hráč má maximálně dva pokusy na zničení brány správnou odpovědí. Otázky jsou při každé hře generovány náhodně a jejich odpovědi jsou taktéž vždy promíchány, což zajišťuje jedinečnost každé hry.

Body získané průchodem jednotlivých místností se sčítají a po úspěšném dokončení celé hry je hráč zapsán do tabulky nejlepších hráčů v případě, že jeho výsledný počet bodů je mezi deseti nejlepšími.

Rozehranou hru je možné kdykoliv ukončit, pozastavit, popřípadě uložit do paměti a následně spustit.



Obr. 14. Stručný popis pravidel přímo ve hře

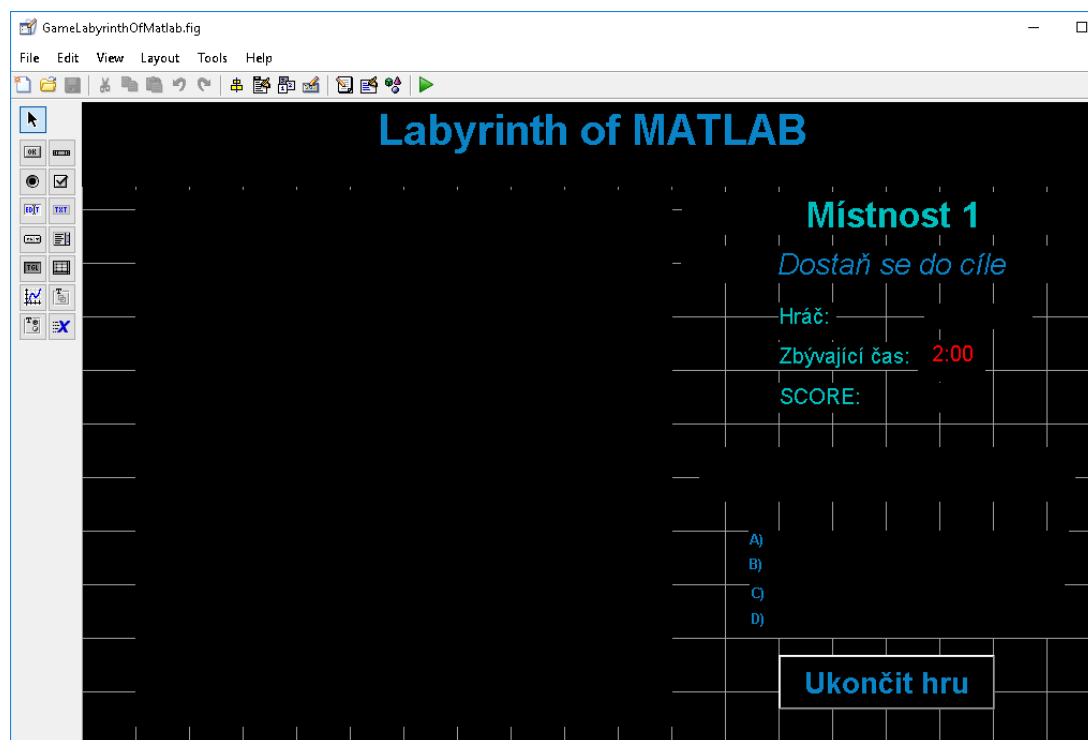
## 4 REALIZACE HRY

Kapitola je věnována popisu hry z programátorské perspektivy.

### 4.1 Popis tvorby hry

Nejprve jsem se seznámila s hrami na MATLAB Central a studentskými pracemi, které jsou již popsány v teoretické části práce.

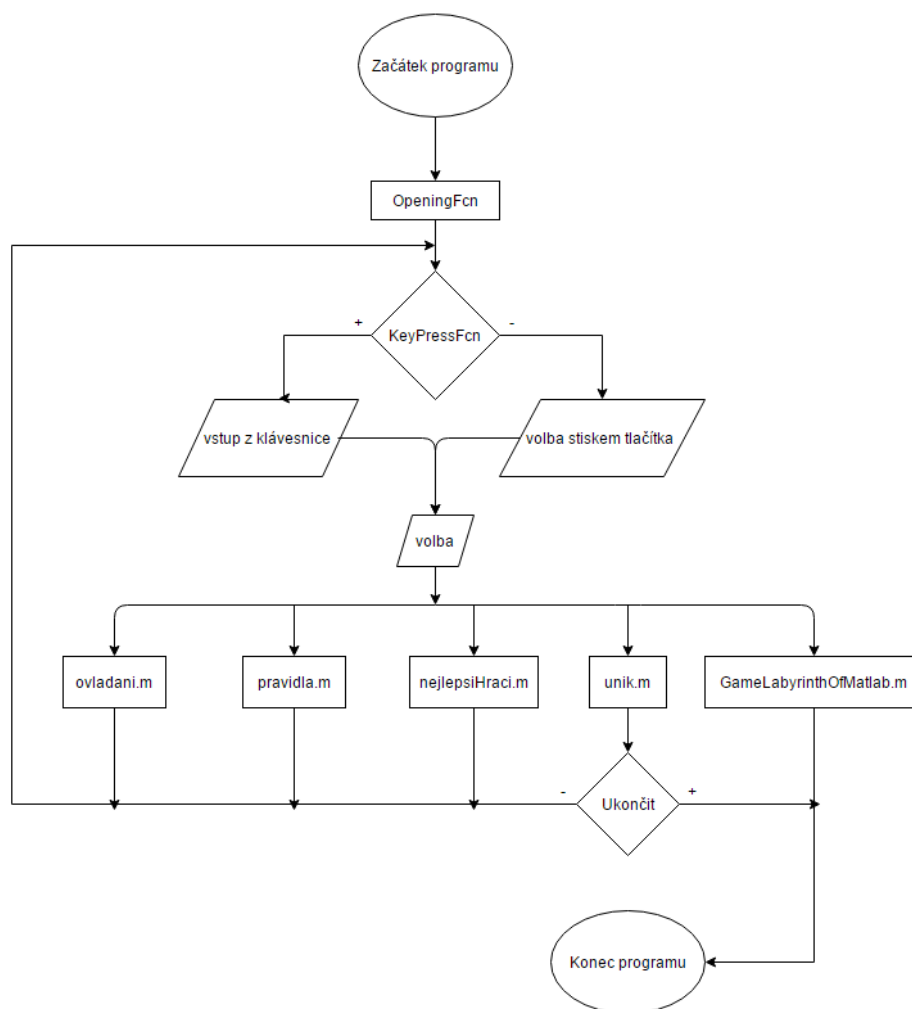
Následně jsem navrhla strukturu hry. Hra byla vyvíjena v programu MALTAB verze R2013a. Vytvořila jsem dva GUI soubory, s kterými jsou spjatý dva hlavní M-soubory, a dále jsem program doplnila o dalších osm M-souborů z důvodu zpřehlednění programu, tj. aby se zdrojový kód neopakoval, tak jsem vytvořila nové funkce v samostatných zdrojových souborech.



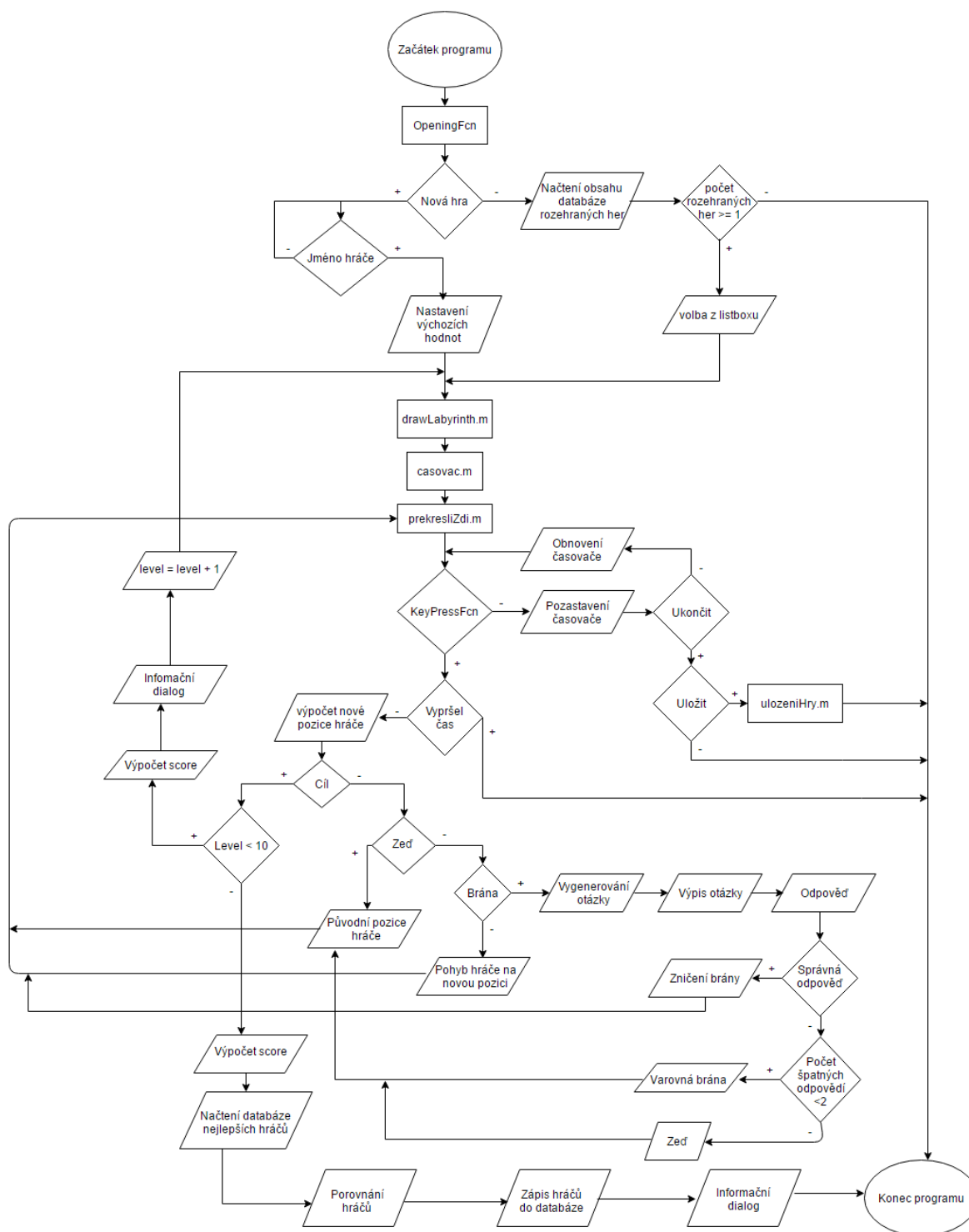
Obr. 15. Grafické okno hry při vytváření pomocí GUIDE

Hlavní M-soubory *LabyrinthOfMatlab.m* a *GameLabyrinthOfMatlab.m* jsou propojeny tak, že při skončení jednoho se spustí druhý, a naopak (vyjma přímého ukončení celého programu v *LabyrinthOfMatlab.m*)

Struktura hlavních M-souborů je popsána následujícími vývojovými diagramy.



Obr. 16. Vývojový diagram funkce *LabyrinthOfMatlab.m*



Obr. 17. Vývojový diagram funkce GameLabyrinthOfMatlab.m

Program také obsahuje tři databáze ve formě CSV souborů, a to databázi otázek a odpovědí, rozehraných her a hráčů s nejlepšími výsledky.

### 4.1.1 Databáze otázek a odpovědí

Databáze otázek a odpovědí vznikla na základě mých vědomostí, znalostí získaných během vývoje této hry a také pomocí doporučené literatury. V tomto CSV souboru je každá položka zapsána jako: otázka, čtyři možnosti odpovědí a číslo, které udává pozici správné odpovědi.

1	Jaký příkaz slouží v MATLABu pro výpis proměnné?	disp()	display()	show()	write()	1
2	Jakým operátorem změníme v MATLABu řádkový vektor [1 2 3] na sloupcový?	'[1 2 3]'	[1 2 3]'	'[1 2 3]'	[1 2 3]'	2
3	Jakou funkci MATLABu zjistíme počet prvků ve vektoru?	sizeof()	vektorSize()	length()	count()	3
4	Jak vytvoříme v MATLABu vektor "v" s ekvidistantně vzdálenými prvky od 1 do 10?	v = 1 : 10	v = 1 : 10	v = 1 : 10	v = 1 : 10	4
5	Jak vytvoříme v MATLABu vektor "v" s ekvidistantně vzdálenými prvky od 1 do 10 s krokem 2?	v = 10:2:1	v = 1 : 2 : 10	v = 1-10:2	v = 1 ~ 10:2	1
6	Jak v MATLABu vybereme z matice M prvek na 5. řádku a 2. sloupci?	M(2,5)	M(5,2)	M(5:2)	M(2:5)	2
7	Jak se v MATLABu nazývá proměnná, která označuje poslední výsledek?	this	last	ans	result	3
8	Jaký příkaz slouží v MATLABu pro zjištění velikosti matice?	lengthof()	sizeof()	count()	size()	4
9	Jak v MATLABu vybereme z matice M prvek na 3. řádku a 2. sloupci?	M(3,2)	M(2,3)	M(2:3)	M(3:2)	1
10	Jak v MATLABu vybereme celý třetí řádek z matice M?	M(:,3)	M(3,:)	M(3:)	M(3,)	2
11	Jak v MATLABu vybereme celý třetí sloupec z matice M?	M(3,:)	M(5,:)	M(:,3)	M(:3)	3
12	Jak v MATLABu přetřansformujeme matici M na sloupcový vektor?	M(/)	M(+)	M(-)	M(')	4
13	Jak v MATLABu smažeme třetí řádek matice M?	M(3,:) = []	M(3,:) = 0	M(:,3) = []	M(:,3) = 0	1
14	Jak vytvoříme v MATLABu matici M o rozměru 4x3 plnou nul?	M = empty(4,3)	M = zeros(4,3)	M = zero(4,3)	M = nonumber(4,3)	2
15	Jak vytvoříme v MATLABu matici M o rozměru 4x3 plnou jedniček?	M = one(4,3)	M = first(4,3)	M = ones(4,3)	M = numberone(4,3)	3
16	Jak vytvoříme v MATLABu diagonální matici M o rozměru 3x3?	M = ear(3,3)	M = mouth(3,3)	M = nose(3,3)	M = eye(3,3)	4
17	Jak vytvoříme v MATLABu matici M o rozměru 3x3 s náhodnými hodnotami s rovnoměrným rozdělením?	M = rand(3,3)	M = random(3,3)	M = randomnumbers(3,3)	linearRandom()	1
18	Jaký příkaz slouží v MATLABu k vytvoření inverze matice?	inverse()	inv()	invert()	matrixinv()	2
19	Jaký operátor se používá v MATLABu k vyjádření nerovnosti?	!=	!=	*=	!=	3
20	Jaký příkaz v MATLABu slouží pro nalezení indexů všech nenulových prvků vektoru nebo matice?	nonull()	noempty()	nozeros()	find()	4

Obr. 18. Ukázka databáze otázek a odpovědí

Před spuštěním samotné hry se nahrají otázky z CSV souboru do matice a hned proběhne jejich promíchání pomocí funkce MATLABu, ukázkou je následující kód:

```

1 % načtení otázek a odpovědí z csv souboru
2     file = fopen('OtazkyAOdpovedi.csv', 'r');
3     C = textscan(file, repmat('%s',1,6), 'delimiter',';', 'CollectOutput',true);
4     C = C{1};
5     fclose(file);
6
7     % náhodné promíchání otázek
8     C = C(randsample(1:length(C),length(C)),:);

```

Před začátkem každého levelu se z této matice vybere dvacet otázek. Odpovědi na každou otázku se navzájem promíchají, opět pomocí funkce MATLABu. Index, označující správnou odpověď musí být tudíž taky změněn, protože správná odpověď promícháním získá novou pozici. Následující algoritmus ukazuje, jak funguje promíchání jednotlivých odpovědí k otázce a nalezení nového indexu správné odpovědi:

```

1 % promíchání odpovědí v otázkách
2     H = Odpovedi;
3     Odpovedi = Odpovedi(:,randperm(size(Odpovedi,2)));
4     sum = size(Odpovedi);
5     sum = sum(2);
6

```

```

7         % nalezení pozice nové správné odpovědi
8         for j=1:length(SpravneOdpovedi)
9             h = H(1,SpravneOdpovedi(j));
10            for i=1:sum
11                if strcmp(h, Odpovedi(1,i))
12                    SpravneOdpovedi(j)=i;
13                end
14            end
15        end

```

Následně se použitých dvacet otázek vymaže z matice nesoucí všechny otázky, aby bylo zabráněno opakujícím se otázkám v dalších místnostech.

#### 4.1.2 Databáze nejlepších hráčů

Databáze hráčů obsahuje maximálně deset údajů. V souboru je zapsáno jméno hráče a následně jeho bodový výsledek ze hry.

	A	B	C
1	Monika	1792	
2	Tester	1556	
3	Jan	1506	
4	Lenka	1499	
5	Tester	1467	
6	Petr	1411	
7	Tester	1394	
8			
9			
10			

*Obr. 19. Ukázka databáze nejlepších hráčů*

Pokud hráč projde všemi deseti místnostmi, nahraje se soubor do matice. Do matice je přidán hráč, který hru momentálně vyhrál. Dále se porovnává bodový zisk jednotlivých hráčů, a podle něj jsou hráči sestupně seřazeni do matice a následně je tato matice opět zapsána do CSV souboru. Do souboru je vždy zapsáno maximálně deset údajů, protože při zobrazení nejlepších hráčů zobrazujeme jen deset nejlepších, a tak by další údaje v souboru byly bezvýznamné. Ukázka algoritmu pro seřazení hráčů:

```

1 % seřazení hráčů podle jejich score
2 nejlepsiHraci = sortrows(nejlepsiHraci, -2);

```



#### 4.1.4.1 Otázky

Většina testerů byla nespokojena kvůli obtížnosti otázek, respektive díky povinnosti odpovídat dvakrát na stejnou otázku, kterou nevěděli a případně si tak zablokovali cestu k cíli. Možným řešením tohoto problému by bylo dvojnásobně zvětšit databázi otázek. Databáze otázek a odpovědí by tedy obsahovala 400 otázek, přičemž na každou místnost by připadlo 40 otázek a každá brána by neobsahovala jednu, ale dvě otázky. Hráč by tak neměl možnost opravit odpověď na otázku, ale musel by odpovídat na úplně jinou otázku.

#### 4.1.4.2 Čas

Více než polovina testerů byla názoru, že časový limit na průchod místností je dostačující. Zbytek by si přálo, aby časový limit byl vyšší než dvě minuty.

Napadli mně tři řešení problému. První řešení by bylo takové, že hráč by měl v první místnosti stejný časový limit jako dosud, tedy dvě minuty, a po průchodu místností by se zbývajícím čas nepřičetl hned do bodového hodnocení, ale přenesl by se do další místnosti. Až po průchodu všemi deseti místnostmi by se čas přepočítal na body.

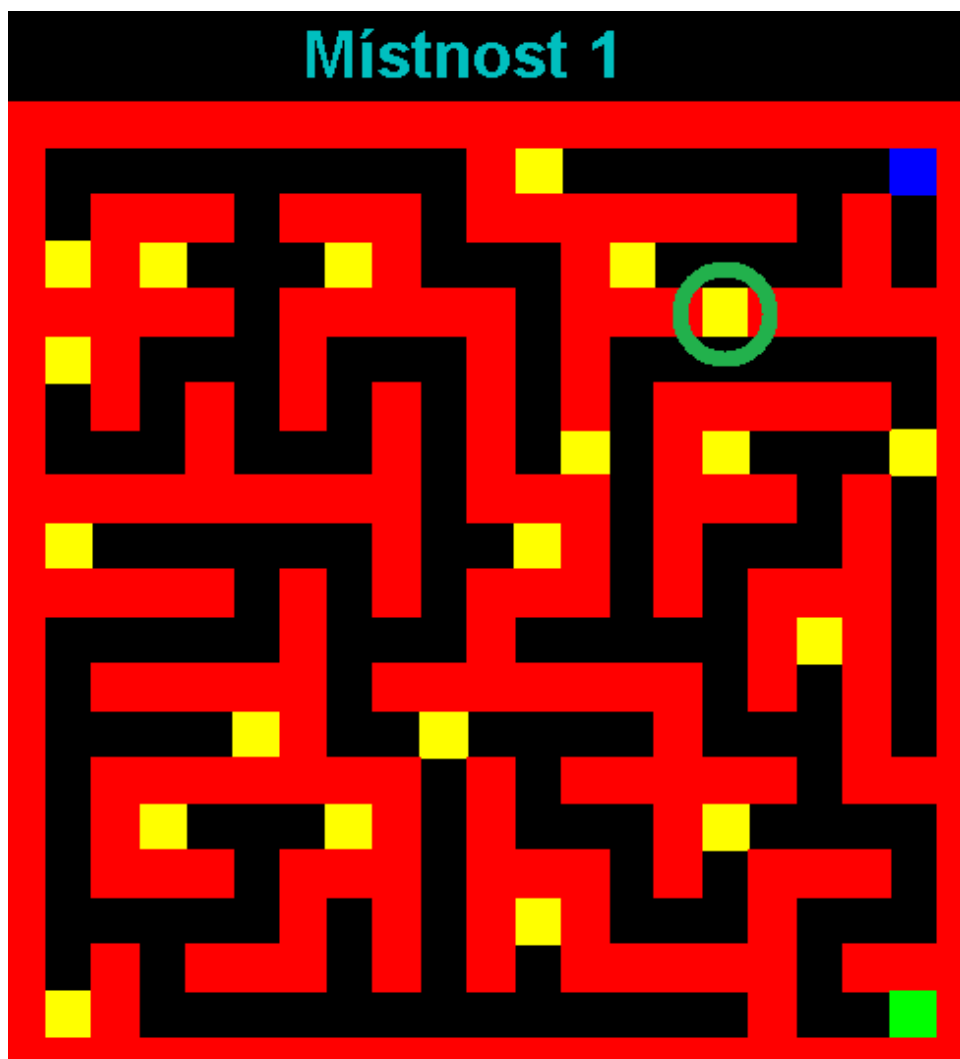
Druhým řešením téhož problému by mohlo být, že by si sám hráč před začátkem hry mohl stanovit časový limit, například dvouminutový, třiminutový nebo čtyřminutový. Vzhledem k nastavení hráče by se tak volila obtížnost hry. V tomto případě, by bylo vhodné v tabulce nejlepších hráčů uvést i obtížnost hry, ve které hráč docílil získaný počet bodů.

Další možností by bylo nastavit ubíhání času jen při odpovídání na otázky. Hráč by tedy nebyl omezen časově v průchodu labyrintem, ale jen při odpovídání.

#### 4.1.4.3 Zablkovaná cesta

Hra je naprogramována tak, aby uživatel hledal cesty k cíli v labyrintu, tudíž aby při zablkování všech cest k cíli byl sám schopen vyhodnotit, zda ve hře pokračovat či ne.

Tento problém byl také častou připomínkou, i když tímto způsobem byla hra vytvořena cíleně. Ovšem i tento problém by se dal jednoduše vyřešit. Řešením by byla funkce, která pro každou místnost uchová důležité pozice bran k cíli, přičemž by průběžně kontrolovala, zda tyto brány jsou zablkovány či ne. Na následujícím obrázku je znázorněno řešení.



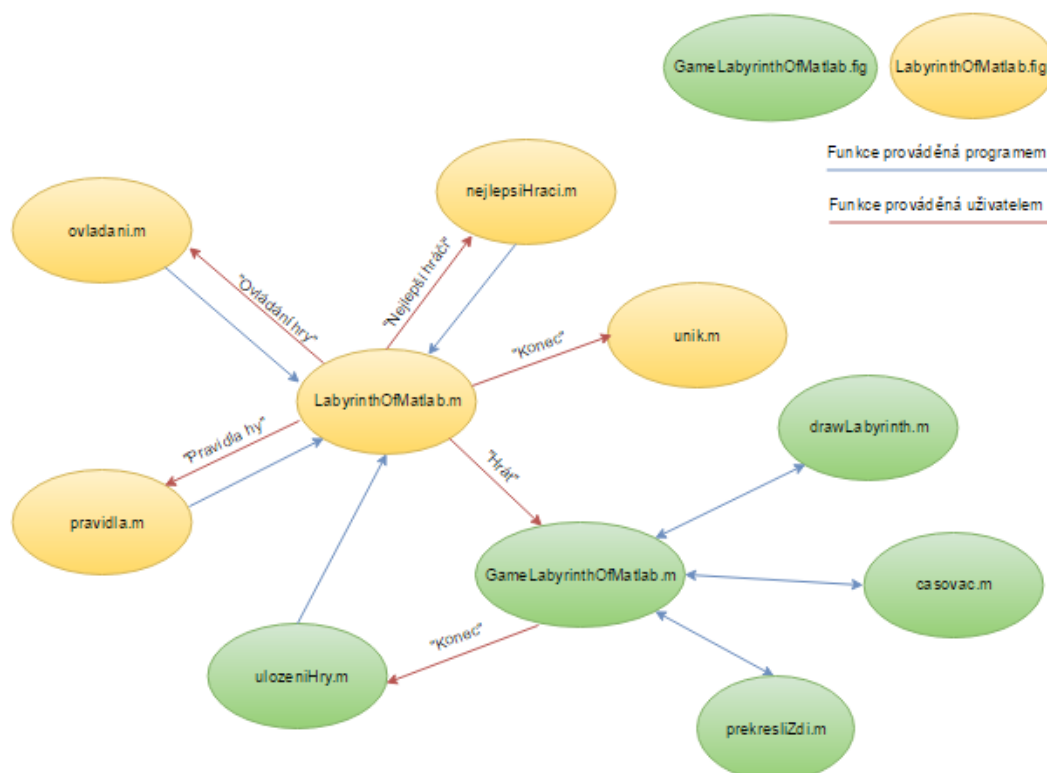
Obr. 21. Znáznornění řešení problému s vyhodnocením zablokování

Na předchozím obrázku lze vidět první místnost, ve které je zvýrazněna důležitá brána. Nová funkce by tedy hlídala tuto bránu podle její pozice a průběžně kontrolovala její průchodnost či zablokování. Pokud by bylo zaregistrováno zablokování, hra by hráče upozornila, že už nemá žádnou možnost se dostat k cíli.

V dalších obtížnějších místnostech už by nezáleželo jen na jedné bráně, jako v první místnosti, ale na více, což by také nebylo problémem kontrolovat, ale algoritmus pro ostatní místnosti by už byl složitější.

## 4.2 Použité funkce

Jak už bylo zmíněno, program obsahuje celkem dva soubory s příponou FIG a deset M-souborů. Vztahy mezi jednotlivými soubory jsou znázorněny na následujícím blokovém schématu:



Obr. 22. Blokové schéma programu popisující vztahy mezi jednotlivými soubory

Mezi hlavní M-soubory patří *LabyrinthOfMatlab.m* a *GameLabyrinthOfMatlab.m*, které obsluhují dialogová okna a proto existují soubory se stejným názvem ale jinou příponou, tj. příponou FIG, které slouží pro zobrazení grafického uživatelského rozhraní.

Červené vazby ve schématu znázorňují možnosti hráče, modré vazby popisují funkce automaticky prováděné programem. Žluté M-soubory pracují v grafickém okně *LabyrinthOfMatlab.fig*, zelené M-soubory v *GameLabyrinthOfMatlab.fig*.

V následujících podkapitolách jsou popsány jednotlivé M-soubory programu z programátorské perspektivy.

### 4.3 LabyrinthOfMatlab.m

Tab. 2. *LabyrinthOfMatlab.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
LabyrinthOfMatlab_OpeningFcn	48 - 116	Nastavení viditelnosti objektů + nahrání a zobrazení obrázku ze souboru
tlacitkoNovaHra_Callback	131 - 138	Reakce na stisk tlačítka Hráti hru + volání funkce <i>GameLabyrinthOfMatlab.m</i>
tlacitkoTabulkaHracu_Callback	142 - 146	Reakce na stisk tlačítka Nejlepší hráči + volání funkce <i>nejlepsi-Hraci.m</i>
tlacitkoPravidlaHry_Callback	150 - 154	Reakce na stisk tlačítka Pravidla hry + volání funkce <i>pravidla.m</i>
tlacitkoUnikove_Callback	158 - 171	Reakce na stisk tlačítka Konec hry + volání funkce unik, popřípadě zobrazení původního hlavního okna
tlacitkoOvladaniHry_Callback	175 - 179	Reakce na stisk tlačítka Ovládání hry + volání funkce <i>ovladani.m</i>
figure1_CloseRequestFcn	183 - 202	Reakce na zavření okna křížkem v pravém horním rohu + volání funkce <i>unik.m</i> , případně zobrazení hlavního okna
figure1_KeyPressFcn	206 - 242	Reakce na stisk klávesy + volání další funkce dané konkrétní klávesou

Tento M-soubor slouží jako hlavní funkce programu, pomocí které se celá aplikace spouští. Hlavní funkcí je *LabyrinthOfMatlab\_OpeningFcn*, jejímž úkolem je nastavit výchozí hodnoty pro grafické uživatelské rozhraní, což znamená načtení a vyobrazení úvodního obrázku a také nastavení viditelnosti objektů, které se v okně mají zobrazit či ne.

Další důležitou funkcí je *figure1\_KeyPressFcn*, která vyhodnocuje akce způsobené stiskem tlačítka klávesnice. Funkce obsahuje jednoduchý *switch* s možnostmi *h* pro hraní hry, *n* pro zobrazení nejlepších hráčů, *p* pro zobrazení pravidel, *o* pro vyobrazení ovládání hry, *k* pro ukončení aplikace a *z* pro návrat do hlavního okna.

Pro stejný účel slouží i jednotlivé funkce zpětného volání každého tlačítka, vstupem zde není stisk klávesy ale kliknutí na příslušné tlačítko. V závislosti na stisknuté klávese nebo výběru myši se spustí další M-soubor.

Kromě tlačítka které spouští hru a otvírá nové grafické okno, všechny ostatní funkce volané z M-souborů po kliknutí na příslušné tlačítko neotevírají nové grafické okno, ale mění pouze viditelnost objektů ve stejném grafickém okně, a proto je zde zavedena důležitá globální binární proměnná *konecna*, kvůli dvojitému použití jednoho tlačítka Konec hry a Zpět. Pokud je spuštěno hlavní okno, hodnota globální proměnné je 1. Pokud se ale nacházíme v jiném M-souboru, odsud spuštěného, tak je hodnota této proměnné nastavena na 0.

Tato proměnná pomáhá řešit obsluhu grafických objektů v jednom dialogovém okně více funkcemi tak, že pomocí hodnoty globální proměnné program pozná, zda má v danou chvíli zaznamenat pokus o ukončení aplikace nebo pokus o návrat do hlavního okna.

#### 4.3.1 pravidla.m

Tab. 3. *pravidla.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
pravidla	1 - 27	Nastavení viditelnosti grafických objektů + výpis pravidel hry

Většina souborů hry pracuje s jedním grafickým oknem a tento M-soubor se používá k úpravě viditelnosti objektů v grafickém okně při vypisování pravidel. Objektům předchozího zobrazení nastaví viditelnost na *off* a objektům nového zobrazení pak na *on*.

Původně tato funkce byla součástí *LabyrinthOfMatlab.m*, ale kvůli dvojitému použití (ovládání přes myš nebo stisk klávesy) a tedy opakujícímu se kódu jsem využila možnost externí funkce.

#### 4.3.2 ovladani.m

Tab. 4. *ovladani.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
ovladani	1 - 35	Nastavení viditelnosti grafických objektů + načtení a vyobrazení obrázku ze souboru + výpis ovládání hry

Tento M-soubor má podobný úkol jako předchozí funkce. Byl vytvořen kvůli opakujícímu se kódu v první verzi hry. Soubor kromě toho že upravuje viditelnost objektů v grafickém okně, pak také nahrává a zobrazí obrázek s vysvětlivkami:

```

1 % načtení obrázku ze souboru a jeho vyobrazení
2 axes(handles.osa2);
3 img2 = imread('vysvetlivky.jpg');
4 image(img2);
5 axis off;
6 axis image;
```

#### 4.3.3 nejlepsiHraci.m

Tab. 5. *nejlepsiHraci.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
nejlepsiHraci	1 - 95	Nastavení viditelnosti grafických objektů + načtení a zobrazení obrázku ze souboru + čtení ze souboru hraci.csv + výpis jednotlivých hráčů

M-soubor *nejlepsiHraci.m* definuje viditelnost jednotlivých objektů v grafickém okně. Dále je pomocí této funkce načten do paměti a zobrazen obrázek s deseti stupni pro vítěze, u kterých se vyobrazuje umístění hráčů. Soubor rovněž obsluhuje načtení a zobrazení

jmen a výsledků nejlepších hráčů, data se načtou ze souboru *hraci.csv*, ve kterém jsou již hráči seřazeni sestupně a jejich maximální počet je deset:

```
1 % seznam nejlepších hráčů ze souboru
2 file = fopen('hraci.csv','r');
3 D = textscan(file, repmat('%s %s',1), 'delimiter',';', 'CollectOutput',true);
4 D = D{1};
5 fclose(file);
```

Tyto údaje jsou zaznamenány zápisem do CSV souboru při úspěšném dokončení hry.

#### 4.3.4 unik.m

Tab. 6. *unik.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
unik	1 - 11	Dotazovací dialog na ukončení hry + vyhodnocení odpovědi dialogu

Tento M-soubor je spuštěn v případě, kdy se hráč snaží ukončit aplikaci, ať už kliknutím na dané tlačítko s popisem Konec hry nebo pomocí křížku v pravém horním rohu grafického okna. Funkce je spuštěna, pouze pokud je zaznamenán pokus o ukončení aplikace, tedy v situaci kdy hodnota globální proměnné *konecna* je 1. Účel použití proměnné byl vysvětlen výše.

Součástí zdrojového kódu funkce je dotazovací dialog, který chce od uživatele potvrdit pokus o ukončení aplikace:

```
1 % Question Dialog
2 konec = questdlg('Opravdu chcete ukončit hru?', ...
3     'Konec hry', ...
4     'Ano', 'Ne', 'Ne');
5
6 if strcmp(konec, 'Ano')
7     delete(handles.figure1);
8 end
```

Pokud je zvolena volba *Ano*, aplikace se ukončí. V opačném případě, tedy při volbě *Ne* aplikace zůstává spuštěna a dialog se ukončí. Jako výchozí volba dialogu je nastavena

možnost *Ne*, pro případ kdy uživatel nevybere možnost řádně, ale ukončí dialog například kliknutím na křížek v jeho pravém horním rohu.

#### 4.4 GameLabyrinthOfMatlab.m

Tab. 7. *GameLabyrinthOfMatlab.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
GameLabyrinthOfMatlab_OpeningFcn	48 - 270	Nastavení viditelnosti grafických objektů + dotazovací dialog na druh hry + dotazovací dialog na jméno hráče + čtení ze souboru rozehraných her + seznam rozehraných her + nastavení výchozích hodnot proměnných + volání funkce <i>drawLabyrinth.m</i>
tlacitkoUkonci_Callback	284 - 307	Pozastavení časovače + dotazovací dialog na ukončení hry a jeho vyhodnocení
Figure2_KeyPressFcn	310 - 715	Vyhodnocení posunu hráče + výpis otázek a zpracování odpovědí

Tento M-soubor řídí veškeré dění samotné hry.

#### 4.4.1 OpeningFcn

První důležitou funkcí M-souboru je *GameLabyrinthOfMatlab\_OpeningFcn*, kde jsou nejen definovány všechny globální proměnné, ale také dotazovací dialog, který se zobrazí ještě před spuštěním hry samotné. Tázací dialog se uživatele ptá, zda chce hrát novou či rozehranou hru. Podle volby jsou dále nastaveny hodnoty proměnných, přičemž při volbě hrát rozehranou hru jsou tyto hodnoty vyňaty z databáze, respektive souboru *rozehrane-Hry.csv*. Následně se zobrazí dialog, který vyžaduje zadání hráčova jména. Bez vyplnění jména se hra nespustí a dialog je zobrazován stále dokola.

Při nestandardním ukončení jednoho ze zmíněných dvou dialogů se opět spustí hlavní okno aplikace, v opačném případě je zavolána funkce *drawLabyrinth*.

#### 4.4.2 drawLabyrinth.m

Tab. 8. *drawLabyrinth.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
drawLabyrinth	1 - 262	Načtení otázek a odpovědí ze souboru + promíchání otázek a odpovědí + nastavení výchozích hodnot pro grafické objekty + vykreslení grafických objektů + volání funkce <i>casovac.m</i> + volání funkce <i>prekresliZdi.m</i>

Vstupními parametry této funkce jsou data ze struktury *handles*, konkrétně aktivní okno, číslo levelu – místnosti, skóre hráče a matice. Tento M-soubor zajišťuje vykreslení všech důležitých prvků hry a práci s daty ve hře. Je zde důležitá proměnná *hra*, která označuje, zda se jedná o novou hru (hodnota proměnné *hra* je rovna 1), nebo zda se jedná o rozehranou hru (hodnota proměnné *hra* je 2). Na základě této proměnné se postupuje individuálně při vykreslování jednotlivých prvků a nastavení ostatních hodnot důležitých pro hru.

Pokud se jedná o novou hru, nejprve se nahrají do matice *C* otázky a odpovědi ze souboru *OtazkyAOdpovedi.csv*, které se následně zamíchají:

```
1      % náhodné promíchání otázek
2      C = C(randsample(1:length(C),length(C)),:);
```

Dále se podle čísla levelu určí rozměr osy pro hru, základní pozice hráče a cíle, pozice jednotlivých bran (otázek) a pozice jednotlivých zdí, protože tyto hodnoty jsou specifické pro každou místnost. Dále je pomocí funkce MATLABu *setappdata* nastavena kompatibilita s pozicemi otázek ve hře. Následně se provede propojení otázek s bránami, respektive ke každé bráně je přiřazena konkrétní otázka, se kterou samozřejmě i její možnosti odpovědi a správná odpověď. Jednotlivé odpovědi u každé otázky jsou poté promíchány, s čímž souvisí i nalezení nové pozice správné odpovědi. Následně je použitých 20 otázek smazáno z matice *C*. V kódu dále následuje nastavení viditelnosti textů u otázek a vykreslení bran, hráče, cíle a zdí. Veškeré grafické prvky ve hře jako jsou brány, hráč, cíl a zdi jsou vykresleny pomocí funkce MATLABu *rectangle*, která vykreslí obdélník na dané pozici, o daných rozměrech a barvách. Pro vykreslení zdí je zde volán M-soubor *prekresliZdi.m*. Také se zde zobrazují hodnoty proměnných do grafického uživatelského rozhraní jako je čas, číslo místnosti, jméno a skóre hráče. V poslední řadě je zde realizováno časování pomocí objektu *timer*, který se odvolává na M-soubor *casovac.m*.

V případě že je zvolena volba hrát rozehranou hru, funkce vykonává trochu jiné úkony než při nové hře. Rozdíl je ve vykreslování bran, kde program pracuje s vektorem špatných odpovědí, podle kterého vykreslí buď žlutou, nebo oranžovou (varovnou) bránu. Ostatní grafické prvky, jako je pozice hráče, cíle a zdí jsou vyňaty z matice, do které byly dříve nahrány z databáze rozehraných her. Tato matice je vstupním parametrem. V této matici jsou také údaje o jméně a skóre hráče, číslu místnosti, zbývajícimu času a vůbec všechny údaje, které jsou potřebné pro hru. Matice obsahuje dokonce i všechny otázky a odpovědi tak, jak byly původně promíchány při prvním spuštění hry ještě před uložením.

#### 4.4.3 prekresliZdi.m

Tab. 9. *prekresliZdi.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
prekresliZdi	1 - 12	Vykreslení grafických objektů

Tento M-soubor pomocí globální proměnné *zdi* nakreslí na příslušných pozicích daných vektorem *zdi* obdélníky, znázorňující neprůchodné zdi:

```
1 % vykreslení zdi
2 help = size(Zdi);
```

```

3 help = help(1);
4 for i=1:help
5     rectangle('Position',Zdi(i,:), 'FaceColor','r','EdgeColor','r');
6 end

```

Kód pro překreslení zdí jsem zvolila jako externí funkci z důvodu volání na ni z různých míst, čímž jsem zabránila opakujícímu se kódu.

#### 4.4.4 casovac.m

Tab. 10. *casovac.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
casovac	1 - 30	Výpočet času

M-soubor slouží, jak už název napovídá, pro řešení časování. Pomocí objektu *timer* ve funkci *drawLabyrinth* voláme tuto funkci každou sekundu. Tato funkce řeší odpočítávání času, tudíž nejprve je zjištěna hodnota aktuálního času (minuty, desítky sekund a sekundy), pak je od těchto hodnot odečtena jedna sekunda, podle základních pravidel počítání času:

```

1 % výpočet času
2 if sec == 0
3     sec = 9;
4     if dSec == 0
5         dSec = 5;
6         min = min - 1;
7     else
8         dSec = dSec - 1;
9     end
10 else
11     sec = sec - 1;
12 end

```

Následně je nová hodnota času vyobrazena na příslušném místě v grafickém uživatelském rozhraní.

Je důležité vědět, že objekt *timer* volající tento M-soubor je nutno vždy při ukončení hry zastavit a odstranit.

#### 4.4.5 KeyPressFcn

Nejdůležitější funkce, ve které probíhá celá hra, je *figure2\_KeyPressFcn* v M-souboru *GameLabyrinthOfMatlab.m*. Jedná se o funkci, která reaguje na stisk tlačítka na klávesnici, tudíž na hraní hry, protože hráč se ve hře pohybuje pomocí šipek klávesnice.

Jsou zde definovány všechny globální proměnné, a hodnota proměnné *hra* je nastavena na 1, pokud hráč v dialogu zvolil možnost Otevřít rozehranou hrou. Důležitá data pro novou anebo rozehranou hru jsou definována v M-souboru *drawLabyrinth.m* a hra tudíž v tomto bodě probíhá stejně. Dále je zde definována struktura *handles*, tudíž objekty aktivního okna. Také zde pracujeme s funkcí MATLABu *getappdata*, díky níž program zpracovává aktuální hodnoty vektoru *PoziceOtazek*.

Následně jsou zjištěny hodnoty aktuálního času a provádí se kontrola, zda čas nevypršel. Pokud časový limit vypršel, zobrazí se oznamovací dialog a hra pro hráče končí. Pokud čas nevypršel, je zjišťována stisknutá klávesa z pole struktury *eventdata.Key*. Je-li zmáčknuta šipka, podle směru šipky se dopočítá potenciální budoucí krok hráče, respektive jeho pozice:

```
1 % pohyb pomocí tlačítek
2         switch key
3             case 'leftarrow'
4                 xshift = -1;
5             case 'rightarrow'
6                 xshift = 1;
7             case 'downarrow'
8                 yshift = -1;
9             case 'uparrow'
10                yshift = 1;
11         end
12
13 % výpočet potenciální pozice hráče
14         poziceX = get(hp, 'XData') + xshift;
15         poziceY = get(hp, 'YData') + yshift;
```

V tomto bodě se porovnává vypočtená pozice hráče s pozicí cíle. Pokud je hráč v cíli a nachází se v poslední místnosti, jsou načtena data ze souboru *hraci.csv*, která jsou porovnávána se skórem hráče. V případě, že hráč dosáhl jednoho z deseti nejlepších výsledků, je zařazen do matice s těmito výsledky. Následně je matice zapsána zpět do souboru *hraci.csv*

s nově seřazenými výsledky. Na obrazovku se zobrazí informační dialog s celkovými výsledky ze hry včetně údaje o ne/zapsání do tabulky nejlepších hráčů.

Pokud je hráč v cíli a nenachází se v poslední místnosti, zobrazí se informační dialog s dosavadními výsledky hry. Následně je spuštěn M-soubor *drawLabyrinth* s novými vstupními parametry, a to s vyšším levellem a výchozím nastavením důležitých hodnot pro začátek nového levelu.

Pokud nová poloha hráče není stejná jako poloha cíle, je tato pozice porovnávána s vektorem zdí. V případě, kdy nová poloha hráče je stejná jako poloha některé zdi, je posun hráče zakázán a jeho pozice zůstává původní bez přepočtu. Jednoduše řečeno, hráč nemůže projít zdí.

Jestliže nová pozice hráče není pozicí žádné ze zdí, je tato poloha porovnávána s pozicemi otázek, respektive s branami. Pokud žádná z bran není ve stejné poloze jako nová pozice hráče, hráč se posune na novou pozici. Pokud ale ve stejné poloze brána je, znamená to, že hráč narazil na otázku. V tomto případě se v grafickém okně zobrazí otázka, spolu s možnostmi odpovědi, příslušná k dané bráně. Nyní se pomocí funkce *waitforbuttonpress* čeká pomocí smyčky na stisk klávesy nebo myši. Pokud je stisknuta klávesa, která zároveň odpovídá správné odpovědi na danou otázku, brána zmizí (překreslí se), je aktualizován vektor *PoziceOtazek* a hráč může v příštím kroku tímto místem projít. Pokud je stisknuta klávesa, která neodpovídá správné odpovědi na otázku, nebo je-li stisknuto tlačítko myši, vektor špatných odpovědí se upraví podle počtu špatných odpovědí na danou otázku. Pokud je hodnota vektoru špatných odpovědí na dané pozici odpovídající dané otázce rovna 1, brána se překreslí na varovnou. Pokud je tato hodnota rovna 2, brána se přemění na zeď, tudíž je smazána z vektoru otázek a přidána do vektoru zdí. Díky tomu je následně volána funkce *prekresliZdi.m*, aby nakreslila případně nově vzniklou zeď.

Jelikož je tato funkce vyvolávána stiskem tlačítka, provádí se opakovaně na základě podnětů z klávesnice uživatele.

#### 4.4.6 CloseRequestFcn

Tato funkce je součástí M-souboru *GameLabyrinthOfMatlab.m* a je shodná s funkcí zpětného volání pro ukončovací tlačítko ve hře.

Pokud tedy uživatel stiskne během hry ukončovací tlačítko nebo křížek v pravém horním rohu, zastaví se čas a zobrazí se dotazovací dialog, který chce po uživateli potvrdit ukon-

čení hry. Jestliže hráč zvolí volbu *Ne*, hra pokračuje a čas se znovu spustí. Této možnosti lze využít například jako pauzy ve hře. Pokud hráč zvolí volbu *Ano* a chce tedy hru ukončit, vyskočí další tázací dialog, který se uživatele táže, zda chce nebo nechce uložit rozehranou hru. Pokud uživatel zvolí volbu *Ne*, hra se ukončí, neuloží a spustí se hlavní okno aplikace. Jestliže hráč chce rozehranou hru uložit a zvolí volbu *Ano*, zavolá se M-soubor *ulozeniHry.m* a po jeho vykonání se hra ukončí, uloží a spustí se hlavní okno aplikace.

#### 4.4.7 ulozeniHry.m

Tab. 11. *ulozeniHry.m*

Funkce	Pozice v souboru (čísla řádků)	Účel použití
ulozeniHry	1 - 157	Dotazovací dialog pro uložení hry a vyhodnocení odpovědí na dialog + čtení ze souboru + zápis důležitých dat do matice + zápis matice do souboru

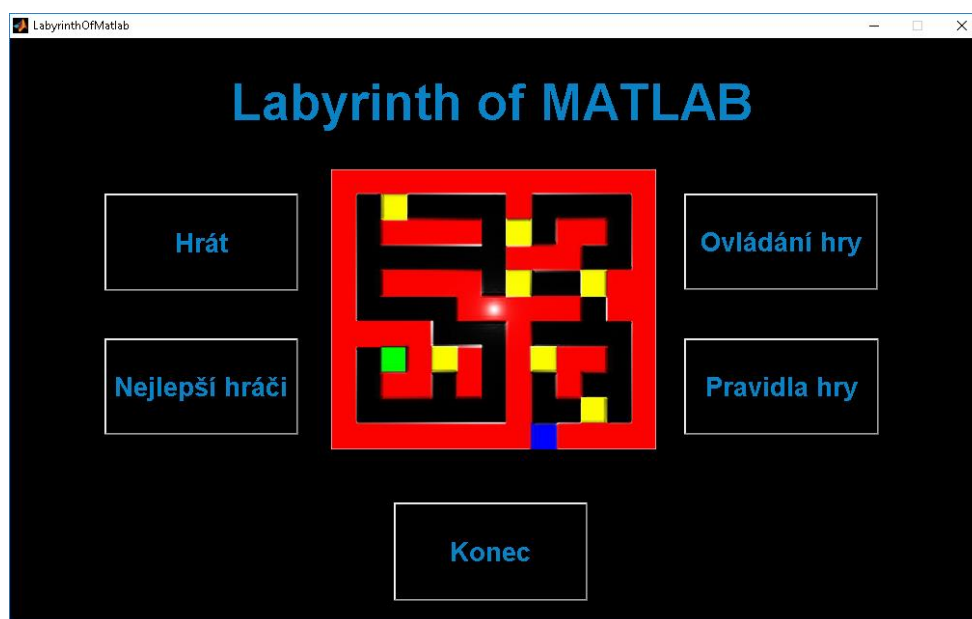
Tento M-soubor slouží pro uložení hry do databáze rozehraných her. Funkce načte všechna data ze souboru *rozehraneHry.csv* do matice, přičemž do ní přidá nový řádek s údaji o aktuálně ukládané hře. Mezi tyto údaje patří jméno a skóre hráče, pozice hráče, číslo místnosti, zbývajíc čas, vektor správných a špatných odpovědí, vektory pozic otázek, zdí, bran, otázek a odpovědí a nakonec také údaje o aktuálním datu a času, pomocí něhož lze následně snadno danou hru dohledat. Nová matice s údaji o hrách se zapíše zpět do souboru *rozehraneHry.csv*.

## 5 HRA Z POHLEDU UŽIVATELE

Uživatel spustí aplikaci v jakékoliv verzi MATLABu pomocí příkazu *LabyrinthOfMatlab*. Po zadání příkazu se zobrazí hlavní okno aplikace uprostřed obrazovky.

### 5.1 Hlavní okno

V hlavním okně si může uživatel vybrat, zda chce hrát hru, přečíst si pravidla hry nebo zjistit jak se hra ovládá, může se také podívat na nejlepší hráče, či hru ukončit.

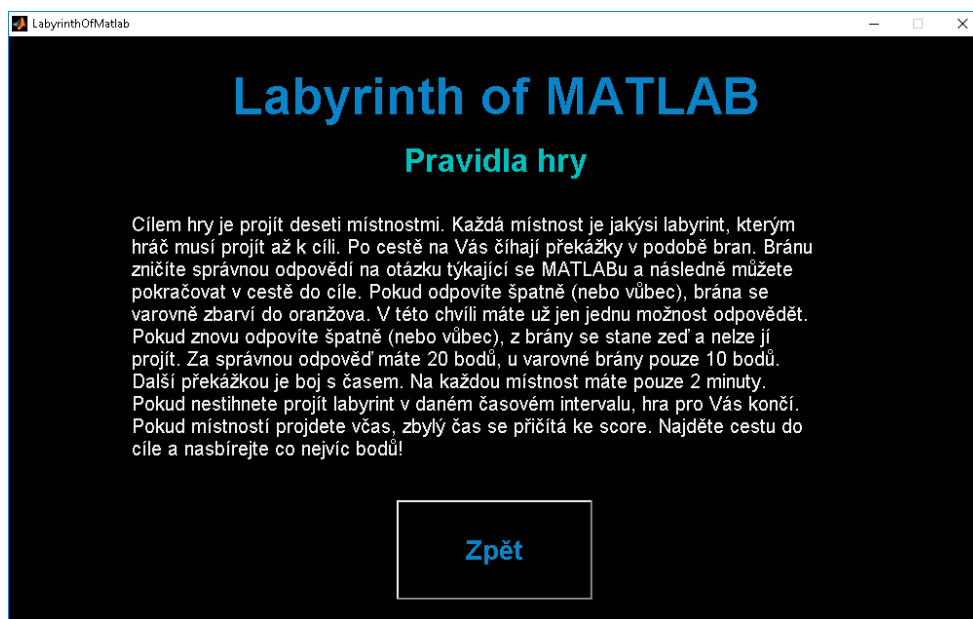


Obr. 23. Hlavní okno aplikace

Uživatel může výběr provést buď pomocí myši, tedy najetím a kliknutím na dané tlačítko, nebo pomocí klávesnice, a to tak, že stiskne takové tlačítko klávesnice, které odpovídá prvnímu písmenu jeho volby, např. pro hru stiskne uživatel klávesu *h*, nebo pro konec hry klávesu *k*.

### 5.2 Pravidla hry

Při výběru *Pravidla hry* v hlavním okně se uživateli překreslí okno a zobrazí se pravidla hry.

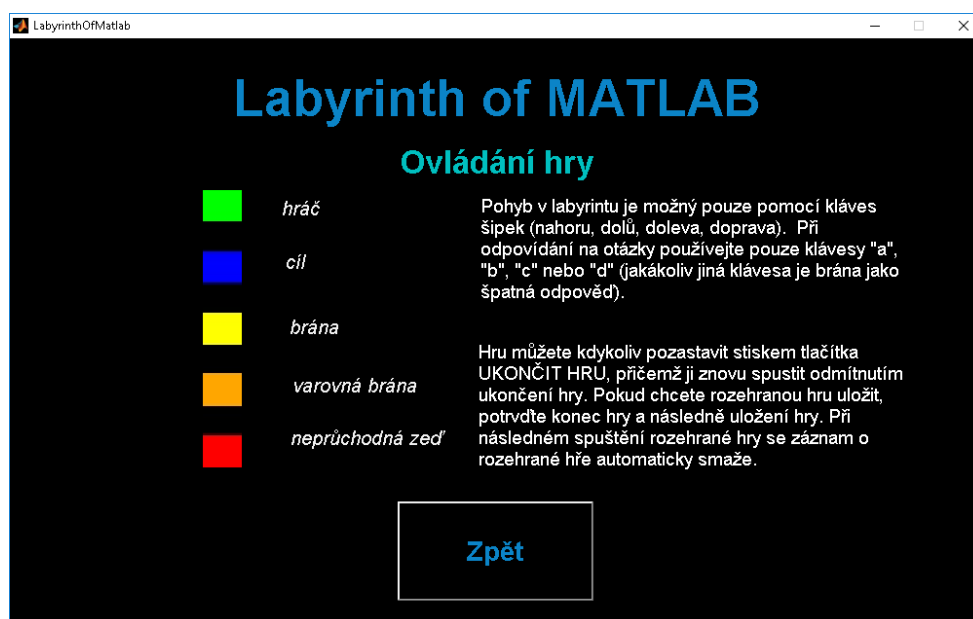


Obr. 24. Pravidla hry

V této situaci má hráč jedinou možnost, a to vrátit se zpět do hlavního okna. To lze opět pomocí myši nebo klávesnice, použitím klávesy *z*. Při kliknutí na křížek v pravém horním rohu uživatel aplikaci neukončí, nýbrž se také vrátí do hlavního okna, stejně jako pomocí ostatních možností zde zmíněných.

### 5.3 Ovládání hry

Při této volbě se uživateli překreslí obsah dialogového okna, s tím rozdílem, že se dozví, jak se hra ovládá.

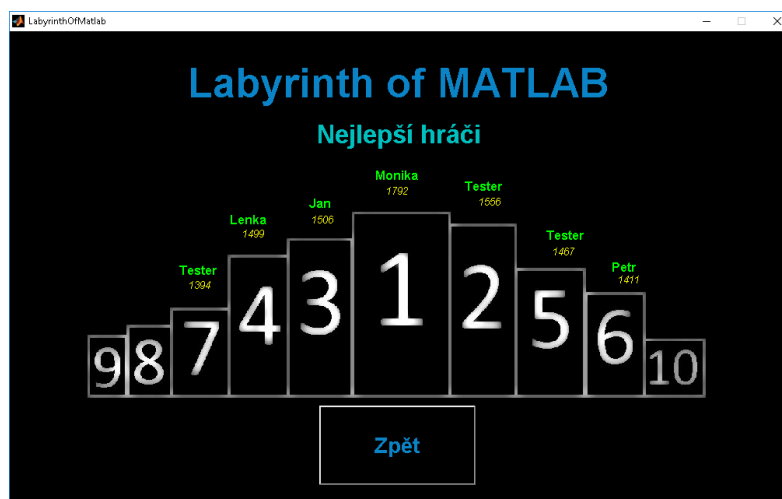


Obr. 25. Ovládání hry

Uživatel se dostane zpět do hlavního okna aplikace obdobně jako v přechozím případě.

## 5.4 Nejlepší hráči

Při výběru možnosti *Nejlepší hráči* v hlavním okně se uživateli překreslí hlavní okno na informační okno s výpisem deseti nejlepších hráčů. Pokud bude hráč úspěšný a zvítězí ve hře s velkým počtem bodů, jeho jméno a skóre se tu objeví.



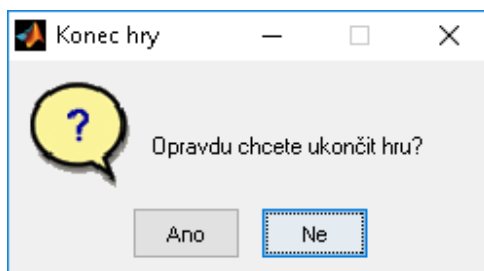
Obr. 26. Výpis nejlepších hráčů

V tomto konkrétním případě, jak můžeme vidět na obrázku, hru dosud hrálo jen sedm hráčů, proto jsou některé stupně vítězů prázdné.

Obdobně jako v přechozích dvou případech volby se uživatel vrátí zpět, do hlavního okna aplikace.

## 5.5 Ukončení hry

Pokud chce hráč ukončit aplikaci, učiní tak buď klávesou *k* nebo kliknutím na tlačítko *Ukončit hru*, či kliknutím na křížek v pravém horním rohu. V této situaci je uživatel tázán dialogem, zda chce opravdu hru ukončit.

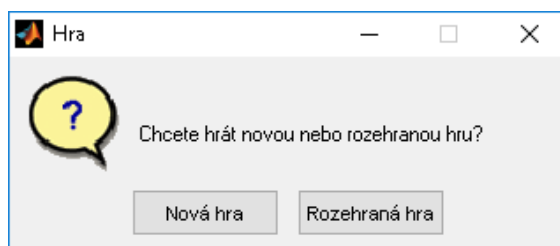


Obr. 27. Tázací dialog při ukončení aplikace

Při zvolení volby *Ne* nebo zavření dialogu křížkem, v pravém horním rohu, aplikace zůstane nadále spuštěna. Pouze při výběru volby *Ano* se celá aplikace řádně ukončí.

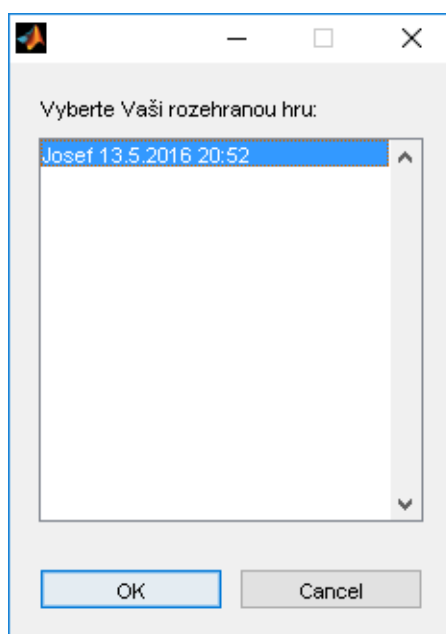
## 5.6 Hra

Při výběru této volby se uživateli, ještě před spuštěním samotné hry, objeví dotazovací dialog pro výběr hry.



Obr. 28. Tázací dialog pro výběr hry

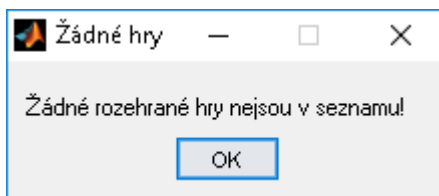
Pokud hráč už hru hrál, nedohrál ji a uložil si ji, může zvolit volbu *Rozehraná hra*, odkud svou rozehranou hru spustí. V této situaci se objeví další dialog pro výběr z rozehraných her.



Obr. 29. Seznam rozehraných her

Zde si uživatel vybere svou rozehranou hru dle svého jména a data a času uložení a po stisknutí potvrzovacího tlačítka se jeho hra spustí.

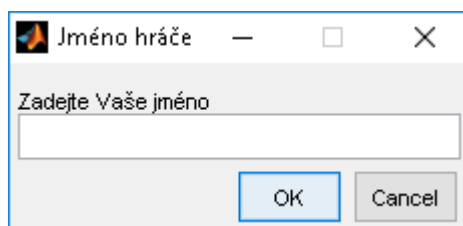
Může se stát, že se uživatel spletl, myslí si, že rozehranou hru uložil, ale není tomu tak. V takovém případě nemusí být žádné rozehrané hry v databázi rozehraných her a uživateli se objeví následující dialog.



Obr. 30: Žádné rozehrané hry

Pokud uživatel nechce hrát rozehranou hru, ať už z důvodu, že žádná v databázi rozehraných her není, nebo z jiného důvodu ukončí některý z předchozích dvou dialogů, zobrazí se mu hlavní okno aplikace.

V této situaci si uživatel nejspíše znovu zvolí volbu *Hrát* v hlavním okně, a vybere možnost *Nová hra* v tázacím dialogu. Nyní je uživatel vyzván k napsání svého jména, respektive přezdívky, pod kterou bude ve hře vystupovat.



Obr. 31: Dotaz na jméno hráče

Tento krok je ošetřen, takže uživatel musí napsat nějaké jméno a nenechat tuto položku prázdnou, jinak dialog bude vyskakovat neustále. Při vyrušení této možnosti se opět spustí okno hlavní aplikace. V případě, kdy hráč napíše své jméno, a potvrdí dialog stiskem tlačítka *OK*, spustí se hra.

Hráč se pohybuje šipkami v labyrintu a snaží se dostat do cíle a po cestě sbírá body odpovídáním na otázky. Na následujícím obrázku lze vidět jak okno, ve kterém hra probíhá, vypadá.

V levé části okna je zobrazen labyrint, ve kterém hra probíhá. V pravé části okna lze vidět číslo místnosti, chcete-li levelu, ve kterém se hráč nachází.

Pod číslem místnosti jsou vyobrazeny instrukce pro hráče. Jelikož je hráč na obrázku v situaci, kdy odpovídá na otázku, je zde popisek *Zodpověz otázku*. Pokud se hráč dostal do cíle, uvidí popisek *Cíl*, v ostatních případech je popisek *Dostaň se do cíle*.

Následně je vyobrazeno jméno hráče, odpočítávající se čas a aktuální skóre hráče v místnosti.

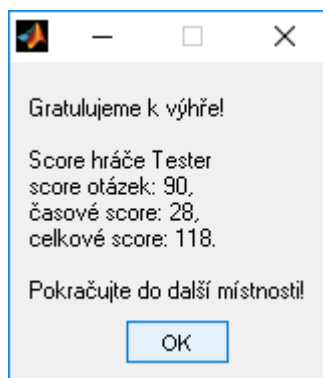
Pod těmito údaji je vyobrazena otázka s možnostmi odpovědí, je-li hráč u brány. Pokud hráč u brány není, je toto místo prázdné.

V poslední řadě je zde tlačítko *Ukončit hru* pro ukončení hry uživatelem.



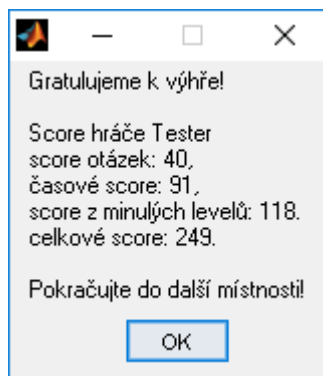
Obr. 32. Hra *Labyrinth of MATLAB*

Ve hře může nastat několik situací. První z nich je, že hráč úspěšně projde místností až do cíle. V tomto případě se objeví informační dialog o hráčově aktuálním skóre po úspěšném průchodu místností.



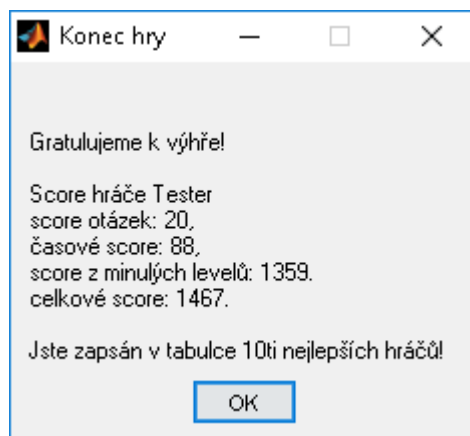
Obr. 33. Informace o stavu hráče po dokončení prvního levelu

Po potvrzení tohoto dialogu se spustí další level – místnost. Body, které hráč nasbíral v jednotlivých místnostech, se průběžně sčítají, a tak má hráč po úspěšném zvládnutí levelu vždy přehled o dosud dosaženém skóre.



*Obr. 34. Informace o průběžném stavu hráče po úspěšném průchodu místnosti*

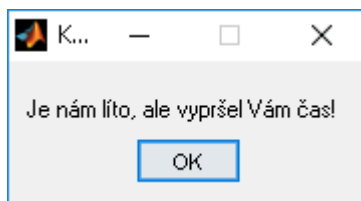
Ve chvíli, kdy hráč projde všemi místnostmi, se mu zobrazí tentýž informační dialog i s informací, zda je nebo není zapsán v tabulce nejlepších hráčů, tedy zda patří mezi deset hráčů s nejlepším skóre.



*Obr. 35. Informace o konečných výsledcích hráče*

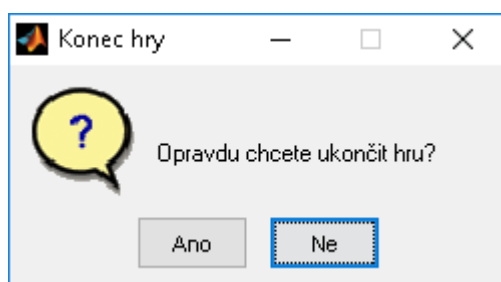
Nyní se opět spustí hlavní okno aplikace po zavření informačního dialogu uživatelem.

Další situace, která může ve hře nastat je, že hráč nestihne místností projít v časovém intervalu, tudíž mu vyprší časový limit. V tomto případě vyskočí hráči na obrazovku informační dialog o vypršení času a hráči tak nezbyvá nic jiného, než hru ukončit.



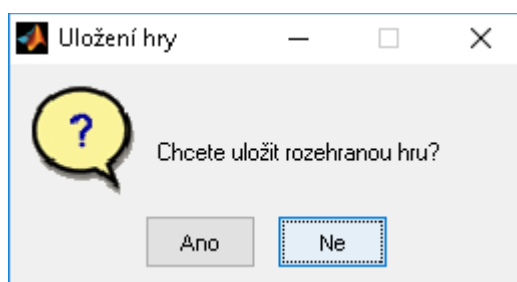
*Obr. 36. Informace o vypršení časového limitu*

Jak už bylo naznačeno, další situací, která může ve hře nastat, je ukončení hry, ať už z důvodu vypršení časového limitu nebo svobodného rozhodnutí hráče. V tomto případě hráč ukončí hru najetím na tlačítko *Ukončit hru*, přičemž se mu zobrazí dotazovací dialog.



*Obr. 37. Tázací dialog při ukončení hry*

Pokud zvolí volbu *Ne* nebo vyruší dialog křížkem v pravém horním rohu, hra pokračuje dále. Lze tedy využít jako pauza ve hře. Pokud ale uživatel zvolí volbu *Ano* pro ukončení hry, zobrazí se další dotazovací dialog, ve kterém má možnost rozehranou hru uložit.



*Obr. 38. Tázací dialog pro uložení hry*

Pokud uživatel zvolí volbu *Ne* nebo zavře dialog křížkem v pravém horním rohu, hra se ukončí bez uložení a zobrazí se hlavní okno aplikace. V opačném případě, při volbě *Ano* se hra uloží do databáze rozehraných her, následně se hra ukončí a spustí se hlavní okno aplikace. Rozehranou hru si pak uživatel může kdykoliv spustit pomocí postupu zmíněného na začátku této podkapitoly *Hra*.

## ZÁVĚR

Cílem práce bylo vytvořit prostředí pro počítačovou hru ve 2D v programu MATLAB založenou na procházení místností a odpovídání na otázky týkající se MATLABu, kdy při správném zodpovězení otázky se odstraní překážka v průchodu místností. Hra dostala název Labyrinth of MATLAB, protože jednotlivé místnosti jsou ve formě labyrintu, kde jsou nastraženy pro hráče překážky ve formě tzv. bran.

Hra je určena pro uživatele, kteří mají základní znalosti programu MATLAB, protože databáze otázek a odpovědí je právě o tomto software. Díky tomu, že otázky jsou formou databáze v samostatných souborech nezávislých na programu, je možné program použít pro ověřování znalostí libovolného předmětu po vytvoření databáze otázek z tohoto předmětu.

Teoretická část práce obsahuje úvod k programu MATLAB. Jsou zde také uvedeny ukázky vytvořených her, které posloužily jako podklad pro realizaci grafického návrhu hry s použitím GUIDE, nástroje pro tvorbu grafického uživatelského prostředí v programu MATLAB.

V praktické části je popsán samotný vývoj hry, od návrhů až po konečný koncept. Hra je zde popsána jak z programátorské, tak uživatelské perspektivy.

Během vypracování programu jsem se setkávala s dílčími úkoly jako například práce s časem, práce s databázemi či řešení ukládání rozehrané hry a její opětovné spuštění. Výsledkem mého úsilí je současná podoba hry, která se líbila všem jejím testerům.

Výsledky této práce spolu s analýzou literárních podkladů uvedenou v teoretické části práce mohou sloužit jako vstupní materiál pro práci s programem MATLAB, protože díky informacím prezentovaným v textu práce a procvičování teoreticky nabytých znalostí pomocí vytvořeného programu si studenti upevní své znalosti zábavnou formou.

## SEZNAM POUŽITÉ LITERATURY

- [1] 2048. *MathWorks* [online]. @bmtran, c1994-2016 [cit. 2016-05-18]. Dostupné z: <https://www.mathworks.com/matlabcentral/fileexchange/46028-2048>
- [2] Sudoku. *MathWorks* [online]. Stepen Sahrn, c1994-2016 [cit. 2016-05-18]. Dostupné z: <https://www.mathworks.com/matlabcentral/fileexchange/34988-sudoku>
- [3] MATRIS v0.10.10.09sb for Windows. *MathWorks* [online]. Mingjing Zhang, c1994-2016 [cit. 2016-05-18]. Dostupné z: <https://www.mathworks.com/matlabcentral/fileexchange/28985-matris-v0-10-10-09sb-for-windows>
- [4] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. MATLAB pro začátečníky. 2. vydání. Praha: BEN - technická literatura, 2005. ISBN 80-7300-175-6.
- [5] DUŠEK, František. MATLAB a Simulink – Úvod do používání. 1. vydání. Pardubice: Univerzita Pardubice, 2001. ISBN: 80-7194-273-1
- [6] PERŮTKA, Ing. Karel. MATLAB - Základy pro studenty automatizace a informačních technologií. 1. Vydání. Zlín: UTB ve Zlíně, 2005. ISBN 80-7318-355-2.
- [7] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. MATLAB - tvorba uživatelských aplikací. 1. vydání. Praha: BEN - technická literatura, 2004. ISBN 80-7300-133-0.
- [8] MATLAB Space Invaders VideoGame. *MathWorks* [online]. Stepen Sahrn, c1994-2016 [cit. 2016-05-18]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/39277-matlab-space-invaders-videogame>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

MATLAB Matrix Laboratory

GUI Graphical User Interface – Grafické uživatelské rozhraní

GUIDE Graphical User Interface Development Enviroment

CD-ROM Compact Disk – Read Only Memory

PPAŘ Programová podpora automatického řízení

CSV Comma-Separated Value – hodnoty oddělené čárkou

**SEZNAM OBRÁZKŮ**

<i>Obr. 1. Pracovní plocha MATLABu .....</i>	<i>12</i>
<i>Obr. 2. Grafický výstup funkce plot .....</i>	<i>20</i>
<i>Obr. 3. Prázdné grafické okno při tvorbě GUI .....</i>	<i>21</i>
<i>Obr. 4. Ukázka Property Inspector dialogu pro tlačítko .....</i>	<i>22</i>
<i>Obr. 5. Ukázka části kódu vygenerovaného MATLABem .....</i>	<i>23</i>
<i>Obr. 6. Hra 2048.....</i>	<i>24</i>
<i>Obr. 7. Hra Sudoku .....</i>	<i>25</i>
<i>Obr. 8. Hra Matris .....</i>	<i>26</i>
<i>Obr. 9. Hra Matlab Space Invaders .....</i>	<i>27</i>
<i>Obr. 10. Hra Člověče, nezlob se .....</i>	<i>28</i>
<i>Obr. 11. Hra Riskuj.....</i>	<i>29</i>
<i>Obr. 12. Hra Pexeso .....</i>	<i>30</i>
<i>Obr. 13. Hra Pozor na kameny .....</i>	<i>31</i>
<i>Obr. 14. Stručný popis pravidel přímo ve hře .....</i>	<i>34</i>
<i>Obr. 15. Grafické okno hry při vytváření pomocí GUIDE .....</i>	<i>35</i>
<i>Obr. 16. Vývojový diagram funkce LabyrinthOfMatlab.m .....</i>	<i>36</i>
<i>Obr. 17. Vývojový diagram funkce GameLabyrinthOfMatlab.m.....</i>	<i>37</i>
<i>Obr. 18. Ukázka databáze otázek a odpovědí.....</i>	<i>38</i>
<i>Obr. 19. Ukázka databáze nejlepších hráčů .....</i>	<i>39</i>
<i>Obr. 20. Ukázka databáze rozehraných her .....</i>	<i>40</i>
<i>Obr. 21. Znázornění řešení problému s vyhodnocením zablokování.....</i>	<i>42</i>
<i>Obr. 22. Blokové schéma programu popisující vztahy mezi jednotlivými soubory .....</i>	<i>43</i>
<i>Obr. 23. Hlavní okno aplikace .....</i>	<i>55</i>
<i>Obr. 24. Pravidla hry.....</i>	<i>56</i>
<i>Obr. 25. Ovládání hry.....</i>	<i>56</i>
<i>Obr. 26. Výpis nejlepších hráčů.....</i>	<i>57</i>
<i>Obr. 27. Tázací dialog při ukončení aplikace .....</i>	<i>57</i>
<i>Obr. 28. Tázací dialog pro výběr hry .....</i>	<i>58</i>
<i>Obr. 29. Seznam rozehraných her .....</i>	<i>58</i>
<i>Obr. 30: Žádné rozehrané hry .....</i>	<i>59</i>
<i>Obr. 31. Dotaz na jméno hráče.....</i>	<i>59</i>
<i>Obr. 32. Hra Labyrinth of MATLAB.....</i>	<i>60</i>

<i>Obr. 33. Informace o stavu hráče po dokončení prvního levelu.....</i>	<i>60</i>
<i>Obr. 34. Informace o průběžném stavu hráče po úspěšném průchodu místnosti .....</i>	<i>61</i>
<i>Obr. 35. Informace o konečných výsledcích hráče .....</i>	<i>61</i>
<i>Obr. 36. Informace o vypršení časového limitu .....</i>	<i>62</i>
<i>Obr. 37. Tázací dialog při ukončení hry.....</i>	<i>62</i>
<i>Obr. 38. Tázací dialog pro uložení hry.....</i>	<i>62</i>

**SEZNAM TABULEK**

<i>Tab. 1. Parametry hry.....</i>	<i>33</i>
<i>Tab. 2. LabyrinthOfMatlab.m .....</i>	<i>44</i>
<i>Tab. 3. pravidla.m.....</i>	<i>45</i>
<i>Tab. 4. ovladani.m .....</i>	<i>46</i>
<i>Tab. 5. nejlepsiHraci.m.....</i>	<i>46</i>
<i>Tab. 6. unik.m .....</i>	<i>47</i>
<i>Tab. 7. GameLabyrinthOfMatlab.m .....</i>	<i>48</i>
<i>Tab. 8. drawLabyrinth.m .....</i>	<i>49</i>
<i>Tab. 9. prekresliZdi.m.....</i>	<i>50</i>
<i>Tab. 10. casovac.m .....</i>	<i>51</i>
<i>Tab. 11. ulozeniHry.m.....</i>	<i>54</i>

## **SEZNAM PŘÍLOH**

**PŘÍLOHA P I: CD - ROM**