Vývoj prostředí pro výukovou aplikaci v MATLABu

Radek Majar

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Radek Majar**
Osobní číslo:	**A17058**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Prezenční**
Téma práce:	**Vývoj prostředí pro výukovou aplikaci v MATLABu**
Téma práce anglicky:	**The Development of an Environment for Teaching Applications in MATLAB**

Zásady pro vypracování

1. Seznamte se s bakalářskými pracemi, které se zabývají výukovými aplikacemi v programu MATLAB a zpracujte literární průzkum o programu MATLAB.
2. Sestrojte strukturu programu výukové aplikace v MATLABu.
3. Vytvořte programové prostředí pro výukovou aplikaci.
4. Vytvořte vlastní databázi otázek z MATLABu.
5. Uveďte popis tvorby výukové aplikace a použité funkce.

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

Seznam doporučené literatury:

1. DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB – tvorba uživatelských aplikací. 1. vydání. Praha: BEN – technická literatura, 2004. 216 s. ISBN: 80-7300-133-0.
2. DUŠEK, František. MATLAB a Simulink – Úvod do používání. 1.vydání. Pardubice: Univerzita Pardubice, 2001. 146 s. ISBN: 80-7194-273-1
3. KARBAN, Pavel. Výpočty a simulace v programech Matlab a Simulink. Praha: BEN-technická literatura, 2007. ISBN: 978-80-251-1448-3.
4. KOZÁK, Štefan a Slavomír KAJAN. Matlab – Simulink. 1. vyd. Bratislava: Slovenská technická univerzita, 1999. 125 s. ISBN: 80-227-1213-2.
5. PERUTKA, Karel. MATLAB – Základy pro studenty automatizace a inform. technologií, FT UTB Zlín 2005, ISBN 80-7318-355-2.

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

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

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 6.5.2021 Radek Majar, v. r.

ABSTRAKT

Bakalářská práce se věnuje vývoji prostředí pro výukovou aplikaci v programu MATLAB.

Teoretická část popisuje prostředí programu MATLAB a jiné studentské práce, zabývající se výukovými aplikacemi v MATLABu.

Praktická část práce obsahuje popis vývoje výukové aplikace, která je přiložena na disku CD-ROM. Aplikace je popsána z pohledu programátora a uživatele.

Klíčová slova: MATLAB, Aplikace, Programování

ABSTRACT

The Bachelor thesis deals with the development environment for educational application in the MATLAB.

The theoretical part describes the environment of MATLAB and other student theses dealing with interactive applications in MATLAB.

The practical part contains a description of the development of the educational application, which is included on the CD-ROM. Application is described from the perspective of the programmer and the user.

Keywords: MATLAB, Application, Programming.

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

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

OBSAH

ÚVOD

Cílem této bakalářské práce je vytvoření výukové aplikace ve vývojovém prostředí programu MATLAB. Aplikace by měla dále sloužit jako výuková pomůcka.

V první části bakalářské práce je popsán program MATLAB a jeho využití. Dále je zde popsáno jeho prostředí, vlastnosti, funkce, práce se soubory a editor GUIDE pro tvorbu GUI objektů.

V další části jsou popsány bakalářské práce studentů z minulých let, které se zabývají výukovými aplikacemi v programu MATLAB.

V praktické části je první popsána výuková aplikace z pohledu programátora, včetně popisu všech funkcí a ukázek kódu. V poslední části je popsána výuková aplikace z pohledu obyčejného uživatele, spolu s popisem jsou uvedeny obrázky všech oken, které můžeme vidět ve hře.

I. TEORETICKÁ ČÁST

1 MATLAB

MATLAB je interaktivní programové prostředí a skriptovací programovací jazyk pro vědecké a technické výpočty, vizualizaci, vývoj algoritmů a analýzu dat.

1.1 O programu

Program MATLAB je vyvíjen společností MathWorks. Společnost byla založena v roce 1984. Program je vyvíjen pro operační systémy Microsoft Windows, Mac OS a Linux.

Pomocí aplikačních knihoven a toolboxů můžeme MATLAB použít v nejrůznějších oblastech při řešení úloh.

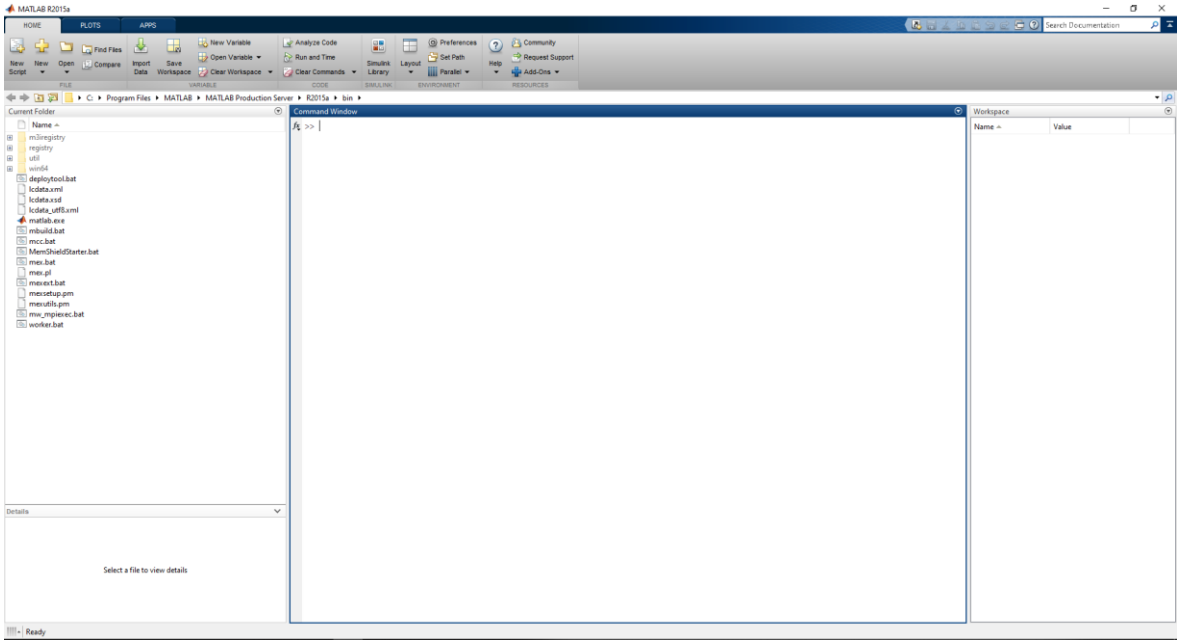
Příklady oblastí:

- Matematika, Statistika, Optimalizace
- Návrh řídicích systémů a analýza
- Zpracování signálu a komunikace
- Testování a měření
- Zpracování obrazu a videa
- Modelování a finanční analýza
- Modelování fyzikálních soustav
- Výpočetní biologie [6]

1.2 Popis prostředí

Po spuštění programu MATLAB se zobrazí pracovní plocha, která je rozdělena na několik částí.

1. *Menu*
2. *Current Folder*
3. *Command Window*
4. *Workspace*
5. *Details*



Obrázek 1. Úvodní obrazovka

1.2.1 Menu

Menu slouží pro uživatele jako přehled možností práce s programem. Dělí se na tři záložky.

1.2.1.1 Home

Nachází se zde základní ovládání a nastavení MATLABu

1. *FILE* – V této záložce jsou funkce pro vytváření, otevírání, hledání a porovnávání souborů. Můžeme zde také vytvořit nové objekty (např. skripta, funkce, GUI)
2. *VARIABLE* – Zde můžeme najít importování dat, uložení *workspace*, vytváření, otevírání a smazání proměnných.
3. *CODE* – Umožňuje přístup k dříve použitým příkazům, dále také jejich mazání v *command window* a v *command history*.
4. *SIMULINK* – Zde můžeme spustit *simulink*
5. *ENVIRONMENT* – V této záložce najdeme nastavení programu. Můžeme zde nastavit rozložení výchozí obrazovky, nastavení výchozí cesty pro vyhledávání a nastavení MATLABu.
6. *RESOURCES* – V záložce najdeme nápovědu MATLABu, přístup ke komunitě a technické pomoci. Také zde můžeme spravovat přídavné doplňky.

1.2.1.2 Plots

Tato záložka se zabývá vytvářením grafů.

1. *SELECTION* – Okno pro výběr proměnné z *workspace.*
2. *PLOTS* – Zde můžeme vidět grafy, které lze vykreslit na základě zvolené proměnné z *workspace.*
3. *OPTIONS* – Zde si můžeme zvolit, jestli se graf má vykreslit do současného nebo do nového okna.

1.2.1.3 Apps

Na této kartě najdeme správu aplikací, označených jako *Apps.*

1. *FILE* – V této záložce můžeme vyhledat a naistalovat nové aplikace.
2. *APPS* – Zde můžeme vidět všechny aktuálně dostupné aplikace.

1.2.2 Current Folder

Current Folder zobrazuje aktuální adresář a jeho obsah. Adresáře můžeme procházet přes průzkumníka Windows.

1.2.3 Command Window

Command window slouží pro komunikaci mezi uživatelem a programem. Zapisujeme zde příkazy a spouštíme funkce. Program zde vypisuje výsledky a systémové hlášení.

1.2.4 Workspace

V okně *workspace* můžeme vidět aktuální definované proměnné a informace o nich. Jako je například jejich název, rozměr a hodnota.

1.2.5 Details

V okně *details* můžeme vidět důležité informace o souboru, který jsme zvolili v *current folder.*

1.3 Užitečné příkazy a funkce

Při práci je užitečné znát příkazy, které se netýkají řešení výpočetního problému, ale jsou užitečné při řešení problémů s MATLABem.

- *pwd* – Zobrazí aktuální adresář.
- *dir* – Zobrazí obsah aktuálního adresáře.
- *cd* – Změní aktuální pracovní adresář.
- *help* – Vypíše seznam všech hlavních témat, ke kterým existuje nápověda. Pokud za příkaz *help* napíšeme druhý argument, tak se vypíše, pokud existuje, nápověda k danému argumentu, například pokud napíšeme příkaz *help help*, vypíše se nápověda k používání nápovědy. Pokud chceme zobrazit nápovědu v novém okně, použijeme příkaz *helpwin.*
- *clc* – Příkaz vymaže celé příkazové okno.
- *clear* – Pro smazání určité proměnné napíšeme příkaz *clear proměnná.* V případě, že chceme smazat celý *workspace*, použijeme příkaz *clear all.*
- *who* – Vypíše seznam všech proměnných v paměti.
- *whos* – Funguje stejně jako *who*, s tím rozdílem, že vypíše u každé proměnné její základní vlastnosti.

Na jeden řádek můžeme psát více příkazů. Pokud chceme, aby se příkaz provedl a vypsal výsledek, použijeme pro oddělení příkazů čárku. V případě, že použijeme středník, příkazy se provedou, ale nevypíše se výsledek. Příkaz můžeme psát i na více řádků, stačí na konec řádku napsat tři tečky. [1]

Pro psaní komentářů napíšeme znak % na začátek řádku.

1.4 Proměnné

V MATLABu nemusíme deklarovat proměnné, protože MATLAB si je automaticky deklaruje sám při prvním výskytu proměnné. Proměnné můžeme vytvořit zadáváním příkazů do *command window* nebo se vyskytují v M-souborech, které obsahují sadu příkazů. MATLAB rozlišuje velká a malá písmena. Proměnná musí vždy začínat písmenem, nesmí obsahovat tečku a může mít maximálně 31 znaků. [1]

1.4.1 Lokální proměnné

Lokální proměnnou nemusíme deklarovat. MATLAB automaticky deklaruje proměnnou při prvním volání. Pokud používáme funkce, můžeme používat lokální proměnnou jen v dané funkci.

1.4.2 Globální proměnné

V případě, že chceme pracovat s jednou proměnnou ve více funkcích, musíme ji nejprve deklarovat pomocí příkazu *global*, protože každá funkce má své lokální proměnné oddělené od ostatních funkcí.

1.4.3 Proměnná persistent

Proměnné *persistent* můžeme používat pouze v jedné funkci. Ostatní funkce k ní nemají přístup, kromě funkce, ve které byla proměnná deklarována. Proměnnou *persistent* musíme deklarovat ve funkci, stejně jako je to u proměnné *global*. Proměnná se deklaruje pomocí příkazu *persistent*. Obě proměnné mají trvalé úložiště, liší se v tom, že persistentní proměnné lze používat jen v jedné funkci, ve které jsou deklarované. Výhodou je, že persistentní proměnné nemůžeme přepsat jinými funkcemi nebo z *command window*. [8]

1.5 Datové typy

V MATLABu máme několik datových typů

1.5.1 Číselné datové typy

Číselné datové typy můžeme rozdělit do dvou oblastí, a to na datové typy, které pracují s celými čísly a s reálnými čísly. Datové typy, které pracují s celými čísly, můžeme dále rozdělit na typy, které pracují s nezápornými celými čísly, jako jsou například *unt8*, *uint16*, *uint32* a *uint64*. Mezi typy, které pracují i se zápornými čísly patří *int8*, *int16*, *int32* a *int64*. S reálnými čísly pracují datové typy *double* a *single*.

1.5.1.1 Datový typ double

Datový typ *double* je jeden z nejpoužívanějších datových typů v MATLABu. *Double* pracuje s reálnými čísly a plovoucí desetinnou čárkou. To znamená, že číslo je složené z celé a z desetinné části. V paměti je uloženo tak, že může dojít ke změně přesnosti, tzn., že se může změnit počet desetinných míst. To se v praxi využívá kvůli vysoké přesnosti výpočtů.[1]

1.5.1.2 Datový typ Single

Datový typ *single* je podobný typu *double*, protože také využívá plovoucí desetinnou čárku, ale s jednoduchou přesností. Výhodou je, že tyhle čísla zabírají méně paměťového prostoru, ale mají menší přesnost a menší rozsah.

1.5.1.3 Datový typ uint

V MATLABu máme 4 datové typy *uint – uint8*, *uint16*, *uint32* a *uint64*. Jedná se o nezáporná celá čísla. Čísla za zkratkou značí, jaký má datový typ rozsah čísel. Pro čísla datového typu *uint* nejsou definované žádné matematické operace, kromě funkce *sum*. Tato čísla můžeme použít pro indexování nebo při logických a relačních operací.

- Datový typ *uint8* – jsou to 8-bitová nezáporná celá čísla v rozsahu 0 až 255. Změna datového typu se provádí pomocí funkce *uint8*. V paměti zabírají 1 bajt.
- Datový typ *uint16* – jsou to 16-bitová nezáporná celá v rozsahu 0 až 65 535. Změna datového typu se provádí pomocí funkce *uint16*. V paměti zabírají 2 bajty.
- Datový typ *uint32* – jsou to 32-bitová nezáporná celá v rozsahu 0 až 4 294 967 295. Změna datového typu se provádí pomocí funkce *uint32*. V paměti zabírají 4 bajty.
- Datový typ *uint64* – jsou to 64-bitová nezáporná celá v rozsahu 0 až 18446744073709551615. Změna datového typu se provádí pomocí funkce *uint64*. V paměti zabírají 8 bajtů. [1]

1.5.1.4 Datový typ Int

Stejně jako *uint*, tak i *int* se dělí na 4 datové typy *int – int8*, *int16*, *int32* a *int64*. *Int* se od *uint* liší tím, že můžeme využít i záporná čísla. Čísla za zkratkou opět značí, jaký má datový typ rozsah čísel. Pro čísla datového typu *int* nejsou definované žádné matematické operace, kromě funkce *sum*. Tato čísla můžeme použít pro indexování nebo při logických a relačních operací.

- Datový typ *int8* – jsou to 8-bitová celá čísla v rozsahu čísle -128 až 127. Změna datového typu čísla se provádí pomocí funkce *int8*. V paměti jeden prvek zabírá 1 bajt.
- Datový typ *int16* – jsou to 16-bitová celá čísla v rozsahu čísle -32 768 až 32767. Změna datového typu čísla se provádí pomocí funkce *int16*. V paměti jeden prvek zabírá 1 bajt

- Datový typ *int32* – jsou to 32-bitová celá čísla v rozsahu čísle -2147483648 až 2147483647. Změna datového typu čísla se provádí pomocí funkce *int32*. V paměti jeden prvek zabírá 1 bajt
- Datový typ *int64* – jsou to 64-bitová celá čísla v rozsahu čísel -9223372036854775808 až 9223372036854775807. Změna datového typu čísla se provádí pomocí funkce *int64*. V paměti jeden prvek zabírá 1 bajt [1]

1.5.2 Textové datové typy

V MATLABu máme 2 textové datové typy.

1.5.2.1 Char

Datový typ *char* slouží k ukládání jednoho znaku. Definuje se pomocí apostrofů.

1.5.2.2 String

Datový typ *string* může mít neomezené množství znaků. Definuje se pomocí uvozovek. Velikost řetězce odpovídá počtu jeho prvků. Jedná se o vektor obsahující jednotlivá písmena jako prvky. Uživatel vidí dané znaky, ale v paměti jsou znaky uloženy jako číselné znaky ASCII tabulky. [2]

Užitečné příkazy:

- *strjoin* – Spojuje více řetězců do jednoho.
- *strcmp* – Porovnává řetězce mezi sebou.

1.5.3 Struct

Struktura obsahuje více různorodých datových typů, takzvaný kontejner. Jedná se o mnohorozměrné pole. Struktura obsahuje název struktury. Pole struktury jsou názvy polí, do kterých se ukládají data. K jednotlivým polím struktury se přistupuje přes tečku.

1.5.4 Cell

Jedná se o indexované datové kontejnery, tedy jedná se o vektor nebo matici s prvky, které nemusí být stejného datového typu. Každá buňka, může obsahovat libovolný datový prvek.

1.5.5 Handle funkce

V tomhle datovém typu jsou uloženy informace pro odkazování na funkci obsaženou v MATLABu nebo na funkci, kterou vytvořil uživatel. V *handle* jsou uloženy všechny

potřebné informace o funkci a informace potřebné ke spuštění. Pro spuštění slouží příkaz *feval.*

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 [1]

1.6 Operátory

Operátory můžeme rozdělit na relační a logické.

1.6.1 Relační operátory

Relační operátory slouží k porovnání dvou hodnot stejného datového typu nebo dvou polí stejné velikosti. Výsledkem porovnání je logická 1 nebo 0 (*true* nebo *false*).

Mezi relační operátory patří:

1. Větší než (>)
2. Menší než (<)
3. Větší nebo rovno (>=)
4. Menší nebo rovno (<=)
5. Rovná se (==)
6. Nerovná se (~=)

1.6.2 Logické operátory

V MATLABu máme 5 základních logických operátorů. Výsledkem porovnání je buď logická jednička (*true*) nebo logická nula (*false*).

Mezi logické operátory patří:

1. Negace (~)

2. Logický součin pro pole (&)
3. Logický součet pro pole (|)
4. Logický součin pro podmínku (&&)
5. Logický součet pro podmínku (||)

Logické operátory jsou seřazeny od nejvyšší priority po nejnižší.

1.7 Druhy souborů

Během práce s MATLABem se můžeme setkat s několika typy souborů. Často používané soubory jsou popsány níže.

1.7.1 Soubory typu M

Jedná se o textové soubory, do kterých můžeme uložit složitější algoritmy. M soubory jsou uložené na disku. Pro spuštění M-souboru stačí napsat do *command window* název požadovaného M-souboru (M soubor se musí nacházet ve složce, ve které MATLAB vyhledává).

- skripty – jedná se o M-soubor, ve kterém se postupně volají příkazy, tak, jako bychom je psali do *command window*. Script můžeme pustit přes *command window* zadáním názvu skriptu bez přípony, nebo napsáním do jiného skriptu.
- funkce – jedná se o M-soubor, který začíná klíčovým slovem *function*. Název funkce je stejný jako název M-souboru. Funkce má oproti skriptu mnoho výhod. Na rozdíl od skriptu, kde se příkazy provádí postupně, se ve funkci provedou najednou. Funkce může mít vstupní a výstupní parametry.

1.7.2 Soubory typu FIG

V souborech typu FIG jsou uloženy informace o obsahu a nastavení grafického okna. Ve FIG-souboru můžeme najít různé ovládací prvky a také zde můžeme, pomocí objektu *axes,* zobrazit grafy nebo obrázek. FIG-soubor vytvoříme pomocí příkazu *GUIDE*.

1.7.3 Soubory typu MAT

Do souborů typu MAT ukládá MATLAB data na disk. Tyhle soubory mohou podporovat libovolný datový typ. Pomocí funkce *save* můžeme uložit data do souboru, pro čtení slouží funkce *load*.

1.8 Podmínky a cykly

1.8.1 Podmínka If

Podmínka *if* se používá k větvení programu na základě testované proměnné. Základní příkaz pro podmínku je *if*. Za *if* se píše logický výraz, který má vždy logický výsledek. Můžeme mít více podmínek po sobě, oddělují se pomocí příkazu *elseif*. V logických výrazech můžeme testovat vždy jinou proměnnou. Pokud je splněna nějaká z podmínek, ostatní podmínky se již nekontrolují a program pokračuje dál za příkazem *end*. Příkaz *end* ukončuje blok podmínky. V případě, že není splněna ani jedna z podmínek, program bude pokračovat za *else*. Příkazy *elseif* a *else* nejsou povinné.

Ukázka podmínky *if.*

```
if logický výraz
      příkazy
elseif logický výraz
      příkazy
else
end
```

1.8.2 Switch

Pro mnohonásobné větvení používáme *switch*. Výraz za slovem *switch* se porovnává s výrazy za slovem *case*. Pokud je nalezena shoda, provede se příkaz v dané větvi. Pokud shoda není nalezena, provede se příkaz ve větvi *otherwise*. *Switch* se stejně jako *if* ukončuje příkazem *end.*

Ukázka kódu:

```
switch výraz
      case výraz,
            příkaz, …, příkaz
      case {výraz1, výraz2, …
            příkaz, …, příkaz
      …
      Otherwise
            Příkaz, …, příkaz
end
```

1.8.3 Cyklus for

Cyklus *for* používáme v případě, kdy známe počet 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 počáteční hodnotu, druhé číslo velikost kroku a třetí konečnou hodnotu. Pokud se bude počáteční hodnota rovnat konečné hodnotě, tak se *for* cyklus ukončí. *For* cyklus se ukončuje příkazem *end.*

Ukázka cyklu *for:*

```
for proměnná=výraz
      příkaz
      …
      příkaz
end
```

1.8.4 Cyklus while

Cyklus *while* používáme v případě, že neznáme počet opakování. Pokud je výraz za slovem *while* vyhodnocen jako logická jednička, provede se příkaz uvnitř cyklu, dokud nebude výraz vyhodnocen jako logická nula. Cyklus můžeme ukončit pomocí příkazu *break*. Příkaz *break* bývá většinou zapsán v další podmínce uvnitř cyklu a použije se až tehdy, kdy je podmínka splněna.

Ukázka cyklu *while*:

```
while logický výraz
      příkazy
end
```

1.9 Pole a matice

MATLAB je nejvíce zaměřen na maximální využití polí, matic a vektorů. Pole je sada čísel, které se nazývají elementy pole, na které se odkazujeme pomocí indexů. Dimenze pole je seznam všech potřebných indexů pro specifikaci elementu pole. Matice je dvoudimenzionální pole, pro které platí speciální pravidla pro přidání elementu, násobení a další operace. Dimenze matice se označují jako řádek a sloupec. Vektor je jako matice, ale jedna dimenze musí mít index 1. [1]

1.9.1 Vyváření polí

Pole vytvoříme pomocí hranatých závorek. Sloupce se oddělují mezerou nebo čárkou, řádky se oddělují středníkem nebo novým řádkem.

```
>> maticeA = [ 2 8; 4 6 ]
maticeA =
     2     8
     4     6
>> maticeB = [ 3,7
5,9 ]
maticeB =
     3     7
     5     9
```

Vektor můžeme vytvořit pomocí symbolu dvojtečky. Syntaxe je *prvniPrvek:krok:posledniPrvek.*

Vektor můžeme vytvořit i pomocí funkce *linspace*. Syntaxe kódu je *linspace(cislo01, cislo02, pocetCisel). Cislo01* je první číslo ve vektoru a *cislo02* je poslední číslo ve vektoru. *PocetCisel* je počet čísel o stejné vzdálenosti od sebe, která mají být vytvořena mezi čísly *cislo01* a *cislo02* včetně těchto dvou čísel.

Užitečné příkazy pro práci s maticemi:

- *size* – Zobrazí velikost každé dimenze.
- *lenght* – Zobrazí nejdelší dimenzi.
- *ndims* – Počet dimenzí.
- *eye* – Příkaz pro vytvoření jednotkové matice.
- *ones* – Příkaz pro vytvoření matice, která je plná jedniček.
- *zeros* – Příkaz pro vytvoření matice, která je plná nul.
- *diag* – Příkaz pro vytvoření diagonální matice.
- *triu* – Příkaz pro vytvoření horní trojúhelníkové matice.
- *tril* – Příkaz pro vytvoření dolní trojúhelníkové matice.
- *rand* – Příkaz pro vytvoření matice, která je naplněna náhodnými čísly. [1]

1.9.2 Operace s poli

Pokud chceme provádět operace s celými poli najednou, používáme k tomu standartních symbolů pro operace. V případě že chceme provádět operace po prvních, přidáme před symbol operace tečku.

```
>> maticeC = maticeA + maticeB
maticeC =
     5    15
     9    15
>> maticeC = maticeA * maticeB % maticove nasobeni
maticeC =
    46    86
    42    82
>> maticeC = maticeA .* maticeB % nasobeni po prvcich
maticeC =
     6    56
    20    54
```

Užitečné funkce pro práci s maticemi:

- *inv* – Výpočet inverzní matice.
- *det* - Výpočet determinantu matice.
- *etg* – Výpočet vlastních čísel a vlastních vektorů matice.
- *help matfun* – Příkaz pro zobrazení nápovědy.

1.9.3 Index polí a matic

Indexy jsou celá čísla a zapisujeme je za název matice do závorek. Indexy můžeme popsat jeden prvek, daný rozsah prvků nebo náhodný rozsah prvků.

Indexování polí a matic můžeme rozdělit na dva způsoby:

1. Po prvcích
2. Podle dimenze

Pokud vytvoříme matici A, která bude mít 3 řádky a 5 sloupců, tak matice A bude mít celkem 15 prvků. Budeme-li chtít například zobrazit poslední prvek v matici A, můžeme použít indexaci po prvcích.

```
maticeA(15)
```

Nebo můžeme použít indexaci podle dimenze.

```
maticeA(3,5)
```

Pokud chceme získat celý řádek nebo sloupec, použijeme dvojtečku. Například pro získání celého posledního sloupce použijeme příkaz:

```
maticeA(:,5)
```

1.10 Komplexní čísla

V MATLABu můžeme pracovat i s komplexními čísly. Pokud je imaginární část nulová, pak s ní MATBLAB nepočítá a číslo je automaticky reálné. Imaginární číslo v MATLABu značíme pomocí znaku i nebo j.

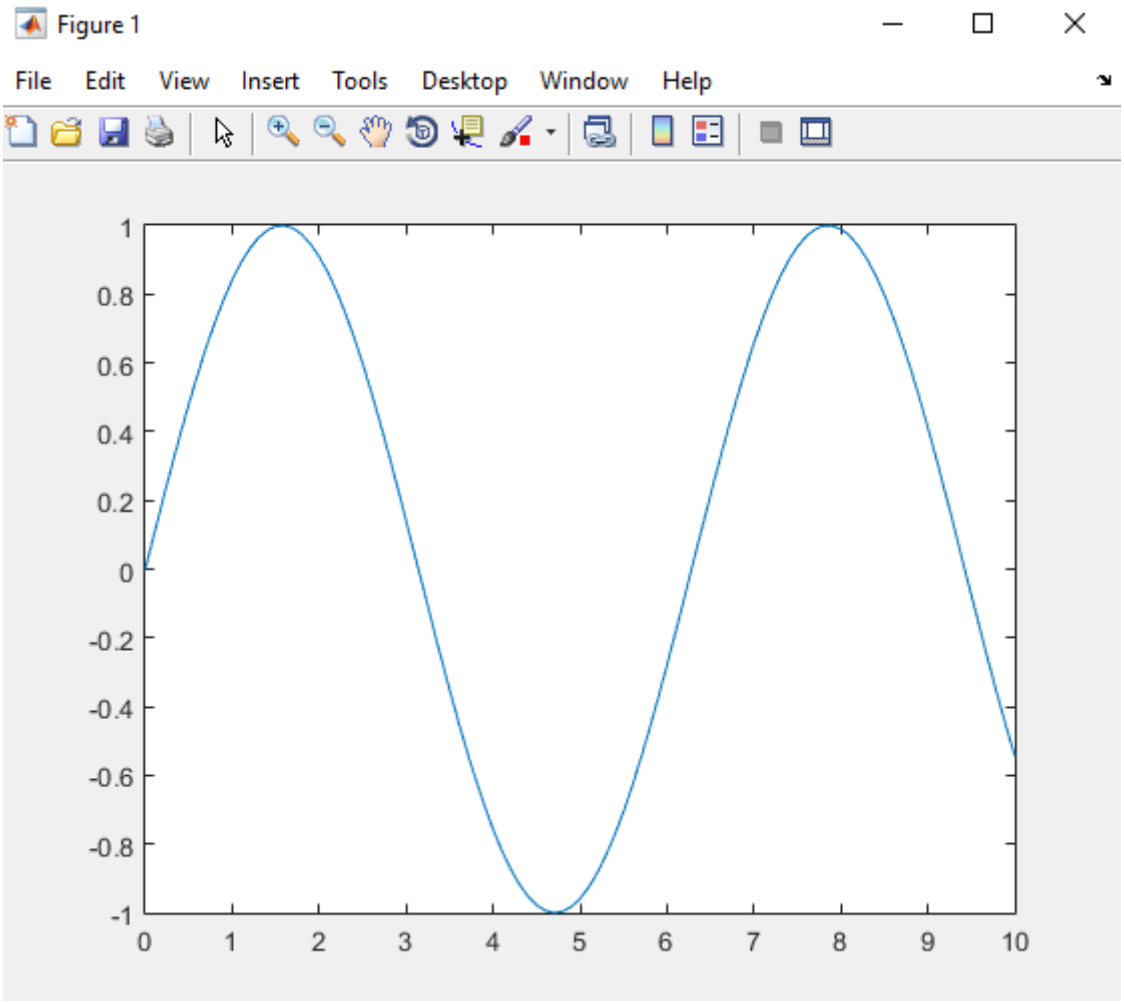
Užitečné příkazy pro práci s komplexními čísly:

- *real* – Reálná část komplexního čísla.
- *imag* – Imaginární část komplexního čísla.
- *abs* – Absolutní hodnota komplexního čísla.
- *angle* – Argument komplexního čísla.
- *isreal* – Vrací logickou jedničku, v případě že číslo obsahuje reálnou část. [1]

1.11 Grafika ve 2D

Pro grafický výstup se nejčastěji používá funkce *plot*.

```
1 t = 0:0.1:10;
2 sinus = sin(t);
3 plot(t, sinus)
```



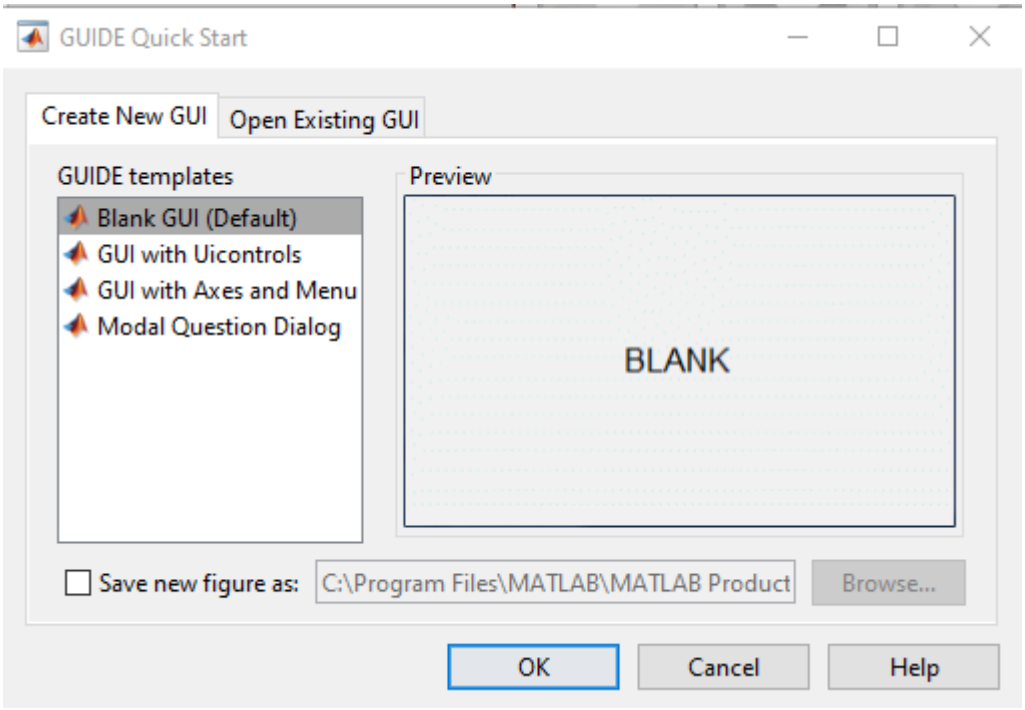
Obrázek 2 Graf funkce sinus

V grafickém okně můžeme vidět výsledek příkazu *plot*. Máme zde několik nástrojů. Díky těmto nástrojům můžeme například napsat text do grafického okna, nakreslit úsečku, zvětšovat, zmenšovat nebo otáčet s výstupem.

V menu můžeme uložit jako obrázek výstup z grafického okna. Pomocí příkazů nebo pomocí menu můžeme nastavit nadpis, popisy os, legendu, typ grafu, jeho vzhled a barvu.[1]

1.12 Grafické uživatelské rozhraní

Pro vytvoření grafického uživatelského rozhraní v MATLABu slouží nástroj *GUIDE*. Tento nástroj pustíme zadáním příkazu *guide* do *command window* nebo pomocí menu v záložce *New -> Graphical User Interface*. V obou případech se nám zobrazí následující dialog.

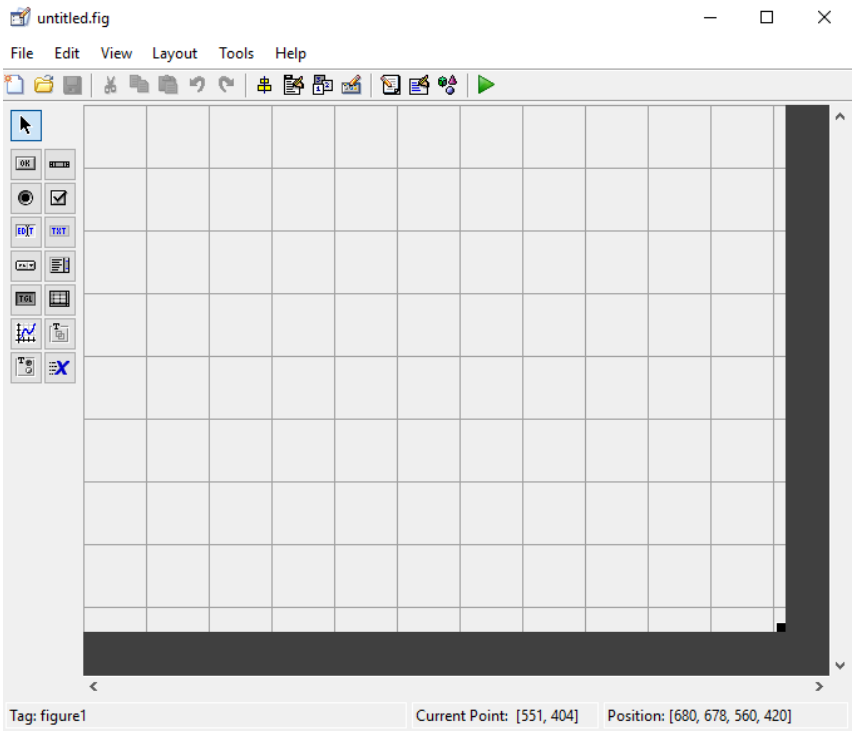


Obrázek 3 GUIDE Quick Start

Můžeme si vybrat ze dvou karet. Karta *Create New GUI* nabízí možnost vytvoření nového dialogu a karta *Open Existing GUI* nabízí možnost otevření již existující GUI.

Na první kartě máme dále na výběr ze čtyř možností. Šablona *Blank GUI (default)* vytvoří prázdné dialogové okno, bez jakýchkoliv ovládacích prvků. Druhá šablona *GUI with Uicontrols* nabízí možnost vytvoření dialogového okna s tlačítky, textem, radio butony a editačními okny. Třetí šablona *GUI with Axes and Menu* nabízí možnost vytvoření dialogového okna, na kterém se nachází tlačítko, pop-up menu a osy. Poslední šablona, *Modal Question Dialog,* nabízí možnost vytvoření dialogového okna s otázkou, obsahuje text, tlačítka a ikonu. Ikona je vložena pomocí os, na kterých je vložený obrázek.

Po vybrání možnosti *Blank GUI (Default)* se nám objeví následující dialog, do kterého můžeme přidávat vlastní objekty.



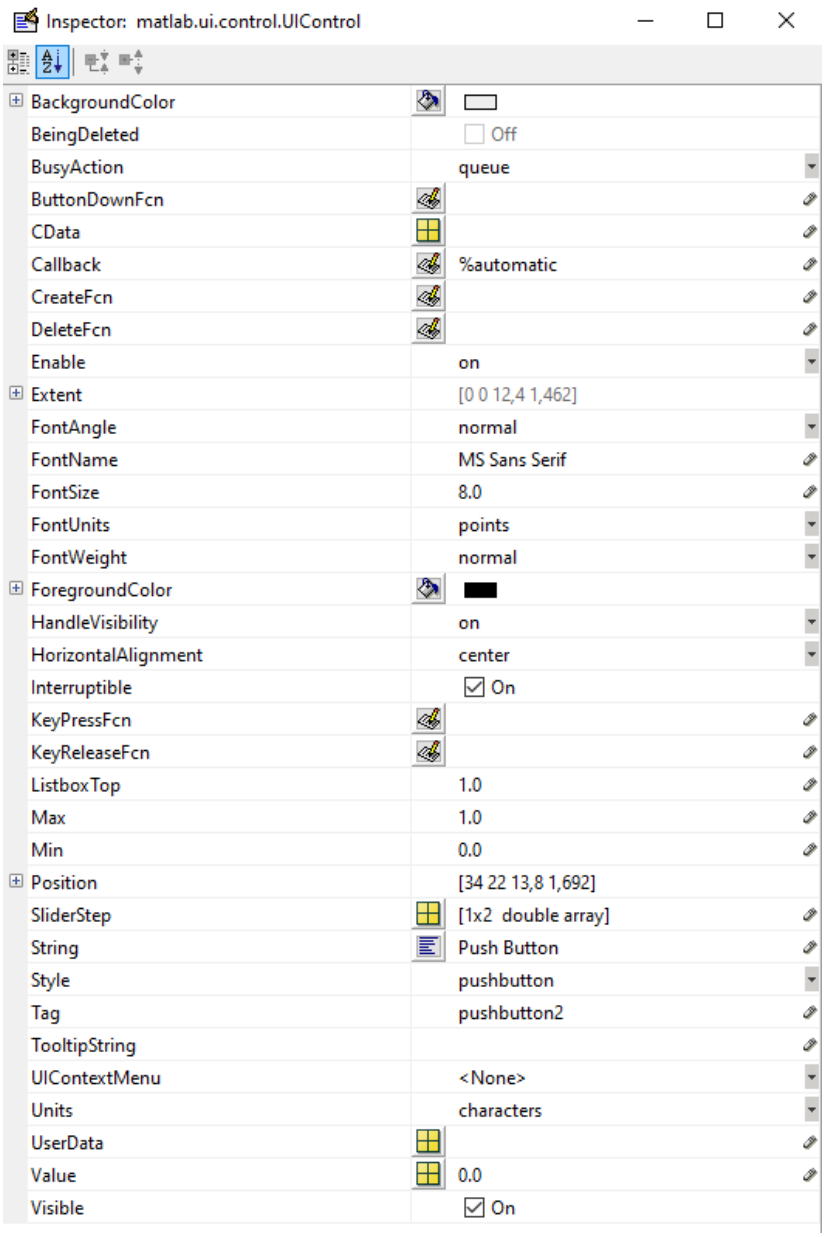
Obrázek 4 Prázdné grafické okno

V levé části okna se nachází lišta s ovládacími prvky. Ovládací prvky umisťujeme do grafického okna pomocí myši, kde dále můžeme změnit jejich velikost.

Lišta obsahuje celkem 14 objektů a jednu ikonu *Select* pro označení již existujících objektů.

Najdeme zde klasické tlačítko (*Push Button*), posuvník (*Slider*), zatrhávací tlačítko ve tvaru tečky (*Radio Button*), zatrhávací tlačítko ve tvaru fajfky (*Check Box*), editační okno (*Edit Text),* statický text (*Static Text*), nabídku položek v rozbalovacím menu (*Pop-up Menu*), nabídku položek v seznamu (*Listbox*), dvoustavové tlačítko (*Toggle Button*), tabulku (*Table*), zobrazení os nebo obrázků (*Axes*), panel (*Panel*), skupinu tlačítek (*Button Group*).

Pokud chceme upravit další vlastnosti ovládacích prvků, klikneme na daný prvek 2x levým nebo pravým tlačítkem myši a vybereme možnost *Property Inspector*. Otevře se následující okno s možnostmi nastavení. Pro každý ovládací prvek jsou možnosti nastavení jiné. Pro *Push Button* je to například: barva pozadí, velikost písma, barva písma, umístění, font písma a tag, pomocí kterého budeme na daný objekt volat.



Obrázek 5 Nastavení vlastností

Po uložení všech ovládacích prvků, jejich nastavení a uložení grafického okna, uloží jako FIG-soubor a k němu se automaticky vytvoří M-soubor se stejným názvem. M-souboru jsou již předepsány funkce, do kterých lze dopsat doplňující kód.

V GUIDE dále můžeme nastavit k jednotlivým objektům funkce zpětného volaní. Jednotlivé funkce najdeme v záložce *View->View Callbacks*. Například když vybereme funkci *CloseRequestFcn*, tak se nám M-souboru vytvoří automaticky funkce, do které můžeme napsat doplňující kód.

```
% --- Outputs from this function are returned to the command line.
function varargout = untitled2_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 when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
delete(hObject);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Obrázek 6 Ukázka kódu v M-souboru

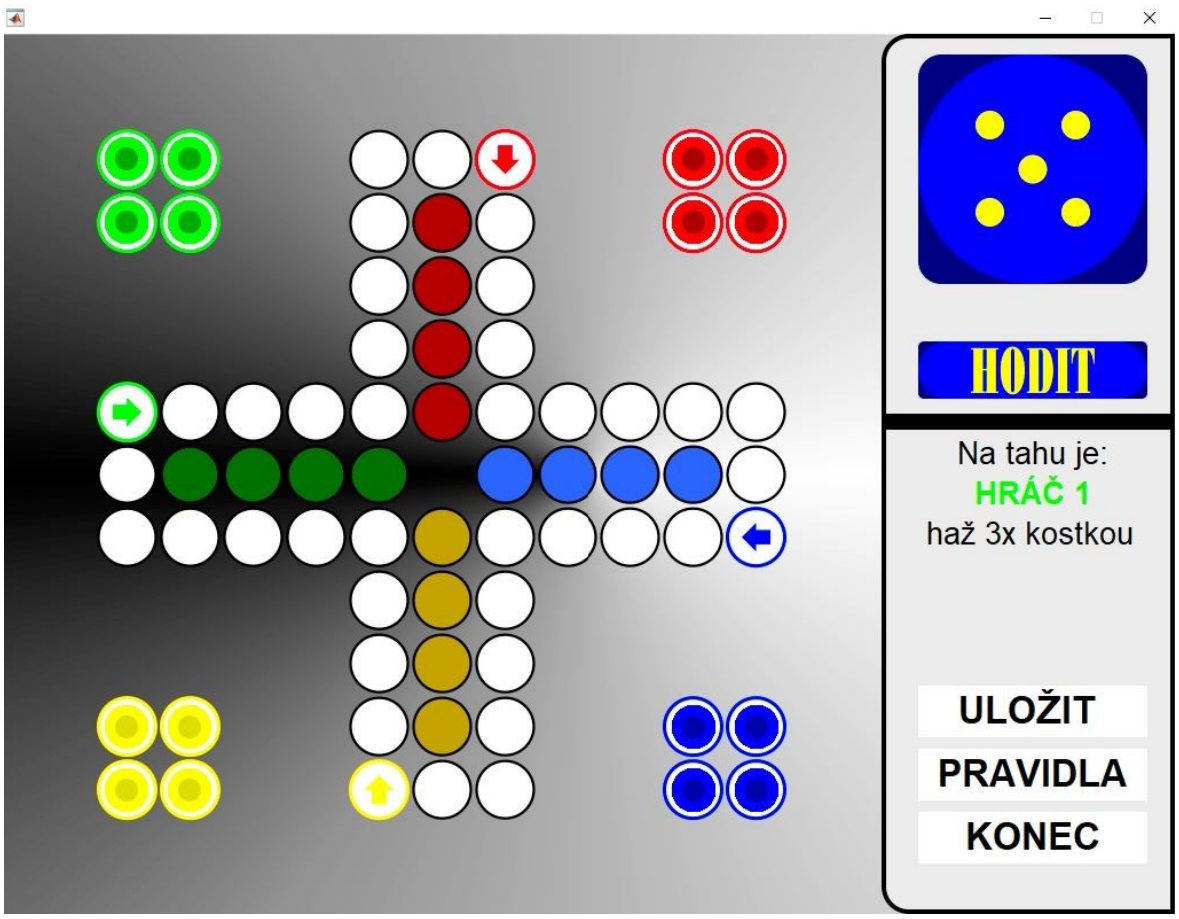
2 HRY V MATLABU

2.1 Člověče, nezlob se

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

Hra je určena pro 2-4 hráče. Protivník může být reálný hráč nebo počítač. Cílem hry je se svými čtyřmi figurkami projít celé hrací pole, až do domečku. Domeček je označen barvou hráče. Vyhrává hráč, který má v domečku všechny čtyři figurky. Hráči se střídají v házení hrací kostky. Pokud hráč nemá v hracím poli žádnou figurku, může házet 3x. Pokud hodí šestku, může dát jednu figurku do hracího pole. Pokud máme figurku v hracím poli, pohybujeme se o počet políček, které hodíme na kostce.

Během hry se můžeme podívat do pravidel. Hru můžeme uložit a pokračovat v ní později. V pravé části hry vidíme informace o hře. Hra je pěkně graficky zpracována.



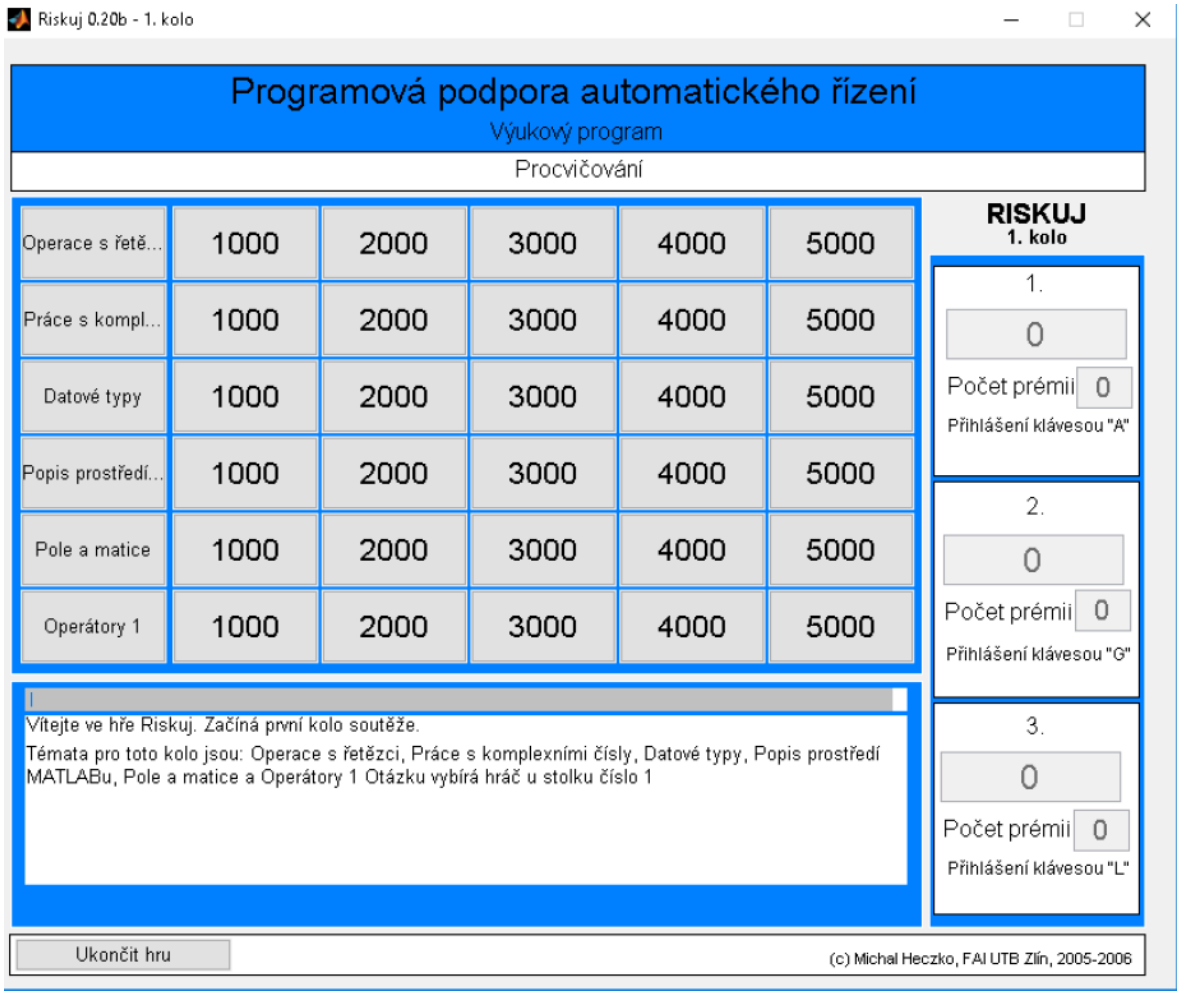
Obrázek 7 Člověče, nezlob se

2.2 Riskuj

Autorem této hry je Michal Heczko. Hru vytvořil v roce 2006, jako součást své bakalářské práce.

Hra je určena pro 1-3 hráče. Hra je inspirována televizním, zábavným pořadem Riskuj. Okruhy témat jsou zaměřeny na otázky z MATLABu. Hráč, který je na řadě si vybírá téma a hodnotu otázky. Pokud odpoví správně, body se mu přičtou k jeho skóre. Ve hře je časový limit jak na jednotlivé otázky, tak na délku kola. Po vypršení časového limitu vyhrává hráč s nejvíce body.

Hra slouží jako výuková aplikace pro program MATLAB. Uživatel si díky ní může ověřit své znalosti z MATLABu zábavnou formou.



Obrázek 8 Hra Riskuj

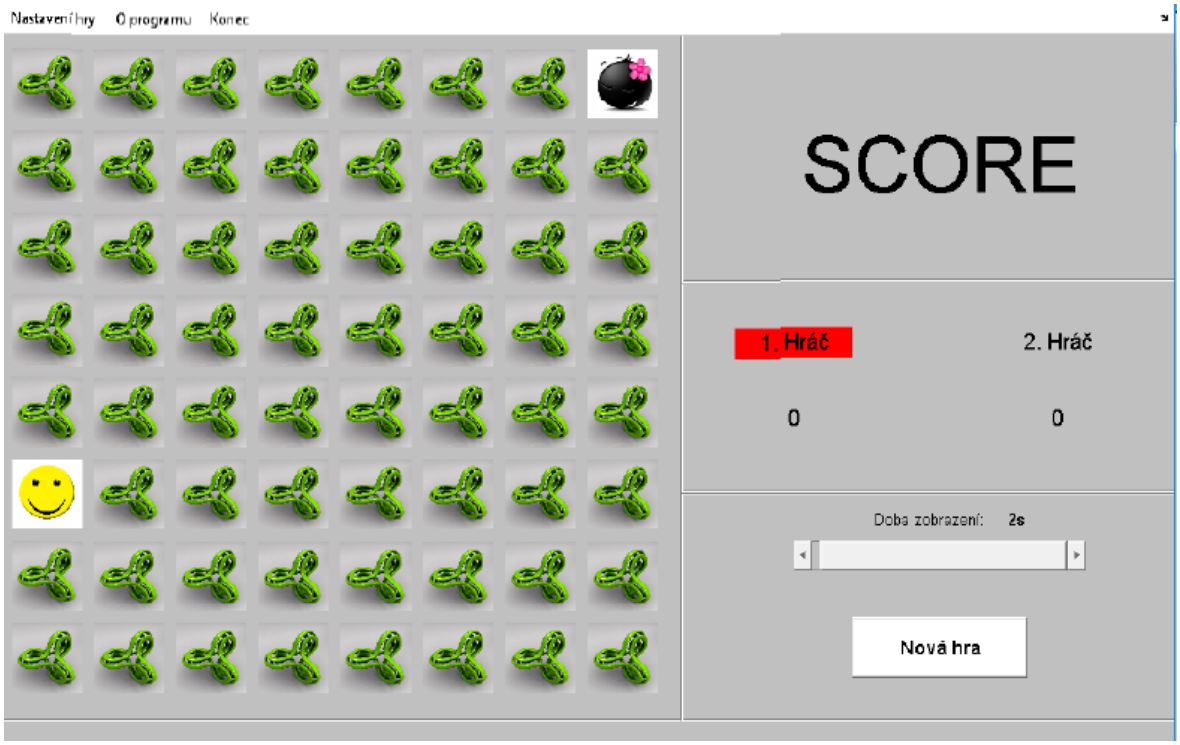
2.3 Pexeso

Tomáš Sedlák vytvořil v roce 2009 v MATLABu hru Pexeso.

Hra je určena pro 2 hráče. Před spuštěním hry můžeme zvolit velikost pexesa (4x4, 6x6 nebo 8x8), vzhled rubu pexesa (zámek, semafor, smyčky nebo klasický), vzhled líce pexesa (abeceda, smajlíci nebo dopravní značky).

Cílem hry je najít páry kartiček, které k sobě patří. Pokud hráč otočí dvě kartičky, které k sobě patří, jsou kartičky smazány z hracího pole, přičte se bod danému hráči a hráč může vybírat další dvě kartičky. Pokud otočí dvě kartičky, které k sobě nepatří. Kartičky se otočí zpět a hraje druhý hráč. Hra končí, jakmile jsou nalezeny všechny páry kartiček. Vyhrává hráč s více body. Pro zvolení obtížnosti hry, je možné nastavit čas pro prohlížení karet od 0,5 do 2 sekund.

Autor si dal práci s různými možnostmi nastavení. Chybí zde možnost hrát proti počítači a také by se hra mohla vylepšit přidáním výukové tématiky.



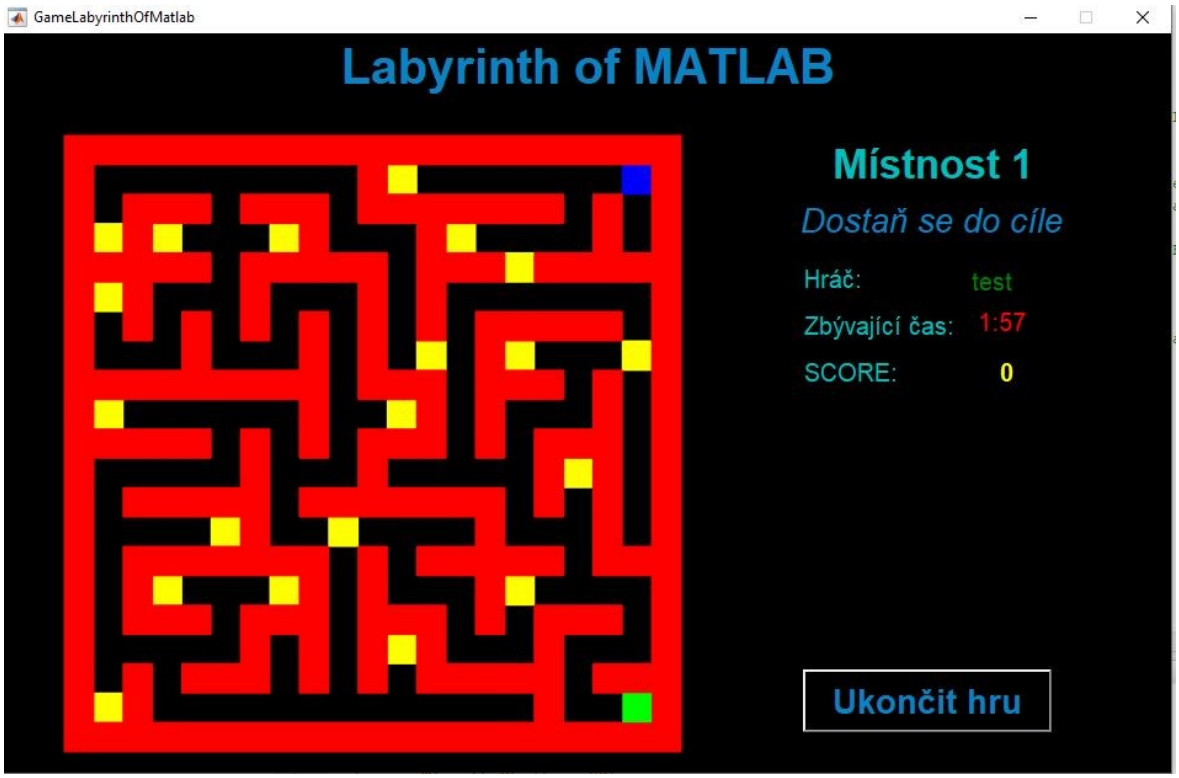
Obrázek 9 Pexeso

2.4 Labyrinth of Matlab

Lucie Šarmanová vytvořila v roce 2016, hru Labyrinth of Matlab.

Cílem hry je projít celým labyrintem, až do cíle. Hra má deset úrovní. Hráč se po labyrintu pohybuje pomocí šipek. V případě, že hráč stoupne na žluté políčko, zobrazí se otázka týkající se MATLABu. Po správném zodpovězení na otázku žluté políčko zmizí a hráč může pokračovat dále. Pokud odpoví špatně, z políčka se stane nepropustná zeď.

Hra obsahuje celkem deset úrovní a každá úroveň je časově omezena. Hráč má možnost si hru uložit a později načíst. Dále se v menu hry může podívat na pravidla a seznam deseti nejlepších řešitelů. Hra obsahuje celkem 200 otázek z MATLABu, které jsou uloženy v CSV souboru.



Obrázek 10 Labyrinth of MATLAB

2.5 Kvíz s otázkami z automatizace

Aplikaci Kvíz s otázkami z automatizace vytvořila v roce 2018 Petra Ducháčková. Jedná se o výukovou aplikaci pro předmět automatizace. Aplikace byla vytvořena v programu MATLAB.

Po spuštění hry si můžeme vybrat jedno z šestnácti polí. Na jednotlivá pole můžeme kliknout myší. Po kliknutí na pole se zobrazí otázka se čtyřmi možnými odpověďmi. Vždy je jen jedna odpověď správná. Pokud hráč odpoví správně, tak pole, které jsme předtím vybrali, zmizí a můžeme si vybrat další pole. Pokud hráč zvolí špatnou odpověď, tak pole taky zmizí a hráč přijde o jeden život. Hra končí, jakmile hráč odkryje všechna políčka nebo přijde o všechny životy. Cílem hry je odkrýt skrytý obrázek.

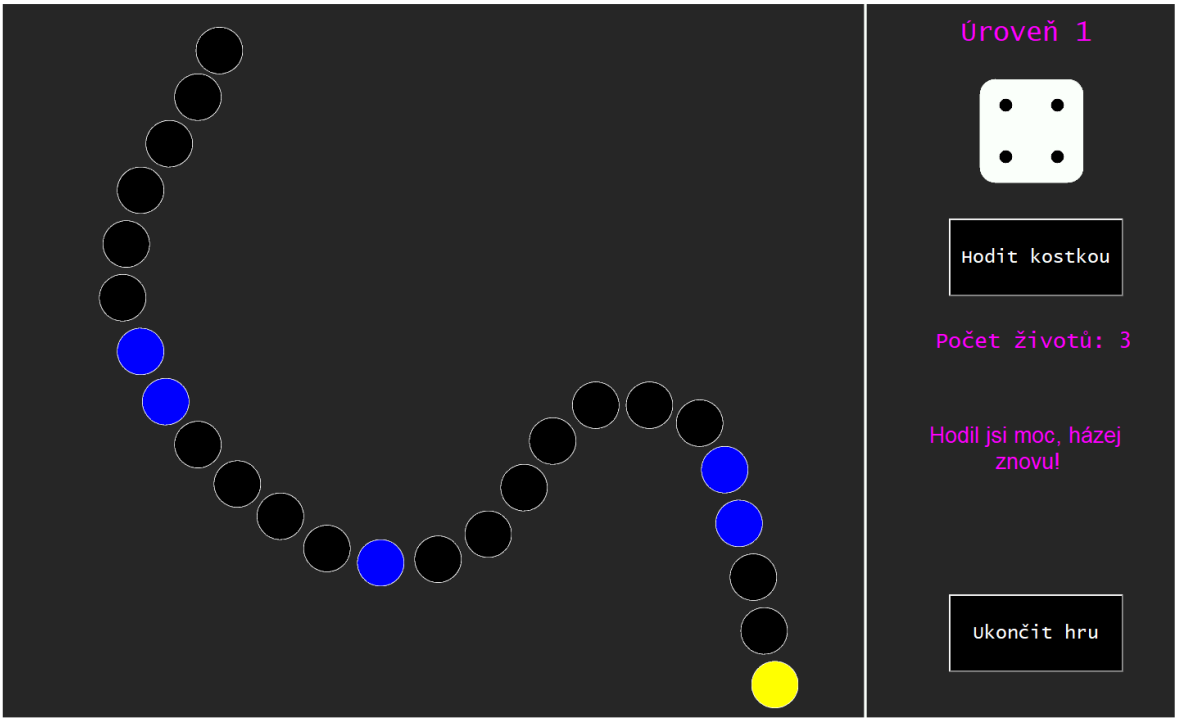
V menu hry má hráč možnost si přečíst pravidla nebo zobrazit vzorové příklady, kde jsou vyřešeny příklady z automatizace. Hra obsahuje celkem šestnáct textových dokumentů, v každém z nich je deset otázek.

Obrázek 11 Kvíz s otázkami z automatizace

2.6 MATLAB Quiz

V roce 2019 vytvořil Dušan Gavenda hru MATLAB Quiz.

Cílem hry je dojít po cestě z políček až do cíle. Jsou zde tři druhy políček: žlutá značí aktuální pozici hráče, modrá je bariéra a černá. Po políčcích se hráč pohybuje pomocí hodu kostkou. Pokud vstoupí na bariéru, tak se zobrazí otázka. Na zodpovězení otázky má hráč omezený čas. Pokud odpoví na otázku správně, tak bariéra zmizí a hráč může pokračovat dál v cestě. V případě že hráč odpoví na otázku dvakrát špatně, tak přijde o jeden život. Hra končí, jakmile hráč projde všech pět levelů nebo pokud ztratí všechny životy. Hra obsahuje celkem 100 otázek týkajících se programu MATLAB.



Obrázek 12 MATLAB Quiz

II. PRAKTICKÁ ČÁST

3 SPECIFIKACE HRY

Kapitola je věnována popisu hry a jejím pravidlům.

3.1 Popis hry

MATLAB vědomostní hra je určena pro všechny studenty, kteří se zabývají programem MATLAB. Hra obsahuje dva herní režimy (hru pro jednoho hráče a hru pro dva hráče).

Hra obsahuje celkem 120 otázek týkající se programu MATLAB. Po spuštění hry jsou otázky náhodně vybrány z databáze otázek, aby se neopakovaly a po každém zapnutí hry byla hra jiná.

Hru je možné spustit pomocí příkazu *menu* v oblasti *command window* v programu MATLAB.

3.2 Pravidla hry

Cílem hry pro jednoho hráče je obsadit celkem 66 políček okolo hradu. Políčka jsou rozdělena do třech úrovní. V první úrovni hráč získá za obsazené políčko 100 bodů, ve druhé 200 bodů a ve třetí 300 bodů. Hráč má k dispozici na začátku hry celkem tři životy. Hráč má možnost získat bonusový život v každé úrovni, pokud bude mít jen jeden život a požadovaný počet bodů v dané úrovni. Hra končí v případě, že hráč ztratí všechny životy nebo obsadí všechna červená políčka okolo hradu. Když hráč zvolí políčko, zobrazí se otázka se čtyřmi možnostmi. Vždy je jen jedna možnost správná. Pokud hráč odpoví špatně, přijde o život a políčko zůstane červené. V případě, že hráč zodpoví na otázku správně získá body podle dané úrovně a políčko se obarví na zelenou barvu.

Hra pro dva hráče obsahuje celkem 60 políček. Hráč jedna je označený zelenou barvou a hráč dvě modrou barvou. Hra končí, jakmile jsou všechna červená políčka obsazena. Vyhrává hráč s více body na konci hry.

4 REALIZACE HRY

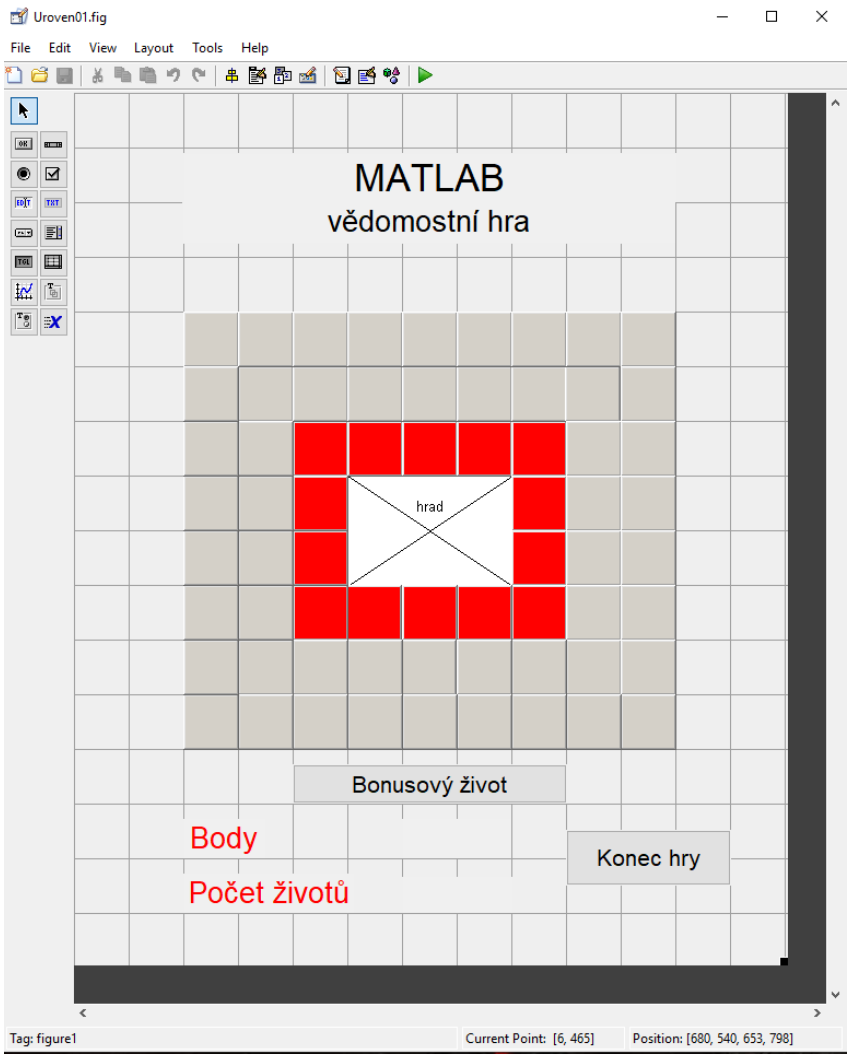
Kapitola je věnována popisu hry z pohledu programátora.

4.1 Popis tvorby hry

Nejprve jsem se seznámil s bakalářskými pracemi, které se zabývají programem MATLAB. Práce jsou popsány v teoretické části.

Hra byla vyvíjena v programu MATLAB verze R2015a a obsahuje celkem 16 M-souborů a 7 FIG-souborů.

První jsem vytvořil GUI soubor hry, kde jsem navrhl vzhled hry pro jednoho hráče. Dále jsem vytvořil další GUI soubory pro menu, pravidla hry, zobrazení otázek a seznam nejlepších hráčů. K jednotlivým GUI souborům MATLAB automaticky vytvořil M-soubory. Později jsem do hry přidal soubory potřebné pro hru se dvěma hráči.

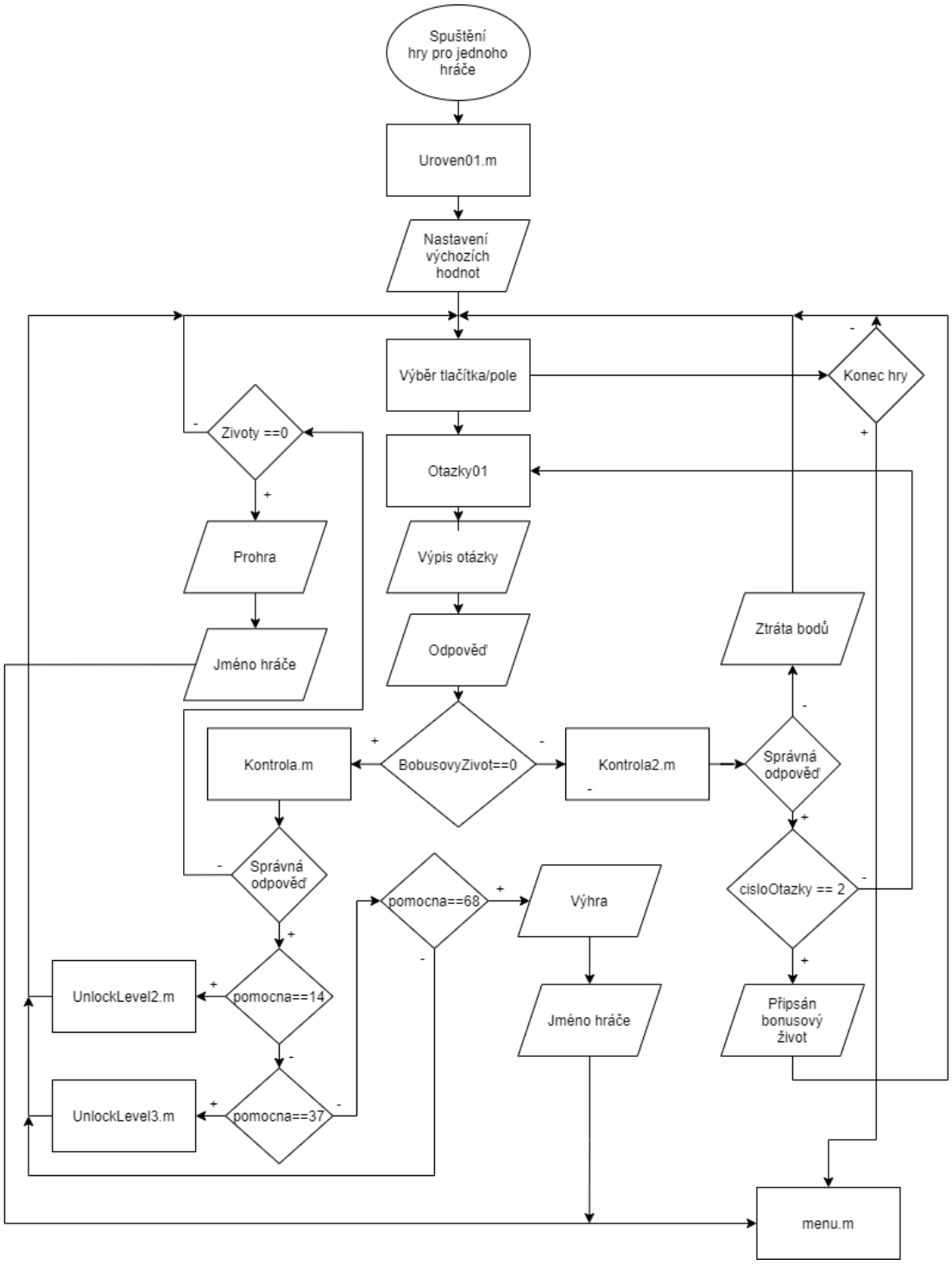


Obrázek 13 Návrh hry pro jednoho hráče

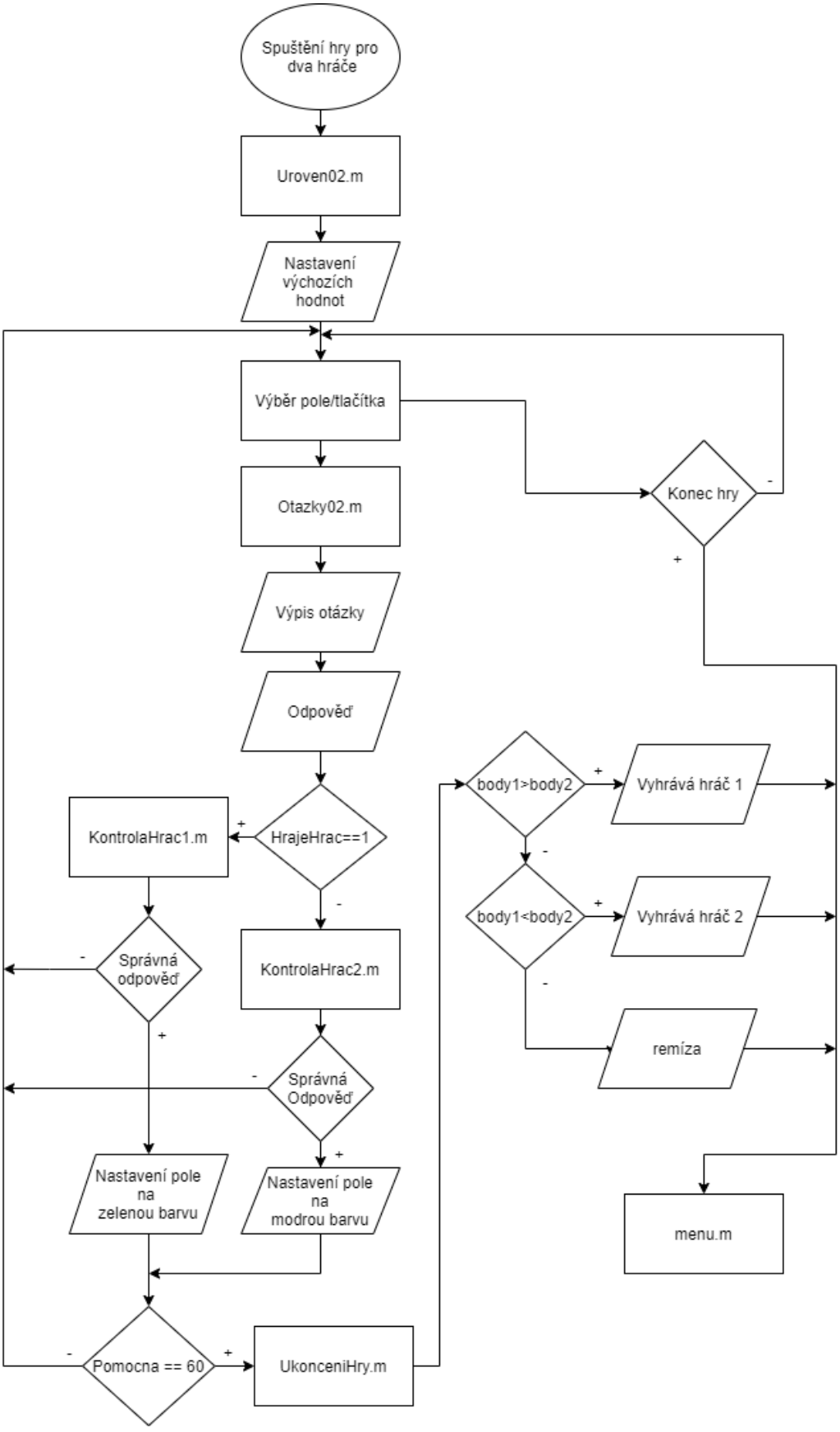
Obrázek hradu jsem navrhnul v programu Inkscape.

Hra obsahuje dvě databáze. V textovém souboru je uložen seznam otázek včetně odpovědí. Pro seznam nejlepších hráčů jsem použil soubor CSV.

Struktury M-souborů pro hru jednoho a dvou hráčů jsou popsány pomocí vývojových diagramů níže.



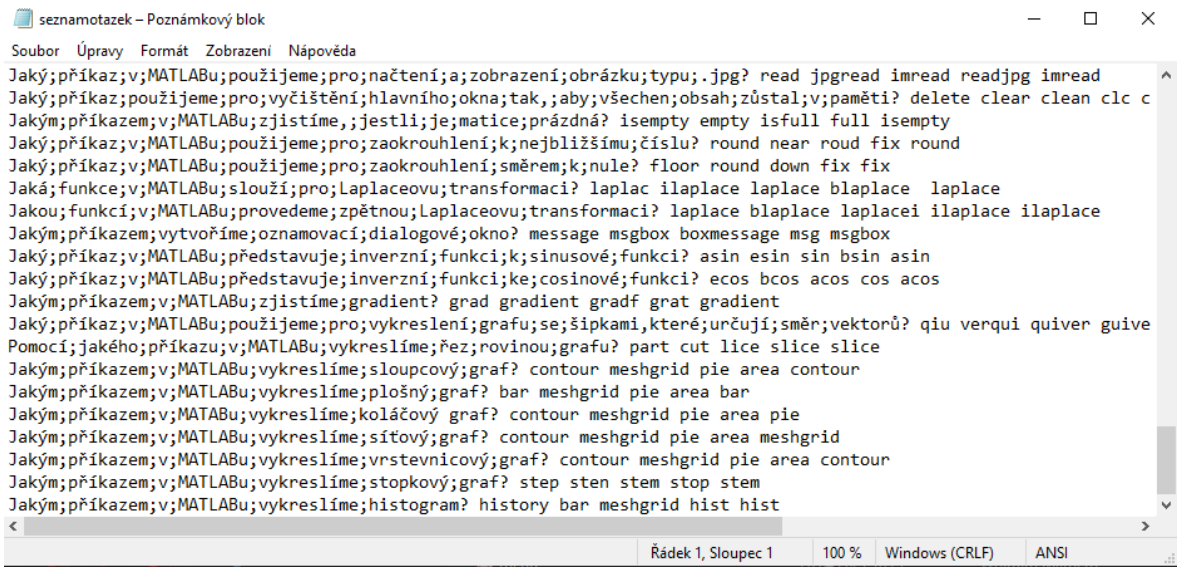
Obrázek 14 Vývojový diagram pro hru jednoho hráče



Obrázek 15 Vývojový diagram pro hru dvou hráčů

4.1.1 Databáze otázek a odpovědí

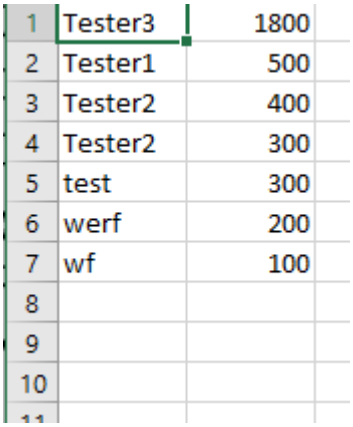
Databázi otázek a odpovědí jsem vytvořil podle mých znalostí, které jsem získal během vývoje této hry a také pomocí doporučené literatury. V textovém souboru je vždy na jednom řádku zapsána otázka, čtyři možné odpovědi a správná odpověď na otázku. Otázka je v textovém souboru zapsána jako jeden řetězec, kde je místo mezerníku použitý znak *;*.



Obrázek 16 Databáze otázek

4.1.2 Databáze seznamu nejlepších hráčů

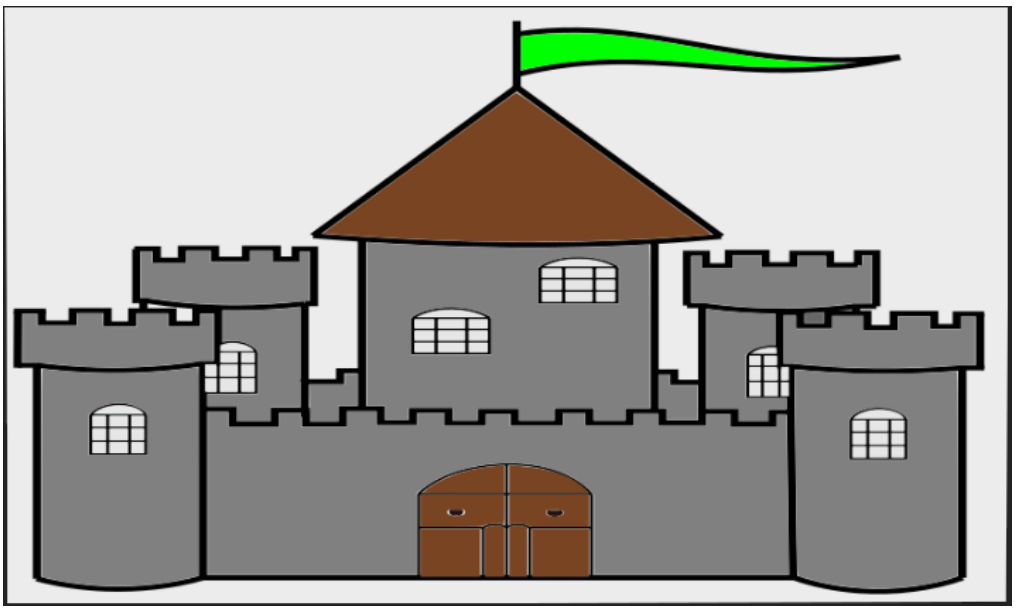
V databázi nejlepších hráčů je uloženo jméno hráče a počet bodů, které hráč získal během hry. Databáze je omezena na 10 hráčů. Údaje jsou uloženy v CSV souboru.



Obrázek 17 Databáze nejlepších hráčů

4.1.3 Inkscape

Pro vytvoření obrázku hradu jsem použil program Inkscape. Inkscape je volně dostupný editor, vhodný pro vytváření vektorové grafiky, jako jsou například ilustrace, perokresby, grafy, loga, ale i složitější obrazy. [7]



Obrázek 18 Hrad

4.1.4 Testování funkčnosti hry

Hru jsem průběžně testoval a vylepšoval. Na doporučení kolegů, kteří hru testovali, jsem přidal do hry další dvě úrovně a možnost získat bonusový život, aby byla hra zajímavější. Původní návrh hry měl jenom jednu úroveň. Žádný z testujících kolegů neměl problém se spuštěním hry.

4.2 Menu.m

Tento soubor slouží jako rozcestník pro spuštění hry, zobrazení pravidel nebo ukončení hry. Obsahuje funkce: *StartButton_Callback, PravidlaButton_Callback, KonecButton_Callback, nejHraci_Callback a HraPro2_Callback.*

4.3 Hra pro jednoho hráče

4.3.1 Uroven01.m

V tomhle M-souboru se nastavuje a spouští celá hra pro jednoho hráče.

Uroven01_OpeningFcn: - hlavním úkolem této funkce je deklarovat a definovat globální proměnné *zivoty, pomocna, body, nradek, x, nasobic, BonusovyZivot a cisloOtazky*. Dále se zde nastavuje obrázek hradu a volá se funkce *LockLevel*.

Funkce *LockLevel* – deaktivuje červená pole pro druhou a třetí úroveň. A také se zde deaktivuje tlačítko pro bonusový život po zapnutí hry. Nejprve pomocí funkce *findobj* najdu konkrétní pole a pak pomocí funkce *set* nastavím *Enable* na *off*.

Ukázka kódu:

```
Odpoved01 = findobj('Tag','pole15');
    set(Odpoved01,'Enable','off');
```

Životy se ve hře nastavují pomocí globální proměnné *zivoty*, po zapnutí hry se nastaví na hodnotu3.. Jak můžeme vidět níže v kódu.

```
global zivoty;
zivoty = 3;
setappdata(0,'zivoty',zivoty);
set(handles.PocetZivotu,'String',zivoty);
```

Globální proměnná *pomocna* slouží pro určení počtu zabraných políček ve hře. Po zapnutí hry se nastaví na hodnotu 0.

Ukázka kódu:

```
global pomocna;
pomocna = 0;
setappdata(0,'pomocna',pomocna);
```

Počet bodů se zapisuje do globální proměnné *body*.

Ukázka kódu:

```
global body;
body = 0;
setappdata(0,'body',body);
set(handles.PocetBodu,'String',body);
```

V globální proměnné *nradek* jsou náhodně uloženy čísla od 1 do 120. Pro vygenerování náhodných čísel v matici, které se neopakují, jsem použil příkaz *randperm*. A globální proměnná *x* určuje index v matici *nradek.*

Ukázka kódu:

```
nradek = randperm(120);
setappdata(0,'nradek',nradek);
x = 1;
setappdata(0,'x',x)
```

Pro zobrazení obrázku hradu ve hře jsem použil příkaz *imshow.*

```
obrHradu = imread('Hrad.jpg');
axes(handles.hrad);
imshow(obrHradu);
```

M-soubor dále obsahuje funkce, které jsou popsány níže.

KonecButton_Callback – funkce, která se ptá hráče, jestli chce ukončit hru. Po kliknutí na tlačítko „KonecButton“ se zobrazí pomocí funkce *questdlg* dialogové okno, které se ptá hráče, jestli chce ukončit hru. Pokud hráč zvolí *Ano*, tak se okno s hrou zavře a zobrazí se menu hry. Pokud hráč zvolí *Ne*, tak se dialogové okno zavře a hráč může pokračovat ve hře.
Ukázka kódu:

```
function KonecButton_Callback(hObject, eventdata, handles)
konec01 = questdlg('Ukončit hru ?', ...
    'Ukončení hry', ...
    'Ano','Ne','Ne');
if strcmp(konec01, 'Ano')
    delete(Uroven01);
    menu;
end
```

figure1_CloseRequestFcn – funkce funguje na podobném principu jako funkce *KonecButton_Callback*. Pokud hráč klikne na červený křížek v pravém horním rohu, tak se zobrazí dialogové okno s otázkou, jestli chce ukončit hru. Pokud hráč zvolí *Ano* tak se okno s hrou zavře. Pokud hráč zvolí *Ne*, tak se dialogové okno zavře a hráč může pokračovat ve hře. Třetí možnost *Zpět do menu* otevře menu hry.

Ukázka kódu:

```
konec01 = questdlg('Opravdu chcete ukončit hru ?', ...
    'Ukončení hry', ...
    'Ano','Ne','Zpět do menu','Zpět do menu');
```

```
if strcmp(konec01, 'Ano')
    delete(hObject);
elseif strcmp(konec01, 'Zpět do menu')
    delete(hObject);
    menu;
end
```

pole01_Callback – po zavolání této funkce se okno s hrou nastaví jako neviditelné. Nejprve pomocí funkce *findobj* najdu požadovaný objekt a pak pomocí funkce *set*, nastavím objekt na neviditelný. Dále se globální proměnná *pomButton* nastaví na jedničku. V proměnné *pomButton* je uloženo číslo pole, na které hráč klikl. Po kliknutí na pole se otevře okno s otázkou pomocí funkce *Otazky01*. Funkce *pole02_Callback* až *pole66_Callback* fungují na stejném principu jako *pole01_Callback.*

Ukázka kódu:

```
global pomButton;
Hrafig = findobj('Type','Figure','Tag','figure1');
set(Hrafig,'Visible','off');
pomButton = 1;
setappdata(0,'pomButton',pomButton);
Otazky01;
```

4.3.2 Otazky01.m

V tomhle souboru dochází k otevření a načtení jednotlivých otázek, které jsou uloženy v textovém dokumentu. Pomocí příkazu *fopen* jsem otevřel textový dokument „otázky.txt“, kde je uložen seznam otázek a pomocí příkazu *strsplit*, jsem rozdělil řetězec obsahu. Do pomocné proměnné *C,* jsem uložil náhodnou otázku z databáze otázek. Otázka je v textovém dokumentu zapsána jako jeden řetězec, kde je místo mezerníku použitý znak *;*. Pro nahrazení *;* v otázce jsem použil příkaz *strrep*, který najde znak *;* v řetězci a nahradí ho mezerníkem.

Ukázka kódu:

```
radek = 0;
seznamOtaz = fopen('seznamotazek.txt');

while ~feof(seznamOtaz)
    radek = radek+1;
    data = fgetl(seznamOtaz);
    pom = strsplit(data);
    for i = 1:6
```

```
        C{radek,i} = pom(i);
        C{radek,i} = C{radek,i}{1};
    end
    C{radek,1} = strrep(C{radek,1},';',' ');
end{radek,1} = strrep(C{radek,1},';',' ');
End
```

V globální proměnné *nradek*, jsou náhodně uloženy čísla 1 – 120, které určují řádek v textovém souboru, kde jsou uloženy otázky. Globální proměnná x určuje pozici v poli *nradek*. Po každém kliknutí na červené políčko ve hře se proměnná x zvětší o 1.

Ukázka kódu:

```
setappdata(0,'otazky',C);
global x;
x = getappdata(0,'x');
global nradek;
nradek = getappdata(0,'nradek');
nahodnyradek = nradek(x);
fclose(seznamOtaz);

x =(x+1);
setappdata(0,'x',x);
```

Po získání dat ze souboru, se pomocí příkazu *set*, nastaví jednotlivé odpovědi do jednotlivých tlačítek a správná odpověď se uloží do proměnné *spravnaodpoved*.

Ukázka kódu:

```
C = getappdata(0,'otazky');
set(handles.otazka,'String',C{nahodnyradek,1});
set(handles.odpovedA,'String',C{nahodnyradek,2});
set(handles.odpovedB,'String',C{nahodnyradek,3});
set(handles.odpovedC,'String',C{nahodnyradek,4});
set(handles.odpovedD,'String',C{nahodnyradek,5});
spravnaodpoved = C{nahodnyradek,6};
setappdata(0,'odpoved',spravnaodpoved);
```

odpovedA_Callback – funkce ve které se získávají hodnoty globálních proměnných *zivoty*, *pomocna*, *body*, *pomButton a BonusovyZivot*. Nastavuje se zde správný výsledek otázky do proměnné *výsledek* a výběr odpovědi (v tomhle případě *odpovedA*) hráče do proměnné *vyber*.

V případě že se globální proměnná *BonusovyZivot* rovná 0, tak se zavolá funkce *Kontrola(vysledek,vyber,zivoty,pomocna,body,pomButton)*, jestli se nebude rovnat 0, tak se zavolá funkce *Kontrola2(vysledek,vyber,zivoty,body)*. V obou případech se ve funkcích vyhodnotí, jestli hráč odpověděl správně na otázku. Rozdíl mezi funkcemi *Kontrola* a *Kontrola2* je vysvětlen níže. Okno s otázkou se zavře pomocí příkazu *closereq*.

Funkce *odpovedB_Callback, odpovedC_Callback, odpovedD_Callback* fungují na stejném principu jako funkce *odpovedA_Callback.*

Ukázka kódu:

```
function odpovedA_Callback(hObject, eventdata, handles)
global zivoty;
global pomocna;
global body;
global pomButton;
global BonusovyZivot;
zivoty = getappdata(0,'zivoty');
pomocna = getappdata(0,'pomocna');
body = getappdata(0,'body');
pomButton = getappdata(0,'pomButton');
vysledek = getappdata(0,'odpoved');
vyber = get(handles.odpovedA,'String');
closereq;
if BonusovyZivot == 0;
Kontrola(vysledek,vyber,zivoty,pomocna,body,pomButton);
else
Kontrola2(vysledek,vyber,zivoty,body)
End
```

`figure1_CloseRequestFcn` – funkce, která zabrání hráči vypnout hru, aniž by odpověděl na otázku výběrem jedné z odpovědí. Pokud hráč klikne na červený křížek v pravém horním rohu, tak se zobrazí zpráva, že hráč musí zvolit jednu z odpovědí. Zpráva je zobrazována pomocí funkce *msgbox.*

Ukázka kódu:

```
error = msgbox('Okno nelze zavřít. Vyberte jednu z odpovědí');
uiwait(error);
```

4.3.3 Kontrola.m

V tomhle souboru se vyhodnocuje, jestli má hráč nárok na bonusový život, jestli odpověděl správně na otázku a nastavuje se zde kolik bodů získá za správnou odpověď.

První se kontroluje, jestli proměnná *vyber* (v proměnné je uložena možnost, kterou zvolil hráč) se shoduje s proměnnou *vysledek* (v proměnné je uložena správná odpověď na otázku). Pomocí funkce *if* zjišťuji, jestli uživatel vybral možnost, která se shoduje se správnou odpovědí. Pokud se shodují přičte se k proměnné *body* proměnná *celkem*. V proměnné *celkem* jsou body vynásobené proměnnou *nasobic*. K proměnné *pomocna* se přičte jednička a políčko na které uživatel klikl se nastaví na zelenou barvu. V případě že se vybraná možnost neshoduje se správným výsledkem od proměnné *zivot* se odečte jednička.

```
if  strcmp(vysledek,vyber)
    pomocna=(pomocna+1);
    setappdata(0,'pomocna',pomocna);
    celkem = 100*nasobic;
    body = (body+celkem);
    setappdata(0,'body',body);

    switch(pomButton)
        case 1
            Odpoved01 = findobj('Tag','pole01');
            set(Odpoved01,'Enable','off');
            set(Odpoved01,'Backgroundcolor','g');
…
        case 66
            Odpoved01 = findobj('Tag','pole66');
            set(Odpoved01,'Enable','off');
            set(Odpoved01,'Backgroundcolor','g');
    end
else
    zivoty = (zivoty-1);
    setappdata(0,'zivoty',zivoty);
```

Pokud se proměnná *zivot* bude rovnat nule, zobrazí se pomocí funkce *msgbox* zpráva „PROHRA - došel vám počet životů“. Po kliknutí *OK* se objeví další dialogové okno, pomocí kterého se hráč může zapsat do tabulky nejlepších hráčů. Pole pro zápis jména je ošetřeno tak, aby nebylo možné provést zápis do tabulky nejlepších hráčů bez zadání jména hráče.

Ukázka kódu:

```
if zivoty == 0
   Prohra = msgbox('PROHRA - došel vám počet životů.');
   uiwait(Prohra);
      jmenoHrace = '';
   while strcmp(jmenoHrace, '') == 1
        jmenoHrace = inputdlg('Zadejte Vaše jméno', 'Jméno hráče', [1 26]);
end
   UlozeniHrace(jmenoHrace, body);
   delete(Uroven01);
   menu;
end
```

Pro odemknutí bonusového života musí být splněny tři podmínky. Globální proměnná *lock* se musí rovnat 0, hráč musí mít poslední život a mít požadovaný počet bodů pro danou úroveň. V první úrovni musí mít hráč minimálně 800 bodů ve druhé 3100 bodů a ve třetí 6100 bodů. Proměnná *lock* určuje, jestli už byla možnost bonusového života využita v dané úrovni. Pokud hráč splní všechny tři podmínky, tak se tlačítko *Bonusový život* aktivuje. A proměnná *lock* se nastaví na hodnotu 1.

Ukázka kódu:

```
if lock == 0
    if zivoty == 1
        if nasobic == 1;
            if body > 700
                    Odpoved01 = findobj('Tag','BonusZivot');
            set(Odpoved01,'Enable','on');
            lock = 1;
            end
        elseif nasobic == 2;
                        if body > 3000
                    Odpoved01 = findobj('Tag','BonusZivot');
            set(Odpoved01,'Enable','on');
            lock = 1;
                        end
        elseif nasobic == 3;
                        if body > 6000
                    Odpoved01 = findobj('Tag','BonusZivot');
            set(Odpoved01,'Enable','on');
            lock = 1;
```

```
            end
        end
    end
end
```

Jestliže se proměnná *pomocna* bude rovnat hodnotě 14, tak se zobrazí pomocí příkazu *msgbox* okno s informací, že hráč postupuje na další úroveň. Proměnná *nasobic* se nastaví na 2 a proměnná *lock* opět na 0. Dále se k proměnné *pomocna* přičte jednička, aby nedošlo k chybě v případě, že by hráč odpověděl na první otázku v nové úrovni špatně. A nakonec se zavolá funkce *UnlockLevel2*.

Ukázka kódu:

```
if pomocna == 14
    nasobic = 2;
    lock = 0;
    pomocna=(pomocna+1);
    setappdata(0,'pomocna',pomocna);

    Level2 = msgbox('Postupujete do další úrovně');
   uiwait(Level2);
   UnlockLevel2;
```

Jakmile se bude proměnná *pomocna* rovnat hodnotě 37, tak se bude postup opakovat jako v případě, kdyby se proměnná *pomocna* rovnala hodnotě 14. S tím rozdílem že se *nasobic* nastaví na hodnotu 3 a zavolá se funkce *UnlockLevel3*.

V případě že se proměnná *pomocna* bude rovnat hodnotě 68, tak se pomocí příkazu *msgbox* zobrazí dialog s informací, že hráč vyhrál. A dál bude postup stejný, jako v případě že by hráč prohrál.

Pokud se proměnná *pomocna* nebude rovnat hodnotě 14,37 ani 68, tak se okno s hrou nastaví jako viditelné.

Ukázka kódu:

```
else
   Hrafig = findobj('Type','Figure','Tag','figure1');
   set(Hrafig,'Visible','on');
end
```

4.3.4 Kontrola2.m

V tomhle souboru se vyhodnocuje, jestli hráč odpověděl 3x správně na otázky pro získání bonusového životu.

První se kontroluje, jestli proměnná *výběr* (v proměnné je uložena možnost, kterou zvolil hráč) se shoduje s proměnnou *výsledek* (v proměnné je uložena správná odpověď na otázku). Pomocí funkce *if* zjišťuji, jestli uživatel vybral možnost, která se shoduje se správnou odpovědí. Pokud hráč odpoví správně na otázku, tak se hodnota proměnné *cisloOtazky* zvýší o jedna a zobrazí se další otázka. Jakmile hráč odpoví 3x správně na otázky, tak se přičte hráči bonusový život a hra pokračuje dalším výběrem pole. Pokud hráč odpoví na jednu ze tří otázek špatně, tak se mu odečtou body podle úrovně, ve které se hráč momentálně nachází.

4.3.5 UnlockLevel2.m a UnlockLevel3.m

Tyhle M-soubory slouží pro odemknutí druhé a třetí úrovně. Pomocí příkazu *set* se nastavují jednotlivá pole na viditelné a barva pozadí na červenou.

Ukázka kódu:

```
Odpoved01 = findobj('Tag','pole15');

set(Odpoved01,'Enable','on');

set(Odpoved01,'Backgroundcolor','r')
```

4.3.6 NejlepsiHraci.m

V tomhle M-souboru se nachází funkce, která načítá seznam deseti nejlepších hráčů ze souboru seznamJmen.csv, ve kterém jsou již hráči seřazeni podle počtu bodů, od hráče s nejvíce body po hráče s nejmíň body.

Dále se zde nachází funkce *KonecButton_Callback,* která smaže okno *NejlepsiHraci* a zobrazí menu hry a funkce *figure1_CloseRequestFcn*, která funguje na stejném principu s rozdílem, že nabídne možnost ukončení celé hry.

4.3.7 UlozeniHrace.m

V tento M-soubor slouží pro ukládání hráčů a jejich počet bodů do databáze nejlepších hráčů. Funkce načte všechna data ze souboru „seznamJmen.csv“ do matice. Dále se do matice přidá nový řádek s údaji o aktuálním hráči. Následně se hráči seřadí podle počtu bodů od nejlepšího po nejhoršího hráče, a nakonec se matice zapíše zpět do souboru „seznamJmen.csv“.

4.4 Hra pro dva hráče

4.4.1 Uroven02.m

V tomhle M-souboru se nastavuje a spouští celá hra pro dva hráče

Uroven01_OpeningFcn: - hlavním úkolem této funkce je deklarovat a definovat globální proměnné*, pomocna, body1, body2, nradek, x. HrajeHrac,* a nastavují se zde obrázky hradů pro hráče jedna a dva.

V proměnné *body1* jsou uloženy body hráče 1 a v proměnné *body2* jsou uloženy body hráče 2.

Pomocná proměnná *HrajeHrac* určuje, který hráč je na řadě.

Proměnné *nradek* a *x* plní stejnou funkci jako v souboru Uroven01.m.

M-soubor dále obsahuje funkce *figure1_CloseRequestFcn, KonecButton_Callback, a pole01_Callback až pole60_Callback.* Funkčnost jednotlivých funkcí je stejná jako u souboru Uroven01.m.

4.4.2 Otazky02.m

V tomhle M-souboru se načítají jednotlivé otázky z textového dokumentu a odpovědi se přiřazují do jednotlivých tlačítek, stejně jako je to v souboru Otazky01.m.
odpovedA_Callback – funkce ve které se získávají hodnoty globálních proměnných *pomocna*, *body1*, *body2*, *pomButton* a *HrajeHrac*. Nastavuje se zde správný výsledek otázky do proměnné *výsledek* a výběr odpovědi (v tomhle případě *odpovedA*) hráče do proměnné *vyber*. Pomocí podmínky *if* zjišťuji, který hráč odpovídá na otázku. Pokud se proměnná *HrajeHrac* bude rovnat 1, tak se zavolá funkce KontrolaHrac1, v opačném případě se zavolá funkce KontrolaHrac2.

4.4.3 KontrolaHrac1.m a KontrolaHrac2.m

V M-souboru KonrtolaHrac1.m se vyhodnocuje, jestli hráč 1 správně odpověděl na otázku. Pomocí funkce *if* se kontroluje, jestli proměnná *vysledek* se shoduje s proměnnou *vyber.* Pokud se proměnné shodují tak pole, které hráč 1 vybral, se nastaví na zelenou barvu, hodnota proměnné *pomocna* se zvýší o jedna a pomocná proměnná *HrajeHrac* se nastaví na hodnotu 2. Jakmile se proměnná *pomocna* bude rovnat číslu 60, tak se zavolá funkce

UkonceniHry. V případě že hráč odpoví špatně na otázku, tak pole zůstane červené a na řadě bude hráč 2.

KontrolaHrac2.m funguje na stejném principu jako KontrolaHrac1.m s rozdílem že, pokud hráč odpoví správně na otázku, tak se pole nastaví na modrou barvu a pomocná proměnná *HrajeHrac* se nastaví na hodnotu 1.

4.4.4 UkonceniHry.m

V tomhle M-souboru se vyhodnocuje, který hráč vyhrál. Pomocí funkce *if* se porovnávají proměnné *body1* a *body2*. Pokud hodnota v proměnné *body1* je vyšší než hodnota v proměnné *body2*, tak se pomocí příkazu *msgbox* zobrazí dialogové okno s informací o tom že hráč 1 vyhrál hru. Pokud je hodnota v proměnné body2 vyšší než hodnota v proměnné body1, tak opět zobrazí pomocí příkazu *msgbox* dialogové okno s informací, že vyhrál hráč 2. V případě že se hodnoty rovnají, tak se zobrazí okno s informací, že hra skončila remízou.

Ukázka kódu:

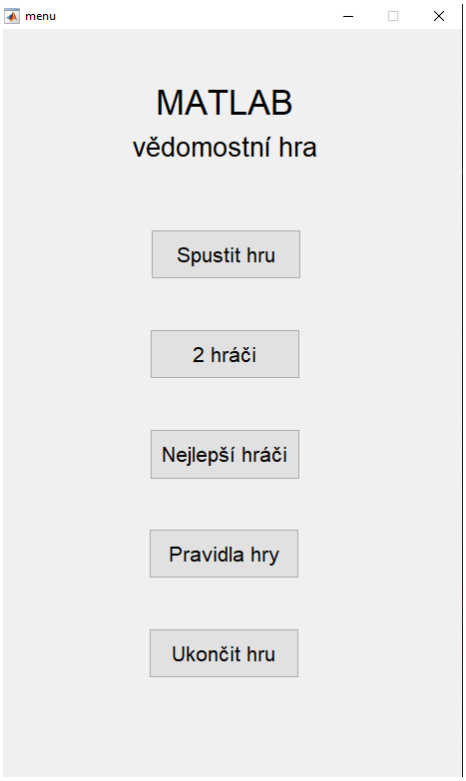
```
if body1 > body2
   vyhra = msgbox('VÝHRA - Hráč 1 vyhrál hru');
   uiwait(vyhra);
   delete(Uroven02);
   menu;
elseif body1 < body2
       vyhra = msgbox('VÝHRA - Hráč 2 vyhrál hru');
   uiwait(vyhra);
   delete(Uroven02);
   menu;
else
   remiza = msgbox('Hra skončila remízou');
   uiwait(remiza);
   delete(Uroven02);
   menu;
end
```

5 POPIS HRY Z POHLEDU UŽIVATELE

Hra se spustí po zadání příkazu *menu* do příkazového okna *command window* v programu MATLAB. Po zadání příkazu *menu* se spustí menu hry.

5.1 Menu hry

V menu hry má uživatel nabídku z pěti možností. Hráč si může vybrat, jestli chce spustit hru pro jednoho hráče nebo pro dva hráče. Dále si hráč může nechat zobrazit seznam nejlepších hráčů, pravidla hry nebo ukončit hru. Hra se ovládá pomocí myši.



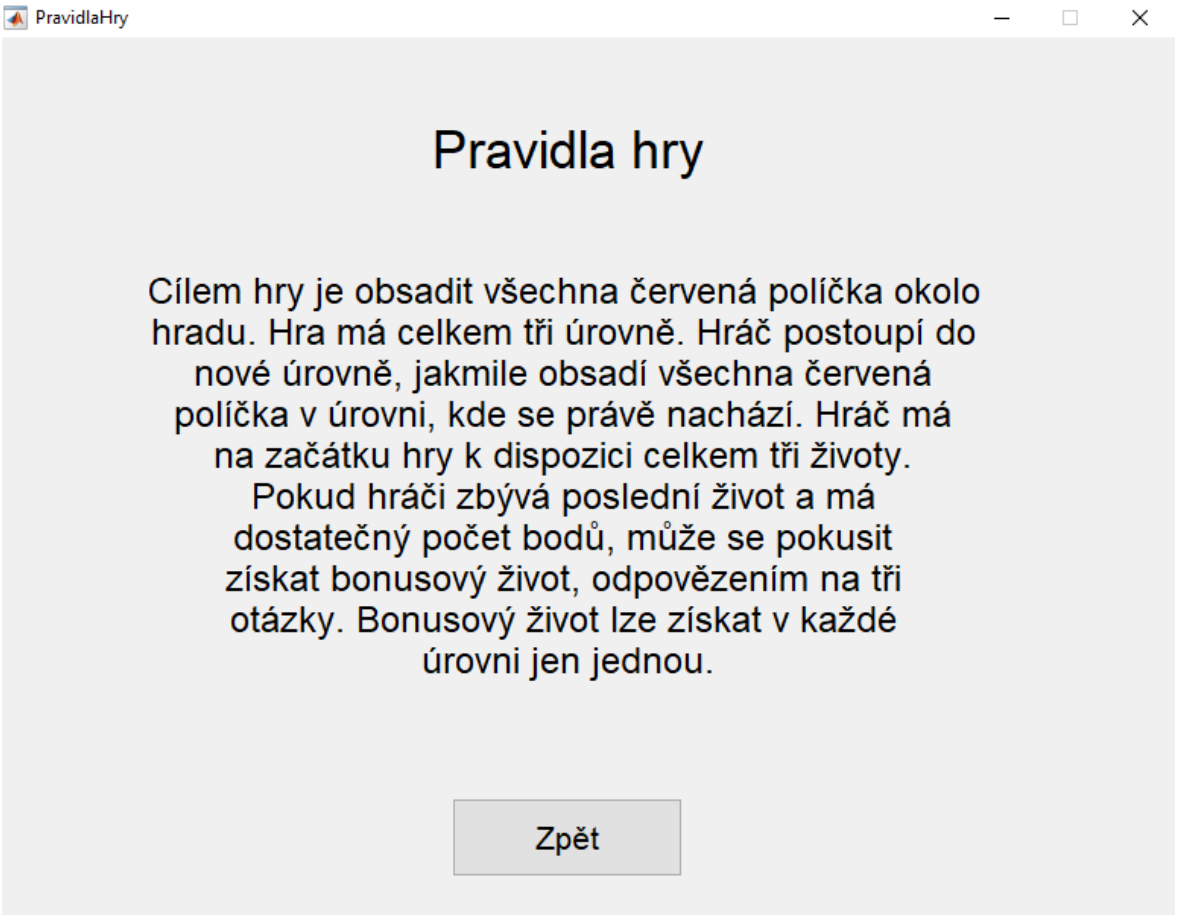
Obrázek 19 Menu hry

5.2 Ukončit hru

Po kliknutí na tlačítko *Ukončit hru,* se okno menu zavře. Zavřít okno s hrou, lze i pomocí červeného křížku v pravém horním rohu.

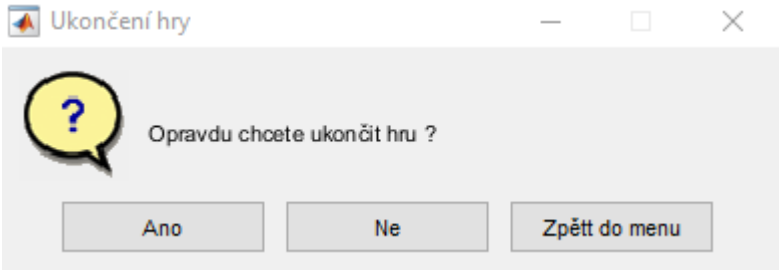
5.3 Pravidla hry

Pokud hráč klikne na tlačítko *Pravidla hry*, menu hry se zavře a otevře se nové okno, kde si hráč může přečíst pravidla hry. Pod pravidly hry se nachází tlačítko *Zpět*. Po kliknutí na tlačítko *Zpět* se okno s pravidly zavře a otevře se menu hry.



Obrázek 20 Pravidla hry

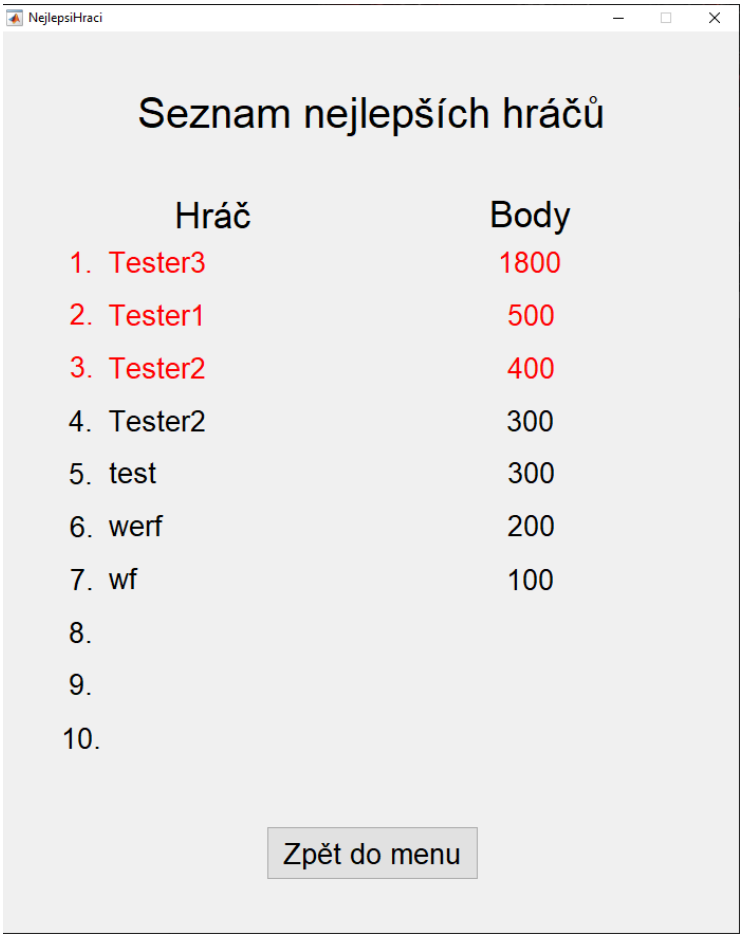
Pokud hráč klikne na červený křížek v pravém horním rohu, tak se zobrazí se dialogové okno s otázkou, jestli chce opravdu ukončit hru. Pokud hráč zvolí *Ano* okno s hrou se vypne, v případě že zvolí možnost *Zpět do menu*, tak se okno s pravidly zavře a otevře se menu hry.



Obrázek 21 Dialog ukončení hry

5.4 Nejlepší hráči

Po stisknutí tlačítka *Nejlepší hráči,* se zobrazí nové okno s tabulkou deseti nejlepších hráčů.

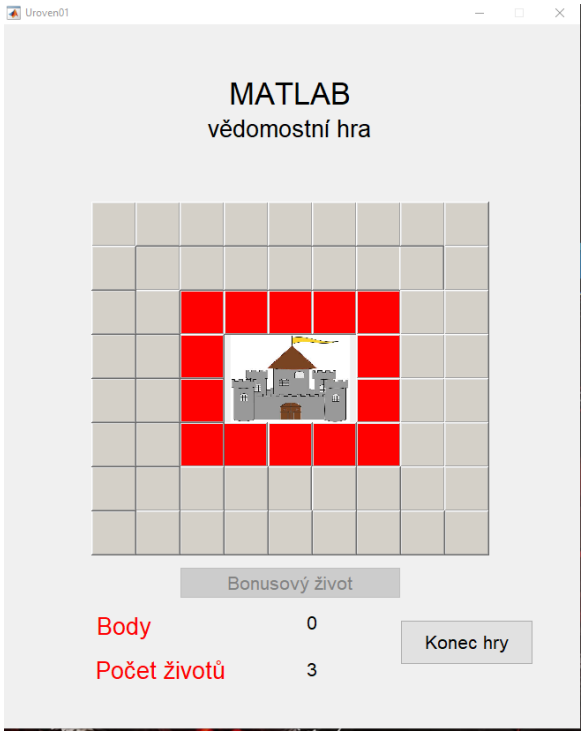


Obrázek 22 Nejlepší hráči

Do menu se hráč může opět dostat pomocí tlačítka *Zpět do menu* nebo křížkem v pravém horním rohu.

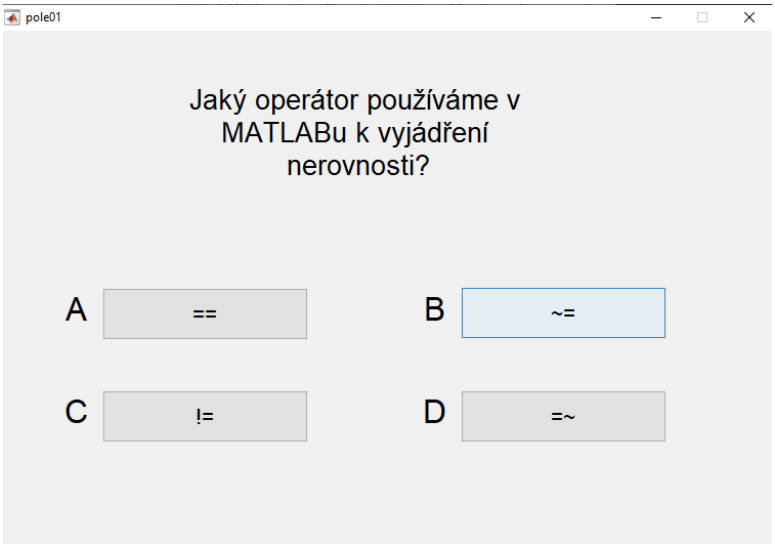
5.5 Hra pro jednoho hráče

V případě, že hráč vybere možnost *Spustit hru*, tak se menu hry zavře a otevře se nové okno s hrou pro jednoho hráče. Ve spodní části hry může hráč vidět počet zbývajících životů a počet získaných bodů. Dále se zde nachází tlačítko pro získání bonusových životů. Tlačítko se hráči zpřístupní, jakmile má jen jeden život a požadovaný počet bodů pro danou úroveň. Hráč pomocí myší vybírá políčka okolo hradu. Políčka, na které hráč může kliknout, jsou označená červenou barvou. Na políčka označené šedou barvou kliknout nemůže.



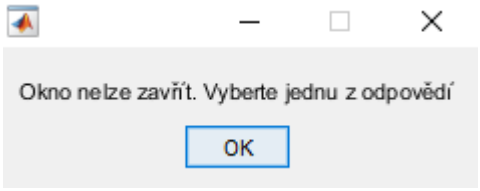
Obrázek 23 Hra pro jednoho hráče

Po stisknutí na červené políčko se otevře nové okno s otázkou. Hráč může vybrat ze čtyř odpovědí na otázku. Na otázku je vždy jen jedna správná odpověď.



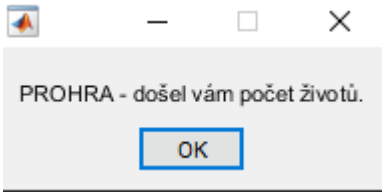
Obrázek 24 Dialog s otázkou

Pokud se hráč pokusí okno s otázkou zavřít pomocí křížku v pravém horním rohu, tak se zobrazí okno s informací, že musí vybrat jednu z odpovědí.



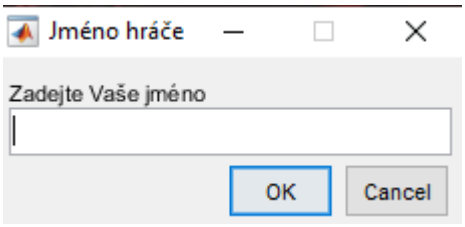
Obrázek 25 Informační dialog

Po výběru jedné z možných odpovědí se okno s otázkou zavře a otevře se okno s hrou. Pokud hráč odpověděl správně na otázku, tak se políčko změní na zelené a přičtou se hráči body podle úrovně. V první úrovni se hráči přičítá sto bodů, ve druhé dvě stě bodů a ve třetí úrovni tři sta bodů. V případě že hráč odpoví špatně na otázku, tak políčko zůstane červené a odečte se hráči jeden život. Pokud hráč přijde o všechny životy, tak se otevře okno s informací o prohře.



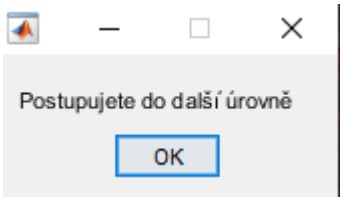
Obrázek 26 Prohra

Po stisknutí tlačítka *OK*, se otevře nové okno s polem, kde hráč může napsat své jméno. Pokud hráč nechá pole prázdné a zmáčkne *OK*, tak okno zmizí a objeví se znovu, dokud hráč nenapíše své jméno. Pokud hráč zmáčkne tlačítko *Cancel*, tak okno zmizí a otevře se menu hry.



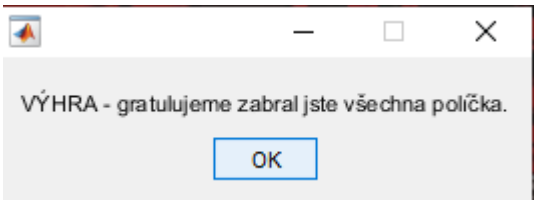
Obrázek 27 Dotaz na jméno hráče

Pokud hráč obsadí všechna červená políčka v první úrovni, tak se zobrazí dialog s informací, že hráč postupuje do další úrovně. Po obsazení všech červených políček ve druhé úrovni se zobrazí dialog s informací, že hráč postupuje do třetí úrovně.



Obrázek 28 Postup do další úrovně

Pokud hráč obsadí všechna políčka okolo hradu, tak se zobrazí informační okno o výhře. Po stisknutí tlačítka *Ok* bude hráč vyzván opět k zadání svého jména do pole, jako v případě prohry. Po zadání jména nebo po stisknutí *Cancel* se dialog zavře a otevře se menu hry.

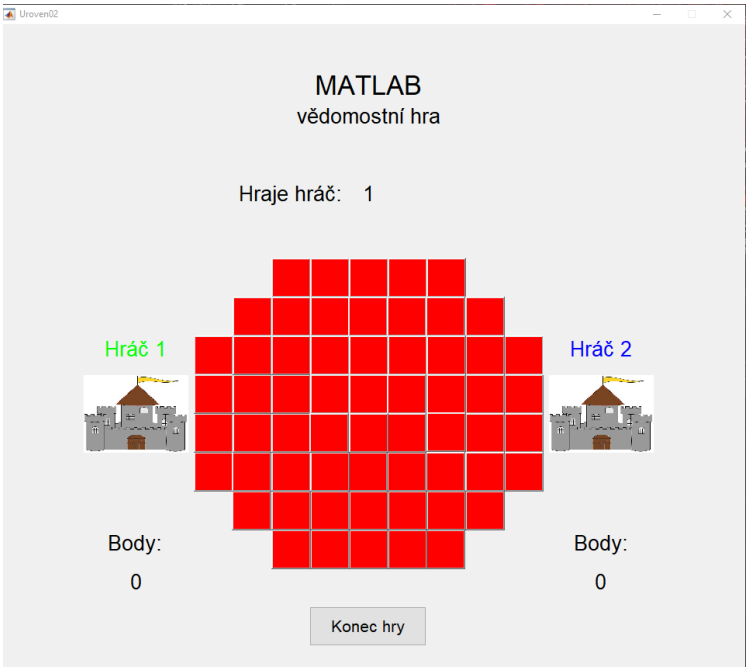


Obrázek 29 Výhra

Hráč může hru kdykoliv ukončit a vrátit se do menu pomocí tlačítka *Konec hry* nebo pomocí křížku v pravém horním okně.

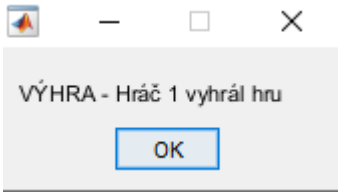
5.6 Hra pro dva hráče

Po stisknutí tlačítka *2 hráči*, se otevře okno s hrou pro dva hráče. V horní části okna je zobrazeno, který hráč je na řade. V levé části okna, jsou zobrazeny body hráče číslo jedna a v pravé části body hráče číslo dva. Hráči se střídají v obsazování červených políček. Pokud hráč 1 klikne na červené políčko a odpoví správně na otázku, tak se políčko změní na zelené. V případě že, hráč 2 odpoví správně na otázku, tak se políčko změní na modré.



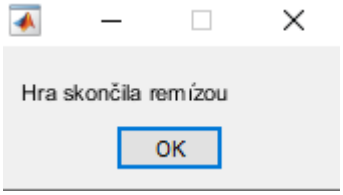
Obrázek 30 Hra pro dva hráče

Hra skončí, ve chvíli, kdy jsou všechna červená políčka zabrána. Vyhrává hráč, který má na konci hry více bodů. Pokud hráč jedna má na konci hry více bodů než hráč dvě, tak se zobrazí dialog s informací o jeho výhře, v opačném případě se objeví dialog o výhře hráče dvě.



Obrázek 31 Hráč 1 vyhrál

Pokud mají hráči na konci hry stejný počet bodů zobrazí se dialog s informací o remíze. Po stisknutí tlačítka *OK* se okno s hrou zavře a zobrazí se menu hry.



Obrázek 32 Remíza

ZÁVĚR

Cílem práce bylo vytvořit výukovou aplikaci v programu MATLAB, založenou na postupném obsazování políček okolo hradu. Hra dostala název „MATLAB – vědomostní hra“, protože obsahuje databázi otázek týkajících se programu MATLAB.

Hra je určena pro uživatele, kteří mají základní znalosti o programu MATLAB. Jelikož se otázky nachází v samostatném textovém souboru, tak je možné databázi otázek jednoduše nahradit databází otázek z jiného předmětu. Hra obsahuje celkem 120 otázek.

V teoretické části práce je popsán program MATLAB. Po přečtení teoretické části by měl mít uživatel dostatečné znalosti pro práci s MATLABem, aby sám dokázal vytvořit jednoduchou aplikaci. Dále jsou v teoretické části popsány ostatní bakalářské práce, které se zabývají programem MATLAB.

V praktické části je popsán samotný vývoj výukové aplikace. Aplikace je nejdříve popsána z pohledu programátora a následně z pohledu běžného uživatele.

Vytvořená aplikace může sloužit jako výukový materiál pro studenty, kteří si s její pomocí můžou procvičit práci s MATLABem zábavnou formou.

SEZNAM POUŽITÉ LITERATURY

[1] PERŮTKA, Karel. MATLAB - základy pro studenty automatizace a informačních technologií. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. ISBN 80-7318-355-2.

[2] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. MATLAB: tvorba uživatelských aplikací. Praha: BEN - technická literatura, 2004. ISBN 80-7300-133-0.

[3] DUŠEK, František. MATLAB a SIMULINK: úvod do používání. Pardubice: Univerzita Pardubice, 2000. ISBN 80-7194-273-1.

[4] KARBAN, Pavel. Výpočty a simulace v programech Matlab a Simulink. Brno: Computer Press, 2006. ISBN 978-80-251-1448-3.

[5] KOZÁK, Štefan a Slavomír KAJAN. Matlab - Simulink. 1. vyd. Bratislava: Slovenská technická univerzita, 1999. ISBN: 80-227-1213-2

[6] *MATLAB®* [online]. [cit. 2021-4-18]. Dostupné z: https://www.humusoft.cz/matlab/details/

[7] *What is Inkscape?* [online]. [cit. 2021-4-20]. Dostupné z: https://inkscape.org/cs/about/

[8] *Mathworks* [online]. [cit. 2021-4-20]. Dostupné z: https://www.mathworks.com/help/matlab/ref/persistent.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MATLAB	Matrix Laboratory
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH