Návrh a realizace počítačové hry v software MATLAB

Miroslav Zapletal

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:	**Miroslav Zapletal**
Osobní číslo:	**A17076**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Prezenční**
Téma práce:	**Návrh a realizace počítačové hry v software MATLAB**
Téma práce anglicky:	**Design and implementation of a computer game in MATLAB software**

Zásady pro vypracování

1. Proveďte literární průzkum z oblasti popisu software MATLAB a prací zabývajícími se 2D počítačovými hrami v MATLAB.
2. Navrhněte novou hru, sestavte vývojový diagram, vytvořte grafický návrh hry.
3. Realizujte hru, dialogy vytvořte v AppDesigner.
4. Vytvořte uživatelský manuál, nápovědu a popis funkcí.
5. Otestujte hru a dle doporučení testerů proveďte případné korekce.

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

Seznam doporučené literatury:

1. KARBAN, Pavel. Výpočty a simulace v programech Matlab a Simulink. Praha: BEN-technická literatura, 2007. ISBN: 978-80-251-1448-3.
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. KOZÁK, Štefan a Slavomír KAJAN. Matlab – Simulink. 1. vyd. Bratislava: Slovenská technická univerzita, 1999. 125 s. ISBN: 80-227-1213-2.

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

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 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 bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji 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 – 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 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis studenta

ABSTRAKT

Abstrakt česky

Bakalářská práce je věnována vývoji 2D aplikace vytvořenou ve vývojovém prostředí *MATLAB APP DESIGNER*.

Teoretická část je složena z popisu vývojového prostředí *MATLAB* a popisu v tomto programu již vytvořených 2D her, které jsou volně dostupné na *MATLAB Central*, nebo se jedná o studentské práce.

Praktická část obsahuje postup tvorby nové hry a popis z uživatelského či programátorského hlediska.

Klíčová slova: *MATLAB*, *APP DESIGNER*, 2D, Hra, Programování.

ABSTRACT

Abstrakt ve světovém jazyce

This bachelor thesis deals with the development of a 2D application created in the MATLAB APP DESIGNER development environment.

The theoretical part consists of a description of the MATLAB development environment and a description of 2D games that have already been created in this program and that are available as free downloads on MATLAB Central, or are student works.

The practical part includes a description of the process of creating a new game and its description from the user's or programmer's point of view.

Keywords: *MATLAB*, *APP DESIGNER*, 2D, Game, Programing.

Rád bych tímto poděkoval svému vedoucímu bakalářské práce, panu Ing. Karlu Perůtkovi, Ph.D., za rady, tipy, připomínky a v neposlední řadě i návrhy k funkčnosti hry samotné.

Dále bych rád poděkoval i své rodině, která mě v době mého studia podpořila a poskytla mi dostatečný prostor a podporu.

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

OBSAH

ÚVOD

Cílem této bakalářské práce je zpracování literární rešerše zaměřené na program *MATLAB*, popis již existujících her v programu *MATLAB* a vytvoření nové 2D hry v *APP DESIGNERu*. Výsledkem této práce je počítačová 2D hra *Snake Quiz*, která svým obsahem splňuje i edukační účely.

První část je zaměřena na popis vývojového prostředí *MATLAB* doplněnou o nedílné součástí obecné charakteristiky z oblastí: Druhů souborů, Typů Proměnných, Datových Typů, Operátorů, Větvení kódu, Cyklů, Matic, Vektorů a *APP DESIGNERu*. Je proveden také průzkum z oblasti již existujících 2D her, mezi něž patři: *Super Mario*, *2048*, *Tetris*, *Člověče nezlob se*.

Druhá část práce se skládá z: Popisu tvorby hry, Testování funkčnosti hry, Funkce souborů, Vývojových diagramů, Blokového schéma souborů, Popisu funkcí aplikací a Ukázek kódů. Jsou zde i uvedeny principy funkcí a princip jednotlivých databází, které se v programu objevují.

V neposlední řadě je i v závěru druhé části práce uvedena aplikace z pohledu uživatele, aby měl uživatel možnost se s hrou seznámit ještě v okamžiku, kdy nedošlo k jejímu spuštění.

I. TEORETICKÁ ČÁST

1 MATLAB

MATLAB je zkratka odvozená od dvojice slov MATrix LABoratory. K prvnímu představení tohoto programového prostředí společností The MathWorks došlo v roce 1984.

1.1 O programu

Matlab je interaktivní prostředí, které se využívá především pro zpracování vědeckých či technických výpočtů. [1]

Matlab byl zpočátku zaměřen pouze na operační systémy *OS Unix*. V dnešní době je ovšem jeho podpora mnohem pestřejší a pracuje na platformách *Microsoft Windows*, *Mac OS* a *Linux*.

Jeho jádro je složeno z více než 500 matematických funkcí, které jsou vloženy do jeho efektivního algoritmu. Poté je možnost, pomocí těchto funkcí a jejich kombinace vytvořit prakticky libovolně složitou funkci, která by řešila daný problém, na který by jednoduchá funkce nestačila. [2]

Jeho využitelnost spadá pod několik oblastí, a z tohoto důvodu jsou zde rozšíření pro jeho rozsáhlejší funkčnost. Těmto rozšířením se říká toolbox. [3]

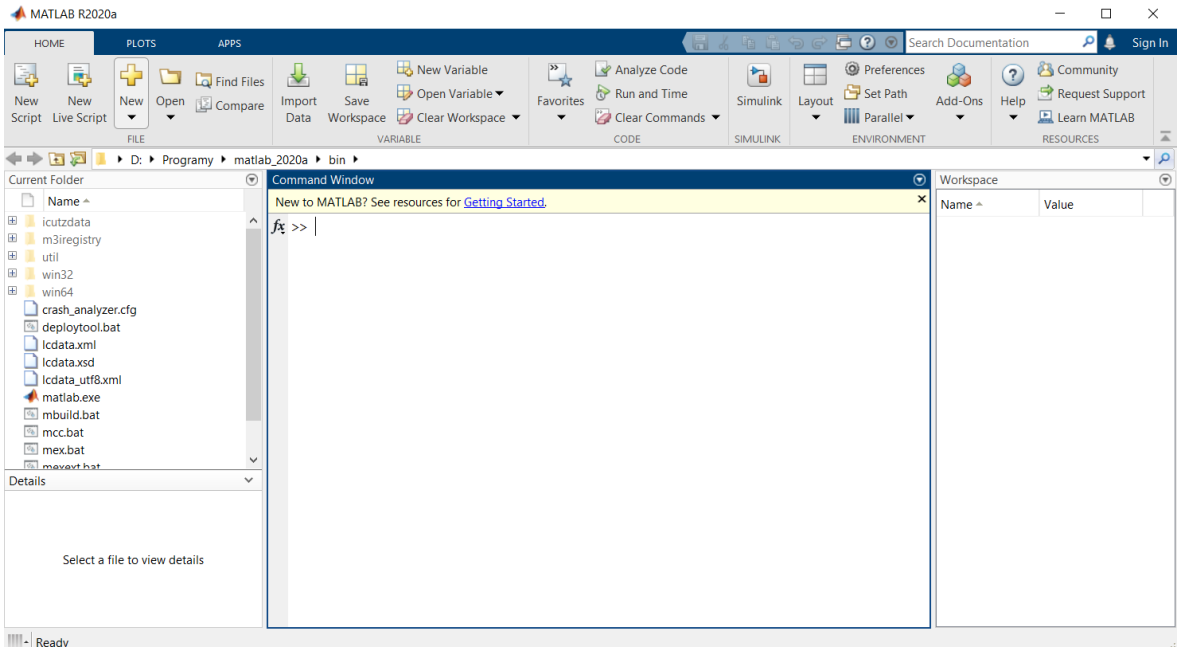
Příklady využití, pro které lze toolbox využít:

- Spreadsheet Link (for Microsoft Excel)
    - Slouží pro propojení mezi programem *Microsoft Excel* a *Matlab*. Pro běžného uživatele je zde výhodou známé uživatelské rozhraní *Microsoft Excel* s výhodou výpočetního výkonu, který *MATLAB* nabízí. [3]
- Financial Toolbox
    - Poskytuje funkce spojené s matematickým modelováním a statistikou finančních dat. Umožnuje odhadnout rizika, analyzovat výnosovou křivku a zjistit výnosnost investice. [3]
- Database Toolbox
    - Využití tohoto *toolboxu* spočívá v poskytnutí funkce k propojení mezi programem *MATLAB* a databázemi, které jsou kompatibilní s rozhraním *ODBC* a *JDBC*. [3]

Zde jsem uvedl jen malou část z jinak široké škály *toolboxu*. Tyto *toolboxy* jsou zveřejněné přímo na stránkách *Mathworks*.

1.2 Popis prostředí

Po spuštění programu *MATLAB* dojde k zobrazení dialogového okna vyobrazeného na Obr.1.



Obr. 1 Aplikace *MATLAB* po spuštění

V grafickém rozhrání MATLAB nalezneme tyto prvky:

1.2.1 Menu

Menu je po spuštění složeno ze tří záložek (*HOME*, *PLOTS*, *APPS*). Pokud bude kód editován, tak se zobrazí další tři funkcionální záložky (*EDITOR*, *PUBLISH*, *VIEW*) sloužící pro editaci kódu.

1.2.1.1 Home

Tato záložka slouží pro ovládání a nastavení MATLAB.

FILE

Tato kategorie slouží pro otevření, vytvoření, hledání a porovnávání souborů. Také nabídne možnost vytvořit skript.

VARIABLE

Umožňuje otevřít, vymazat, a smazat proměnné. Můžeme též i importovat data nebo uložit aktuální pracovní prostor.

CODE

Zde jsou možnosti, které se týkají spuštění, smazání a kontroly syntaxe samotného kódu.

SIMULINK

Skládá se pouze z jednoho rozšiřujícího prvku *SIMULINK*.

ENVIRONMENT

Zde se nachází možnost pro změnu vzhledu, rozložení a celkového nastavení *MATLAB*.

RESOURCES

Zde, je možné naleznout nápovědu ve formě off-line dokumentace a online spojení s rozsáhlou uživatelskou komunitou.

1.2.1.2 Plots

V záložce najdeme možnosti grafického zobrazení jedné či více proměnných.

SELECTION

V tomto okně dochází k selekci námi vykreslované proměnné

PLOTS

K selektované proměnné nabídne pestrý výběr grafů, které v rámci dané proměnné jdou vykreslit.

OPTIONS

Slouží k přizpůsobení zobrazení nového grafu a to tak, že se znázorní do nového nebo stávajícího okna.

1.2.1.3 Apps

S touto záložkou jsou spjaty pouze 2 podzáložky sloužící k využívání dodatečných aplikací.

FILE

Zde je možnost k nalezení a nainstalování nové aplikace. Také je možnost si aplikaci vytvořit dle vlastních preferencí.

APPS

Slouží k zobrazování námi nainstalovaných aplikací.

1.2.2 Command Window

V tomto okně je přehled použitých příkazů a zobrazování výsledků.

1.2.3 Workspace

Obsahuje všechny dostupné proměnné, které jsou vytvořené nebo vložené z datových souborů. Obsah pracovního prostoru je možné zobrazit, upravovat nebo celý smazat.

1.2.4 Current folder

Jedná se o aktuálně vybraný adresář, se kterým program pracuje. Soubory a složky zde obsažené můžeme upravovat, mazat, přesouvat a přejmenovávat.

1.2.5 Details

Zobrazuje všechny podrobnosti o souboru v aktuálním adresáři.

1.3 Druhy soborů

V *MATLAB* je možné se setkat s několika typy soborů, které se od sebe liší příponou a využitím. Níže jsou uvedeny nejčastěji se objevující typy.

1.3.1 Soubory typu FIG

Tento typ souboru se z uživatelského hlediska využívá v případě vytváření grafického uživatelského prostředí, protože je nositelem informace o obsahu a parametrech *GUI*.

1.3.2 Soubory typu M

Pokud bude docházet k sekvenčnímu používání řady příkazů pomocí *Command Window*, pak právě pomocí M souborů můžeme zvýšit efektivitu naší práce. Do souboru si zapíšeme posloupnost používaných příkazů případně i část programu a po uložení tento nově vytvořený soubor stačí jeho jménem pomocí *Command Window* zavolat a dojde k provedení předpřipravené sady instrukcí, které *MATLAB* vykoná (je ovšem potřeba, aby *Active directory*, bylo nastaveno v místě umístění souboru). [4]

K vytvoření M souboru se používá *MATLAB editor*, ale je možnost použít i jiný alternativní textový editor.

Tyto soubory je možné zapisovat těmito způsoby:

1.3.2.1 Skripty

Skriptové soubory jsou programové soubory typu M. Jedná se o řadu příkazů, které se spouští současně. Pracují s daty v pracovním prostoru, a proto nemají žádný výstup ani vstup.

1.3.2.2 Funkce

Soubory funkce jsou také programové soubory typu M jen s rozdílem, že umožnují přijímat vstupy a vracet výstupy.

1.3.3 Soubory typu MAT

Jsou to soubory binárního typu, využívající se pro ukládání proměnných uživatelského prostoru. Používá se dvojice příkazů pro zápis *save* a pro čtení *load*.

1.4 Proměnné

Oproti jiným programovacím jazykům není v *MATLABu* zapotřebí deklarovat či jinak definovat proměnné. Při jakémkoliv výskytu nové proměnné v *Command Window* dojde k dynamickému typování. Po vložení první hodnoty do nově vytvořené proměnné se v ten moment přiřadí podle obsahu i datový typ. [5]

1.4.1 Lokální proměnné

Jedná se o výchozí typ proměnných vytvořených v *Command Window*. Poté se nacházejí výhradně ve *Workspace*. V případě použití funkcí je dáno, že každá funkce má možnost pracovat pouze se svými lokálními proměnnými.

1.4.2 Globální proměnné

V tomto případě je na rozdíl od lokálních proměnných potřeba danou globální proměnnou definovat pomocí příkazu *global name*, kde *name* je název globální proměnné pro daný *Workspace*. Díky globálním proměnným je možnost při použití funkcí pracovat s jednou proměnnou ve více funkcích.

Pokud bude potřeba danou globální proměnnou odstranit, tak použijeme příkaz *clear name*.

1.5 Datové typy

MATLAB umožňuje použití patnácti základních datových typů, které je následně možné zapsat formou pole nebo matice.

1.5.1 Číselné datové typy

Všechny numerické datové typy *MATLAB* definuje jako hodnoty typu *double* tedy s přesností na setiny. [6]

1.5.1.1 Single

Single je desetinný datový typ, který v paměti zabírá polovinu místa oproti double a to konkrétně 4 byty. Má poloviční přesnost než double, což přinese zrychlení výpočetních procesů ovšem na úkor přesnosti. [6]

1.5.1.2 Double

Double je výchozím datovým typem *MATLABu*, který pracuje s přesností dvou desetinných míst. Datový typ double si v paměti vyhradí 8 bytů. [6]

1.5.1.3 Int8, Int16, Int32, Int64

Tento datový typ je předurčen pro celočíselné hodnoty, u kterých je potřeba zachovat informaci, zda se jedná o kladné či záporné číslo [6]. Rozsah hodnot je dán podle jeho označení, např. *int16* má rozsah od hodnoty -32768 až po +32767.

1.5.1.4 uint8, uint16, uint32, uint64

Datový typ *uint* je určen pro všechny celočíselné hodnoty s rozsahem, který je dán svým označením, např. uint16 má rozsah od hodnoty 0 až po 65535. U tohoto datového typu není brán zřetel na znaménko, a proto bude vždy hodnota kladná. [6]

1.5.2 Datový typ CHAR

Jedná se o textový datový typ, který se využívá pro řetězce a znaky. Jsou v proměnné uloženy uvnitř apostrofů. V paměti je ovšem tento znak reprezentován jeho hodnotou ASCII tabulky. [6]

1.5.3 Datový typ STRUCT

Tento datový typ *struct* je specifický svým užitím. Je využíván ve chvíli, kdy je potřeba seskupit data skládající se ze dvou nebo více datových typů. K seskupování těchto dat je využíváno polí a k jejich přístupu je potřeba využít tečkovou notaci. [6]

1.5.4 Datový typ CELL

Jedná se o datový typ s indexovanými datovými kontejnery nazývanými *cells*. Jedná se tedy o vektor či matici. Pole buněk může obsahovat text, kombinaci textu a čísel, nebo i samotné číselné pole o různých velikostech. [6]

1.5.5 Datový typ HANDLE FUNKCE

Handle funkce slouží pro předání informace o funkci, díky čemuž je možné přistupovat k jedné funkci pomocí jiné funkce. [6]

1.6 Operátory

V *MATLABu* lze operátory rozdělit do tří kategorií.

1.6.1 Aritmetické operátory

Tyto operátory slouží především pro aritmetické výpočty daných proměnných. V Tab.1 jsou uvedeny základní používané aritmetické operátory.

Operátor	Význam
+	Součet
-	Rozdíl
*	Součin
/	Podíl

Tab.1. Aritmetické operátory

Součet dvou celých čísel pomocí aritmetického operátoru je vyobrazen na Obr.2.

```
>> a=2,b=3,vysledek=a+b

a =

     2


b =

     3


vysledek =

     5
```

Obr.2 Součet dvou čísel

1.6.2 Relační operátory

Relační operátory se využívají v případech, kdy je potřeba porovnat konkrétní prvky. [7] Výstupem tohoto porovnání relačními operátory bude buď pravda (*true*) nebo nepravda (*false*). Základní relační operátory jsou uvedeny v Tab.2.

Operátor	Význam
==	rovnost
>=	větší nebo rovno než
>	větší než
<=	menší nebo rovno než
<	menší než
~=	Nerovná se

Tab.2. Relační operátory

1.6.3 Logické operátory

V *MATLABu* je možnost použití 5 předdefinovaných základních logických operátorů, které využívají logických hodnot *true* nebo *false*. Tyto Logické operátory jsou uvedeny v Tab.3.

Operátor	Význam
&	Logický součet pro pole
&&	Logický součet pro podmínku
|	Logický součin pro pole
||	Logický součin pro podmínku
~	Negace

Tab.3. Logické operátory

1.7 Větvení kódu

Při programování může dojít k situaci, kdy bude potřeba na základě aktuálního stavu reagovat na nejrůznější situace. Na tyto situace je možné reagovat právě pomocí podmínky a přepínače.

1.7.1 Podmínka if, else, elseif

Podmínka *if* slouží k posuzování logického stavu jedné či více proměnných pomocí logických operátorů tak, aby bylo možné program větvit na základě tohoto stavu.

V případě použití podmínky *if* je nutnost uvést podmínku, která se skládá z relačního či logického operátoru. Tato podmínka při vyhodnocení získá hodnotu *true* nebo *false*. V případě vyhodnocení podmínky jako *false* bude program pokračovat dále v podmínce a bude vykonávat ty příkazy, které budou obsaženy v pravdivé podmínce. Pokud ani jedna podmínka nebude vyhodnocena kladně, program vykoná soubor příkazů obsažených v části podmínky *else*. Jelikož uvnitř podmínky může být nespočet příkazů, je potřeba za posledním z nich uvést konec celé podmínky pomocí *end*. Po splnění podmínky přestane program vyhodnocovat ostatní podmínky a bude pokračovat od příkazu *end*. [8]

```
>> if a==b
    disp('A a B mají stejnou hodnotu');;
elseif a>b
    disp('A má větší hodnotu než B');;
else
    disp('B má větší hodnotu než A');;
end
B má větší hodnotu než A
```

Obr.3 Podmínka *if*

Na Obr.3 dojde k porovnání hodnot *a, b* pomocí podmínky *if*. Po vyhodnocení je výsledek pomocí *disp* vyobrazen přímo do *Command Window*.

1.7.2 Switch

Stejně jako *if* slouží k větvení programu. *Case* vyhodnocuje jednu proměnnou, která se nachází přímo za deklarací *switch*. Takto zadaná proměnná je následně porovnávána s *case*, které jsou součástí *switch*. Pokud tedy je hodnota vstupní proměnné shodná s hodnotou *switch*, dojde k vykonání příkazů uvedených pro daný *case*. V opačném případě, kdy nedojde k nalezení příslušné case tak jsou vykonány příkazy uvnitř *otherwise*. *Otherwise* je tedy univerzální část *case*, která se využívá v případě nenalezení dané hledané hodnoty. Není ovšem podmínkou ke správné funkčnosti, aby *otherwise* byla použita.[9]

Příklad uvedený na Obr.4 porovnává hodnoty *a,b*. V jednotlivých *case* jsou následně pomocí *disp* vyobrazeny výsledky přímo do *Command Window*.

```
>> switch a
case a==b
disp('A a B mají stejnou hodnotu');
case a>b
disp('A má větší hodnotu než B');
otherwise
disp('B má větší hodnotu než A');
end
B má větší hodnotu než A
```

Obr.4 Podmínka *switch*

1.8 Cykly

V *MATLABu* jsou dva druhy cyklů. Tyto cykly mají odlišný způsob použití.

1.8.1 Cyklus while

Cyklus *while* se používá v případě, že dané příkazy uvedené v cyklu budeme chtít použít více než jednou, ovšem nebude předem jasné, jaký bude počet opakování, ale bude známá podmínka, pro kterou se bude vykonávat. Této podmínce se říká řídící podmínka cyklu. Jeho užití může být zkombinováno spolu s použitím podmínky *if*. Ukončení cyklu se provádí pomocí *end* je i možnost cyklus předčasně ukončit pomocí *break* [10, 11].

Ukázka na Obr.5 slouží k transformaci řádkové matice na matici sloupcovou pomocí cyklu while a podmínky *if*. Tyto prvky následně vypíše do *Command Window*.

```
>> a = [1 2 3 4 5];
    i = 1;
while true
    if i==length(a)
        disp(a(i));
        break;
    else
        disp(a(i));
        i=i+1;
    end
end
     1

     2

     3

     4

     5
```

Obr.5 Cyklus *while*

1.8.2 Cyklus for

Cyklus *for* se využívá v situaci, kdy přesně víme, kolikrát se budou dané příkazy opakovat. Analogicky stejně jako u cyklu *while* je možné vyvolat předčasné ukončení pomocí *break*. [11, 12]

Na Obr.6 je uveden příklad cyklu, který vypíše všechny prvky matice, pomocí *disp* přímo do *Command Window*. Délka samotné matice je určena pomocí příkazu *length*.

```
>> a = [1 2 3 4 5];
for i = 1:length(a)
       disp(a(i));
end
     1

     2

     3

     4

     5
```

Obr.6 Cyklus *for*

1.9 Matice a vektory

Práce s vektory a maticemi jsou jedny z hlavních důvodů používání *MATLABu*, protože se jedná o nezbytné funkce pro výpočetní operace.

Matice jsou obdélníková či čtvercová pole reálných či komplexních čísel. Naproti nim je skalár, který je speciální typ matice o rozměru 1x1. Skalár tedy obsahuje pouze jedem prvek. Vektory jsou dvojího typu, a to sloupcové, popřípadě řádkové. [13]

Obecně je matice značena jako počet sloupců *n* a počet řádků *m*. Po vytvoření a uložení jednoho prvku do matice bude její velikost 1x1. [13]

1.9.1 Vektory

K vytvoření vektoru slouží hranaté závorky, ve kterých jsou následně uvozeny jednotlivé elementy, které jsou od sebe odděleny čárkou nebo mezerou. [14]

Vytvoření řádkového vektoru o 6 prvcích:

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

Pokud bude potřeba vypsat daný prvek vektoru, pak využijeme index prvku uvedený ve dvojici závorek.

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

1.9.2 Matice

Matice se vytváří analogicky k vektorům. Jediným rozdílem je potřeba nadefinovat i další prvky, které budou na jiném indexu řádku. Při zápisu je tedy potřeba využít opět hranaté závorky, do kterých zapíšeme jednotlivé elementy oddělené mezerou nebo čárkou a další datové elementy zapíšeme po uvedení středníku. [14]

Vytvoření matice 3x3:

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

Oproti vektoru je potřeba při vypisování prvku navíc uvést další údaj. Zápis tedy bude index řádku a index sloupce oddělený čárkou. Prvek vyskytující se na druhém řádku třetího sloupce bude zobrazen takto:

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

V případě potřeby vypsání celého druhého řádku, bude formální zápis:

>> matice(2,:)

MATLAB též umožňuje automatizované vytvoření matic pomocí speciálních funkcí. Zde jsou uvedeny některé příkazy pro jejich zavedení:

zeros(m, n)-slouží pro vytvoření matice, které bude mít obsah složený pouze z nul. [15]

ones(m, n)-po zadání příkazu se vytvoří matice, které bude obsahovat pouze jedničky. [15]

rand(n,m)-vytvoří se matice, obsahující náhodná čísla v intervalu od 0 do 1. [15]

magic(n)-jedná se o čtvercovou matici, která má součet všech hodnot na řádcích, diagonálách a sloupcích stejný. [15]

1.9.3 Operace

V jiných programovacích jazycích byla nutnost, při práci s maticemi a vektory používat navíc i cykly. Jelikož má *MATLAB* integrovaný nespočet funkcí pro práci s poli umožňuje práci s nimi i bez použití cyklů. [11]

Máme zde tedy možnost na matice aplikovat dostupné aritmetické operace, vytvořit inverzní matici a vytvořit transponovanou matici.

Pro vytvoření rozdílů dvou matic, využijeme tento modelový zápis, který má stejný zápis, jako v případě dvou obyčejných proměnných:

>> matice = matice1 - matice2

K určení hodnot mezi dvojicí matic po násobení stačí použít opět aritmetický operátor při tomto zápisu:

>> matice = matice1 * matice2

Na Obr.7 je příklad vytvoření matice s následným přičtením hodnoty pět ke všem prvkům.

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

a =

     1     2     3
     4     5     6
     7     8     9


ans =

     6     7     8
     9    10    11
    12    13    14
```

Obr.7 Práce s maticí

Případnou tvorbu posloupností s konstantním krokem nám umožní operátor *:*. Jeho zápis pro výpis prvků od jedné do osmi s krokem jedna bude vypadat takto:

```
>>Vektor=1:1:8;
```

Pokud chceme vytvořit transponovanou matici tak stačí za název proměnné uvést apostrof:

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

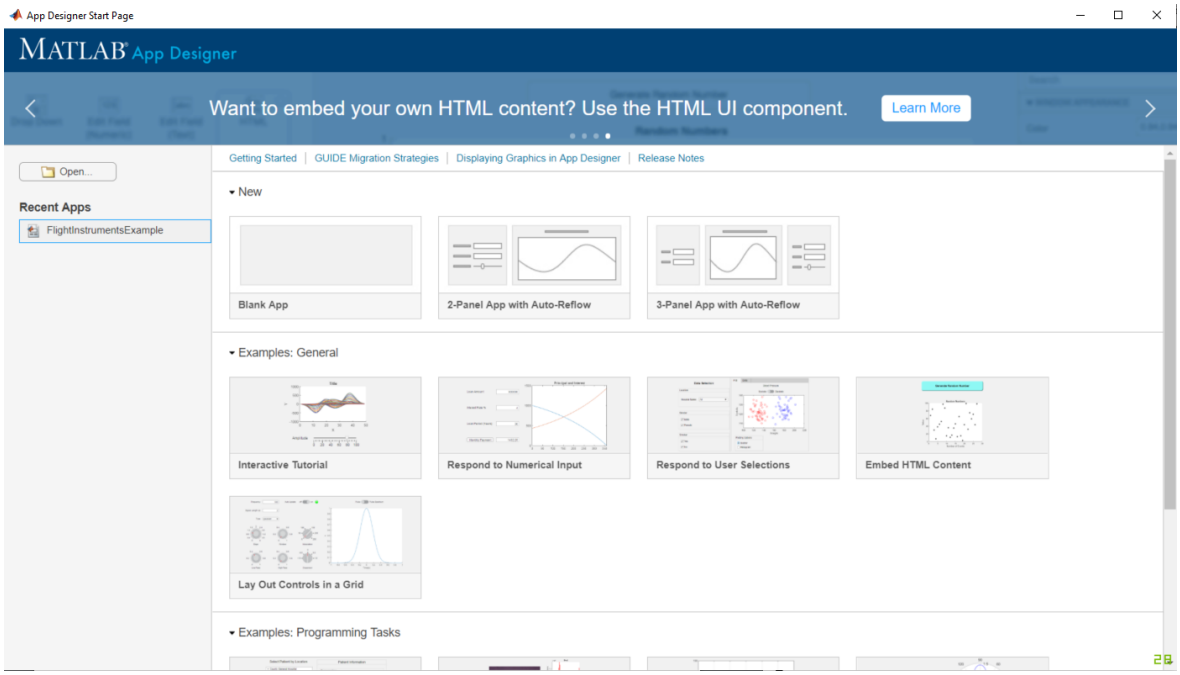
1.10 App designer

MATLAB App Designer je poměrně nová funkce vývojového prostředí *MATLAB*, která se poprvé objevila v roce 2016. Jedná se o nástupce Grafického uživatelského prostředí. *GUI* je i v současné verzi MATLAB (R2020b) stále dostupné a je možné jej zavolat pomocí příkazu *guide*. *App Designer* nabízí nespočet nových funkcí (např. vlastní ikony, tutoriál pro vytvoření nové aplikace, sdílení webových aplikací), moderní design a pro začátečníky je mnohem přívětivější než jeho předchůdce.[16]

Jedná se o interaktivní vývojové prostředí určené pro navrhování grafického rozložení aplikací a v neposlední řadě také k samotnému funkcionálnímu programování nově vznikající aplikace.[16]

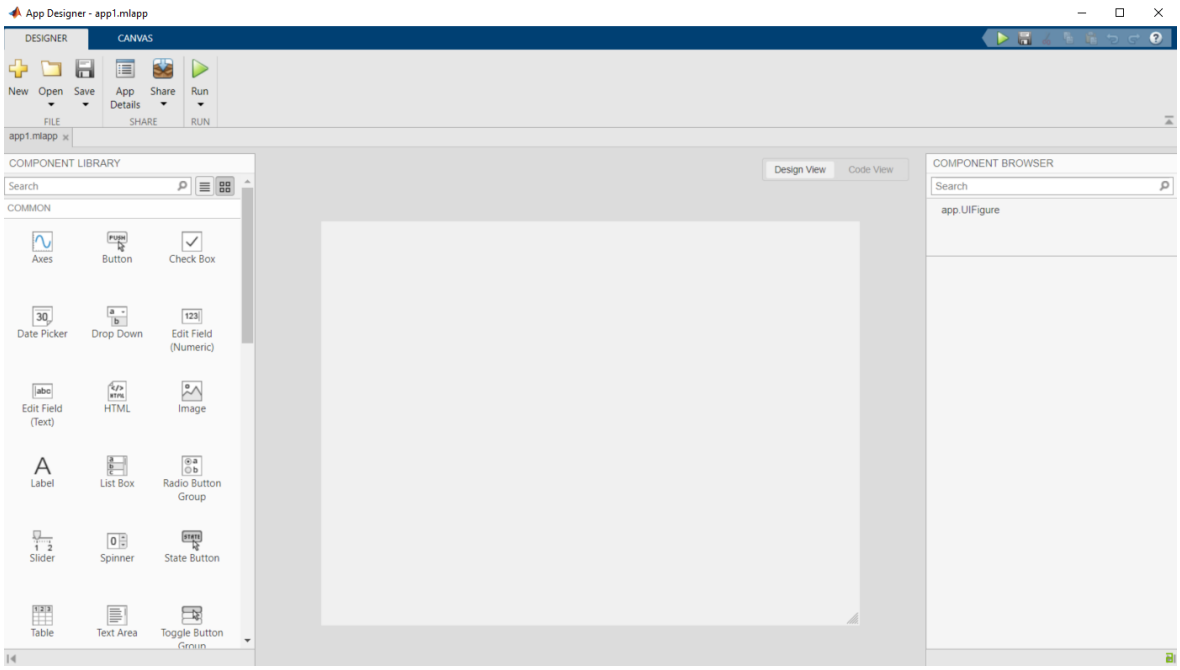
App Designer je možné spustit pomocí zadání příkazu *appdesigner* do příkazové konzole *MATLABu*, nebo pomocí hlavního menu v záložce *Home-> New-> App*.

Po spuštění grafického okna viz Obr.8, které povede k vytvoření nové aplikace dojde k zobrazení základních grafických rozložení pro její návrh, příklady již vytvořených aplikací a interaktivní příklady sloužící k pochopení funkčnosti samotného *App Designer*.



Obr.8 Průvodce *App Designer*

Po výběru prázdné aplikace se zobrazí pracovní oblast pro tvorbu aplikace vyobrazené na Obr.9, která je rozdělena na několik částí.



Obr.9 Nově vytvořená aplikace v *App Designer*

Jeho součástí je i *MATLAB* editor a poskytuje velkou škálu komponent, které je možné využít pro tvorbu nově vznikající aplikace. Mezi jeho další přednosti patří také správce rozložení komponent, umožňující automaticky reagovat na změnu velikosti okna aplikace, podle které dojde k přeformátování. Nachází se zde i možnost distribuování aplikací ve formě

archivovaných instalačních souborů a v případě použití *MATLAB Compiler* se možnosti rozšíří navíc i o vytvoření desktopových nebo webových aplikací. [16]

1.10.1 Menu

Skládá se ze dvou záložek pro grafický návrh aplikace a dvou záložek pro programovou část návrhu aplikace.

V záložce Designer, která je společná pro grafickou a programovou část se nachází tři kategorie (*FILE, SHARE, RUN*), ve kterých jsou rozděleny jednotlivé nástroje pro práci s programovým prostředím. V části *FILE* je možné provádět základní úkony spojené s vytvořením, načtením a uložením nově vznikající aplikace. Pokud bude potřeba dosavadní práci sdílet, tak se přesuneme do *SHARE*, kde se nachází dvě možnosti. V *App details* jsou všechny informace o dané aplikaci a autor sám, může přidat stručný popisek, který si druhá strana zobrazí při otevření a v stejnojmenné možnosti *Share* jsou všechny možnosti distribuce aplikace. Kategorie *Run* slouží ke spuštění aplikace.

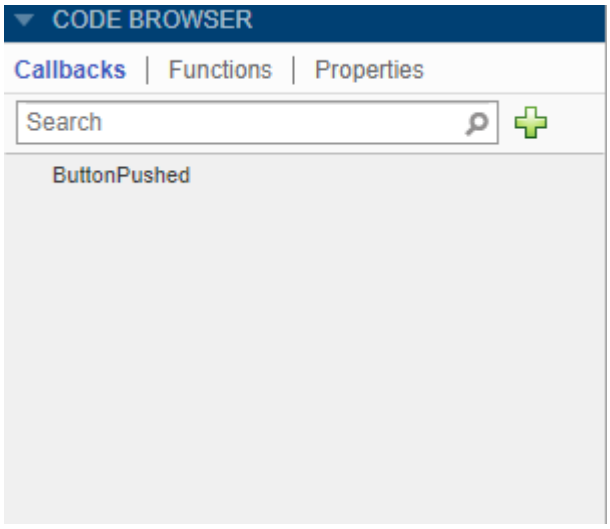
Po přesunutí do další části menu, a to konkrétně *CANVAS*, jak už název napovídá jedná se o nástroje sloužící pro práci s grafickým uživatelským prostředím navrhované aplikace. Nachází se zde 6 kategorií (*FILE, ALIGN, ARRANGE, SPACE, VIEW, RUN*). *File* slouží k základním činnostem, které jsou spojené s vytvořenou prací. Nalezneme zde tedy možnost uložení souboru, případně uložení jako M-FILE a též i dodatečné nastavení zobrazení dvou nebo tří panelů grafického uživatelského prostředí. K nastavení ohraničení konkrétního prvku, slouží *Align*. K upravování velikostí jednotlivých objektů poté využijeme možností kategorie *ARRANGE*. *SPACE* můžeme využít k formátování prostoru, mezi jednotlivými prvky a *VIEW* nám umožní pracovat s dostatečnou přesností na pracovní ploše, díky možnosti zobrazení pomocné mřížky, organizéru umístění prvku na mřížce a nápovědy při změnách rozměru prvku. Samotné *Run* slouží poté už ke spuštění aplikace.

V menu editace programového kódu aplikace se nachází záložka Editor obsahující 7 kategorií, které uživateli umožní snadnější způsob psaní kódu. Nalezneme zde záložku *INSERT*, umožňující přidání nových funkčních bloků, volání jednotlivých funkcí i přidání vstupního argumentu. *NAVIGATE* umožňuje snadné prohledávání programu, dle zadaných kritérii. Záložka *EDIT* obsahuje možnost komentování, popř. zrušení tohoto komentáře a nastavení odsazování jednotlivých textových úseků. *VIEW* nabízí pouze možnost pro zobrazování chybových hlášek kódu. Pokud budeme potřebovat zobrazit tipy tak využijeme záložku

RESOURCES. Funkčnost *FILE* a *RUN* zůstává stejná jako v případě grafického uživatelského prostředí.

1.10.2 Code Browser

Tato karta viz Obr.10 se skládá ze tří záložek. V první z nich nalezneme možnost, které umožní přidat, smazat a přejmenovávat volanou funkci. Po zvolení přednastaveného prvku v *Callback* případně *Functions*, dojde k přesunutí se v kódu na pozici, kde se daná funkce nachází. Stejnou funkčnost obsahuje i záložka *Functions* jen s rozdílem, že se vztahuje na pomocné funkce. *Properties* poté umožňuje nastavení vlastních vlastností pro danou aplikaci.



Obr.10 Příklad funkce stisknutého tlačítka v *Code Browser*

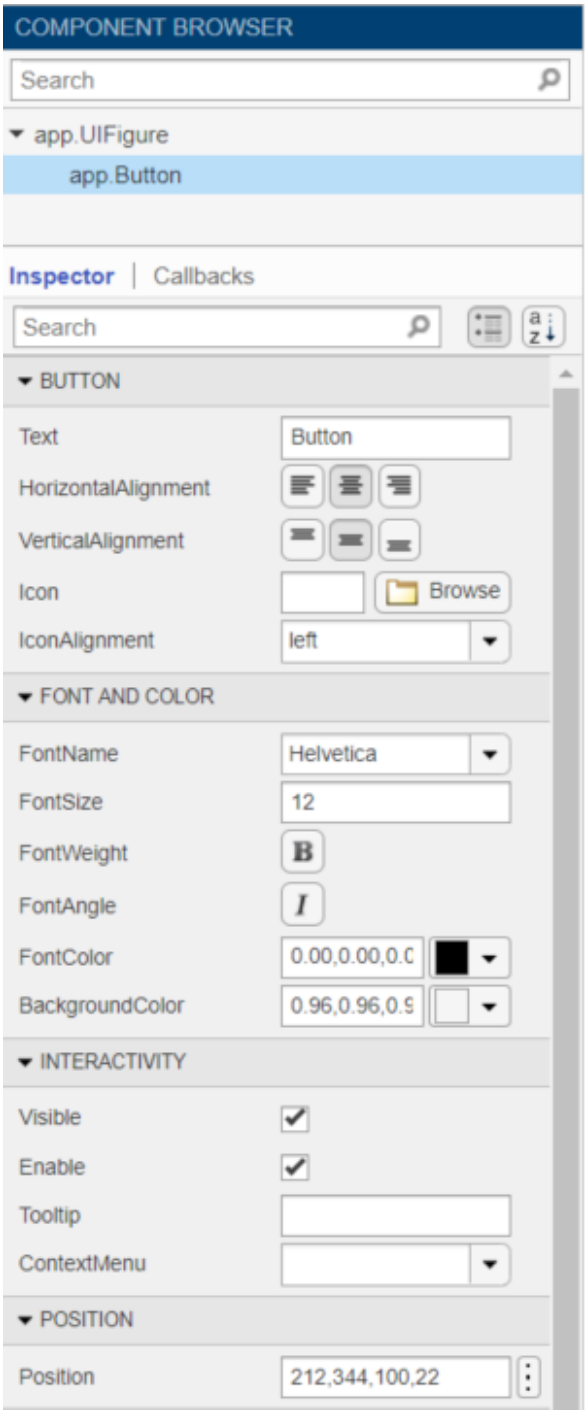
Samotný zdrojový kód aplikace s vytvořenou funkcí, která se bude vykonávat po stisknutí tlačítka je vyobrazen na Obr.11.

```
classdef app1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure  matlab.ui.Figure
        Button    matlab.ui.control.Button
    end


    methods (Access = private)

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
            app.Button.Position = [271 297 100 22];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
```

Obr.11 Příklad zdrojového kódu aplikace

1.10.3 Component Browser

Skládá se z *Context Menu*, které obsahuje soubor všech umístěných komponent dané aplikace. Jednotlivě je možné tyto prvky smazat či přejmenovat. Záložka *Inspector* zobrazuje všechny vlastnosti pro aktuálně vybraný prvek, pro který je poté možné tyto preference pozměnit. [17] Pro názornou ukázku vzhledu této karty slouží Obr.12.



Obr.12 Příklad *Property Inspector* panelu pro tlačítko

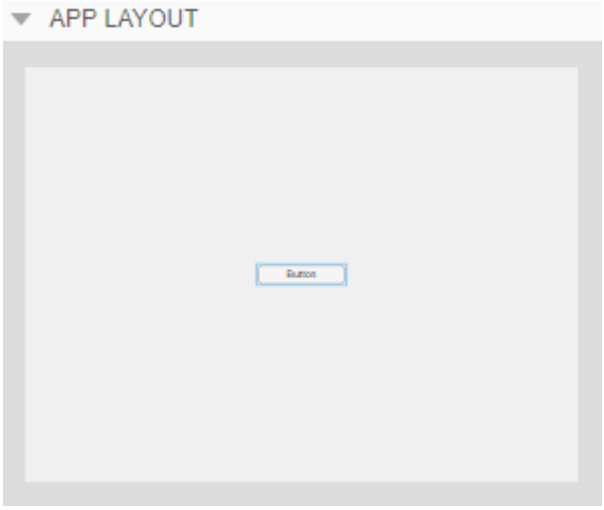
1.10.4 Component Library

Obsahuje nespočet komponentů, které můžeme využít při tvorbě aplikací. Tyto grafické prvky jsou přehledně uspořádané do jednotlivých kategorií. Jsou zde například přepínače, tlačítka, zaškrtávací políčka, měřidla, zadávací pole, posuvné i otočné potenciometry. Díky

široké škále prvků nebude problém tedy vytvořit aplikaci, která bude svou funkčností a vzhledem připomínat měřící panel podobný laboratorní úloze. [17]

1.10.5 App Layout

Jedná se o miniaturu aplikace, kterou je možné nalézt v programové části aplikace. Po zvolení vybraného prvku aplikace v *App Layout* dojde ke zvýraznění tohoto prvku v *Code Browser*. Jedná se o praktickou vlastnost, která v případě rozsáhlého projektu dokáže ušetřit spoustu času při vyhledávání daného prvku v kódu. [17]



Obr.13 Příklad *App Layout* panelu pro tlačítko

2 POČÍTAČOVÉ HRY V MATLABU

V rámci této bakalářské práce, bylo potřeba prozkoumat již vytvořené hry, a to v rámci studentských či volně přístupných prací z *MATLAB Central*.

2.1 Super Mario

Tato hra je dokonalou napodobeninou hry *Super Mario*, bohužel tento projekt zůstal ve stádiu Demo verze. Autorem je Mingjing Zhang, který tuto hru představil veřejnosti v roce 2013 a provedl téhož roku i jednu aktualizaci s rozšířením o melodie. [18]

Tato hra oproti originální hře poskytuje hráči pouze možnost procházení se po herní oblasti, poskytuje i originální melodii a totožné animace. Figurka se ovládá pomocí klávesnice ve směru doleva či doprava je zde i umožněno vyskočit a utíkat. Bohužel není možnost interagovat s objekty, nepřáteli a případnou mincí. Hra nemá žádný konec, tudíž po dosažení konce mapy je potřeba hru manuálně vypnout.

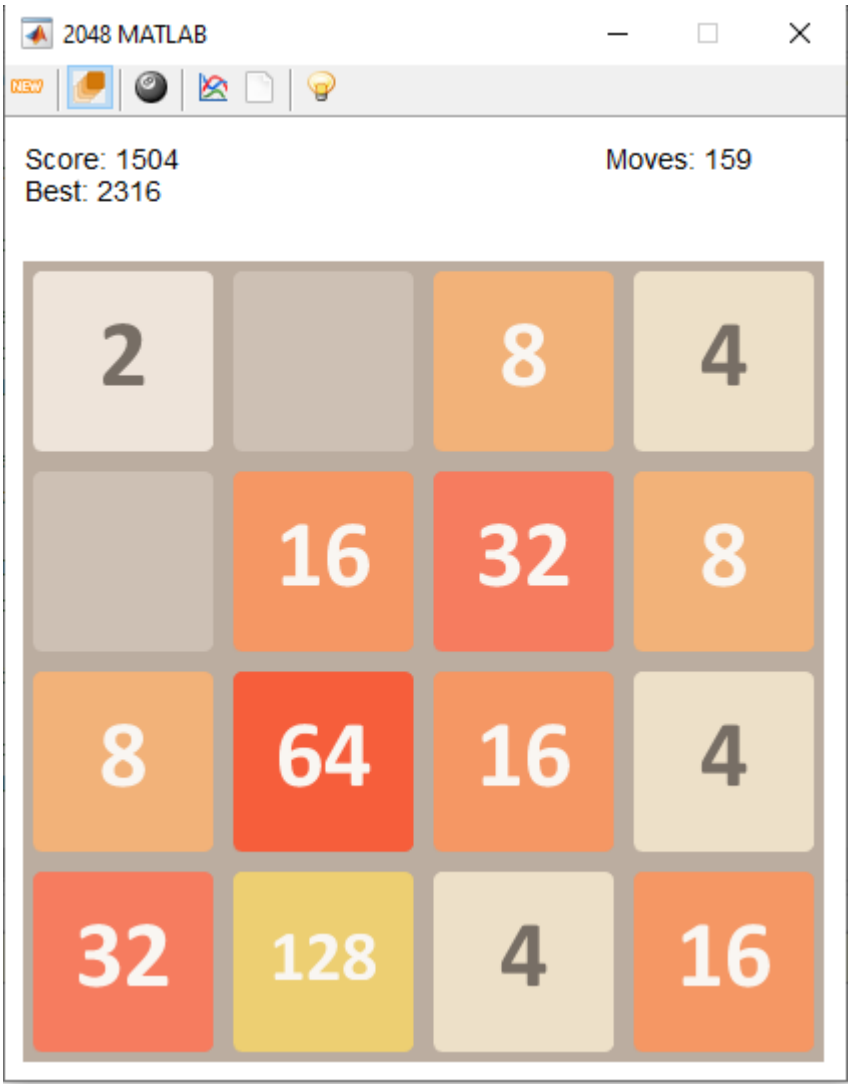
Obr.14 *Super Mario* [18]

2.2 2048

Autorem hry *2048* je Gabriele Cirulli, který hru představil začátkem roku 2014. Tato hra byla vytvořena v programovacím jazyku JavaScript.

Hra *2048* vytvořená v programu *MATLAB*, byla uvolněna vývojářem s přezdívkou jiro, který ji vydal o necelé dva roky později, a to v roce 2016. [19]

Cílem hry je dostat v matici 4x4 právě jednu buňku, která bude obsahovat číslo 2048. Toho se dá dosáhnout součtem dvou vedlejších buněk, ovšem stejné hodnoty. K pohybu těchto dlaždic v matici využíváme směrové klávesy klávesnice. Při stisku klávesy se políčka vydají tím směrem, jaká směrová klávesa byla zvolena. K ukončení pohybu dochází ve chvíli, kdy se jednotlivá políčka zastaví o stěnu či jiné políčko. Po tomto pohybu navíc přibude do hracího pole buňka s nejnižší hodnotou, a to 2.

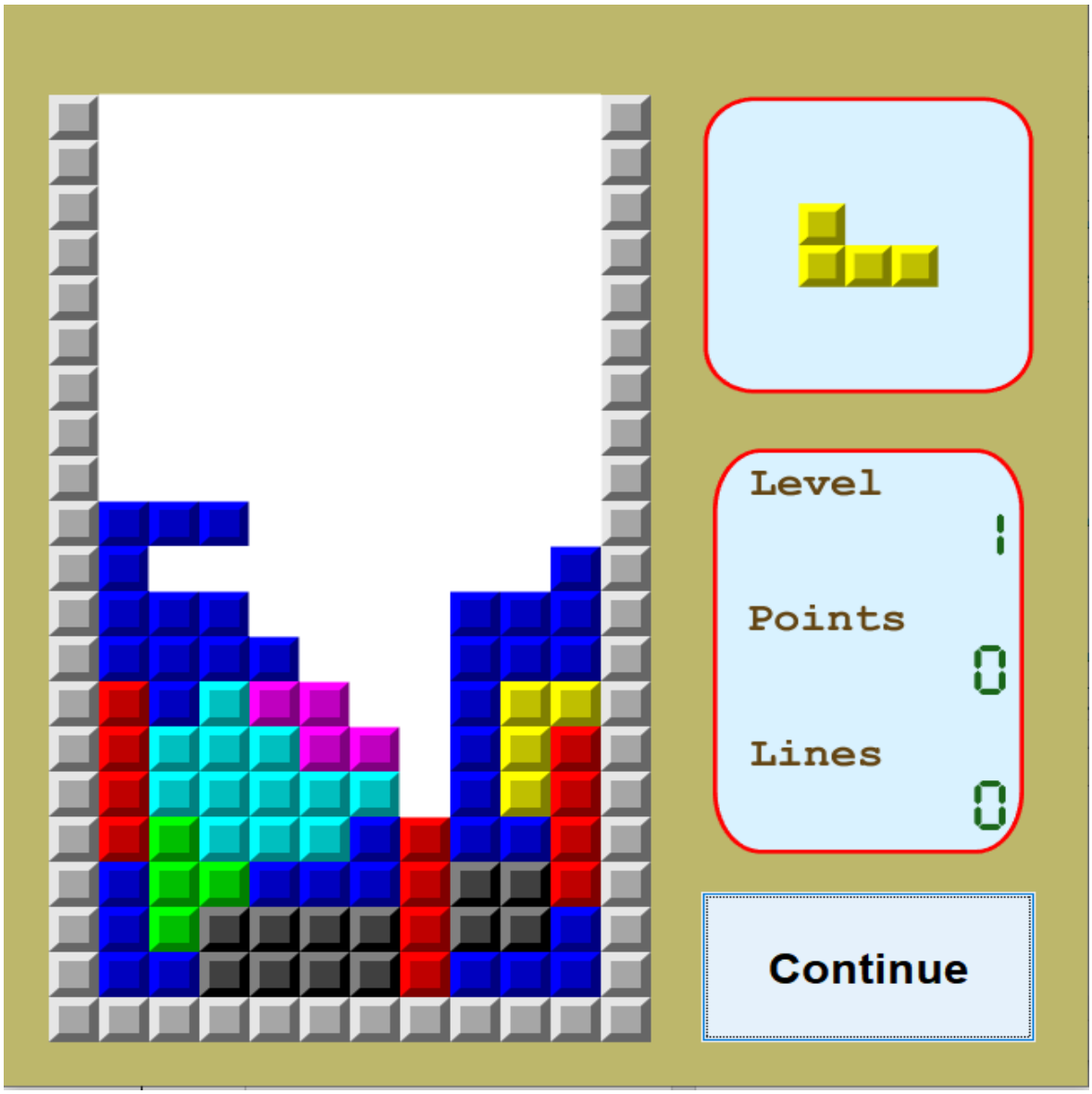


Obr.15 *2048* [19]

2.3 Tetris

Tuto hru zprostředkoval komunitě *MATLAB* uživatel Matt Fig v roce 2012.

Cílem hry je dosáhnout co nejvyššího skóre získávaného zaplněním jedné či více řad náhodně padajících těles. V případě, kdy k tomuto zaplnění dojde, hráči se připíše skóre a zmizí tyto zaplněné řady z hrací plochy. Vše nad touto právě zmizelou řadou se přesune na spodní místo a proces je nutné opakovat. S postupně se zvyšujícím skóre se hráči bude zvyšovat i intenzita rychlosti padajících těles. Hra bude končit v případě, kdy hráč nebude schopný okamžitě reagovat a umisťovat padající tělesa, což povede k postupnému dorůstání herních objektů k horní ose. V případě, kdy se těleso dotkne této osy, dojde k ukončení hry. Hra je hezky zpracována a je obsažena pouze v jednom M-souboru. [20]



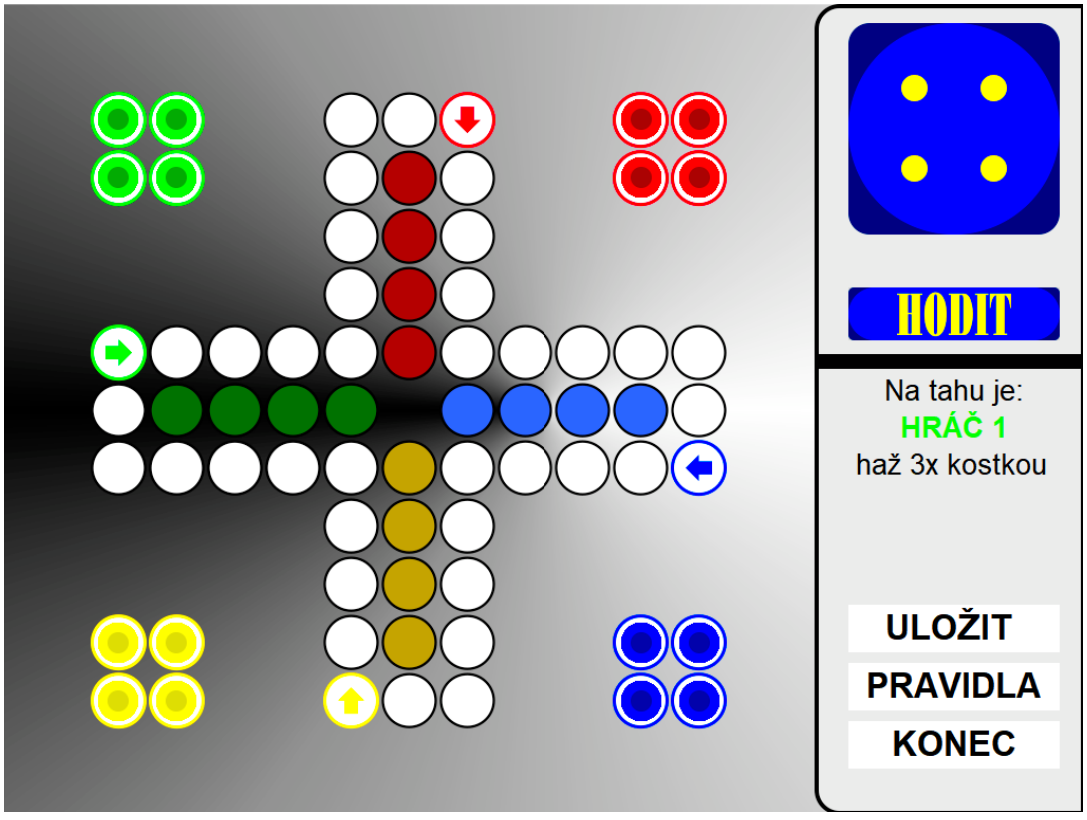
Obr.16 *Tetris* [20]

2.4 Člověče nezlob se

Tato hra vznikla v rámci bakalářské práce na téma Počítačová hra ve 2D v *MATLAB* je realizovaná pomocí *GUI* ve vývojovém prostředí *MATLAB*, kterou zrealizoval David Fiala v roce 2014. Tato hra je analogicky zkonstruovaná k její původní deskové verzi.

Hra je intuitivně zpracována pro dva až čtyři hráče, přičemž je možné hru hrát spolu s počítačem. Hra byla konstruovaná ovšem tak, aby vždy byl ve hře jeden člověk. Při spuštění hry musí hráč hodit hodnotu šest z třech hodů kostkou. Pokud se mu to nepovede, bude na řadě další hráč. Ve chvíli, kdy na kostce padne hodnota šest musí hráč provést selekci, kterou figurku umístí na hrací pole a jelikož na kostce byla hodnota šest tak musí hrát ještě jednou. Ve hře může nastat i situace, kdy hodnota hodu je rovna posunu figurky na pozici, kde se již může nacházet jiný hráč a dojde k jeho navrácení do domovské pozice. Součástí propracovaného grafického prostředí je animace hodu kostky a zvukové projevy typické pro deskovou verzi. Cílem hry je dostat čtyři figurky příslušné barvy přes celé hrací pole do finálních polí, které jsou pro každého hráče separované.

Hra je složena z 53 M-souborů, 17 obrázků a 7 zvukových souborů vyskytujících se v této hře. [21]



Obr.17 *Člověče nezlob se* [21]

II. PRAKTICKÁ ČÁST

3 SPECIFIKACE HRY

V této kapitole je detailně popsán popis hry a její pravidla.

3.1 Popis hry

Snake Quiz je hra určená pro jednoho hráče. Hra je složena ze 2 částí.

V první části je možné nalézt hru had, ve které hráč musí nasbírat co nejvíce možného jídla. Ovládání je realizováno pomocí klávesnice. Díky tomuto jídlu se hráči postupně zvyšuje skóre v počtu jednoho bodu. Hra je navržená tak, aby s rostoucím skóre docházelo k samovolnému zvyšování rychlosti pohybu hada. Tato vlastnost je brána jako postupné zvyšování úrovně hráči, aby bylo možné na konci hry v tabulce výsledků vyobrazit pouze deset nejlepších hráčů hry *Snake Quiz*. Tato část může skončit dvěma způsoby. První z nich nastane ve chvíli, kdy se hráč nestihne vyhnout statické překážce. Druhý konec této minihry nastává v okamžiku, kdy například rychlost pohybu hada je tak vysoká, že hráč nestíhá adekvátně korigovat směr pohybu hada a dojde ke kolizi s jeho tělem.

Do kvízu si hráč přenáší skóre z předchozí části. Tyto dostupné otázky může následně směnit za lehkou nebo těžkou otázku. Pokud si hráč zvolí lehkou otázku pak po jejím správném zodpovězení dojde k přičtení dvou bodů k jeho současnému skóre. Pokud je hráč zkušenější, může si vybrat z těžší sady otázek, která mu přidá body rovnou čtyři, samozřejmě v případě správné odpovědi. Správná odpověď je signalizována pomocí zelené diody. Červená dioda signalizuje odpověď špatnou.

Hra je určená pro široký okruh hráčů. Výhodu, kterou přináší v druhé části hry lehká sada otázek budou pociťovat především uživatelé, kteří přišli poprvé do kontaktu s programovým prostředím *MATLAB* a rádi by se dozvěděli něco nového. Naproti ní je těžká sada otázek pro zkušené uživatele *MATLAB*, kteří by si rádi zopakovali užitečné příkazy, otázky a rozšířili své zkušenosti. Za špatně zodpovězenou otázku se hráči žádné body odečítat nebudou.

Tvůrce:	Miroslav Zapletal
Zaměření:	Uživatelé MATLAB
Žánr:	Single-player
Doba hraní:	Cca 5 minut

Tab.4. Popis hry

3.2 Pravidla hry

Cílem hry je dosáhnout nejvyššího možného bodového ohodnocení z herní části *Snake* a *Quiz*.

V herní části hry had musí hráč nasbírat co nejvíce jídla ve formě červených kuliček a při tomto sběru by neměl opustit herní pole. Pokud ovšem hráč toto herní pole opustí, přesune se do další části. Analogicky by hráč neměl během pohybu hada docílit jeho záměrnému pojídání sebe sama.

V kvízové části uživatel odpovídá na otázky dle výběru obtížnosti. Jedná se tedy o těžké a lehké otázky. Hráč má na odpověď pouze jeden pokus. Pokud hráč odpoví na otázku správně dojde k rozsvícení zelené diody. V případě špatné odpovědi se rozsvítí červená dioda u aktuálně zvolené otázky, a navíc dojde k vyobrazení správné odpovědi spolu se signalizací pomocí zelené diody.

Jednotlivé otázky jsou čerpané z jedné společné databáze pro oba druhy obtížnosti. Databáze je tvořena 105 otázkami. Generování otázek do kvízové části má při každé hře unikátní hodnoty a předchází monotónnosti hry. Již při spuštění je vygenerován vektor obsahující čísla otázek, které budou uživateli postupně přiděleny v závislosti na jeho volbě obtížnosti otázky.

Bodové ohodnocení hráče se sčítá po celou dobu hry a hráč má neustálý přehled o aktuálním bodovém zisku.

Uložení výsledků proběhne ve chvíli, kdy hráč úspěšně dokončí celou hru *Snake Quiz*.

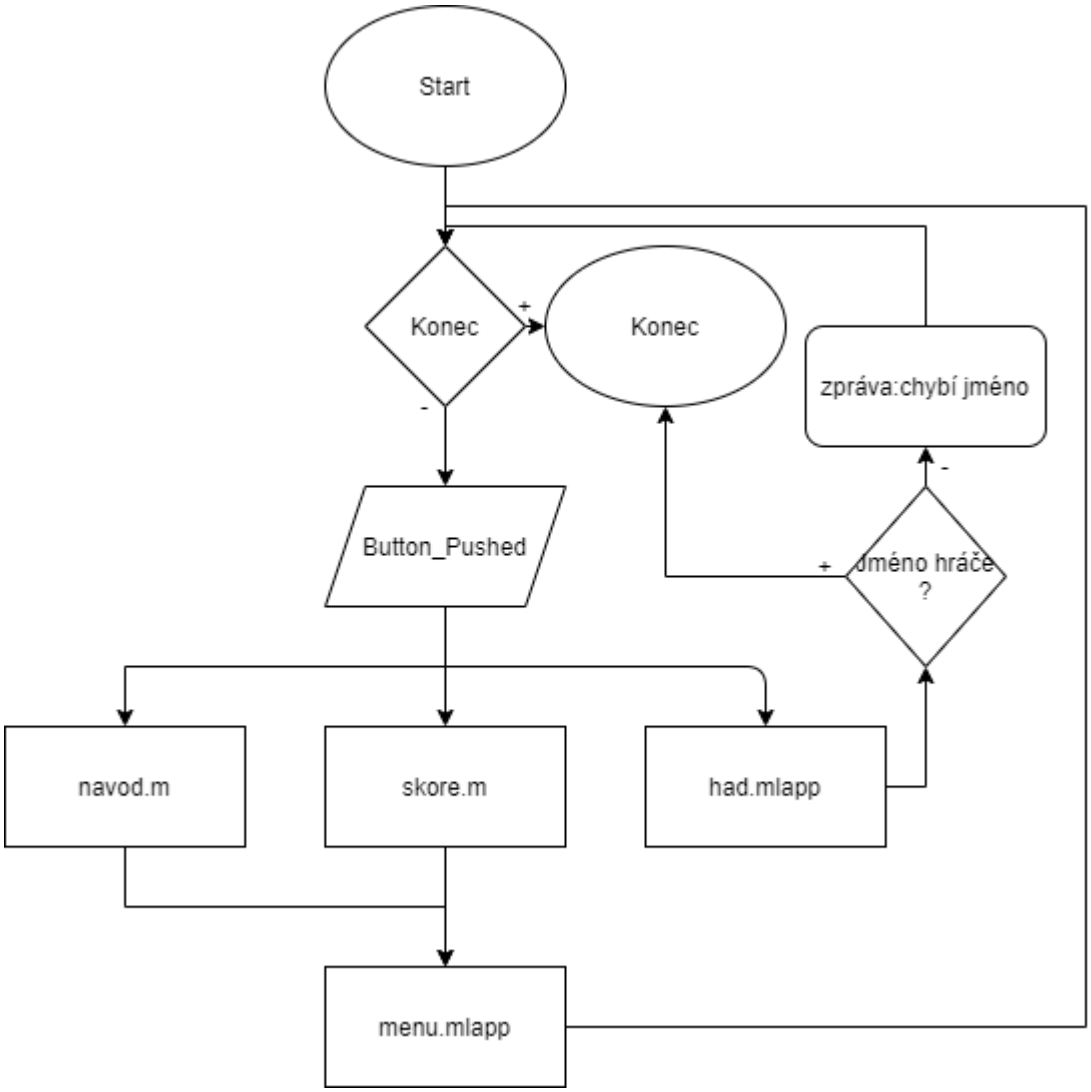
4 REALIZACE HRY

V této kapitole je popsána hra z programátorského hlediska.

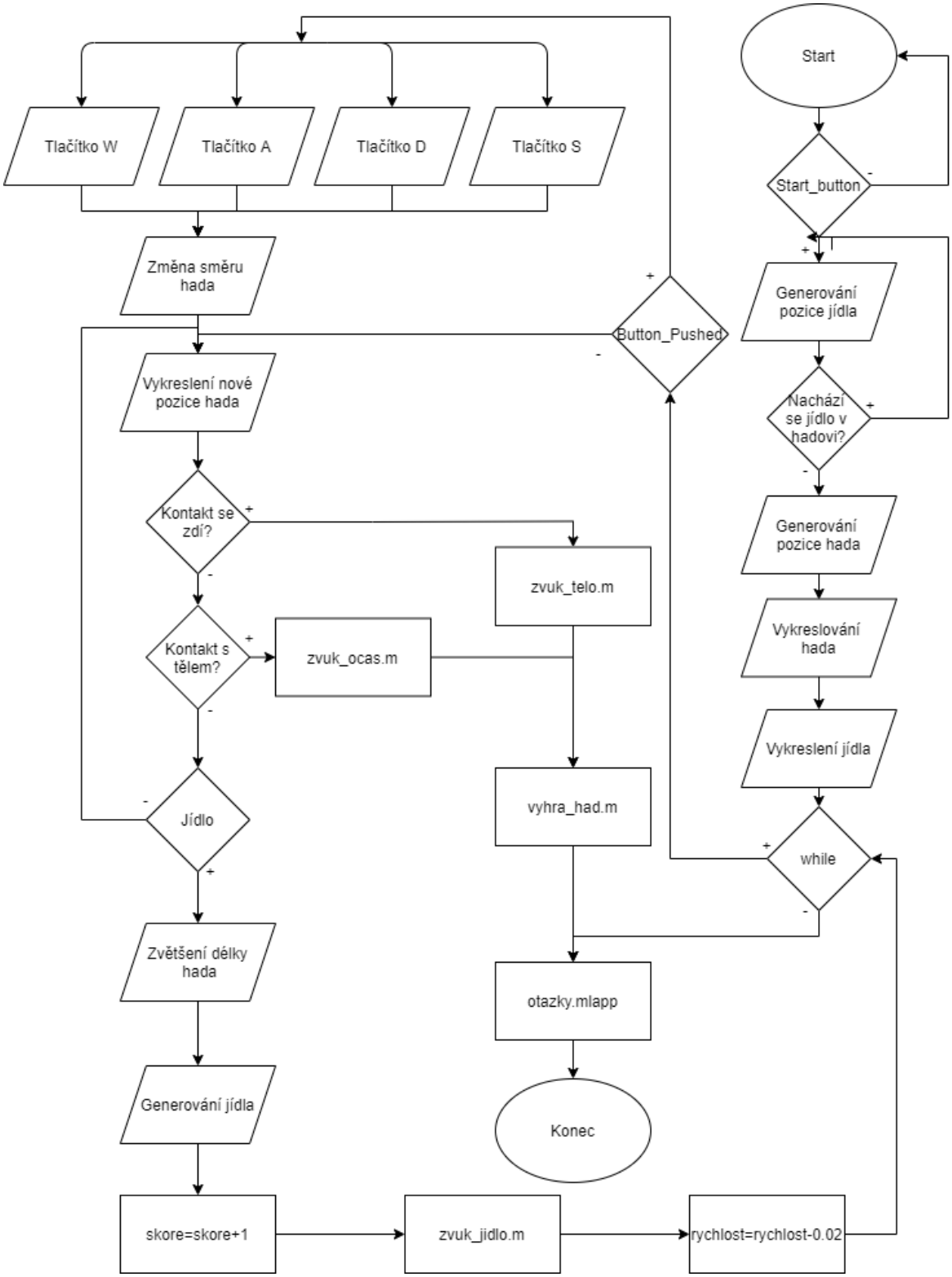
4.1 Popis tvorby hry

V první fázi jsem si studoval studentské práce a tvorbu veřejnosti na stránkách *MATLAB Central*, které byly součástí teoretické části. Po nastudování všech potřebných informací jsem se začal seznamovat s nástrojem *App Designer* v programovém prostředí *MATLAB* verze *R2020b* a kompatibilitu ověřoval ve verzi *R2020a*.

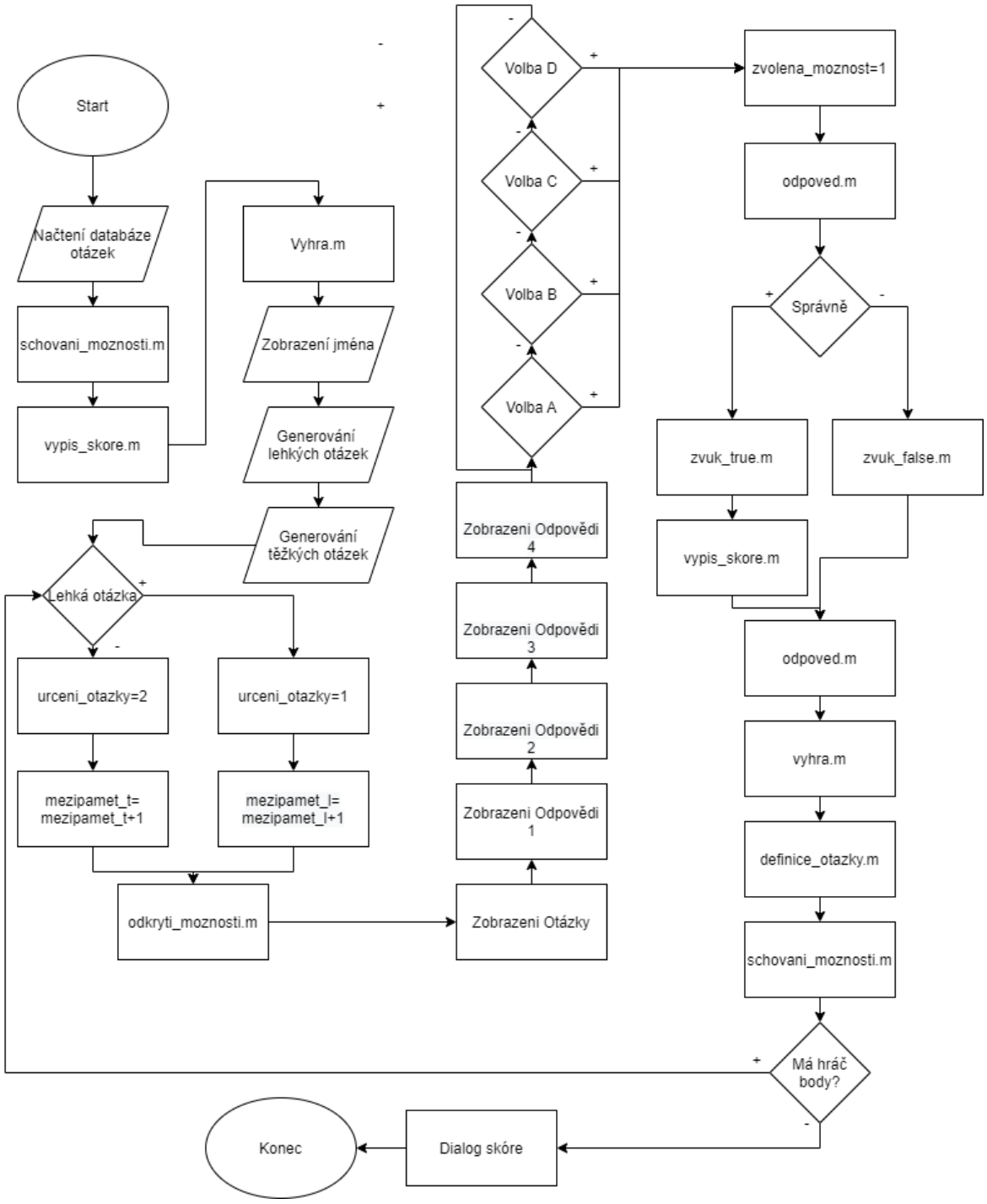
Vývojové diagramy jednotlivých aplikací jsou následně uvedené na Obr.18, Obr.19 a Obr.20.



Obr.18 Vývojový diagram aplikace *menu.mlapp*



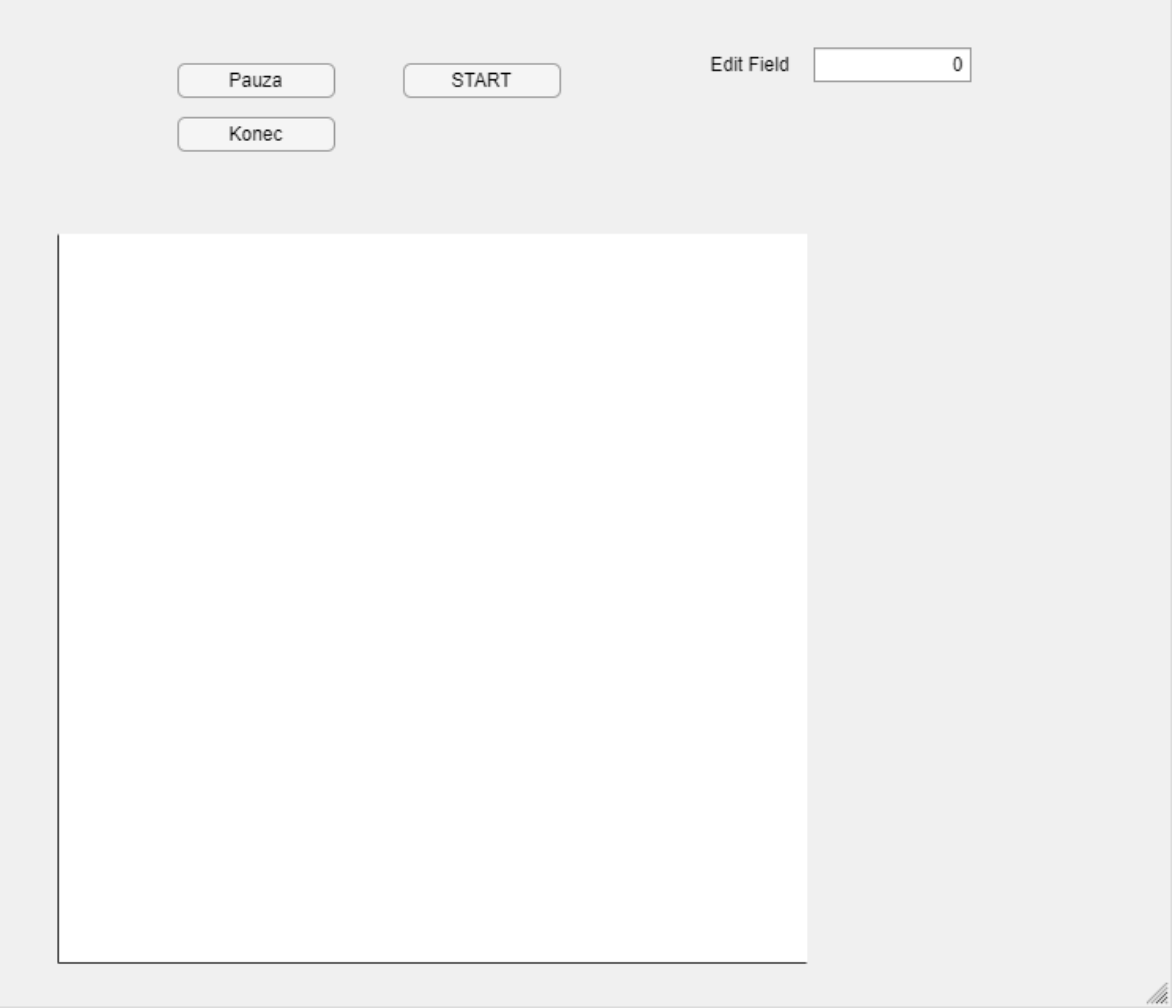
Obr.19 Vývojový diagram aplikace *had.mlapp*



Obr.20 Vývojový diagram aplikace *otazky.mlapp*

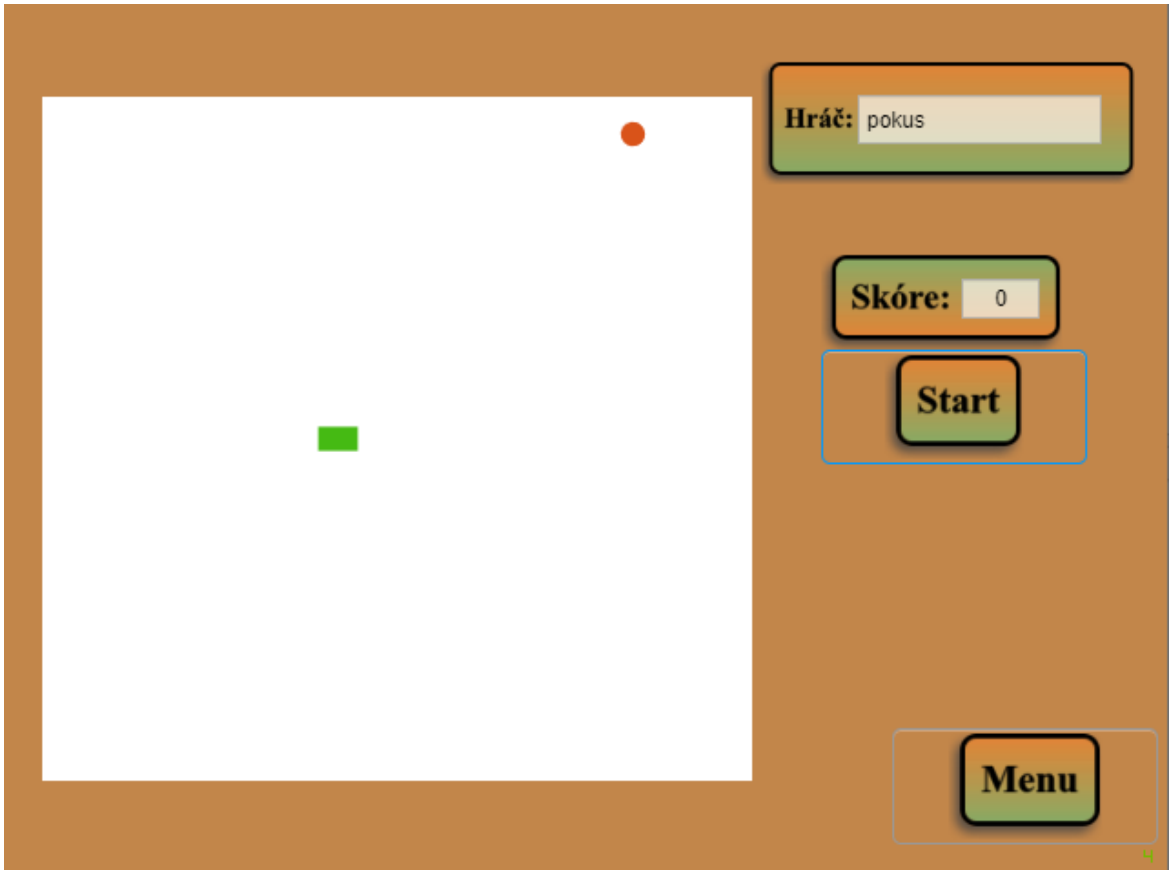
4.1.1 Had

Aplikace had, byla vytvořena jako první soubor při tvorbě této hry. Po založení nového projektu jsem realizoval rozvržení prvků aplikace tak, aby bylo možné tyto prvky následně naprogramovat k celkové funkčnosti. Prvotní vzhled aplikace je možné vidět na Obr.21.



Obr.21 Snake-prvotní rozložení

S přibývajícími vědomostmi jsem se rozhodl využívat i M-soubory pro zpřehlednění programu. Aplikaci jsem tedy rozšířil o tři M-soubory, které slouží pro přehlednost kódu ze strany uživatele. Postupnými úpravy se aplikace dostala po grafické stránce do podoby, kterou je možné vidět na Obr.22.



Obr.22 *Snake*-konečné rozložení

Na ovládací a zobrazovací prvky bylo použito černé orámování a lineární přechod barev, který dodal barevnou pestrost. Tyto jednotlivé ovládací prvky jsou vloženy do aplikace ve formě obrázku. Aplikace se rozrostla o schopnost zobrazování jména hráče a návrat do menu.

4.1.2 Databáze otázek a odpovědí

Databáze vyobrazená na Obr.23 se skládá ze sedmi sloupců, které v uvedeném pořadí mají význam: Otázka, čtyři odpovědi, pozice správné odpovědi a typ otázky.

1	Jaký příkaz využijeme pro výpis proměnné?	disp()	show()	enable()	who()	1	Lehké
2	Pomocí které funkce můžeme zjistit počet prvků matice či vektoru?	show()	length()	mod()	ceil()	2	I
3	Pomocí které proměnné můžeme vypsat poslední výsledek, který byl vypočítán?	clf	abc	ans	abs	3	I
4	Jaký příkaz využijeme pro zjištění velikosti matice?	ceil()	table()	readtable	size()	4	I
5	Pomocí kterého příkazu je možné vypsat v matici A prvek obsažený na 2 řádku a ve 3 sloupci?	A(3,2)	A(2,:)	A(2,3)	A(:,3)	3	I
6	Jak docílíme vypsání celého 3 řádku matice A?	A(:,3)	A(3,:)	A(3,all)	A(3)	2	I
7	Jak docílíme vypsání celého 3 sloupce matice A?	A(:,3)	A(3,:)	A(all,3)	A(3)	1	I
8	Jak docílíme vytvoření matice A velikosti 3x2 obsahující pouze nuly?	A = ones(3	A = zeros(	A = poly(3	A = null(3,	2	I
9	Které soubory v MATLABu obsahují funkce či skripty?	N soubory	F soubory	M soubory	P soubory	3	I
10	Pomocí kterého příkazu zobrazíme nápovědu?	call	say	tell	help	4	I
11	Pomocí kterého příkazu vykreslíme graf?	fig()	disp()	plot()	show()	3	I
12	Pomocí kterého příkazu zobrazíme v pracovním prostoru proměnné a jejich obsah?	whitch	who	tell	say	2	I
13	Pomocí kterého příkazu zjistíme nejmenší prvek vektoru?	min	max	midl	minim	1	I
14	Pomocí kterého příkazu zjistíme největší prvek vektoru?	min	max	maxim	midl	2	I
15	Pomocí kterého příkazu zjistíme průměr z prvků?	equal	midl	mean	maen	3	I
16	Pomocí kterého znaku uvedeme aritmetickou posloupnost?	=	<	>	:	4	I
17	Pomocí kterého operátoru uvedeme přiřazení hodnoty prvku?	:	<	=	>	3	I
18	Pomocí kterého operátoru uvedeme porovnání hodnot prvků?	:	==	=	<>	2	I
19	Pomocí jaké dvojice znaku vytvoříme pole?	[]	<>	()	{}	1	I
20	Pomocí jaké dvojice znaku vytvoříme buněčné pole?	[]	{}	()	<>	2	I
21	Pomocí kterého příkazu je možné vypsat v matici A prvek obsažený na 4 řádku a v 5 sloupci?	A(5,4)	A(4)(5)	A(4,5)	A{4,5}	3	I
22	Pomocí kterého příkazu je možné vypsat v matici A prvek obsažený na 5 řádku a ve 2 sloupci?	A(2,5)	A{5,2}	A[5,2]	A(5,2)	4	I
23	Pomocí kterého příkazu je možné smazat v matici A celý 3 řádek?	A(:,3) = []	A(3,:) = ()	A(3,:) = []	A(3,:) = {}	3	I

Obr.23 Databáze otázek a odpovědí

Databáze je tvořena z lehké a těžké sady otázek a odpovědí. Jedná se o jeden CSV soubor, který obsahuje celkem 105 otázek. Tyto otázky jsem vytvářel v průběhu vytváření hry, díky které jsem se dostal do kontaktu s velkou spoustou užitečných funkcí, příkazů a aritmetických operátorů.

Samotné vkládání otázek do hry uvedené na Obr.24 probíhá ve chvíli, kdy hráč dokončí hru had a vejde do aplikace *Quiz*. Kombinované otázky jsou načteny do matice, ze které se průběžně jednotlivé otázky čerpají.

```
58
59 -        %načtení všech otázek do matice
            app.vse = readtable('C:\Users\mirdi\Documents\MATLAB\Otazky.csv');
```

Obr.24 *Quiz*-Vkládání otázek do lokální proměnné

Programová část vyhodnotí počet otázek, které budou v průběhu hry hráči přiděleny. Na základě znalosti množství otázek se vygenerují dva vektory, uvedené na Obr.25 obsahující náhodné čísla otázek pro obě náročnosti. Tímto způsobem program zamezuje monotónnosti vybíraných otázek a je velmi malá pravděpodobnost, že by ve dvou hrách byly stejné otázky ve stejném pořadí.

```
77
78 -        %          nacteni otazek lehkych
            cislo_ot_l=randsample((1:52),otazek);
79
80 -        %          nacteni otazek tezkych
            cislo_ot_t=randsample((53:105),otazek);
```

Obr.25 Quiz-Generování čísel otázek

4.1.3 Kvíz

Vytváření kvízové části probíhalo ve chvíli, kdy byla aplikace had kompletní. Při návrhu aplikace jsem již dbal na zachování rozložení jednotlivých prvků a snažil jsem se jednotlivé kroky tvorby promyslet. Na Obr.26 je vzhled aplikace, který vznikl v návaznosti na dokončení aplikace *had.mlapp*.

Obr.26 *Quiz*-prvotní rozložení

Vkládání dat do textových polí je uvedeno na Obr.27, které díky vygenerovaným indexům otázek, které proběhlo na Obr.25, přiřadí vždy správné odpovědi k dané otázce do jednotlivých textových oblastí aplikace.

```
213 -           otazka= app.vse{cislo_ot_l(mezipamet_l),1} ;
214 -           odpoved1= app.vse{cislo_ot_l(mezipamet_l),2} ;
215 -           odpoved2= app.vse{cislo_ot_l(mezipamet_l),3} ;
216 -           odpoved3= app.vse{cislo_ot_l(mezipamet_l),4} ;
217 -           odpoved4= app.vse{cislo_ot_l(mezipamet_l),5} ;
218 -           app.odpoved5= app.vse{cislo_ot_l(mezipamet_l),6} ;
219
220             %vypis otazky do jednotlivych text area
221 -           app.TextArea.Value=otazka;
222 -           app.TextArea_2.Value=odpoved1;
223 -           app.TextArea_3.Value=odpoved2;
224 -           app.TextArea_4.Value=odpoved3;
225 -           app.TextArea_5.Value=odpoved4;
```

Obr.27 *Quiz*-Vkládání dat do textových polí

Aplikace je rozšířena o 7 M-souborů, které umožňují analogicky k hadovi lepší čitelnost kódu a snazší přístup k jednotlivým skriptům. Na Obr.28 je současný vzhled aplikace.

Obr.28 *Quiz*-konečné rozložení

4.1.4 Databáze nejlepších hráčů

Do databáze nejlepších hráčů jsou data vkládány vždy po dokončení hry. Data jsou průběžně ukládána do globálních proměnných, které se získávají v průběhu celé hry. Tyto údaje jsou

následně zpracována v M souboru *Data*, který je volán ve chvíli, kdy hráč dokončí úspěšně hru poslední otázkou kvízové části. Na Obr.29 je ukázka zápisu jednotlivých údajů do CSV souboru a vypsání celkového dosaženého skóre hráči. Předchozí data z minulých her jsou statické a nedochází k jejich pozdějším úpravám.

```
1 -    global jmeno otazek lehkych tezkych skore;
2 -    a=jmeno;
3 -    b=otazek;
4 -    c=lehkych;
5 -    d=tezkych;
6 -    e=skore;
7 -    c=sprintf('%s;%f;%f;%f;%f',a,b,c,d,e);
8 -    writematrix(c,'C:\Users\mirdi\Documents\MATLAB\Skore.csv',"WriteMode",'append');
9 -    c=sprintf('Gratuluji!!!\nHráč: %s\nCelkové dosažené skóre je: %1.0f\nNyní v menu
10 -   uiwait(msgbox(c));
```

Obr.29 Ukládání dat do CSV souboru

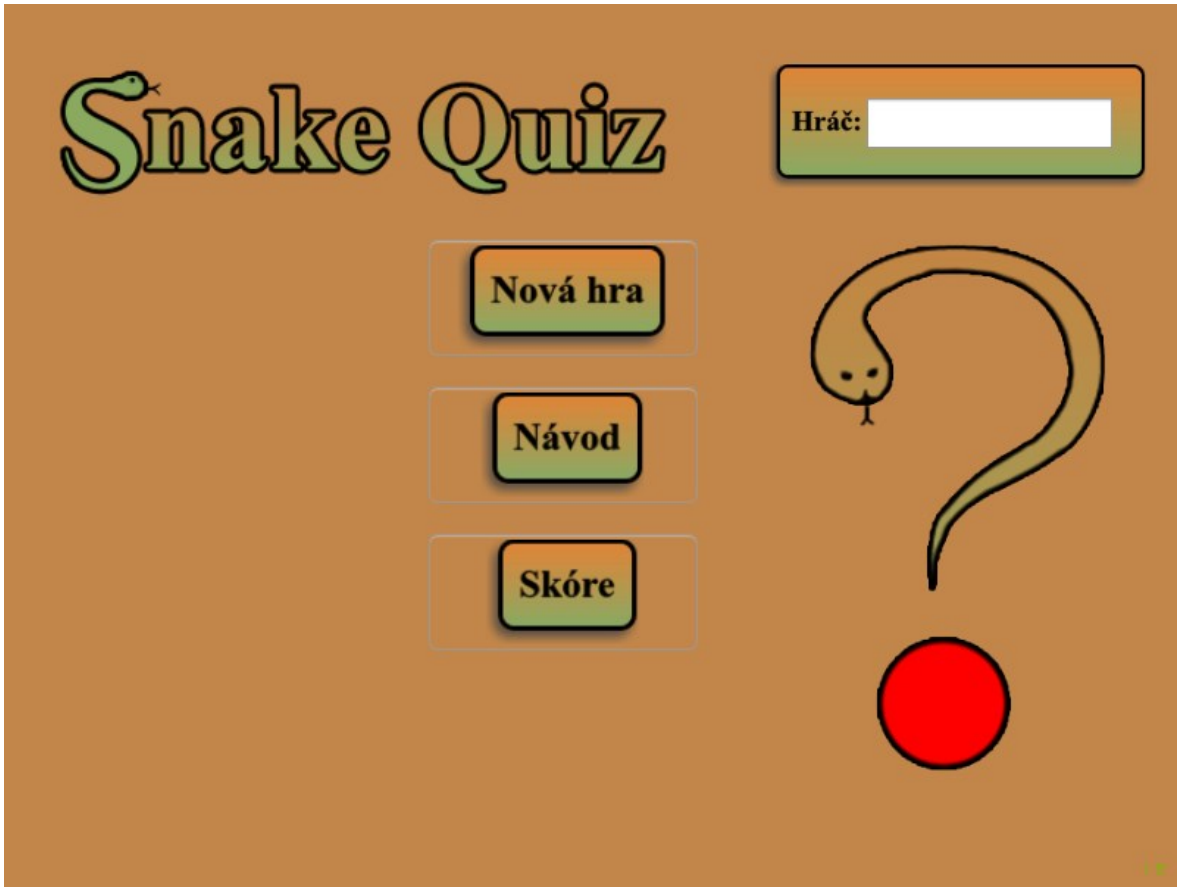
Samotný CSV soubor, obsahující údaje o jednotlivých hráčích je vyobrazen na Obr.30. Skládá se z pěti sloupců a jejich význam je následující: Jméno hráče, Skóre ze hry had, Množství lehkých, Těžkých otázek a Celkové skóre.

	A	B	C	D	E
1	Pokus	1.000000	1.000000	0.000000	0.000000
2	Petr	2.000000	1.000000	1.000000	4.000000
3	Pavel	2.000000	1.000000	0.000000	4.000000
4	Sabina	3.000000	0.000000	0.000000	3.000000
5	Marek	6.000000	0.000000	1.000000	10.000000
6	Tonda	4.000000	2.000000	1.000000	12.000000
7	Pokus2	3.000000	1.000000	1.000000	9.000000
8	Had	1.000000	1.000000	0.000000	3.000000
9	Nikita	2.000000	0.000000	1.000000	6.000000
10	Tamara	7.000000	0.000000	6.000000	31.000000
11	pokus	1.000000	0.000000	0.000000	1.000000
12	pokus	9.000000	1.000000	1.000000	15.000000
13	PokusHok	6.000000	1.000000	3.000000	20.000000

Obr.30 Skóre

4.1.5 Menu

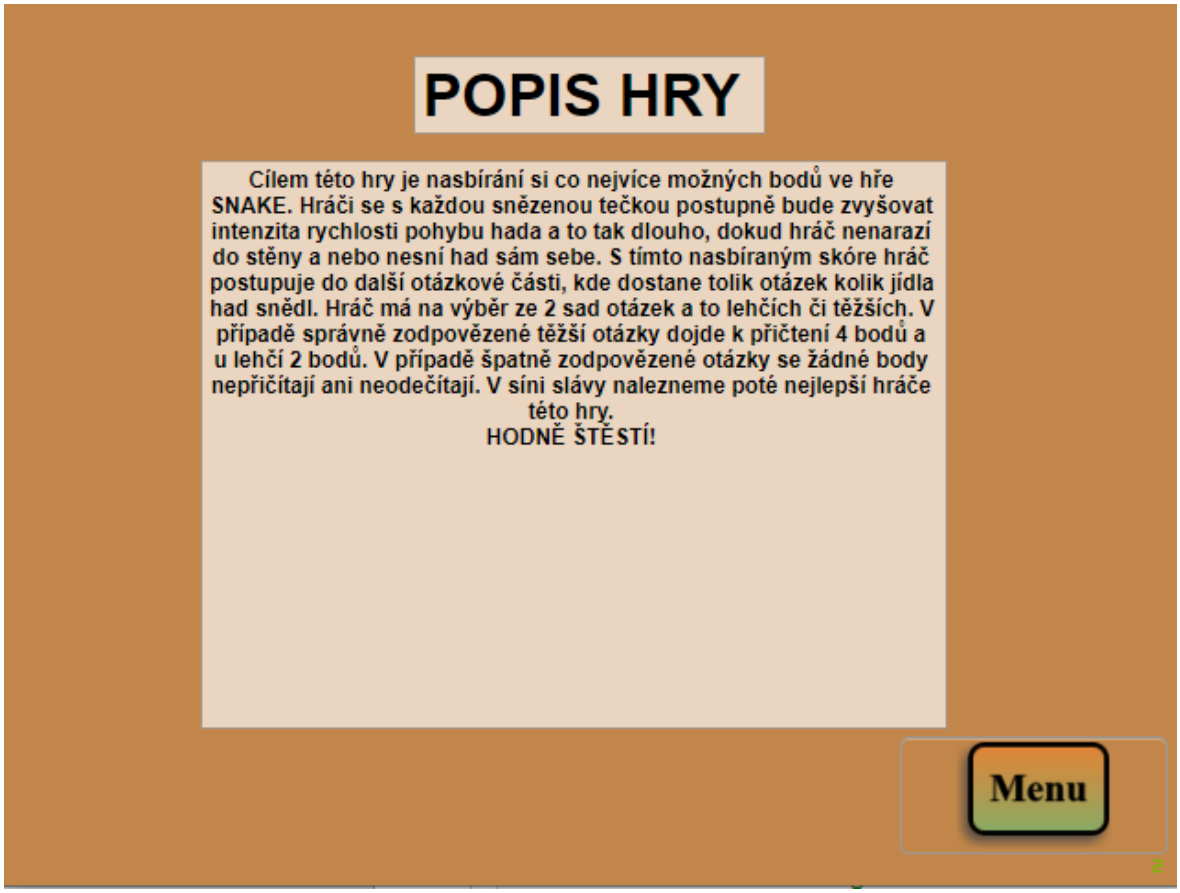
Ve chvíli, kdy jsem měl vytvořené výše uvedené hlavní části programu tak jsem mohl začít vytvářet samotné menu aplikace. Zde je možné nalézt tři tlačítka a *edit box* pro zadání jména hráče. Vzhled samotného menu je vyobrazen na Obr.31.



Obr.31 *Menu*

4.1.5.1 Návod

Po stisknutí tlačítka *Návod* v aplikaci *menu.mlapp* se spustí nová aplikace, která obsahuje stručný popis hry. Aplikace se skládá ze dvou textových polí a jednoho tlačítka. Textové pole mají přednastavenou hodnotu zobrazovaných údajů a vypnutý editační režim těchto polí. Není tedy možné, aby hráč záměrně pozměňoval jednotlivé dialogy. Graficky kompletní aplikaci je možné vidět na Obr.32.



Obr.32 Popis

4.1.5.2 Skóre

Tlačítko *Skóre*, obsažené v aplikaci *menu.mlapp*, po stisknutí spustí novou aplikaci. Ta se skládá z textového pole a tabulky. Tabulka obsahuje až 10 pozic pro nejlepší hráče hry. Selekce pak probíhá z datového souboru na Obr.27, kdy dojde k uložení obsahu celého CSV souboru se záznamy o jednotlivých hrách do lokální matice. Následně se celá tato matice setřídí dle nejvyšších hodnot celkového skóre, obsažených v 5 sloupci. Jelikož se může stát, že v CSV souboru nemusí být 10 a více hráčů, je zde i ochrana umožňující výpis menší hodnoty než 10 výsledků. Samotný kód pro zmíněné seřazení je uveden na Obr.33.

```
23      %    Setřízení záznamů do tabulky
24 -    app.zaznamy = readcell('C:\Users\mirdi\Documents\MATLAB\Skore.csv');
25 -    setrizene=sortrows(app.zaznamy,5,'descend');
26 -    delka=size(setrizene,1);
27 -    radky=[1:1:delka];
28      %setřízení hráčů
29 -    if delka<10
30 -        vybrane=setrizene(radky,[1 5]);
31 -        app.UITable.Data=vybrane;
32 -    elseif delka>=10
33 -        vybrane=setrizene([1 2 3 4 5 6 7 8 9 10],[1 5]);
34 -        app.UITable.Data=vybrane;
35
36 -    end
```

Obr.33 Skóre-třídění jednotlivých záznamů

Aplikace skóre, která ve výpisu obsahuje 7 testovaných subjektů a 3 mé výsledky, je zobrazena na Obr.34.

Obr.34 Skóre

4.1.6 Testování funkčnosti hry

Hru jsem průběžně testoval a zdokonaloval, ovšem na nejdůležitější nedostatky jsem narazil ve chvíli, kdy ovládání a samotný průběh byl svěřen testerům. Tyto nedostatky jsem následně opravil, aby hra byla plně funkční. Celkem bylo se hrou seznámeno sedm osob, ovšem pouze jedna z nich měla zkušenosti s vývojovým prostředím *MATLAB*.

Většina mích testerů, byla zaujata pouze hrou had, se kterou neměli žádný osobní problém. Problém většiny spočíval v kvízové části, která dělala největší problém. Větší část odpovědi pouze náhodně volila a menšina se i o *MATLAB* dále začala zajímat. Doporučení většiny tedy spočívalo ve změně tématu otázek.

V následujících bodech se budu snažit vyřešit poznatky mích testerů, kteří se hrou měli osobní zkušenost.

4.1.6.1 Otázky

Jednou z možností by byla možnost dodělat další aplikaci nastavení, která by sloužila k nastavení cesty k databázi s požadovaným okruhem otázek. V aplikaci nastavení by bylo tlačítko pro potvrzení volby. Následně v části kvíz by došlo k přidání textového pole, které by zobrazovalo název právě používané databáze. V části zobrazování skóre by bylo potřeba přidat rozřazovací podmínku, která by vypisovala výsledky do kategorii pro databáze, které se ve hře objevili. Tato aplikace by byla volně spustitelná pomocí nového tlačítka v menu.

4.1.6.2 Had

Mnoho z mích testerů navrhovalo možnost rozšíření hada i o překážky, které by se volně vyskytovali na hracím poli a v rámci tohoto počinu by došlo i ke zrušení stěn. Tento nápad by bylo též možné realizovat v sekci nastavení, kterou jsem zmínil víše. Hráč by tedy mohl kromě vložení vlastní databáze i přednastavit herní pole hada dle vlastních preferencí. Byla by zde možnost vypnutí statické stěny okolo herního pole, nastavení statických překážek uvnitř herního pole a případně i generování náhodných překážek před začátkem hry, které by se umístili na herní pole v náhodném rozvržení. Jednotlivé volby by následně pro hráče znamenali i upravené hodnocení. V případě vypnutých statických překážek okolo herního pole by pro hráče znamenalo bodový zisk o polovinu menší, náhodné překážky by hráči zvýšili skóre o +50 % a statické přednastavené překážky by znamenali +20 % bodů.

4.2 Funkce souborů

Na Obr.35 je vyobrazeno blokové schéma používaných souborů. Je zde 5 hlavních aplikací, které jsou v diagramu vyplněny červenou barvou. K hlavním souborům patří *menu.mlapp*, *had.mlapp*, *otazky.mlapp, skore.mlapp*, *navod.mlapp*.. Tyto aplikace jsou dále propojeny se 17 M soubory, které jsou v blokovém schématu šedé.

Propojení souborů, obsažená ve schématu jsou dvojího charakteru. Černá propojení signalizují uživatelsky ovládané funkce a modrá značí programově obsluhované funkce.

Obr.35 Blokové schéma aplikace *Snake Quiz*-vztahy mezi soubory

Jednotlivé aplikace, M soubory a použité funkce jsou detailně popsané v následujících podkapitolách

4.3 menu.mlapp

Tato aplikace se skládá z funkcích uvedených v Tab.5 a níže uvedených M souborů.

Název Funkce	Poloha v souboru	Význam
StartupFcn	20-25	Funkce vykonaná při spuštění aplikace. Slouží pro volání funkce *Definice_menu.m.*
ButtonPushed	28-45	Dojde k vykonání v případě stisknutí tlačítka *Nová hra*. Otestuje jestli hráč zadal jméno a volá aplikaci *had.mlapp.*
Button_2Pushed	48-53	Dojde k vykonání v případě stisknutí tlačítka *Návod*. Slouží pro uzavření aplikace a přechod do aplikace *navod.mlapp.*
Button_3Pushed	56-61	Dojde k vykonání v případě stisknutí tlačítka *Skóre*. Slouží pro uzavření aplikace a přechod do aplikace *skore.mlapp.*

Tab.5. Funkce v aplikaci *menu.mlapp*

4.3.1 Definice_menu.m

Tento M soubor slouží k prvotnímu definování globálních proměnných, které se využívají v průběhu hry *Snake Quiz*.

4.3.2 zvuk_error.m

Tento M soubor slouží k přehrání zvukového souboru, která hráče informuje o chybě formou zvukového znamení a chybové hlášky. Tento soubor je volán pouze v případě neuvedeného jména hráče a jeho obsah je možné vidět na Obr.36.

```
1 -    zvuk_chyba = audioread('error.wav');
2 -    sound(zvuk_chyba,50000);
3 -    msgbox('Nejdříve zadejte své jméno','Chyba','error');
```

Obr.36 *zvuk_error.m*

4.3.3 zvuk_potvrzeni.m

M soubor *zvuk_potvrzeni.m* slouží k přehrání zvukového souboru, který imituje zvuk kliknutí. Tato funkce je volána při každém stisku tlačítka.

4.4 skore.mlapp

Název Funkce	Poloha v souboru	Význam
StartupFcn	20-34	Funkce vykonaná při spuštění aplikace. Složí pro načtení skóre hráčů z CSV souboru a seřazení jednotlivých záznamů dle nejvyšší hodnoty. Tyto záznamy jsou následně vypsány do tabulky zobrazované v aplikaci.
ButtonPushed	37-42	Dojde k vykonání v případě stisknutí tlačítka *Menu*. Po stisknutí dojde k zavření okna a nastane návrat do aplikace *menu.mlapp.*
UIFigureClose-Request	45-48	Dojde k vykonání v případě zavření aplikace pomocí tlačítka na zavření okna. Po stisknutí křížku se zavře aktuální okno a otevře se aplikace *menu.mlapp.*

Tab.6. Funkce v aplikaci *skore.mlapp*

4.5 navod.mlapp

Název Funkce	Poloha v souboru	Význam
zptButtonPushed	15-20	Dojde k vykonání v případě stisknutí tlačítka *Menu*. Po stisknutí dojde k zavření okna a nastane návrat do aplikace *menu.mlapp*.
UIFigureClose-Request	23-26	Dojde k vykonání v případě zavření aplikace pomocí tlačítka na zavření okna. Po stisknutí křížku

		se zavře aktuální okno a otevře se aplikace *menu.mlapp*.

Tab.7. Funkce v aplikaci *navod.mlapp*

4.6 had.mlapp

Název Funkce	Poloha v souboru	Význam
startupFcn	30-36	Funkce vykonaná při spuštění aplikace. Slouží pro počáteční definování lokálních proměnných, které slouží pro určení směru pohybu, uložení průběžného skóre, přednastavení počáteční rychlosti pohybu a definici rozměru hracího pole
StartPushed	39-148	Funkce, vykonávaná po stisknutí tlačítka *Start*. Dojde ke generování náhodných souřadnic jídla, přednastavení počáteční pozice hada, reakci na změnu směru pohybu hada, nastavení rozměru hracího pole, reakci v případě srážky hada se statickou překážkou a s vlastním tělem, způsob změny délky hada při snězení jídla a samotné vykreslování hada do hracího pole
UIFigureWindowKeyPress	151-171	Jedná se o funkci, která obsluhuje stisk klávesy. Po stisku herní klávesy dojde k nastavení nového směru hada. Ukázka vypracovaného kódu je uvedena na Obr.37.
MenuPushed	174-179	Dojde k vykonání v případě stisknutí tlačítka *Menu*. Po stisknutí dojde k zavření okna a nastane návrat do aplikace *menu.mlapp*.
UIFigureCloseRequest	182-186	Dojde k vykonání v případě zavření aplikace pomocí tlačítka na zavření okna. Po stisknutí křížku

		se zavře aktuální okno a otevře se aplikace *menu.mlapp*.

Tab.8. Funkce v aplikaci *had.mlapp*

```
152 -           key = event.Key;
153
154
155 -           if(key=='w' & app.smer~=3)
156               app.smer = 0;
157 -         end
158 -         if (key=='a'  & app.smer~=2)
159 -             app.smer = 1;
160           end
161 -         if(key=='d'  & app.smer~=1)
162 -             app.smer = 2;
163 -         end
164           if(key=='s'  & app.smer~=0)
165 -             app.smer = 3;
166 -         end
167 -          if(key=='q')
168               close all force;
169 -         menu;
170 -         end
```

Obr.37 Zobrazování výsledků-správná odpověď

4.6.1 vyhra_had.m

Jedná se o M soubor, který je spuštěn ve chvíli, kdy hráč následkem nárazu ve hře *Snake* již dále nemůže pokračovat. Dojde k vykreslení dialogového okna, které hráče má informovat o aktuálním bodovém zisku. Hra čeká na potvrzení tohoto dialogového okna, kterého je docíleno příkazem *uiwait* a poté dojde k přesunu do aplikace *otazky.mlapp*.

Vypisované prvky jsou použity z obsahu globálních proměnných. Tato funkce je na Obr.38.

```
1 -   global jmeno otazek;
2 -   a=jmeno;
3 -   e=otazek;
4 -   c=sprintf('Gratuluji!!!\nHráč: %s\nAktuální skóre je: %1.0f\nPo potvrzení z
5 -   uiwait(msgbox(c));
```

Obr.38 *vyhra_had.m*

4.6.2 dotazeni.m

Tento M soubor slouží především k úpravám generovaného jídla. V průběhu testování s testery jsem narazil na problém generování potravy pro hada dvojího charakteru.

První z nich se projevoval tak, že jednotlivé kuličky se generovali mimo hrací pole. Tento problém jsem vyřešil na Obr.39.

```
1 -    global jidlo had;
2      %ochrana
3 -    if jidlo(1,1)==0
4 -        jidlo(1,1)=1;
5 -    elseif jidlo(1,2)==0
6 -        jidlo(1,2)=1;
7 -    elseif jidlo(1,1)>=20
8 -        jidlo(1,1)=19;
9 -    elseif jidlo(1,2)>=20
10 -       jidlo(1,2)=19;
11 -   end
```

Obr.39 *dolazeni.m*-ochrana generovaného jídla 1

Druhý problém nastával v generování potravy uvnitř těla hada. Bylo tedy potřeba vytvořit cyklus, který po každém generování potravy zkontroluje, jestli se na těchto souřadnicích had nenachází. Pokud ano, tak dojde ke generování nových souřadnic. Tento problém je vyřešen na Obr.40.

```
15     %cyklus pro zjištění jestli se v hadovi nebude nacházet jídlo, pokud ano
16     %tak vygeneruje nové
17 -       for i=1:delka
18
19 -          if had(i,1:2)==jidlo(1,1:2)
20 -                    jidlo(1) = round(rand*dim);
21 -                   jidlo(2) = round(rand*dim);
22 -          end
23
24 -       end
```

Obr.40 *dolazeni.m*-ochrana generovaného jídla 2

4.6.3 zvuk_jidlo.m

Tento M soubor slouží k přehrání zvukového souboru *papani.ogg*, který je volán při každé nově snězené kuličce.

4.6.4 zvuk_ocas.m

Tento M soubor slouží k přehrání zvukového souboru, který hráče informuje o srážce s tělem hada pomocí zvukového znamení. Je volán pouze v případě kdy hráč narazí při pohybu hada do jeho těla.

4.6.5 zvuk_hranice.m

Tento M soubor slouží k přehrání zvukového souboru *konec_naraz.wav*, který hráče informuje o srážce se statickou překážkou hraniční oblasti hracího pole pomocí zvukového znamení. Tento soubor je volán pouze v případě nárazu hada do statické limitní oblasti herního pole.

4.7 otazky.mlapp

Název Funkce	Poloha v souboru	Význam
startupFcn	57-81	Tato funkce je vykonávaná při spuštění aplikace. Dojde k načtení všech otázek z CSV souboru do matice a následně jsou volány funkce *schovani_moznosti.m, vypis_skore.m, Vyhra.m*. Do textového pole se vloží z globální proměnné hodnota jména a vygenerují se dvě sady čísel otázek podle přenesené hodnoty skóre ze hry had.
ButtonPushed_A Button_2Pushed_B Button_3Pushed_C Button_4Pushed_D	84-112 115-141 144-170 173-198	Funkce je vykonávaná v okamžiku, kdy hráč vybere odpověď na danou otázku. Funkce vyhodnotí, jestli odpověď byla správná. Následně jsou volány pomocné funkce *Odpoved.m, vypis_skore.m, Vyhra.m* a v závislosti na správnosti odpovědi *zvuk_true.m* či *zvuk_false.m*
LehkaButtonPushed	201-230	Ke spuštění této funkce dojde v případě stisknutí tlačítka *Lehká otázka*. Volají se zde funkce *zvuk_potvrzeni.m a odkryti_moznosti.m*. Do pomocné proměnné *urceni_otazky* se vloží hodnota, která slouží k informaci programu, které tlačítko bylo stisknuto. Dále slouží k vložení dat z matice do jednotlivých textových polí.
TezkaButtonPushed	233-259	Ke spuštění této funkce dojde v případě stisknutí tlačítka *Těžká otázka*. Volají se zde funkce

		zvuk_potvrzeni.m a *odkryti_moznosti.m*. Do pomocné proměnné *urceni_otazky* se vloží hodnota, která slouží k informaci programu, které tlačítko bylo stisknuto. Dále slouží k vložení dat z matice do jednotlivých textových polí.
DalsiButtonPushed	262-273	K vykonání této funkce dojde v případě stisknutí tlačítka *Další*. Tato funkce volá obslužné funkce *zvuk_potvrzeni.m, definovani_otazky.m, vypis_skore.m* a *Vyhra.m*.
KonecButtonPushed	276-281	Dojde k vykonání v případě stisknutí tlačítka *Konec*. Po stisknutí dojde k zavření okna a nastane návrat do aplikace *menu.mlapp*.
UIFigureCloseRequest	284-288	Dojde k vykonání v případě zavření aplikace pomocí tlačítka zavření okna. Po stisknutí křížku se zavře aktuální okno a otevře se aplikace *menu.mlapp.*

Tab.9. Funkce v aplikaci *otazky.mlapp*

4.7.1 schovani_moznosti.m

Tento M soubor umožňuje aplikaci skrytí jednotlivých nepotřebných komponent ve chvíli, kdy uživatel nemá zvolenou obtížnost otázky.

4.7.2 odkryti_moznosti.m

Tento M soubor umožňuje aplikaci odkrytí jednotlivých komponent, potřebné pro správnou funkčnost aplikace ve chvíli, kdy uživatel zvolil obtížnost otázky.

4.7.3 Odpoved.m

V tomto M souboru je obsažena celá logika zobrazování výsledku, která v případě zvolení správné odpovědi skryje všechny ostatní nepotřebné ovládací a zobrazovací prvky viz Obr.41, který obsahuje schování prvků při stisku volby A.

```
    app.DalsiButton.Visible=true;
    if app.zvolena_moznost>0 && app.odpoved5==1
        app.Lamp.Visible=true;
        app.Lamp.Color=[0 1 0];
        app.Button.Visible=false;
        app.Button_2.Visible=false;
        app.Button_3.Visible=false;
        app.Button_4.Visible=false;
        app.Lamp_2.Visible=false;
        app.Lamp_3.Visible=false;
        app.Lamp_4.Visible=false;
        app.TextArea_3.Visible=false;
        app.TextArea_4.Visible=false;
        app.TextArea_5.Visible=false;
        app.DalsiButton.Visible=true;
        %     Rozhodující podmínka, sloužící pro zobrazení špatné volby
        if app.odpoved6==2
            app.Lamp_2.Color=[1 0 0];
            app.Lamp_2.Visible=true;
            app.TextArea_3.Visible=true;
        elseif app.odpoved6==3
            app.Lamp_3.Color=[1 0 0];
            app.Lamp_3.Visible=true;
            app.TextArea_4.Visible=true;
        elseif app.odpoved6==4
            app.Lamp_4.Color=[1 0 0];
            app.Lamp_4.Visible=true;
            app.TextArea_5.Visible=true;
        end
```

Obr.41 *odpoved.m*

V případech, kdy je zvolena špatná odpověď, program nechá hráčem zvolenou původní odpověď, a navíc vyobrazí i odpověď správnou. Špatná odpověď je navíc odlišena pomocí červené diody, na rozdíl od správné odpovědi, u které bude svítit dioda zelená.

4.7.4 Data.m

Tento M soubor umožňuje aplikaci ukládat obsah globálních proměnných do CSV souboru s následným zobrazení dialogového okna, které může mít dvě podoby v závislosti na hráčově skóre.

4.7.5 definovani_otazky.m

M soubor *definovani_otazky.m* se požívá v okamžiku, kdy hráč odpoví na otázku a je potřeba z pomocných proměnných odebrat pomocná data používající se pro uložení odpovědi uživatele.

4.7.6 zvuk_true.m

Tento M soubor slouží k přehrání zvukového souboru true.wav, která hráče zvukovým signálem informuje o správném zodpovězení otázky.

4.7.7 zvuk_false.m

Tento M soubor slouží k přehrání zvukového souboru *false.wav*, která hráče zvukovým signálem informuje o špatném zodpovězení otázky.

4.7.8 vypis_skore.m

M soubor *vypis_skore.m* slouží k vypisování celkového skóre do aplikace v závislosti na počtu hodnoty bodů ze hry had a počtu správně zodpovězených otázek v kvízové části viz Obr.42.

```
1 -    global  otazek lehkych tezkych skore | jmeno;
2 -    skore=otazek+lehkych*2+tezkych*4;
3 -                preklad=jmeno;
4 -          app.SkoEditField.Value=skore;
```

Obr.42 *vypis_skore.m*

4.7.9 Vyhra.m

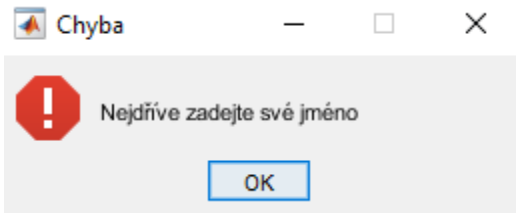
M soubor *Vyhra.m* je volán již na začátku celé aplikace *otazky.mlapp*. Jedná se o kontrolu bodového zůstatku hráče. V případě, že hráč žádné body nemá, dojde ke skrytí ovládacích prvků aplikace, volání obslužné funkce *Data.m*, a hráči bude nabízen pouze návrat zpátky do *menu.mlapp*.

5 HRA Z POHLEDU UŽIVATELE

Hra se spustí po zadání příkazu *menu* do vývojové konzole *Command Window* v MATLAB nebo po spuštění souboru *menu.mlapp* v hlavní složce aplikace.

5.1 Hlavní okno

V hlavním okně, které obsahuje menu si hráč může zobrazit návod, skóre jednotlivých hráčů, popřípadě spustit hru samotnou. Obsahem tohoto okna je i zadávací pole pro uživatelovo jméno. V případě spuštění hry bez uvedení jména, ukáže se hráči dialogové okno na Obr.43.



Obr.43 Chybí jméno

Ovládání je realizováno pomocí výběru daného objektu pomocí stisku levého tlačítka myši.

5.2 Návod

Do této sekce se hráč přesune z aplikace *menu.mlapp* po stisknutí tlačítka *Návod*. V této aplikaci má hráč po přečtení stručného návodu pouze jednu možnost, a to návrat zpět do *menu.mlapp* stiskem tlačítka *Menu*.

5.3 Výsledky hráčů

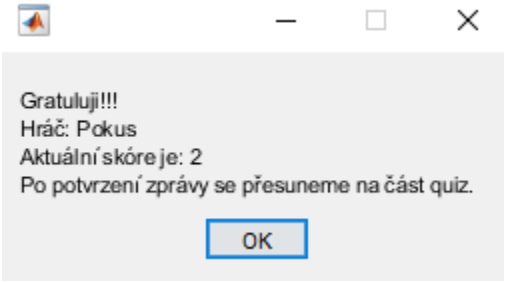
Do aplikace *skore.mlapp* se hráč přesune po stisknutí tlačítka *Skóre* v Menu hry. Hráči se zobrazí výsledky až 10 hráčů, kteří hru dohráli. Uživatel má pouze jednu možnost a to návrat zpět do aplikace *menu.mlapp* pomocí tlačítka *Menu*.

5.4 Hra Snake

Hráč se přesune do hry *Snake* stiskem tlačítka *Nová hra* v aplikaci *menu.mlapp*. Tento přesun je umožněn hráči pouze ve chvíli vyplněného jména hráče.

V případě, kdy hráč bude z nějakého důvodu potřebovat ukončit, má hned 3 způsoby, jak odejít ze hry *Had* a přesunout se zpět do menu aplikace. Pomocí klávesnice hráč ukončí svůj pokus stiskem klávesy *q*. Pro přesun pomocí myši hráč může využít uvnitř aplikace tlačítko *Menu*, nebo stisknout červený křížek uvedený v pravém rohu aplikace.

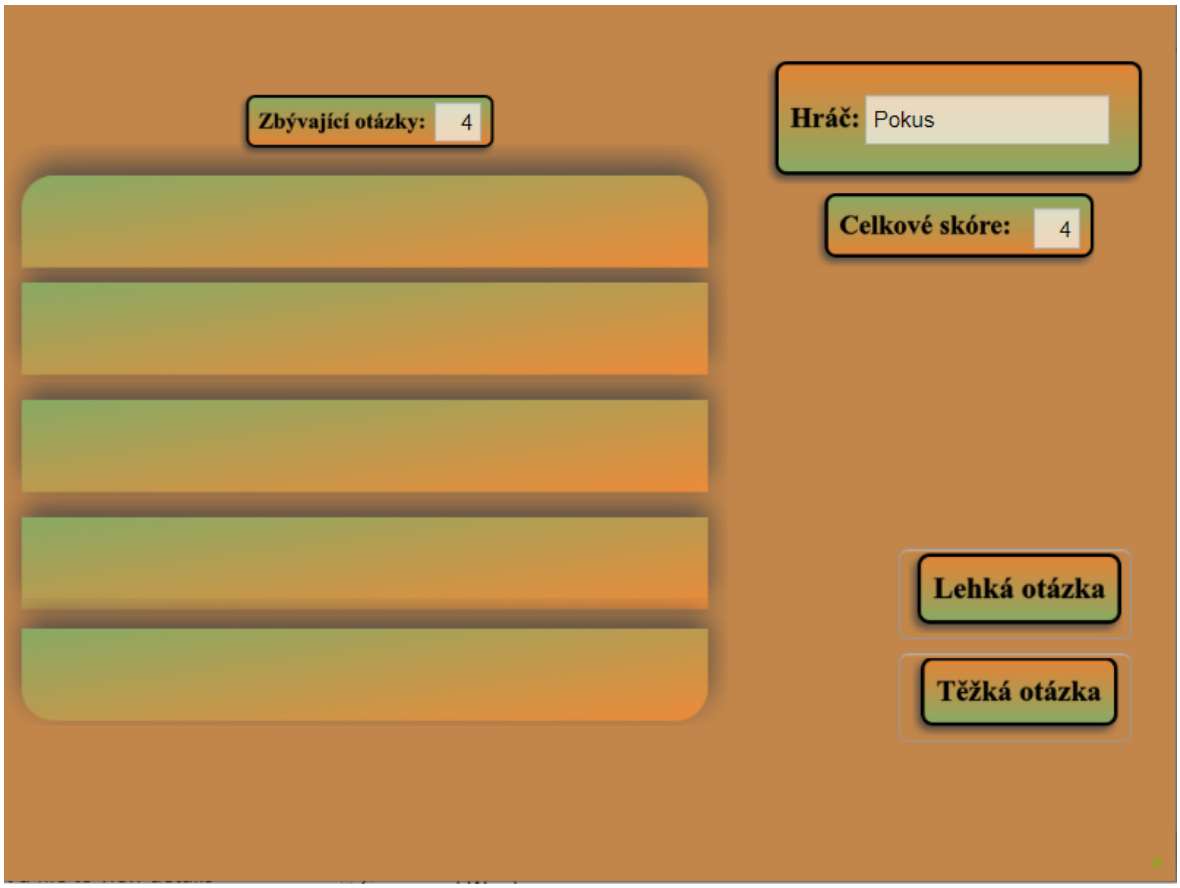
Hra se ovládá pomocí herních kláves *W, A, D, S*. Po neúmyslném ukončení se hráči vyobrazí notifikační okno viz Obr.44, které ho informuje o aktuálním bodovém zisku.



Obr.44 Dialogové okno-Had

5.5 Hra Quiz

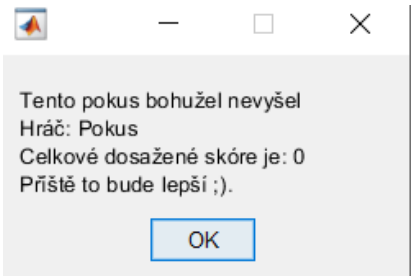
Po potvrzení dialogového okna ve hře *Had* dojde k samovolnému přesunu do aplikace *kvíz*. Spuštěná aplikace je možná vidět na Obr.45.



Obr.45 *otazky.mlapp*-přesun z aplikace *had.mlapp*

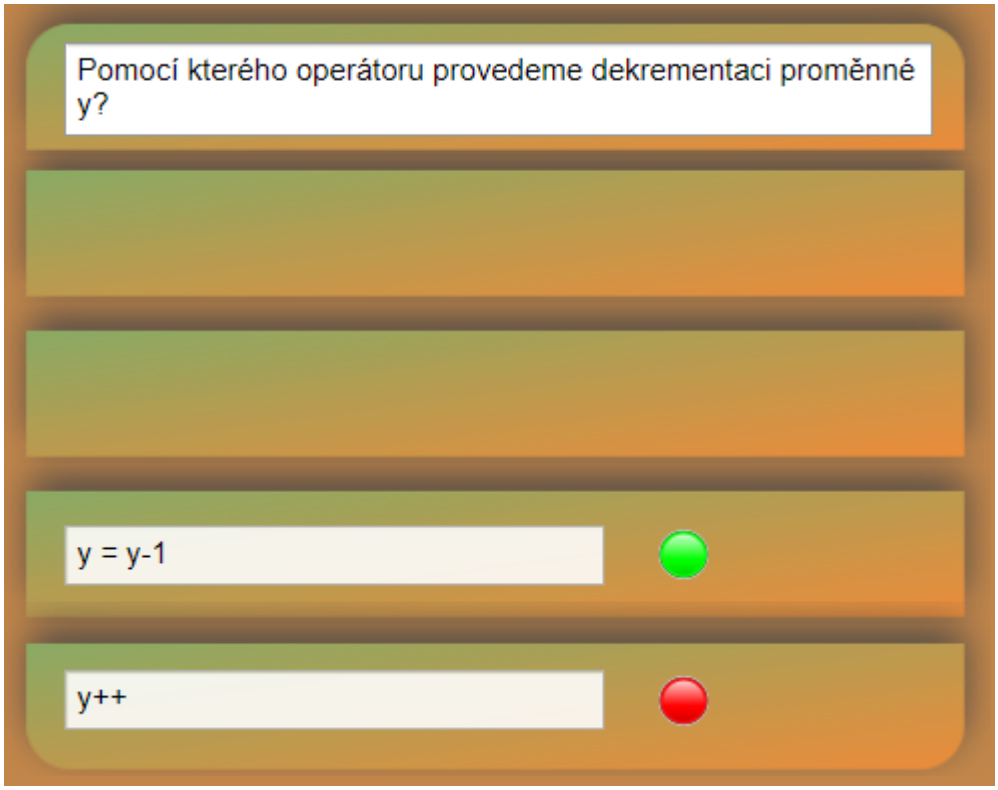
Interakce s jednotlivými prvky je realizována pomocí myši. Uživatel má stále možnost zahodit svůj pokus a přesunout se pomocí tlačítka *Menu*, nebo stisknout červený křížek uvedený v pravém rohu aplikace.

Ve hře se může objevit i situace, kdy hráč dojde s bodovým hodnocení 0 ze hry *had*. Tato situace hráči zobrazí dialogové okno uvedené na Obr.46.



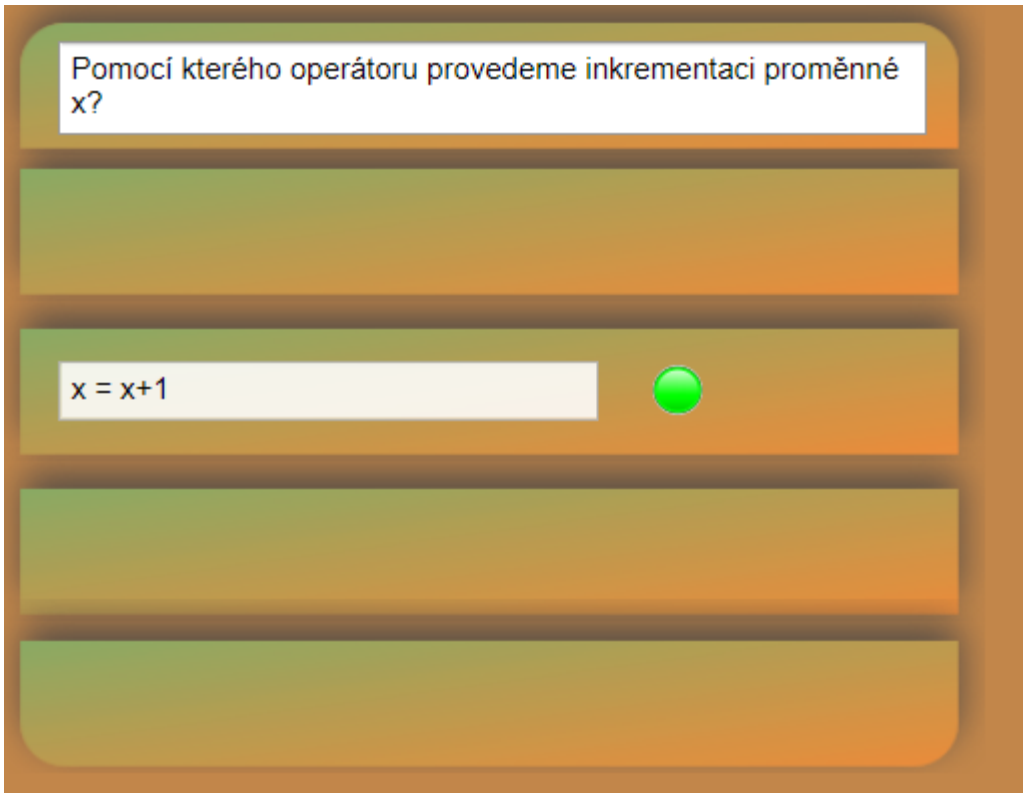
Obr.46 Dialogové okno-Quiz-skóre=0

Po zvolení typu otázky se hráči zobrazí otázka a čtyři odpovědi, na které odpovídá stisknutím tlačítka, symbolizující odpověď. Pokud hráč odpoví špatně, dojde k zobrazení správné možnosti doprovázené zelenou diodou, odpověď uživatele zůstane zachována a rozšířena o červenou diodu a zbylé dvě špatné odpovědi se skryjí. Tato situace je zobrazena na Obr.47.



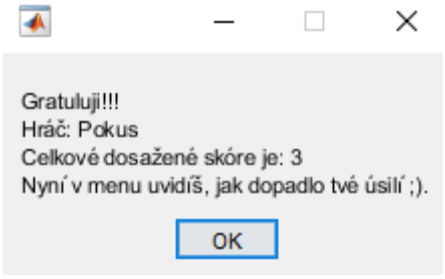
Obr.47 Zobrazování výsledků-špatná odpověď

Situace správné odpovědi je vyobrazená na Obr.48 a analogicky k situaci špatné odpovědi skryje špatné odpovědi a zanechá správnou odpověď uvozenou zelenou diodou.



Obr.48 Zobrazování výsledků-správná odpověď

V případě úspěšného průchodu hrou a zodpovězených otázkách, dojde k zobrazení vyskakovacího okna, které je na Obr.49 zobrazeno.



Obr.49 Dialogové okno Quiz-skóre>0

ZÁVĚR

Cílem této práce bylo vytvoření návrhu a realizace nové počítačové hry ve vývojovém prostředí *MATLAB*. Byla vytvořena tedy hra s názvem *Snake Quiz*, která kombinuje zábavnou 2D hru Had a edukační nástroj ve formě Kvízu.

Zaměření hry je pro uživatele, kteří s programem *MATLAB* právě začínají nebo i pro zkušené uživatele. Této vlastnosti je docíleno dvojí sadou otázek, která hráči umožnuje vybrat si náročnost přidělené otázky z rozsáhlé databáze otázek a odpovědí. Jelikož jsou otázky v samostatném souboru, je možné použít obdobnou databázi s jiným zaměřením pro edukační účely v ostatních předmětech. V návaznosti na úspěšnost uživatele si hráči mohou porovnávat své výsledky a zdokonalovat své znalosti v touze dosáhnout nejlepšího skóre hry.

Teoretická část této práce je zpracovaná formou literární rešerše o vývojovém prostředí *MATLAB*. V této části jsou také uvedeny čtyři již vytvořené 2D hry, které obsahovali několik funkčních prvků, jež jsem později při vytváření hry využil k inspiraci.

Praktická část je věnována praktickému popisu vývoje hry. Jsou uvedeny jednotlivé kroky formování hry v průběhu její tvorby. V této části nalezneme dva způsoby popisu aplikace, a to z úhlu uživatele i programátora.

Výsledkem této práce je edukační aplikace, sloužící převážně pro studenty, kteří s prostředím *MATLAB* přichází prvně do kontaktu. Po prostudování teoretické části by i novému uživateli nemělo dělat problém odpovědět na lehkou sadu otázek objevující se v kvízové části.

SEZNAM POUŽITÉ LITERATURY

[1] *Matlab* [online]. [cit. 2021-02-22]. Dostupné z: https://uvt.cuni.cz/UVT-920.html

[2] ZAPLATÍLEK, Karel a DOŇAR, Bohuslav. *MATLAB pro začátečníky*. Praha: BEN - technická literatura, 2003. s. 9. ISBN 80-7300-095-4. Dostupné také z: https://ndk.cz/uuid/uuid:b2b6fc10-5298-11e3-9ea2-5ef3fc9ae867

[3] *Access MATLAB Add-On Toolboxes. MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/thingspeak/matlab-toolbox-access.html

[4] PERŮTKA, Karel a Institut řízení procesů a aplikované informatiky. *MATLAB - základy pro studenty automatizace a informačních technologií*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. s. 66-67. ISBN 80-7318-355-2. Dostupné také z: https://ndk.cz/uuid/uuid:4b7ca070-004d-11e7-bff9-005056825209

[5] DUŠEK, František, HONC, Daniel a Chemicko-technologická fakulta. *Matlab a Simulink: úvod do používání*. Pardubice: Univerzita Pardubice, 2005. s. 11. ISBN 80-7194-776-8. Dostupné také z: https://ndk.cz/uuid/uuid:be2b6610-f684-11e6-bfcc-001018b5eb5c

[6] Data Types. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/data-types.html?searchHighlight=data%20types&s_tid=srchtitle

[7] KOLÁČEK Jan a Kateřina KONEČNÁ. *Jak pracovat s MATLABem* [online]. [cit. 2021-02-23]. Dostupné z: https://www.math.muni.cz/~kolacek/vyuka/vypsyst/navod.pdf

[8] *If, elseif, else*. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/ref/if.html?searchHighlight=if&s_tid=srchtitle

[9] *Switch, case, otherwise*. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/ref/switch.html?searchHighlight=switch&s_tid=srchtitle

[10] *While*. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/ref/while.html

[11] *Cykly* [online]. [cit. 2021-03-04]. Dostupné z: http://uprt.vscht.cz/majerova/matlab/lekce6.html

[12] *For*. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/ref/for.html?search-Highlight=for&s_tid=srchtitle

[13] ZAPLATÍLEK, Karel a DOŇAR, Bohuslav. *MATLAB pro začátečníky*. Praha: BEN - technická literatura, 2003. s. 9. ISBN 80-7300-095-4. Dostupné také z: https://ndk.cz/uuid/uuid:b2b6fc10-5298-11e3-9ea2-5ef3fc9ae867

[14] *Array vs. Matrix Operations*. *MathWorks* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/matlab_prog/array-vs-matrix-operations.html?s_tid=srchtitle

[15] KARBAN, Pavel. *Výpočty a simulace v programech Matlab a Simulink*. 1. Brno: Computer Press, 2006. ISBN 80-251-1301-9.

[16] *MATLAB App Designer*. *MathWorks* [online]. [cit. 2021-04-07]. Dostupné z: https://www.mathworks.com/products/matlab/app-designer.html

[17] *Manage Code in App Designer Code View*. *MathWorks* [online]. [cit. 2021-04-07]. Dostupné z: https://www.mathworks.com/help/matlab/creating_guis/app-designer-code-generation.html

[18] ZHANG, Mingjing. *Super Mario Bros. Demo*. *MathWorks* [online]. 2013 [cit. 2021-04-07]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/40961-super-mario-bros-demo

[19] 2048 MATLAB Edition. *MathWorks* [online]. 2016 [cit. 2021-04-07]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/46124-2048-matlab-edition

[20] FIG, Matt. *MATLABTETRIS*. *MathWorks* [online]. 2012 [cit. 2021-04-07]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/34513-matlabtetris

[21] *Počítačová hra ve 2D v MATLAB* [online]. Zlín, 2014 [cit. 2021-04-07]. Dostupné z: https://theses.cz/id/0eu0xg/. Bakalářská. Univerzita Tomáše Bati ve Zlíně. Vedoucí práce Perůtka Karel, Ing. Ph.D.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MATLAB	MATrix LABoratory
OBDC	Open Database Connectivity
JBDC	Java Database Connectivity
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Enviroment
CSV	Comma-Separated Value
Wav	Waweform audio file format-zvukový soubor
Ogg	Zvukový soubors

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM