Počítačová hra PEXESO v MATLABu

Computer game PEXESO in MATLAB

Tomáš Sedlák

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ABSTRAKT

První část bakalářská práce obsahuje stručnou rešerši prostředí a ovládání grafického počítačového programu Matlab. Druhá část se zabývá vytvořením počítačové hry PEXESO v prostředí Matlab. Jde o názornou ukázka toho, co vše je možné v tomto programu vytvořit. Bakalářská práce by měla sloužit studentům jako vhodný úvod a vědomostní základ při tvorbě GUI (Graphical User Interface).

Klíčová slova: Matlab, GUI

ABSTRACT

The first part of this bachelor thesis contains brief recherche about enviroment of Matlab - computer graphical programme. The second part deals with creation of computer game PEXESO in Matlab enviroment. It is a demonstration of this programme capabilities. This bachelor thesis should also serve as proper introduction and base knowledge for GUI (Graphical User Interface) creation.

Keywords: Matlab, GUI

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš SEDLÁK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Počítačová hra PEXESO v MATLABu**

Zásady pro vypracování:

1. **Seznamte se s prací v MATLABu a vytvořte o něm stručnou řešerši.**
2. **Vytvořte v MATLABu hru PEXESO za didaktickým účelem v rámci předmětu MATLAB a Simulink.**
3. **Program bude umožňovat volbu hry tzv. malého pexesa tj. 2x 8 obrázků a klasického pexesa.**
4. **Vytvořený program umístěte na CD-ROM jako přílohu vaší práce.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB pro začátečníky. 1. vydání. Praha: BEN – technická literatura, 2003. 144 s. ISBN: 80-7300-095-4.**
2. **DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB – tvorba uživatelských aplikací. 1. vydání. Praha: BEN – technická literatura, 2004. 216 s. ISBN: 80-7300-133-0.**
3. **PERŮTKA, Karel. MATLAB –– Základy pro studenty automatizace a informačních technologií. 1. vyd. Zlín: UTB ve Zlíně, 2005. 304 s. ISBN 80-7318-355-2.**
4. **MATLAB GUI Tutorial. URL: http://www.matlabgui.com [cit. 2009-02-04].**
5. **MATLAB CENTRAL File exchange, Section: Games. URL: http://www.mathworks.com/matlabcentral/fileexchange/?term=tag%3A%22games%22 [cit. 2009-02-04].**

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

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

Rád bych tímto poděkoval vedoucímu bakalářské práce, Ing. Karlu Perůtkovi, Ph.D., za jeho rady, podněty a celkově za odborné vedení mé bakalářské práce.

Motto

“

Kdo chce pohnout světem, musí pohnout sám sebou.

”

Sokrates (469 př. n. l. – 399 př. n. l.)

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 a jeden výtisk bude uložen u vedoucího práce;
- 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 s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování 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.

Ve Zlíně …….…….

podpis diplomanta

OBSAH

ÚVOD 9
I TEORETICKÁ ČÁST 10
1 O PROGRAMU 11
2 POPIS PROGRAMOVÉHO PROSTŘEDÍ 12
2.1 COMMAND WINDOW 12
2.1.1 Užitečné příkazy pro práci v Command Window 13
2.2 COMMAND HISTORY 14
2.3 Typy MATLAB souborů pod MS WINDOWS 14
2.4 NÁPOVĚDA 15
3 PROMĚNNÉ 15
3.1 LOKÁLNÍ PROMĚNNÉ 16
3.2 GLOBÁLNÍ PROMĚNNÉ 16
3.3 PERSISTENTNÍ PROMĚNNÉ 17
3.4 REZERVOVANÁ KLÍČOVÁ SLOVA 17
4 DATOVÉ TYPY 17
4.1 ČÍSELNÉ DATOVÉ TYPY 17
4.2 DATOVÝ TYP CHAR 19
4.3 DATOVÝ TYP CELL 19
4.4 DATOVÝ TYP STRUCT 20
4.5 DATOVÝ TYP – HANDLE FUNKCE @ 21
5 OPERÁTORY 22
5.1 RELAČNÍ OPERÁTORY 22
5.2 LOGICKÉ OPERÁTORY 22
6 PODMÍNKY A CYKLY 23
6.1 PODMÍNKA IF 23
6.2 MNOHONÁSOBNÉ VĚTVENÍ SWITCH 23
6.3 CYKLUS WHILE 24
6.4 CYKLUS FOR 25
7 POLE A MATICE 25
7.1 VYTVÁŘENÍ POLÍ 26
7.2 INDEXACE MATIC A POLÍ 26
7.3 ZÁKLADNÍ OPERACE S POLI 27

8 OPERACE S ŘETĚZCI 29
9 GUI OBJEKTY - UICONTROL 30
10 FUNKCE 32
II PRAKTICKÁ ČÁST 34
11 O PROGRAMU – HŘE PEXESO 35
11.1 HLAVNÍ OKNO 35
11.2 OKNO - NASTAVENI HRY 36
11.3 OKNO - O PROGRAMU 37
12 STRUKTURA PROGRAMU 37
12.1 STRUKTURA – HLAVNÍ OKNO 38
12.1.1 Vytvoření vlastního menu 38
12.1.2 Textové objekty 39
12.1.3 Hrací pole 40
12.1.4 Překreslení hracího pole 44
12.1.5 Posuvník 44
12.1.6 Tlačítko – Nová hra 44
12.2 STRUKTURA OKNA – NASTAVENÍ HRY 45
12.2.1 Textové objekty 46
12.2.2 Objekty typu - radiobutton 46
12.2.3 Zobrazovací osy 47
12.2.4 Objekty typu - edit 47
12.2.5 Tlačítko OK 47
12.3 STRUKTURA OKNA – O PROGRAMU 48
13 SESTROJENÍ MATICE ZOBRAZENÍ 48
ZÁVĚR 50
ZÁVĚR V ANGLIČTINĚ (CONCLUSION) 51
SEZNAM POUŽITÉ LITERATURY 52
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 53
SEZNAM OBRÁZKŮ 54
SEZNAM PŘÍLOH 55

ÚVOD

Úkolem této bakalářské práce je seznámit se a vytvořit stručnou rešerši o programu Matlab a následně vytvořit v tomto programu hru PEXESO pro didaktickými účely pro předmět - *Programová podpora automatického řízení*, který je vyučován na FAI UTB ve Zlíně.

Ačkoliv je Matlab především zaměřen na vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu, měření, zpracování signálů, návrhy řídících a komunikačních systémů, může být tento program použit také pro tvorbu jednoduchých her a animací.

Teoretická část práce ~ rešerše obsahuje souhrn informací o prostředí Matlab, které jsou nezbytné pro vytvoření počítačové hry PEXESO (seznámení s programem, syntaxe, tvorba GUI objektů, podmínek a cyklů, manipulace s proměnnými, maticemi a poli).

Praktická část práce obsahuje hrubý postup tvorby programu s ukázkami, dále pak také mé osobní nápady a řešení jednotlivých problémů, které v průběhu nastaly.

Jednotlivé části:

- O programu
- Popis programového prostředí
- Proměnné
- Datové typy
- Operátory
- Podmínky a cykly
- Pole a matice
- Operace s řetězci
- GUI objekty
- Funkce
- Práce

I. TEORETICKÁ ČÁST

1 O PROGRAMU

Název MATLAB vznikl zkrácením slov MATrix LABoratory (volně přeloženo „laboratoř s maticemi“), což odpovídá skutečnosti, že klíčovou datovou strukturou při výpočtech v MATLABu jsou matice. Vlastní programovací jazyk vychází z jazyka Fortran. [7]

MATLAB je programové prostředí a skriptovací programovací jazyk pro vědeckotechnické numerické výpočty, modelování, návrhy algoritmů, počítačové simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. Nástavbou Matlabu je Simulink – program pro simulaci a modelování dynamických systémů, který využívá algoritmy Matlabu pro numerické řešení především nelineárních diferenciálních rovnic. Matlab používá nadstavby – toolboxy, což jsou nástroje obsahující kód, kterým řeší specifickou oblast použití.

Některé toolboxy:

- Excel link for Matlab
    - Zprostředkovává vzájemnou komunikaci mezi programy MATLAB a MS Excel – je možné používat příkazy MATLABu přímo v MS Excelu.

- Database Toolbox
    - Slouží pro přenos dat mezi programem MATLAB a databázovými formáty IBM DB2, intrese, Informix, Microsoft Access, Microsoft SQL Server, Oracle, Sybase SQL Server a Sybase SQL Anywhere

- Image Proccessing Toolbox
    - Toolbox pro práci s obrázky s možností editace grafických procedur, umožňuje různé transformace obrázků, filtrace, změny kvality vybraných částí

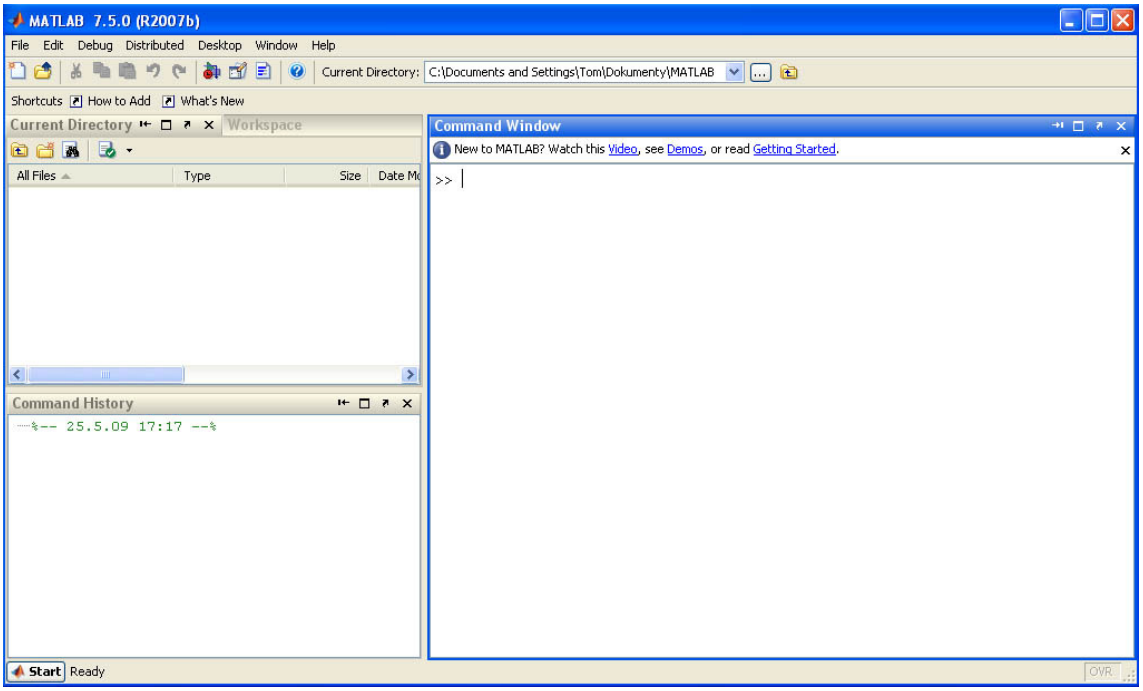
- MATLAB Web Server
    - Umožňuje posílat data pro výpočet MATLABem přes web a zobrazovat výsledky v internetovém prohlížeči

Toolboxů je mnohem více, uvádím proto pro studenty ty nejzajímavější, další můžete najít např. ve skriptech *MATLAB – Základy pro studenty automatizace a informačních technologií,* jejichž autorem je Ing. Karel Perůtka.

Původně byl MATLAB vyvinut pro počítače pracující pod OS Unix, dnes již pracuje také pod OS Windows. V současné době je nejnovější verze tohoto programu V7.8 (R2009a) a je zpoplatněna licencí.

2 POPIS PROGRAMOVÉHO PROSTŘEDÍ

Po spuštění MATLABu se objeví následující dialogové okno:



Obrázek 1: Výchozí okno po spuštění MATLABu

2.1 COMMAND WINDOW

Jde o okno, do kterého se zadávají jednotlivé příkazy. Velmi používané pro ladění a testování skriptů, zvláště pokud nevíte, co jednotlivé řádky nebo části skriptu provádí. Oproti jazyku C++ nemusíte řádek kódu ukončit středníkem – tímto docílíte toho, že se příkaz provede a zároveň vypíše výsledek.

Příklad:

```
a = 15*0,6

a=

10,2
```

Pokud středník přesto použijete a později budete chtít znát hodnotu proměnné, stačí do příkazu napsat název proměnné (samozřejmě bez středníku).

Uživatelský popis

Komentáře nebo popisky začínají znakem % nebo třemi tečkami (…) , to znamená, že text v řádku za těmito znaky MATLAB nevyhodnocuje.

2.1.1 Užitečné příkazy pro práci v Command Window

Jde o příkazy, které se hodí při ladění příkazu, skriptu, části kódu nebo programu.

Příkazy:

who – vypíše seznam všech proměnných základního pracovního prostoru
whos – vypíše vše co who, navíc vypíše také velikost, počet bajtů, které zabírá a datový typ dané proměnné
clear proměnná – vymaže proměnnou *proměnná* ze základního pracovního prostoru
clear all – vyčistí celý základní pracovní prostor
help funkce – zobrazí stručnou nápovědu k funkci *funkce* do okna *Command Window*
doc funkce – zobrazí podrobnou nápovědu k funkci *funkce* v okně *Help*
lookfor slovo – hledá slovo *slovo* ve všech nápovědách na cestě MATLABu
format typ – zobrazí číslo v závislosti na daném *typ*u, volá se vždy před výpisem čísla

2.2 COMMAND HISTORY

V tomto okně je zobrazena historie příkazů.

2.3 TYPY MATLAB SOUBORŮ POD MS WINDOWS

Sobory .FIG

Slouží pro uložení informací o obsahu grafického okna. Toto okno může být dialog s menu a různými ovládacími prvky nebo může jít o okno s objektem typu *axes*, ve kterém může být znázorněn graf či obrázek. Soubor je spustitelný – asociován se souborem *matlab.exe*.
Jde o výchozí formát při ukládání grafických výstupů a GUI vytvořených pomocí nástroje GUIDE.

Soubory .M

Jde o textové soubory, do kterých se píše kód funkcí a příkazů v pořadí jak by měly být volány. Pro editaci stačí *Poznámkový blok*. Zde můžeme psát vlastní skripty, funkce, nebo jen posloupnosti příkazů stejně jako v Command Windows. Soubor je spustitelný – asociován s editorem MATLABu, který slouží také jako ladící program.

Soubory .P

Tyto soubory jsou předzpracované pseudokódové soubory, tedy zakódované soubory obsahující zdrojový kód (soubor .M). Kódování se provádí funkcí *pcode*. Pro dekódování, čtení a úpravy tohoto typu souboru žádný příkaz sice není, můžeme jej ale spouštět stejně jako soubor s příponou .M . Kodér je zde samostatný, proto může nastat problém s kompatibilitou u starších verzí. Verze kodéru je v hlavičce .P souboru.

Soubory MEX

Jsou to subrutiny vytvořené ze zdrojového kódu jazyka C nebo Fortran. Chovají se stejně jako .M soubory a vestavěné funkce, nemají však pevně danou příponu – ta závisí na platformě:
MS WINDOWS - přípona *dll* , u LINUXu – přípona *mexglx* .
V případě, že mají .M a *mex* soubor stejné jméno, má přednost *mex* soubor.

2.4 NÁPOVĚDA

Nejjednodušší přístup k nápovědě je přímo z hlavní nabídky – tj. po kliknutí na *Help*. Může se však stát, že nápovědu nainstalovánu nemáte. Pokud nemáte originální instalační CD při ruce, můžete si otevřít oficiální stránku: *http://www.mathworks.com/access/helpdesk/help/helpdesk.html* , kde najdete kompletní dokumentaci jak k MATLABu, tak i k toolboxům a Simulinku.

Pokud máte nápovědu nainstalovánu a neznáte správný syntax funkce, stačí do Command Window zadat *help funkce* .

3 PROMĚNNÉ

Typy proměnných není třeba dopředu definovat ani deklarovat (pokud nejde o globální a persistent proměnné). MATLAB je definuje a deklaruje sám při prvním použití a uloží je do základního pracovního prostoru. Jedná se o proměnné, které vznikají zadáváním příkazů v Command Window MATLABu a také proměnné skriptů M souborů, které lze kopírovat a spouštět po řádcích v jejich pořadí.

Při použití funkcí se proměnné ukládají do lokálního prostoru dané funkce, což zvyšuje výpočetní rychlost. Tento lokální prostor funkce je oddělen od ostatních lokálních prostorů jiných funkcí, i od základního pracovního prostoru MATLABU.
Pokud tedy potřebujete zjistit hodnotu jisté proměnné uvnitř funkce, je třeba tuto proměnnou použít jako argument výstupu dané funkce. Pokud potřebujete proměnnou upravit, pak je třeba ji použít jako vstupní a výstupní argument funkce.
Další možností je definovat proměnnou jako globální, čímž se uloží do globálního prostoru. Hodnota této globální proměnné se dá poté měnit ve všech funkcích a skriptech.
Proměnnou typu *persistent* lze číst a měnit rodičovskou funkcí, u tohoto typu proměnné je potřeba použít ještě 3 další funkce pro správnou obsluhu.

Proměnné uložené v lokálním prostoru dané funkce jsou po ukončení funkce vymazány, kdežto u proměnných, které lze zadat přímo do Command Window, zůstávají a jejich seznam lze vyvolat pomocí příkazu *who* nebo přesněji pomocí příkazu *whos*.

MATLAB rozlišuje velká a malá písmena, je tudíž rozdíl mezi proměnnou *promenna a proMENNA* – jde tedy celkem o 2 různé proměnné!

Proměnná může mít maximálně 31 znaků, nesmí obsahovat tečku ani diakritiku a musí začínat písmenem.

3.1 LOKÁLNÍ PROMĚNNÉ

Lokální proměnná je výchozí formát proměnných používaných MATLABem v Command Window, skriptech, funkcích a metodách. Není třeba ji definovat ani deklarovat, vše je provedeno automaticky při prvním volání proměnné. U číselných proměnných je výchozím formátem formát *double*. Pokud je třeba upravit rychlost programu, můžeme číselné proměnné deklarovat na začátku skriptu, například pomocí příkazu *zeros*.

Příklad přetypování lokální proměnné z *double* na *string*:

```
LokPromennaDouble = 10;
LokPromennaString = int2str(LokPromennaDouble)
```

3.2 GLOBÁLNÍ PROMĚNNÉ

Tato proměnná se používá kvůli funkcím. Kazdá funkce MATLABu, která je definovaná v .M souboru, má své vlastní lokální proměnné oddělené od ostatních proměnných. Lze však sdílet jednu kopii jedné v pracovním prostoru i u několika funkcí najednou. Je třeba použít příkaz *global* a proměnnou deklarovat. Všechny funkce, které ji mají deklarovánu jako globální, ji mohou používat a měnit tak její obsah.

Vymazat globální proměnnou lze příkazem *clear promenna*, kde *promenna* je název globální proměnné typu *global* – tímto se proměnná odstraní z hlavního pracovního prostoru MATLABu, ale hodnota globální proměnné se nezmění.
Pro odstranění globální proměnné z globálního prostoru globálních proměnných je třeba zadat příkaz: *clear global promenna*, kde *promenna* je název globální proměnné.

3.3 PERSISTENTNÍ PROMĚNNÉ

Tyto proměnné lze používat pouze v jedné funkci, ostatní funkce k ní nemají přímý přístup, kromě té, ve které byla deklarována – tím se zaručí, že nebude přepsána. Po skončení funkce tato proměnná není vymazána. Jestliže funkci, ve které je tato proměnná odstraníte z paměti nebo editujete M-soubor této funkce, smažou se všechny persistentní proměnné této funkce. Proměnnou je potřeba deklarovat před jejím použitím přímo ve funkci - příkazem *persistent promenna*, kde *promenna* je název persistentní proměnné. [3]

3.4 REZERVOVANÁ KLÍČOVÁ SLOVA

Tato slova nelze použít jako názvy proměnných:

ans, beep, bitmax, break, case, catch, continue, else, elseif, end, for, function, global, i, if, Inf, j, NaN, nargin, nargout, otherwise, pi, persistent, realmin, realmax, return, switch, try, varargin, varargout, while

4 DATOVÉ TYPY

4.1 ČÍSELNÉ DATOVÉ TYPY

Pracující s:

a) Celými čísly
   - nezáporná čísla (*uint8, uint16, uint32, uint64*)
   - záporná čísla (*int8 ,int16 ,int32, int64*)
b) Reálnými čísly (*single,double*)

Číselný datový typ double

Je výchozím číselným datovým typem v MATLABu. Číslo uložené v tomto typu proměnné je složeno z celé a desetinné části, kde počet desetinných míst není pevně dán – může se tedy měnit v závislosti na požadované přesnosti.

K přetypování čísla z jiného číselného datového typu do typu double se používá funkce *double*.

Číselný datový typ single

Proměnné tohoto typu jsou podobných vlastností jako proměnné typu double, rozdíl je pouze v nižší přesnosti a rozsahu. Díky tomu tyto proměnné zabírají méně paměťového prostoru a jejich použití, pokud to použití umožňuje, má za následek v časově náročných výpočtech, rychlejší chod programu.
V současné verzi MATLABu jsou již definovány matematické operace i pro tento datový typ.

Datové typy uint8, uint16, uint32 a uint64

Tyto datové typy můžou být čísla v rozsahu 0 až 2 na mocninu čísla uvedeného u daného datového typu mínus 1. Číslo u datového typu (8,16,32,64) zároveň udává počet bitů daného datového typu. Pro *uint8* je to tedy **0 až 255,** počet bitů proměnné je **8.**
V současné verzi MATLABu jsou již definovány matematické operace i pro tyto datové typy. Pokud přejde k překročení meze daného typu, je výsledkem maximum (pro *uint8* je to 255).

Přetypování – příklad:

```
b=85;   %proměnná je typu double - tj. výchozí typ proměnné
c=uint8(b)
whos  %vypíše proměnné i s datovými typy
```

Datové typy int8 ,int16 ,int32, int64

Tyto datové typy mají rozsah podobný jako předchozí datové typy, jen s tím rozdílem, že jdou také do záporných hodnot a maximální hodnota je tedy polovina celkového rozsahu datového typu. Pro *int8* je to tedy **-128 až 127.**
V současné verzi MATLABu jsou již definovány matematické operace i pro tyto datové typy. Pokud přejde k překročení meze daného typu, je výsledkem maximum (pro *int8* je to 127).

Přetypování je obdobné jako je tomu u předchozího příkladu.

4.2 DATOVÝ TYP CHAR

Tento datový typ je určen pro znaky a řetězce. Řetězce typu *char* jsou 16bitové řetězce, rozsah použití příkazu *char* je tedy 0-65535. Řetězce jsou ukončeny prvkem NULL.

Přetypování – příklad:

```
cislo=65  %proměnná cislo je datového typu double
char(cislo)

ans =

A         %ans je typu char
```

Příkaz *char* tedy vrací ASCII znak odpovídající hodnotě v proměnné *cislo*.

Nebo také:

```
bunka{1}='text v promenne typu cell';
char(bunka)

ans =

text v promenne typu cell  %ans je typu char
```

4.3 DATOVÝ TYP CELL

Jde o buněčné pole – cell array. Prvům tohoto pole se říká kontejnery. Obsah těchto kontejnerů se může v buňce lišit. Buňka tedy může obsahovat řetězec, matici, nebo číselné hodnoty.

Přiřazování dat lze provést pomocí indexace:

a) buňky
b) přímo obsahu buňky

Příklad:

Indexování buněk:

```
bunka(1,1) = {[1 3 5; 7 6 2]};
bunka(1,2) = {'Text v bunce'};
```

Indexování obsahu buněk:

```
bunka(1,1) = [1 3 5; 7 6 2];
bunka(1,2) = 'Text v bunce';
```

4.4 DATOVÝ TYP STRUCT

Struktury v MATLABu jsou mnohorozměrová pole k jehož prvkům se přistupuje přes tzv. textové určovatele pole. Struktura se ukládá podobně jako buněčné pole. Prvky datového vektoru struktury se označují jako pole. Každé pole struktury má své jméno, tzv. textový určovatel pole. Pole se od názvu struktury oddělují tečkou. Pole struktury může obsahovat novou, tzv. vnořenou strukturu.

Příklad:

```
SeznamKontaktu.jmeno='Antonín Vydřiduch';
SeznamKontaktu.adresa='Lakomých 254, Prachatice';
SeznamKontaktu.telefon='521123456'
```

Výsledkem je struktura o 3 polích:

```
SeznamKontaktu =

 jmeno: 'Antonín Vydřiduch'
 adresa: 'Lakomých 254, Prachatice '
telefon: '521123456'
```

Další pole přídáme takto:

```
SeznamKontaktu(2).jmeno='Jarmila Vysmátá';
SeznamKontaktu(2).adresa='Smíchov 12, Praha 6';
SeznamKontaktu(2).telefon='511654321';
```

nebo takto:

```
SeznamKontaktu(2)=struct('jmeno','Jarmila vysmátá',
'adresa','Smíchov 12, Praha 6','telefon','511654321');
```

4.5 DATOVÝ TYP HANDLE FUNKCE @

Handle funkce je datový typ MATLABu, který obsahuje informace pro odkazování se na funkci. Když vytvoříte handle funkce, MATLAB do handle uloží veškeré informace potřebné k tomu, aby se funkce spustila nebo vyhodnotila pomocí funkce *feval*. Handle funkce je však více než jen odkaz na danou funkci. Často reprezentuje sadu metod funkce. Když vyhodnocujete handle funkce, MATLAB bere v úvahu pouze ty funkce, které byly uloženy do handlu, když byl vytvořen. Ostatní funkce, které mohou být nyní v cestách MATLABu nejsou uvažovány.

Výhody handlu funkce:

a) Umožňuje přístup k funkci jinými funkcemi
b) Zachytí všechny metody přetížené funkce
c) Umožňuje širší přístup k subfunkcím a privátním funkcím
d) Zvyšuje čitelnost
e) Snižuje počet souborů
f) Zvyšuje výkon programu při opakovaných operacích

Jelikož je handle standardní datový typ MATLABu, lze s ním provádět všechny operace jako s ostatními datovými typy, tj. např. můžete vytvářet pole, struktury nebo buněčná pole handlů funkcí. Přístup k datům je stejný jako u numerických datových typů.
Handle funkce vytváří tak, že před název funkce dáme znak @a to pak přiřadíme do proměnné.

Příklad:

```
handleSinus=@sin
feval(handleSinus,0.23)

ans =

0.2280
```

5 OPERÁTORY

5.1 RELAČNÍ OPERÁTORY

Slouží k porovnání dvou hodnot (proměnných, polí stejné dimenze a délky…). Výsledkem je číslo logického typu, tj. log. 0 - *false* a log.1 - *true*.

Relační operátory jsou tyto:

<	menší než
>	větší než
==	rovná se
<=	menší než nebo rovná se
>=	větší než nebo rovná se
~=	nerovná se

5.2 LOGICKÉ OPERÁTORY

Jde o logické operátory, které se aplikují na jednotlivé prvky, pole (po prvcích), tj. skaláry.

Logické operátory jsou tyto:

~	logická negace – not, pro pole
&	logický součin – and, pro pole
|	logický součet – or, pro pole
&&	logický součin – and, pro podmínky, pro skalár
||	logický součet – or, pro podmínky, pro skalár

Výsledkem při použití těchto operátorů je buď log. 0 – tj. *false* nebo log. 1 – tj. *true.*
Logické operátory se řídí prioriou – viz pořadí výše sestupně od nejvyšší po nejnižší prioritu.
Pokud si nejste jisti, zda jste podmínku nebo logickou operaci provedli správně, závorkujte.

6 PODMÍNKY A CYKLY

6.1 PODMÍNKA IF

Syntaxe:

```
if  logický výraz
     příkazy   %tyto příkazy se provedou, pokud hodnota
               log. výrazu (viz řádek výše) nabývá log.1

elseif  logický výraz  %větvení, může jich být libovolný počet
     příkazy
else
     příkazy
end
```

Příklad:

```
     if ( (a>68) && (a<80) )       % && odpovídá „a současně“
          c=1;
     else if ( (a>80) && (a<92) )
          c=2;
     else
          c=0;
     …
```

Tento příklad lze použít např. při porovnání x-ových souřadnic pozice myši pro jednotlivé oblasti GUI okna, kde proměnná *a* obsahuje aktuální hodnotu x-ové souřadnice kurzoru myši vzhledem k oknu.

6.2 MNOHONÁSOBNÉ VĚTVENÍ SWITCH

Syntaxe:

```
switch výraz
     case výraz
          příkaz, …, příkaz
     case {výraz1,výraz2,výraz3,…}
          příkaz, …, příkaz
     …
     otherwise
          příkaz, …, příkaz
end
```

Pokud tedy existuje proměnná *výraz* za *switch*, pak se vezme jeho obsah a porovnává se s výrazy, které jsou uvedeny za *case*. V případě shody (log.1) provede uvedené příkazy. Pokud se tak nestalo – nebyla nalezena shoda (log.0), provedou se příkazy ve větvi *otherwise*. Pokud existuje více shod, MATLAB provede první, kterou najde (v pořadí jak jsou podmínky *case* napsány), a zbytek přeskočí. Výrazem za slovem *switch* může být řetězec nebo číslo libovolného numerického datového typu.

Příklad:

```
nahodneCislo=ceil(rand(1)*21)-1;

switch nahodneCislo
     case 1
          disp(`nahodne cislo ma hodnotu: `);
          disp(`1`);
     case 5
          disp(`nahodne cislo ma hodnotu: `);
          disp(`5`);
     otherwise
          disp(`Promenna NahodneCislo ma jinou hodnotu`);
end
```

6.3 CYKLUS WHILE

Syntaxe:

```
while logický výraz
     příkazy
end
```

Pokud tedy logický výraz *while* nabývá log.1, pak se cyklicky provádí příkazy, dokud tato podmínka platí. Pokud neplatí, cyklus je ukončen. Je důležité ošetřit, aby nedošlo k zacyklení – logický výraz za *while* by měl někdy nabýt hodnoty log.0. Další možností ukončení tohoto cyklu je pomocí příkazu *break* uvnitř těla cyklu – např. v podmínce *if*. [3]

Příklad:

```
a = 0;
c=1;
while a<20
     c=c*a;
     if c>40
          break
     else
     c=c-1;
     end
     a=a+1;
end
disp(`Konec cyklu while.`)
```

6.4 CYKLUS FOR

Syntaxe:

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

Cyklus *for* provádí příkazy uvnitř cyklu se zadaným počtem opakování. Pro předčasné ukončení cyklu lze použít příkaz *break*, který může být uvnitř cyklu v podmínce *if*. Výraz může být libovolný vektor čísel, nesmí být však komplexní.

Příklad:

```
index=0;
for i=0:0.1:10,
t(index)=I;
sinus(index)=sin(i);
end
plot(t,sinus)       % vykreslení
```

7 POLE A MATICE

MATLAB je navržen tak, že se u něj uvažuje maximálního použití polí,matic a vektorů.Pole je v MATLABu nejdůležitější „stavební kámen“, protože matice, vektor a skalár jsou jeho speciální případy. Vyplývá to ze způsobu uložení v paměti. Pole je tedy sada čísel, které se

nazývají elementy pole, na které se odkazujeme jedním nebo více indexy různých množin indexů. Dimenze pole je počet indexů potřebných pro specifikaci elementu pole. Velikost pole je seznam velikostí množin indexů. Maticí může být například dvojdimenzionální pole, pro které platí speciální pravidla pro přidání elementu, násobení a další operace. Její dvě dimenze se označují jako řádek a sloupec. Vektorem označujeme matici, jejíž jedna dimenze má index 1.

7.1 VYTVÁŘENÍ POLÍ

Příklad:

```
pole = [1 2 3 4 5 6 7 8 9]

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

vektor = 0:0.5:10
```

Často používané příkazy pro práci s maticemi:

size – velikost každé dimenze
length – velikost nejdelší dimenze
ndims – počet dimenzí
find – vrací indexy nenulových prvků
eye – jednotková matice
ones – matice jedniček
zeros – matice nul
diag – diagonální matice
triu – horní trojúhelníková matice
tril – dolní trojúhelníková matice
rand, randn – náhodná čísla

7.2 INDEXACE MATIC A POLÍ

Matlab indexuje od jedničky, nikoliv od nuly, jak tomu je u jiných programovacích jazyků.

Příklad:

```
matice =[
     10   11   12   13   14   15
     16   17   18   19   20   21
     22   23   24   25   26   27
]
```

a) Indexace matice podle dimenze

```
matice(2,4)

ans =

19
```

b) Indexace matice po prvcích

```
matice(12)

ans =

21
```

c) Pomocí dvojtečky a end

```
matice(2,:)         %vypíše druhý řádek matice
matice(:,4)         %vypíše čtvrtý sloupec matice
```

7.3 ZÁKLADNÍ OPERACE S POLI

Příklady:

```
A = [3 6; 4 5]
B = [2 7; 9 6]
```

a) Součet matic

```
C = A + B

C =
     5    13
    13    11
```

b) Rozdíl matic

```
C = A - B

C =
     1    -1
    -5    -1
```

c) Maticové násobení

```
C = A * B

C =
    60    57
    53    58
```

d) Násobení po prvcích

```
C = A .* B

C =
     6    42
    36    30
```

e) Dělení matice zprava

```
C = A / B

C =
    0.7059         0.1765
    0.4118         0.3529
```

f) Dělení matice zleva

```
C = B \ A

C =
    0.1961        -0.0196
    0.3725         0.8627
```

g) Dělení po prvcích

```
C = A ./ B

C =
    1.5000         0.8571
    0.4444         0.8333
```

h) Maticové umocnění

```
C = A ^ 2

C =
     33   48
     32   49
```

i) Umocnění po prvcích

```
C = A .^ 2

C =
      9    36
     16    25
```

8 OPERACE S ŘETĚZCI

Příklady:

```
retezec = 'Tento text je retezec.';
```

a) Úprava části řetězce

```
retezec = strrep(retezec,'je','je upraveny')
retezec =

Tento text je upraveny řetězec.
```

b) Hledání v řetězci

```
strfind(retezec,'e')

ans =

     2     8    13    20    25    27    29
```

nebo

```
strfind(retezec,'upraveny')
% v tomto případě vrátí počáteční souřadnici hledaného slova
%tedy souřadnici písmene u

ans =

    15
```

c) Porovnání řetězců

```
retezec = strcmp('str1', 'str2')
retezec = strcmp('str1', retezec2)
retezec = strcmp(retezec1, retezec2)
```

Příkaz *strcmp* vrací log.1 v případě rovnosti, jinak vrací log.0. [1]

```
retezec = strncmp('str1', 'Str2',cislo)
retezec = strncmp('str1', retezec2,cislo)
retezec = strncmp(retezec1, retezec2,cislo)
```

Příkaz *strncmp* slouží k porovnání části dvou řetězců, kde *cislo* je počet znaků řetězce zleva, které se mezi sebou budou porovnávat. Tento příkaz rozlišuje velká a malá písmena. Vyhodnocení je stejné jako u *strcmp*.

Příkaz *strncmpi* velikost písmen nerozlišuje.

d) Indexace řetězce

```
retezec(7:10)

ans =

text
```

nebo

```
retezec(7:end);
```

9 GUI OBJEKTY - UICONTROL

Mezi tyto objekty patří:

- Check box
- Editable text field
- Frames
- List box
- Pop-up menu
- Push button
- Radio button

- Slider
- Static text label
- Toggle button

Tyto objekty lze tvořit automaticky pomocí příkazu *GUIDE* nebo manuálně – viz níže. [8]

Příklady:

```
uicontrol(nFigure,...
    'Style','checkbox',...
    'Value',1,...
    'Position',[.8 .4 .2 .05],...
    'Units','Normalized',...
    'Callback','checkedF',...
    'Tag','osy') ,...
    'String','Osy');

uicontrol(nFigure,...
    'BackgroundColor',[1 1 1],...
    'Units','characters',...
    'Style','edit',...
    'Position',[77 18 20 2],...
    'Tag','Lic_typ',...
    'String','Editovatelný text');

uicontrol(nFigure,...
    'Style','popupmenu',...
    'String','oranzova|seda|modra|zelena|cerna',...
    'Position',[.8 .3 .2 .05],...
    'Units','Normalized',...
    'Callback','vyber',...
    'Tag','menu');

uicontrol(nFigure,...
    'BackgroundColor',[1 1 1],...
    'Style','text',...
    'Position',[0.5 21 22.5 1],...
    'Tag','text1',...
    'FontSize',8,...
    'String','Zobrazovaný text.');

uicontrol(nFigure,...
    'BackgroundColor',[1 1 1],...
    'Style','pushbutton',...
    'Position',[145 4 30 4],...
    'Tag','tlac1',...
    'Callback','nabidka',...
    'String','Tlacitko');
```

Vytvoření tří radiobuttonů v jedné skupině:

```
radiogr = uibuttongroup('BackgroundColor',[1 1 1] ,...
            'visible','off',...
            'Position',[0.0 0 0.25 1]);

r1 = uicontrol('BackgroundColor',[1 1 1] ,...
       'Style','Radio','String','4x4',...
       'pos',[10 230 60 30] ,...
       'parent',radio_group2,...
       'HandleVisibility','off');

r2 = uicontrol('BackgroundColor',[1 1 1] ,...
       'Style','Radio',...
       'String','6x6',...
       'pos',[10 200 60 30] ,...
       'parent',radio_group2,...
       'HandleVisibility','off');

r3 = uicontrol('BackgroundColor',[1 1 1] ,...
       'Style','Radio',...
       'String','8x8',...
       'pos',[10 170 60 30] ,...
       'parent',radio_group2,...
       'HandleVisibility','off');

set(radiogr,'SelectionChangeFcn',@plocha);
set(radiogr,'SelectedObject',r2);  % Defaultni vyber
set(radiogr,'Visible','on');
```

Tyto radiobuttony jsou v jedné skupině – tj. jen jeden radiobutton může být aktivní. [6]

10 Funkce

Funkce je M-soubor, který začíná slovem *function* a její název je stejný jako název M-souboru. Funkce má vůči skriptu mnoho výhod. Při zavolání se zpracuje, zkompiluje celá najednou. Může mít vstupní a výstupní parametry. Proměnné funkce se ukládají do pracovního prostoru této funkce a jsou tak chráněny proti zásahu z jiných funkcí, skriptů či *Command Window*. Po ukončení funkce se tyto lokální proměnné odstraní z paměti. Voláme-li funkci z *Command Window* nebo z jiného M-souboru, pak je rozebrána na příkazy a uložena v paměti MATLABu do té doby, než se zadá příkaz *clear* nebo než se ukončí MATLAB.

Příklad:

function [prvniVystup,varargou]=vykresleni(prvniVstup,varargin)

```
if nargin == 1
     sinus=sin(prvniVstup);
     plot(prvniVstup,sinus,'r')
     prvniVystup=prvniVstup;
     varargout{1}=sinus;
elseif nargin > 1
     [prvniVystup,varargout]=ostatniVykresleni(prvniVstup,varar
gin);

else
     Error('Vyskytla se chyba - asi jste nezadali vstupni
parametry funkce.')

end
```

II. PRAKTICKÁ ČÁST

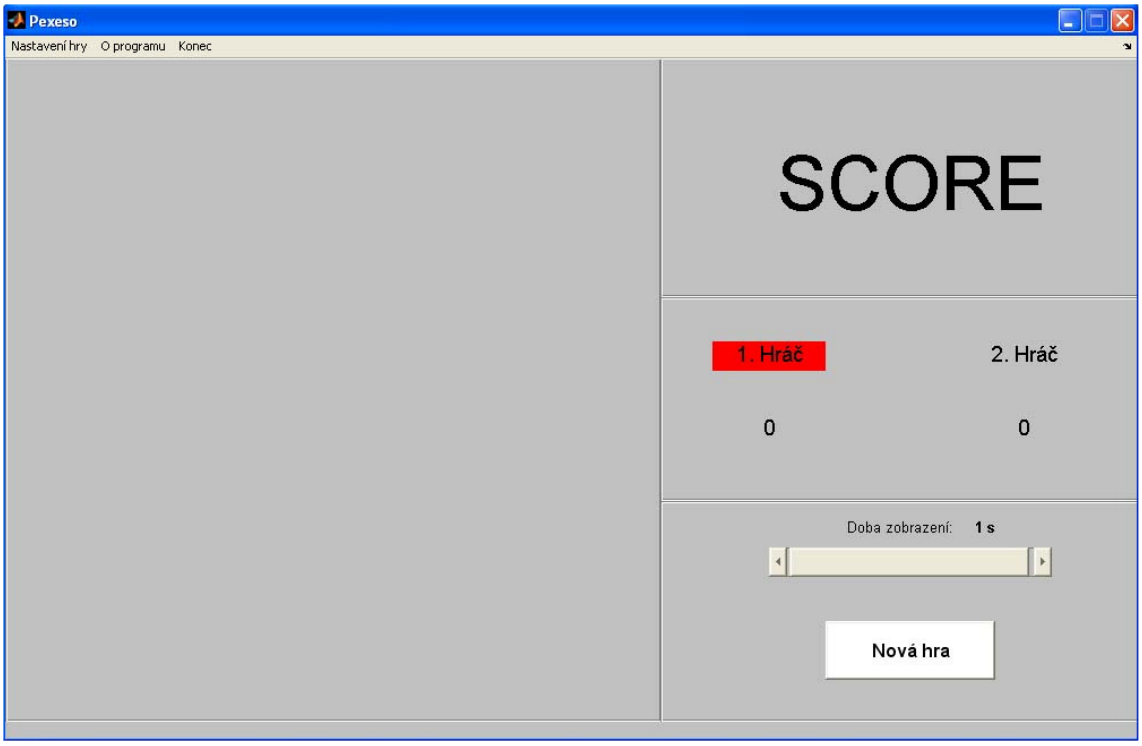
11 O PROGRAMU – HŘE PEXESO

Program byl vytvořen v programu MATLAB - verze 7.5.0.342 (R2007b) bez použití GUIDE. Program je optimalizován pro zobrazení 1024x768 a 1280x800. Hra je určena pro dva hráče. Spouští se souborem *pexeso.m* a obsahuje tři uživatelská okna.

Jsou to:

a) Hlavní okno
b) Okno nastavení hry
c) Okno "O programu"

11.1 HLAVNÍ OKNO



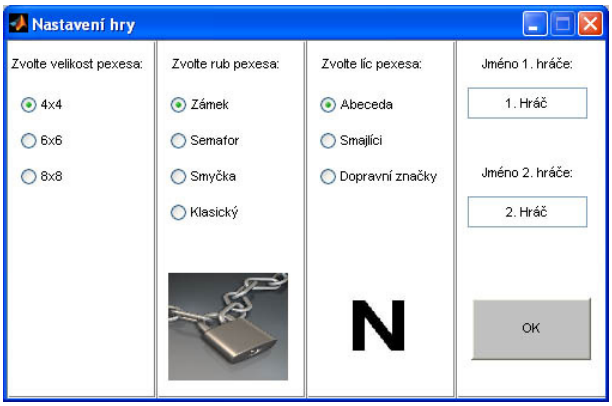
Obrázek 2: Hlavní okno po spuštění programu

Aktuální hráč, který je na tahu je označen červeným obdélníkem.

Toto hlavní okno se skládá z:

- Vlastního menu (*Program,Nastavení hry, O programu*)
- Hrací plochy (levá část)
- Informační části (stav počtu bodů jednotlivých hráčů)
- Posuvníku určujícího dobu zobrazení líců
- Tlačítka *Nová hra*

11.2 OKNO - NASTAVENÍ HRY

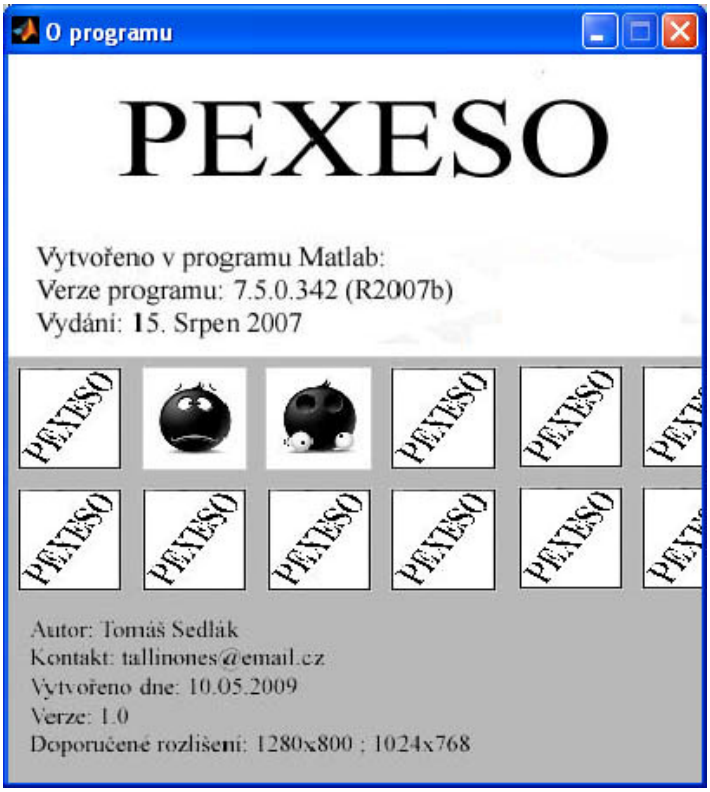


Obrázek 3: Okno nastavení hry

V tomto okně je možné jak měnit velikost pexesa, tak i jeho celkový vzhled. Jako výchozí je nastaveno pexeso o velikosti 4x4, tj. 2x8 karet, rub v podobě visacího zámku a líc s motivem abecedy. Změna rubu nebo líce má za následek vykreslení daného rubu nebo líce pod výběr – viz *obrázek 3*. Ruby a líce je možné kombinovat spolu s velikostí hrací plochy pexesa.
Jednotlivé motivy rubů a líců jsou umístěny v adresáři: *\images\Líc* nebo *\images\Ruby* , je tedy možné je měnit s tím, že se u obrázků doporučuje zachovat rozlišení 130x150 pixelů, resp. odpovídající poměr stran. Pokud bude poměr stran odlišný, bude zobrazený obrázek zkreslený.
Dále je možné zadat jména hráčů a potvrdit tlačítkem *OK*, čímž se vykreslí hrací plocha do levé části hlavního okna – viz *obrázek 2*.

11.3 OKNO - O PROGRAMU

V tomto okně je vykreslen obrázek *info.jpg* popisující program, ve kterém byla hra PEXESO vytvořena, dále pak autora, kontakt na autora, verzi a datum vytvoření pexesa. Po kliknutí se okno samo zavře.



Obrázek 4: Okno "O programu"

12 STRUKTURA PROGRAMU

Program má celkem 3 okna:

a) Hlavní okno
b) Okno s nastavením hry
c) Okno "O programu"

Každé okno je vytvořeno prvkem *figure*, který dané okno popisuje (velikost okna, jeho jméno, menu a další) . [2]

Ukázka z kódu – pexeso.m :

```
hFigure = figure('Units','pixels',...
    'Position',[10 100 1000 600],... %velikost okna
    'MenuBar','None',...
    'Name','Pexeso',... %jméno hlavního okna
    'Tag','HlavniOkno',...
    'NumberTitle','off',...
    'Resize','off',... %zvetsovani a zmensovani hlavního okna zakázano
    'Visible','on',...
    'BackingStore','off',...
    'Renderer','painters',...
    'DoubleBuffer','on');
```

12.1 STRUKTURA – HLAVNÍ OKNO

Hlavní okno obsahuje tyto části:

- Vlastní menu
- Textové objekty
- Hrací pole
- Posuvník
- Tlačítko *Nová hra*

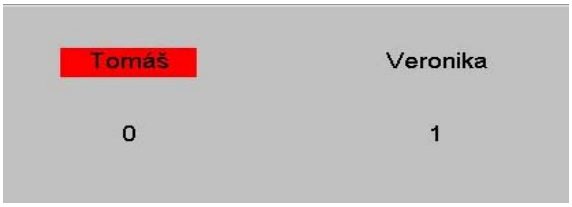
12.1.1 Vytvoření vlastního menu

Ukázka z kódu – pexeso.m :

```
uimenu('Label','Nastavení hry','Callback','mbar1_open');
uimenu('Label','O programu','Callback','mbar2_open');
uimenu('Label','Konec','Callback','exit');
```

Ukončení programu je možné kliknutím na *Konec* v menu okna nebo kliknutím na křížek v pravém horním rohu okna.

12.1.2 Textové objekty



Obrázek 5: GUI objekty typu *text*

Jména hráčů a počet nalezených dvojic jsou *string* řetězce GUI objektu typu *text* – viz *pexeso.m* nebo Kapitola 9.

Řetězce jmen hráčů jsou aktualizovány po stisknutí tlačítka *OK*, které je okně s nastavením hry. Přiřazení viz soubor *zavreni.m* .

Řetězce počtu nalezených dvojic pro jednotlivé hráče se aktualizují ve chvíli, kdy jsou nalezeny dvě stejné hrací karty – viz soubor *odkryj.m* .

Pokud aktuální hráč nenajde 2 stejné karty, stane se aktuálním hráčem druhý hráč.

Ukázka z kódu – odkryj.m :

```
if (isequal(srov1,srov2))==1 % pokud jsou odkryté karty stejné,
            pause(hodnota_posuv); %pauza je dána hodnotou posuvníku
            set(hFigure,'CurrentAxes',dyn_axes(umisteni1));
            image(vymazane); %obsahuje obrazek v barve pozadi plochy
            axis off
            set(hFigure,'CurrentAxes',dyn_axes(umisteni2));
            image(vymazane);
            axis off
            click=0;
            srov1=0;
            srov2=0;

            if (strcmp(active_player,'p1')==1) %přičtení bodu prvnímu hráči
                p1_score=p1_score+1;
                set(score_hrac1,'String',num2str(p1_score));
            else                        %jinak přičte bod druhému hráči
                p2_score=p2_score+1;
                set(score_hrac2,'String',num2str(p2_score));
            end
...
else
...
end
```

Proměnné *srov1* a *srov2* obsahují cestu k obrázku, na který se kliklo.

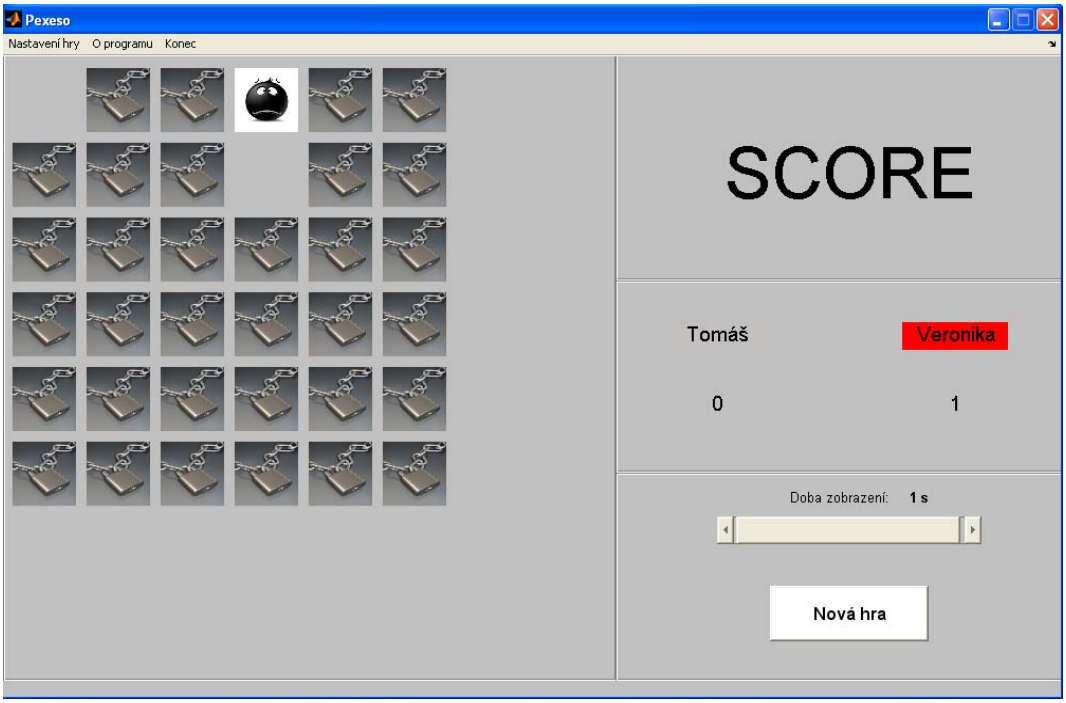
Proměnná *dyn_axes()* obsahuje pole os - *axes*, do kterých se vykreslují ruby, líce nebo pozadí a *umisteni1* a *umisteni2* udává, které obrázky/osy se mají překreslit – tj. „vymazat“.

Proměnné p1_score a p2_score tedy obsahují aktuální skóre hráče v číselném formátu.

12.1.3 Hrací pole

Tato oblast obsahuje dynamicky vytvořené osy – *axes*, do kterých jsou vykreslovány obrázky (ruby,líce nebo pozadí) v závislosti na tom, v jakém stavu se program nachází. Počet těchto os je dán velikostí hrací plochy, která se volí v okně s nastavením hry. Pozice obrázků pexesa se při každém spuštění hry mění, tj. hra je pokaždé „unikátní“.

Ukázka ze hry:



Obrázek 6: Ukázka ze hry

Ukázka z kódu – vykresli.m :

```
iter_axes=1;

while (s_ctverce)>0, %strana hraci plochy pro 4x4 je tedy 4
    dyn_axes(iter_axes)=axes('Units','pixels',...
        'Position',[xpoc+xposun ypoc velx vely],...
        'Tag','PoziceX');
    image(rub,'ButtonDownFcn','ktery');
    axis off
    iter_axes=iter_axes+1;
    y_iter=st_ctverce-1; %pomocna promenna pro indexaci sloupcu

    while (y_iter)>0,
        dyn_axes(iter_axes)=axes('Units','pixels',...
        'Position',[xpoc+xposun ypoc+yposun velx vely],...
        'Tag','PoziceY');
        image(rub,'ButtonDownFcn','ktery');
        axis off
        iter_axes=iter_axes+1;
        y_iter=y_iter-1;
        yposun=yposun-70;
    end

y_iter=st_ctverce-1;
s_ctverce=s_ctverce-1;
xposun=xposun+70;
yposun=-70;
end
```

Uvedený kód plní pole *dyn_axes()* a vytváří osy (*axes*) v hrací ploše v závislosti na velikosti hracího pole a rovnou vykresluje do os zvolený rub. Pokud tedy klikneme na rub, provede se skript souboru *ktery.m* , jenž zjistí, na který řádek a sloupec se kliknulo. Toto je provedeno pomocí příkazu `get(gca,'Currentpoint');` která vrací pozici myši (x,y souřadnice) v době kliknutí na rub. [9]

Pozn.: Volání funkce je v souboru *pexeso.m*, reaguje na každé kliknutí, nicméně vrací hodnotu pouze po kliknutí na jednotlivé osy.

Ukázka z kódu – pexeso.m :

```
set(gca,'buttondownfcn',@ktery);
```

Pozor, osy jsou číslovány po sloupcích, to znamená, že při velikosti hracího pole 4x4 je pátá osa v druhém sloupci, prvním řádku, kdežto při velikosti hracího pole 6x6 je pátá osa v prvním sloupci, pátém řádku. Proto je nezbytné zjišťovat, na kterou osu se kliknulo.

Porovnáním souřadnic *x,y* (viz soubory *ctyri.m*, *sest.m* a *osm.m*) získáme číslo osy, na kterou se kliknulo (proměnná *d_axes*).

Získal jsem tedy číslo osy, na kterou se kliknulo, řádek a sloupec této osy. Zbývá jen přiřadit správný líc.

Proto jsem dynamicky vytvořil matici, která odpovídá velikosti zadaného hracího pole. Tato matice obsahuje čísla od 1 do čísla udávající počet dvojic hracích karet - tj. 4x4 odpovídá 2x8 dvojic. Každé číslo je v matici logicky dvakrát. Tato čísla odpovídají názvu souborů jednotlivých líců (*\images\Líc\Abeceda**1***.jpg*).

Ukázka z kódu – vytvořená matice 4x4:

```
matice =

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

Přiřadil jsem tedy řádek a sloupec kliknutého rubu k řádu a sloupci (výše uvedené) vytvořené matice a získal tak číslo souboru líce. Jednoduchou úpravou pak celou cestu k souboru líce:

Ukázka z kódu – odkryj.m :

```
prvek=matice(klik_radek,klik_sloupec);
prvekstr=num2str(prvek);
prvek=['images\Líc\',TYP_LICE,'\',prvekstr,'.jpg']; %cesta k líci
...
...
set(hFigure,'CurrentAxes',dyn_axes(d_axes));
lic=imread(prvek);
image(lic);
axis off
```

Tímto je karta „obrácena“ a zobrazen líc.

Obrázky pro líc použity z internetových zdrojů – viz [4], [5].

Líc zůstává zobrazen do doby, než jsou zobrazeny líce dva. Poté se vyhodnotí, zda jsou či nejsou líce stejné a zůstanou zobrazeny po dobu danou hodnotou posuvníku v hlavním okně.

Ošetření pro nedočkavé hráče

Pokud by hráč kliknul rychleji, než karty zmizí, zobrazil by se další, třetí, čtvrtý… atd. líc.
Aby se tomuto předešlo, je třeba hlídat počet kliknutí - proměnné *click* a *iter_shody*.
Proměnná *click* se inkrementuje pokaždé, když je kliknuto na rub karty, což zastaví chod programu, pokud bylo kliknuto více než dvakrát – viz *ktery.m* .
Proměnná *iter_shody* slouží spolu s proměnnou *click* k správnému uložení porovnávacích proměnných daného líce a osy u prvního a druhého kliknutí.

Ukázka z kódu – odkryj.m :

```
if (iter_shody==2)||(iter_shody>2)

    iter_shody=0;
    srov_pole=0;
    srov_pole2=0;

    else
end

...
...
...

if (iter_shody==0)&&(click==1)
    srov1=prvek;           %proměnná obsahující cestu k 1.lícu
    umisteni1=d_axes;       %cislo udavajici adresu axes 1. klik. Obrazku

else if iter_shody==1
        srov2=prvek;      %proměnná obsahující cestu k 2.lícu
        umisteni2=d_axes;  %cislo udavajici adresu axes 2. klik. obrazku
…
…
```

následuje vymazání (hráč uhodl) a přičtení bodu aktivnímu hráči, nebo znovu zobrazení líce (hráč neuhodl)

```
…
…

iter_shody=iter_shody+1;

else

end
```

12.1.4 Překreslení hracího pole

Překreslení hracího pole je potřeba v případě, že hráči nejdříve zvolí větší a poté menší velikost hracího pole. Pokud by se pole nepřekreslilo, zůstaly by na hrací ploše osy, resp. obrázky z minulé hry.

Ukázka z kódu – vymaz.m :

```
vel_vykr_pole=length(dyn_axes); %velikost pole os

while(vel_vykr_pole)>0,          %pracuje dokud je co prekreslovat

    %nastaví se focus na danou osu:
    set(hFigure,'CurrentAxes',dyn_axes(vel_vykr_pole));
    image(vymazane); %vykreslí se barva pozadí - „vymaze“ hrací plochu
    axis off

    % odstranění funkce kliknutí u všech obrázků
    set(dyn_axes(vel_vykr_pole),'ButtonDownFcn','');
    vel_vykr_pole=vel_vykr_pole-1; %iterator -1 (prekresl. se zezadu)

end
```

12.1.5 Posuvník

Posuvník je GUI objektem určujícím jak dlouho bude dvojice líců karet odryta, ať už hráč dvojici uhodne, či ne.

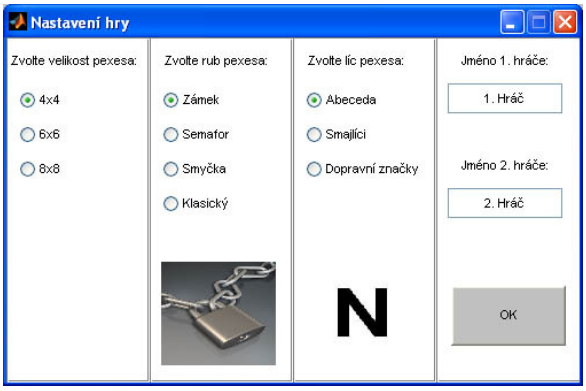
Rozmezí je volitelné od 0,5s do 2s po dvou krocích:

a) Krok posuvu šipkou – 0,1s
b) Krok posuvu táhlem – 0,25s

12.1.6 Tlačítko - Nová hra

Stisk tohoto tlačítka vyvolá okno s nastavením hrací plochy.

12.2 STRUKTURA OKNA – NASTAVENÍ HRY



Obrázek 7: Okno nastavení hry

Okno na obrázku č.7 se zobrazí po kliknutí na tlačítko *Nová hra* v hlavním okně nebo po kliknutí na *Nastavení hry* v menu hlavního okna.

Je ošetřeno, aby se toto okno otevřelo v daný moment maximálně jednou a to:

- vypnutím tlačítka *Nová hra* ve chvíli, kdy se toto okno otevře
- proměnnou *nabidka_otevrena* , která nabývá log. 1 ve chvíli, kdy je okno otevřeno

Ukázka z kódu – nabidka.m :

```
set(button_odeslat,'Callback','');
nabidka_otevrena=1;
```

Po zavření tohoto okna se tlačítku vrátí jeho *Callback* a proměnná *nabidka_otevrena=0* (viz *mbar1_open.m*, *closewindow.m* a *zavreni.m* – v závislosti na tom, jak je okno zavřeno).

Okno nastavení obsahuje tyto části:

- Textové objekty
- Objekty *radiobutton*
- Zobrazovací osy
- Objekty typu *edit*
- Tlačítko *OK*

12.2.1 Textové objekty

Textové popisky nad jednotlivými *radiobuttony* a *edit* GUI objekty jsou *string* řetězce GUI objektu typu *text* – viz *nabidka.m* nebo Kapitola 9.

12.2.2 Objekty typu - radiobutton

Tyto objekty jsou tvořeny pomocí příkazu *uibuttongroup*. Pod ním je pak vytvořena skupina jednotlivých radiobuttonů – jež sou jeho potomky.

Ukázka z kódu – nabidka.m :

```
radio_group2 = uibuttongroup('BackgroundColor',[1 1
1],'visible','off','Position',[0.0 0 0.25 1]);
radio44 = uicontrol('BackgroundColor',[1 1
1],'Style','Radio','String','4x4',...
    'pos',[10 230 60 30],'parent',radio_group2,'HandleVisibility','off');
radio66 = uicontrol('BackgroundColor',[1 1
1],'Style','Radio','String','6x6',...
    'pos',[10 200 60 30],'parent',radio_group2,'HandleVisibility','off');
radio88 = uicontrol('BackgroundColor',[1 1
1],'Style','Radio','String','8x8',...
    'pos',[10 170 60 30],'parent',radio_group2,'HandleVisibility','off');

set(radio_group2,'SelectionChangeFcn',@plocha); % funkce při změně
set(radio_group2,'SelectedObject',radio44);  % Defaultni vyber
set(radio_group2,'Visible','on');
```

Při změně výběru jednotlivého radiobuttonu se spustí funkce plocha, která se nachází v souboru *plocha.m*. Výše uvedený kód slouží pro výběr velikosti hrací plochy.

Ukázka z kódu – plocha.m :

```
vel_plochy=get(get(source,'SelectedObject'),'String');
vel_plochy=vel_plochy(1);
```

Pomocí tohoto kódu získám řetězec *string* vybraného radiobuttonu (např. '4x4').
Protože jde o funkci, zvolil jsem použití globálních proměnných pro uložení výsledku funkce.

Pokud jde o radiobuttony pro zvolení rubu a líce, vykreslí se pod nimi zvolený rub nebo příklad líce.

Ukázka z kódu – rub.m :

```
rub_axes=axes('Position',[0.27 0.05 0.2 0.3]);
rub_image=imread(TYP_RUBU);
image(rub_image); %vykreslení rubu pod radiobuttony
axis off
```

12.2.3 Zobrazovací osy

Do těchto os se vykreslují zvolené ruby nebo líce z radiobuttonů – viz Kapitola 11.2.2.

Ukázka z kódu – lic.m :

```
ran4=num2str(ceil(rand(1)*32)); %náhodné číslo od 1 do 32 převedené do
datového typu string

lic_axes=axes('Position',[0.52 0.05 0.2 0.3]);
lic_image=['images\Líc\',TYP_LICE,'\',ran4,'.jpg'];
lic_image=imread(lic_image);
image(lic_image);
axis off
```

Použitím tohoto kódu se vykreslí náhodný líc zvoleného typu.

12.2.4 Objekty typu - edit

Do těchto objektů je možno psát, tj. zadávat jména hráčů. Ta se po stisknutí tlačítka *OK* a zavření okna s nastavením zobrazí v hlavním okně.

Ukázka z kódu – zavreni.m :

```
jmeno1=get(p1_value,'String');
jmeno2=get(p2_value,'String');

close gcf %zavře aktuálně otevřené okno (Nastavení hry)

...
...

%aktualizace jmen hráčů v hlavním okně:
set(hrac1,'String',jmeno1);
set(hrac2,'String',jmeno2);
```

12.2.5 Tlačítko OK

Stisknutím tohoto tlačítka se uloží nastavení a vykreslí hrací plocha pexesa.

12.3 STRUKTURA OKNA - O PROGRAMU

Toto okno obsahuje se zobrazí po kliknutí na tlačítko *O programu* v menu hlavního okna. Zobrazí se pouze osy - *axes*, do kterých je vykreslen obrázek se všemi popisky. Po kliknutí na tento obrázek se okno samo zavře – viz soubor *about.m.*

Ukázka z kódu – about.m :

```
oFigure = figure('Units','pixels',...
      ...
      ...
      'CloseRequestFcn','delete(gcf);info_otevreno=0;',...
      'DoubleBuffer','on');

info_otevreno=1;
info_plocha_axes=axes('Position',[0 0 1 1]);
info_plocha=imread('images\info.jpg');
image(info_plocha,'ButtonDownFcn','close gcf;info_otevreno=0'); %vykreslení
pozadí
axis off %vypne popisky os
```

Zavřít toto okno lze tedy dvěma způsoby:

- Kliknutím na křížek v pravém horním rohu obrazovky - *CloseRequestFcn*
- Kliknutím na obrázek samotný – *ButtonDownFcn*

Je ošetřeno, aby se toto okno otevřelo v daný moment maximálně jednou a to:

- proměnnou `info_otevreno` , která nabývá log. 1 ve chvíli, kdy je okno otevřeno

Po zavření je proměnná `info_otevreno` nastavena na hodnotu 0, čímž je možné toto okno znovu otevřít.

13 SESTROJENÍ MATICE ZOBRAZENÍ

Proměnná *pocet_paru* má hodnotu danou polovinou velikostí pole (počtem párů).

Postup:

1) Naplnit pole od velikosti počtu párů do hodnoty 1
2) Zdvojit toto pole
3) Náhodně prohodit prvky v poli – tímto se stane matice “unikátní“
4) Rozdělit pole do řádků a naplnit s nimi matici

Ukázka z kódu – vykresli.m :

```
pole=pocet_paru; %nachystani promenne (pole)
pocet_paru1=pocet_paru-1;

while pocet_paru1>0
    pole=[pole pocet_paru1]; %pridani noveho cisla (a=a-1)do pole b
    pocet_paru1=pocet_paru1-1;
end

pole=[pole pole];    %zdvojeni pole - nachystani dvojic

while ran3>0 %cyklus prohazovani prvku v poli
    ran1=ceil(rand(1)*pocet_paru*2); %generuje nahodne cislo od 1 do vel.
pole (pouzito pro prvni souradnici presunu v poli)
    ran2=ceil(rand(1)*pocet_paru*2); %generuje nahodne cislo od 1 do vel.
pole (pouzito pro druhou souradnici presunu v poli)
    saved1=pole(ran1); %hodnota na pozici ran1 v poli b (slouzi jako
zaloha)
    saved2=pole(ran2); %hodnota na pozici ran2 v poli b (slouzi jako
zaloha)
    pole(ran1)=saved2;
    pole(ran2)=saved1;
    ran3=ran3-1;
end

%disp(pole); % zobrazení zamíchaného pole

radek=pole(prv1:pexwidth*prv1); %prvni radek matice
matice=[radek];                 %vytvoreni matice

while (pexwidth*iter)<(2*pocet_paru)
radek=pole(pexwidth*iter+1:pexwidth*(iter+1));
matice=[matice; radek]; %sestavení matice
iter=iter+1;
end
```

Po skončení tohoto kódu matice obsahuje čísla od 1 do hodnoty danou počtem párů hracích karet pexesa. Tímto je dána indexace líců vzhledem k hracímu poli.

ZÁVĚR

V této práci jsem se snažil ukázat, že MATLAB není jen program určený pro vědeckotechnické numerické výpočty, modelování, návrhy algoritmů, počítačové simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů a další, ale je schopný také vytvoření jednoduché grafické hry, která bude stejně dobře hratelná jako jiné hry vytvořené například v prostředí Flash nebo jazyku C.

Zároveň jsem se snažil v teoretické i praktické části shromáždit potřebné znalosti, nápady a řešení potřebné k téměř bezproblémovému vytvoření této hry nebo hry velmi podobné této.

Program MATLAB je tedy k vytvoření této hry vhodný, i když jsem při řešení narazil na problémy, na které jsem v dokumentaci jen stěží hledal přímé řešení – většinou bylo řešením několik jednotlivých kroků.

Bohužel je tato hra a celkově program MATLAB veřejnosti téměř nepřístupný z důvodu vysoké ceny licence (několik desítek tisíc Kč) a je dostupný většinou pouze studentům nebo zaměstnancům firem.

Proto očekávám, že celá tato bakalářská práce bude sloužit především studentům jako vhodný studijní materiál pro budoucí práce a výuku (zdrojový kód je velmi dobře okomentován).

CONCLUSION

In this bachelor thesis I have tried to prove, that MATLAB is not only a programme used for scientific-technical numerical calculations, modelling, algorithm schemes, computer simulations, data analysis and presentation, signal measuring and compiling, schemes of controlling and comunation systems etc. , but is also capable to create a simple graphic game, which would be playable just as other games created in different programming languages or structures - for example in Flash environment or C language.

In teoretical and practical part I have also tried to gather necesary knowledge, ideas and solutions needed for nearly trouble-free creation of this game or a game which is similar to this one.

Even though MATLAB is suitable to create this game, I have found some difficulties during the program forming, for which was really hard to find direct solution in the documentation – so the solution was generally several steps.

However, this game and whole MATLAB is unavailable to public users because of the licence price of MATLAB (several ten thousands of CZK) and is available mostly for schools, students or employees.

So because of that, I expect that this whole bachelor thesis would above all serve as a useful studying material to students for future labours and education (source cod eis well-commented).

SEZNAM POUŽITÉ LITERATURY

Monografie:

[1] DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB pro začátečníky. 1.vydání. Praha: BEN – technická literatura, 2003. 144 s. ISBN: 80-7300-095-4

[2] DOŇAR Bohuslav, ZAPLATÍLEK Karel. MATLAB – tvorba uživatelských aplikací. 1. Vydání. Praha: BEN – technická literature, 2004. 216 s. ISBN: 80-7300-133-0.

[3] PERŮTKA, Karel. MATLAB – Základy pro studenty automatizace a informačních technologií. 1.vyd. Zlín: UTB ve Zlíně, 2005. 304 s. ISBN 80-7318-355-2.

Internetové zdroje:

[4] [FREE-RELAX.RU] (66) – obrázky líců pexesa [online]. [cit. 2009-05-12]. Dostupný z WWW: http://www.flickr.com/photos/25094548@N07/2915043482/in/photostream/

[5] Clip project .info – obrázky líců pexesa [online]. [cit. 2009-05-12]. Dostupný z WWW: http://www.clipproject.info/Clipart_Smilies_Seite_11.html

[6] MATLAB - Documentation [online]. Mathworks. [cit. 2009-04-10]. Dostupný z WWW: http://www.mathworks.com/access/helpdesk/help/techdoc/index.html?/access/helpdesk/help/techdoc/matlab.html&http://www.mathworks.com/access/helpdesk/help/helpdesk.html

[7] MATLAB [online]. Wikimedia Foundation. [cit. 2009-04-10]. Dostupný z WWW: http://cs.wikipedia.org/wiki/Matlab

[8] MATLAB GUI Tutorial. [online]. [cit. 2009-02-04]. Dostupný z WWW: http://www.matlabgui.com

[9] MATLAB CENTRAL File exchange, Section: Games. [online]. [cit. 2009-02-04]. Dostupný z WWW: http://www.mathworks.com/matlabcentral/fileexchange/?term=tag%3A%22games%22

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASCII	American Standard Code for Information Interchange
CD-ROM	Compact Disc-Read Only Memory
GUI	Graphical User Interface
JPG	Joint Photographic Group
MATLAB	MAtrix LABoratory
OS	Operating System

SEZNAM OBRÁZKŮ

OBRÁZEK 1: Výchozí okno po spuštění MATLABu…. 11
OBRÁZEK 2: Hlavní okno po spuštění programu……. 34
OBRÁZEK 3: Okno nastavení hry……. 35
OBRÁZEK 4: Okno "O programu"..... 36
OBRÁZEK 5: GUI objekty typu *text*…. 38
OBRÁZEK 6: Ukázka ze hry ……. 39
OBRÁZEK 7: Okno nastavení hry..... 44

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM (OBSAHUJE HRU PEXESO VYTVOŘENOU V PROGRAMU MATLAB SPOLU S TIMTO DOKUMENTEM)