

► Skládanka

Ročníková práce z programování

Tomáš Vitek; 2.E

Gymnázium Arabská, 2007 – 2008

Obsah

Obsah.....	2
Seznam obrázků, tabulek a úryvků zdrojových kódů	2
Úvod	3
Uživatelský popis	3
Průběh hry	3
Spuštění algoritmu pro výpočet nejkratší cesty	3
Klávesové zkratky	4
Popis prostředí hry	5
Programátorský popis	6
O tvorbě a spuštění aplikace	6
Popis datových struktur.....	6
Vyjádření souřadnic.....	6
Vyjádření vlastností hry	6
Popis jednotlivých algoritmů	8
Výpočet cesty a simulace	8
Ukládání a nahrávání souborů.....	8
Realizace tabulky nejlepších hráčů.....	9
Screenshoty aplikace na různých operačních systémech s různými vzhledy.....	9
Další informace.....	11
Závěr	11

Seznam obrázků, tabulek a úryvků zdrojových kódů

Tabulka klávesových zkratk	4
Obrázek uživatelského prostředí.....	5
Zdrojový kód struktury „souradnice“	6
Zdrojový kód struktury „vlastnostiHry“	7
Příklad souboru s uloženou hrou.....	8
Seznam adres v registru, kam aplikace zapisuje seznam nejlepších hráčů.....	7
Screenshoty aplikace na různých OS s různými vzhledy	9

Úvod

Jako téma své ročníkové práce z programování jsem si zvolil implementaci „**deskové hry**“. A to konkrétně hry Skládanka. V anglickém jazyce se hra jmenuje „**Click & Slide**“, raději než ji tedy překládat jako například „**Klikni a šoupi**“, jsem zvolil název nový a to právě „**Skládanka**“ (princip této hry naleznete níže).

Ovšem jako hlavní úkol při tvorbě aplikace k ročníkové práci jsem si předeslal najít a implementovat algoritmus pro nalezení nejkratší cesty k dokončení hry.

Uživatelský popis

Průběh hry

Hra začíná volbou „**Nová hra**“, či „**Ze souboru**“. Pokud uživatel zvolí volbu „**Ze souboru**“ je pomocí dialogu požádán o zadání adresy obrázku, který je použit ke hře, pokud zvolí pouze „**Nová hra**“, počítač automaticky náhodně jeden vybere ze čtyř výchozích obrázků. Po zvolení obrázku je uživatel požádán o vybrání obtížnosti (počtu políček). Má na výběr z několika přednastavených obtížností, nebo si může zvolit obtížnost vlastní. Nyní je obrázek nařezán na uživatelem zadaný počet políček, jedno z políček je odebráno (vždy se jedná o to pravé dolní) a poté jsou políčka promíchána a zobrazena uživateli na hrací desce. Úkol uživatele je klikáním na sousedy bílého vynechaného políčka (tím prohodí bílé políčko a jeho souseda) dosáhnout toho, aby byl obrázek složen do jeho původní podoby.

Během hry může uživatel využít pauzy - stisknutím tlačítka „**P**“, stejným tlačítkem také hru odpauzuje. V oblasti okna „**Informace**“ (viz níže rozložení uživatelského prostředí) se uživateli zobrazují informace o dané hře – čas a počet tahů, které zatím odehrál.

Uživatel má také možnost využít možnosti zobrazit si zmenšený náhled původního obrázku (pomocí tlačítka „**Náhled**“).

Spuštění algoritmu pro výpočet nejkratší cesty

Spuštění algoritmu pro výpočet nejkratší cesty probíhá pomocí stisknutí tlačítka „**Pomoc!**“, které uživateli otevře dialog, v kterém po potvrzení probíhá výpočet cesty. Pokud uživatel zvolil nejjednodušší obtížnost (3 x 3 políčka), výpočet proběhne téměř okamžitě. Pokud ovšem uživatel zvolil těžší obtížnost, bude upozorněn, že výpočet může trvat dlouho. Při výpočtu se zobrazuje průběh pomocí „**nahrávacího pruhu**“, kdy ovšem celé zaplnění pruhu neznámá 100% výpočtu (neboť není dopředu znám počet výpočetních operací, které je nutné provést), ale pruh se střívaje zaplňuje a vyprazdňuje. Pruhu tedy neukazuje jaká část výpočtů byla provedena, ale pouze to, že počítač pracuje (že nedošlo k „zamrznutí“ aplikace).

Po dokončení výpočtu je uživateli zobrazen výběr, kdy si může zvolit, kolik kroků chce simulovat, či zda chce jen zobrazit seznam kroků, které je nutné provést k nejkratšímu dokončení hry.

Simulace probíhá tak, že jednou za necelou vteřinu dojde automaticky k prohození dvou políček. Během simulace nemůže uživatel hrát. Při simulování je možné tuto simulaci ukončit

pomocí tlačítka „**Stop pomoc!**“, které je umístěno na stejném místě, jako bylo umístěno tlačítko „**Pomoc!**“ před simulací, a dále pokračovat normálně v hraní hry.

Pokud dojde k dokončení hry (ať již odsimulováním, či vlastnoručně uživatelem) a s počtem kroků se uživatel pro danou obtížnost umístil v tabulce deseti nejlepších hráčů, je požádán o zadání jména pro Top 10.

Rovněž je možné kdykoli v průběhu hry uložit svoji hrací pozici do souboru, ze kterého je možné tuto pozici kdykoli nahrát.

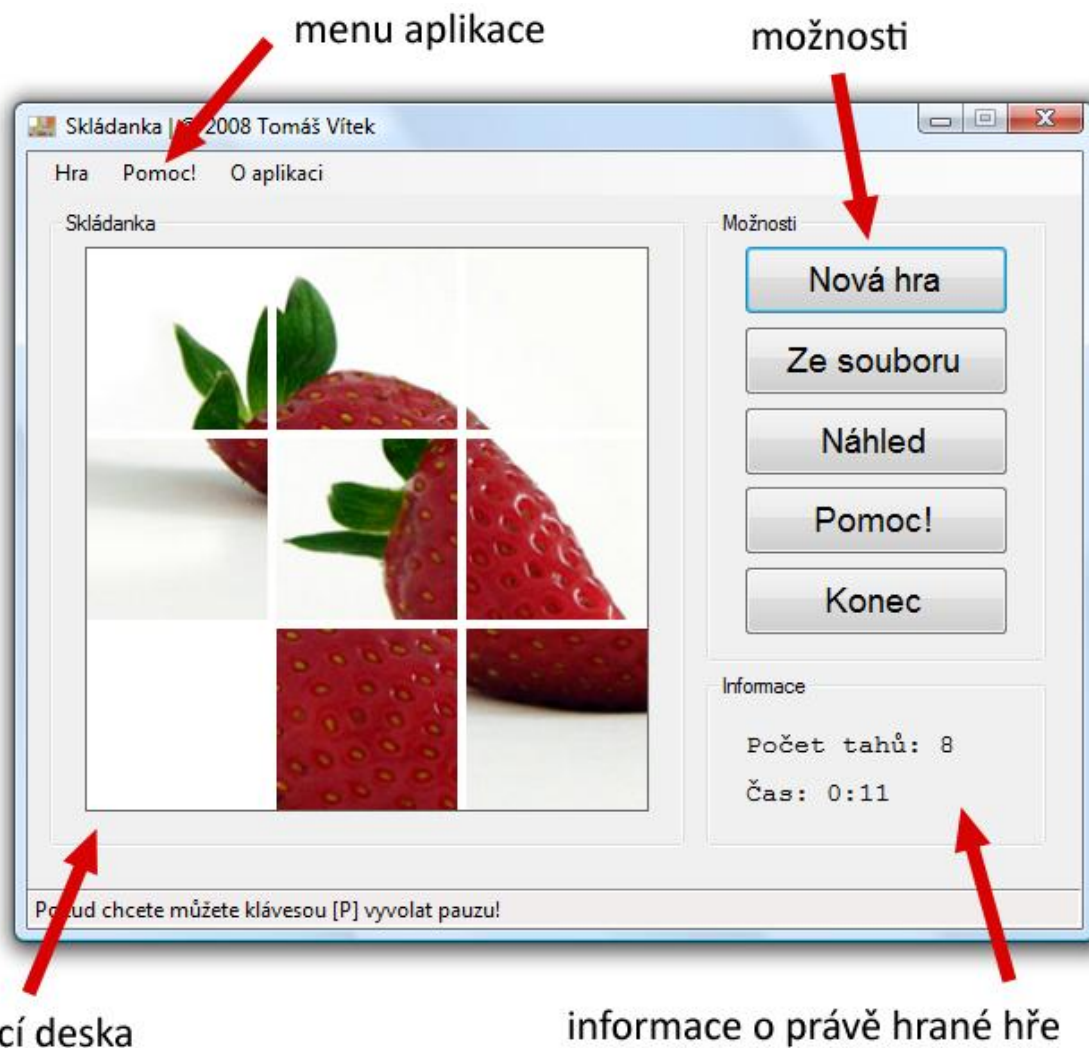
Všechny akce jsou přístupné jak pomocí ovládacích prvků (tlačítek), tak přes menu v horní části aplikace.

Klávesové zkratky

Pro nejdůležitější funkce lze využít následujících klávesových zkratk:

Funkce, či akce	Klávesová zkratka
vstup do menu „ Hra “	F2
vstup do menu „ Pomoc! “	F3
otevření okna „ O aplikaci “ a nápovědy	F1
nová hra	Ctrl + N
nová hra ze souboru	Ctrl + F
otevření tabulky nejlepších hráčů	Ctrl + B
uložení hry	Ctrl + S
nahrání uložené hry	Ctrl + L
zobrazení náhledu	Ctrl + P
otevření okna s výpočtem řešení	Ctrl + V
ukončení hry	Ctrl + K

Popis prostředí hry



„Menu aplikace“

Slouží k výběru akcí.

Skupina tlačítek „Možnosti“

Slouží k výběru a spuštění nejdůležitějších funkcí aplikace.

„Hrací deska“

Slouží k zobrazení aktuálního rozložení políček a pomocí myši k hraní (prohazování políček).

„Informace o hře“

Ukazují aktuální informace o hře – čas a počet odehraných tahů.

Programátorský popis

O tvorbě a spuštění aplikace

Aplikace byla naprogramována pomocí vývojového prostředí **Microsoft Visual Studio 2005** (studentská licence) v **jazyce C#**. Grafické prvky aplikace byly zpracovány pomocí grafického editoru **Adobe Photoshop CS2**. Aplikace samotná je pak uvolněna pod licencí **Creative Commons Attribution-Noncommercial-Share Alike 2.0**, která dovoluje aplikaci měnit, nekomerčně šířit, ale vždy se zachováním jména původního autora v kódu i výstupu aplikace.

Ke spuštění aplikace je nutný operační systém **Microsoft Windows** a nainstalovaný **.NET Framework 2.0**, bez kterého aplikace naprogramované v jazyce **C#** nefungují.

Popis datových struktur

Při tvorbě programu jsem využil dvou struktur:

Vyjádření souřadnic

Neboť se v celé hře jedná o uložení políček rozřezaného obrázku je nutné si nějak standardizovat uložení souřadnic, které jsou používány poměrně často. Já jsem zvolil datovou strukturu, která se v jazyce C# zapíše následujícím způsobem:

```
public struct souradnice
{
    public int x, y;
    public string slovy;
}
```

Struktura je veřejná, abych ji mohl využívat i v jiných formulářích (aplikace se skládá z více formulářů). Integer proměnné vyjadřují **X** a **Y** souřadnici a řetězec „**slovy**“ vyjadřuje (např. u pole směrů) slovní vyjádření daného směru.

Vyjádření vlastností hry

V aplikaci se vyskytuje ještě jedna datová struktura, která je užitá jako **singleton** (je v celé aplikaci použita pouze jedna instance této struktury):

```

struct vlastnostiHry
{
    // pocet kostek (zada uzivatel)
    public souradnice pocetKostek;
    // velikost kostek (vypocita se na zaklade zadani uziv.)
    public souradnice velikostKostek;
    // aktivni kostka (tedy ta „bila“)
    public souradnice aktivni;
    // cas
    public int casMinuty;
    public int casSekundy;
    // pocet tahu
    public int pocetTahu;
    // uz se hraje?
    public bool hrajeme;
    // rozdeleny obrazek
    public Color[:,][:] kostky;
    // poradi kostek (pri prohozeni pouze prohazuji poradi)
    public souradnice[:,] poradiKostek;
    // bitmapa (obrazek) s celym zmensenym slozenym obrazkem
    public Bitmap nahled;
    // lze kreslit?
    public bool kreslit;
    // je nahled k dispozici?
    public bool nahledKDispozici;
    // formular nahledu
    public Form formNahled;
    // formular vypisu cesty
    public Form formPomocVypis;
    // cesta k souboru s obrazkem (pro ukladani do souboru)
    public string cesta;

    // je zapnuta simulace?
    public bool automatZapnuty;
    // kolik odsimulovat kroku
    public int kolikSimulovat;
    // aktuální krok automatu
    public int krokAutomatu;
    // kroky (seznam kroku)
    public int[] krokyAutomatu;

    // vyhral jiz hrac tuto hru?
    public bool vitezstvi;
}

```

Vzhledem k tomu, že je vždy aktivní pouze jedna hra (a ani to z podstaty věci nelze jinak) nevytvářím více instancí této struktury, ale vždy při vytvoření nové hry pouze přemažu důležité proměnné.

Tato struktura je ve hře reprezentována instancí „hra“. Např. tedy k X souřadnici aktivního políčka přiřoupím následovně:

```
hra.aktivni.x
```

Popis jednotlivých algoritmů

Výpočet cesty a simulace

Při tvorbě aplikace bylo nejsložitější vymyslet algoritmus, který zjišťuje nejkratší cestu k cíli. Nakonec jsem zvolil zřejmě jediný možný a to **prohledávání do šířky**. Kdy je tento algoritmus implementován pomocí fronty (v jazyce C# jsem zvolil implementaci fronty pomocí **ArrayListu**). Jako heuristika je zde použit opět **ArrayList**, v které je uložen seznam všech konfigurací, do kterých se prohledávání dostalo v minulosti (konfigurace, resp. stav herní plochy je vyjádřen pomocí řetězce – kdy používám funkce **array2str** na zakódování pole do řetězce a **str2array** obráceně).

Při každém kroku je tázán každý ze čtyř směrů (vlevo, vpravo, nahoru, dolů) a pokud je krok možný (nejde mimo hrací plochu) a dané sestavení herní plochy ještě nenastalo je přidáno do fronty a algoritmus pokračuje další položkou z fronty. Tento postup probíhá dokud nedojde k vyčerpání celé fronty (nebyla nalezena cesta) nebo dokud nebyl nalezen cíl.

Samotné spouštění prohledávání cesty se pouští jako samostatné vlákno (**Thread**), aby nedošlo k zamrznutí aplikace, a také díky druhému vláknu má uživatel možnost během výpočtu (pokud již nechce čekat), tento výpočet stornovat.

Automatické simulování složení obrázku je řešení pomocí komponenty „**timer**“, která jednou za určený časový úsek vždy prohodí dvě políčka.

Ukládání a nahrávání souborů

Ukládání řeším pomocí jednoduchého **saveFileDialogu** a následného zapsání do textového souboru (s příponou ***.save**). Přičemž na prvním řádku je cesta k souboru s obrázkem, na druhém a třetím počty kostek X a Y, na čtvrtém a pátém řádku souřadnice aktivního políčka, na šestém a sedmém řádku čas, na osmém řádku počet tahů a na devátém řádku je v jednom řetězci vyjádřeno uložení jednotlivých částí hracího pole.

Příklad souboru s uloženou pozicí

```
C:\slozka\obrazek.jpg
3
3
1
1
1
22
16
1.1|2.1|1.0|2.0|2.2|0.0|0.1|0.2|1.2
```

Nahrávání uložené pozice potom probíhá jednoduchým **openFileDialogem**, přečtení daného souboru a jeho rozparsováním.

Realizace tabulky nejlepších hráčů

Je řešena pomocí zápisu jména, počtu tahů a času do registru počítače. A to na adresy (rozdělení je podle obtížnosti):

```
HKEY_LOCAL_MACHINE/SOFTWARE/Skladanka/3x3
```

```
HKEY_LOCAL_MACHINE/SOFTWARE/Skladanka/4x4
```

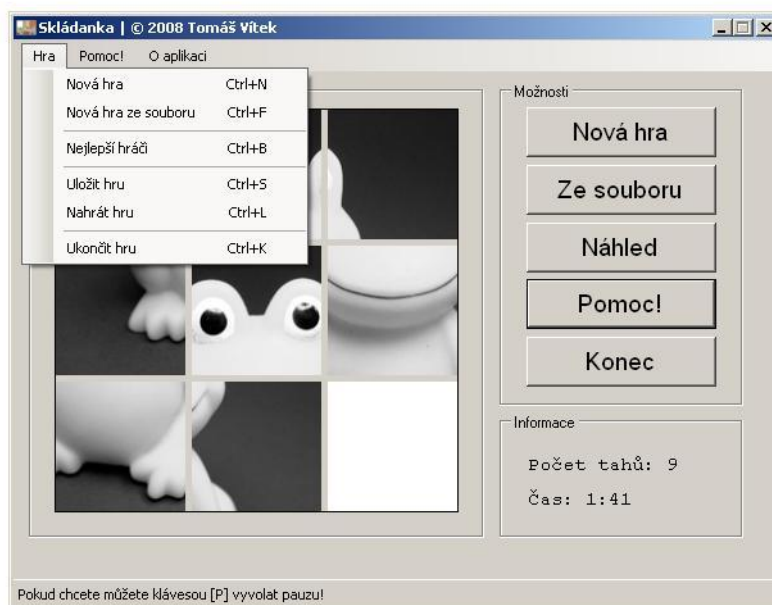
```
HKEY_LOCAL_MACHINE/SOFTWARE/Skladanka/5x5
```

```
HKEY_LOCAL_MACHINE/SOFTWARE/Skladanka/dalsi
```

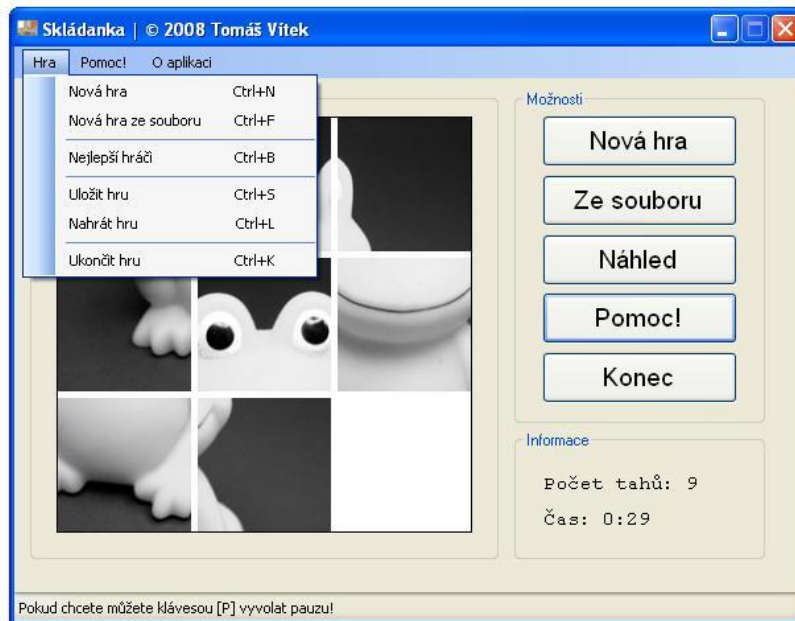
Program nevytváří žádné jiné zápisy do registru, ani žádné záznamy nemaže. V případě delšího nepoužívání programu provede (většina) operační systém promazání automaticky.

Screenshots aplikace na různých operačních systémech s různými vzhledy

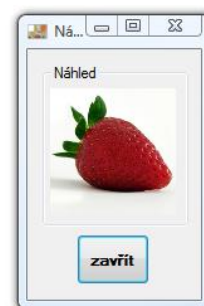
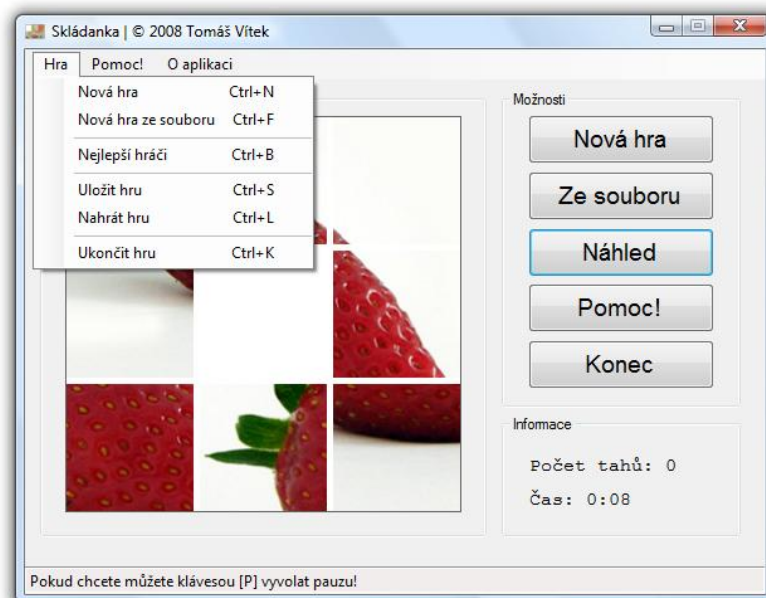
Microsoft Windows XP, „klasický vzhled“



Microsoft Windows XP, „moderní vzhled“



Microsoft Windows Vista, vzhled „Aero“



Další informace

Při programování mne zaskočilo několik věcí, např. že není možné v jazyce C# vytvořit prosté pole objektů Bitmap (s kterými pracuji, pokud pracuji s obrázky). Je sice možné si naprogramovat podobnou funkčnost pomocí třídy **BitmapArray**, ovšem zarazilo mě, že se tato funkčnost nenachází již přímo v jazyce (poznámka: vzhledem k tomu, že vlastním legální licenci pouze verze Visual Studio 2005 – tedy 3 roky staré, je možné, že v současné verzi je již chování uzpůsobeno tomu, jak by člověk předpokládal).

Celkové jsem byl s jazykem C# velice spokojen (a jsem s ním spokojen již dlouhodobě). Neboť umožňuje programátorovi jednoduše (přes **Designer**) „naklikat“ uživatelské prostředí a tak se programátor může v klidu soustředit na algoritmické problémy a vlastní programování.

Jazyk C# (a vývoj v **Visual Studiu**) má sice nevýhodu v podobě nutnosti nainstalovaného **.NET Frameworku**, ovšem myslím, že to již v dnešní době není problém (dle mého názoru např. Java s Virtual Machine je na tom hůře).

Závěr

Domnívám se, že se mi podařilo splnit cíl, který jsem si předsevzal. Myslím, že aplikace je použitelná. Jsem s jejím provedením spokojený. Snažil jsem se, aby byla uživatelsky přehledná a přívětivá, proto jsem se také vyhnul grafickým „zhůvěřilostem“, i když s pomocí Visual Studia a jeho Designeru by tvorba složitější grafiky nebyla problém. Ovšem domnívám se, že okenní aplikace (např. využívající knihovnu **WinForms** – tak jako tato aplikace) mají největší smysl s designem prvků (tlačítek, menu, okrajových lišt) přebírající od operačního systému, neboť předpokládám, že každý uživatel má nastavený vzhled OS podle svých potřeb, a vzhledem k tomu, že jsem vzhled žádného prvku neměnil, vždy dojde k tomu, že bude aplikace vypadat tak, jak je uživatel zvyklý u ostatních aplikací.

Aplikace by se dala určitě ještě více rozšířit, ovšem domnívám se, že již teď obsahuje velmi rozsáhlou funkčnost (s přihlédnutím k cílům aplikace). Například by bylo možné vytvořit aplikaci pro mobilní zařízení (**Windows Mobile**), což je v Visual Studiu také možné a poměrně jednoduché, vzhledem k tomu to mám také v plánu.