Západočeská univerzita v Plzni
Fakulta aplikovaných věd
katedra informatiky a výpočetní techniky

ZÁPOČTOVÁ PRÁCE z UIR

Umělá inteligence pro hru „Tetris“

13. 7. 2012

Pavel Lorenz
A10B0061K

1. Formulace úlohy (zadání)

- implementace hry tetris pomocí libovolného programovacího jazyka
- návrh a implementace algoritmu simulující hráče

2. Analýza úlohy

Tetris je jedna z nejznámějších a nejúspěšnějších počítačových her. Herní plocha má standardně rozměry 20x10 (výška x šířka). Tetrominy jsou kostky skládající se ze 4 čtverečků. V celé hry je 7 standardních tetromin. Na počátku se náhodně vygeneruje na horním okraji obrazovky jedna tetromina s náhodnou rotací a pomocí působení gravitace padá k dolnímu okraji hrací plochy.

Pokud políčka tetromin vytvoří ucelený řádek, tak bude tato řádka odmazána a vyšší řady se propadnou o tento smazaný řádek směrem dolů. Hráč může pohybovat s aktivní tetrominou doleva, doprava či pustit minu na zem. Také je umožněno minu otáčet.

Hra končí v případě, kdy se miny dotknout horního okraje hrací plochy, respektive již není prostor pro utvoření nové tetrominy. Cílem tedy je pomocí kompletace řádek udržet miny co nejdále od horního okraje.

Algoritmus umělé inteligence by měl simulovat lidského hráče a snažit se hrát hru pokud možno co nejdéle. Největším problémem samotné hry je nahodilost nově vytvořených tetromin.

Mým cílem není vytvořit skvělou hru a nejlepší algoritmus pro hraní hry tetris, ale primárně si vyzkoušet implementaci umělé inteligence v praxi.

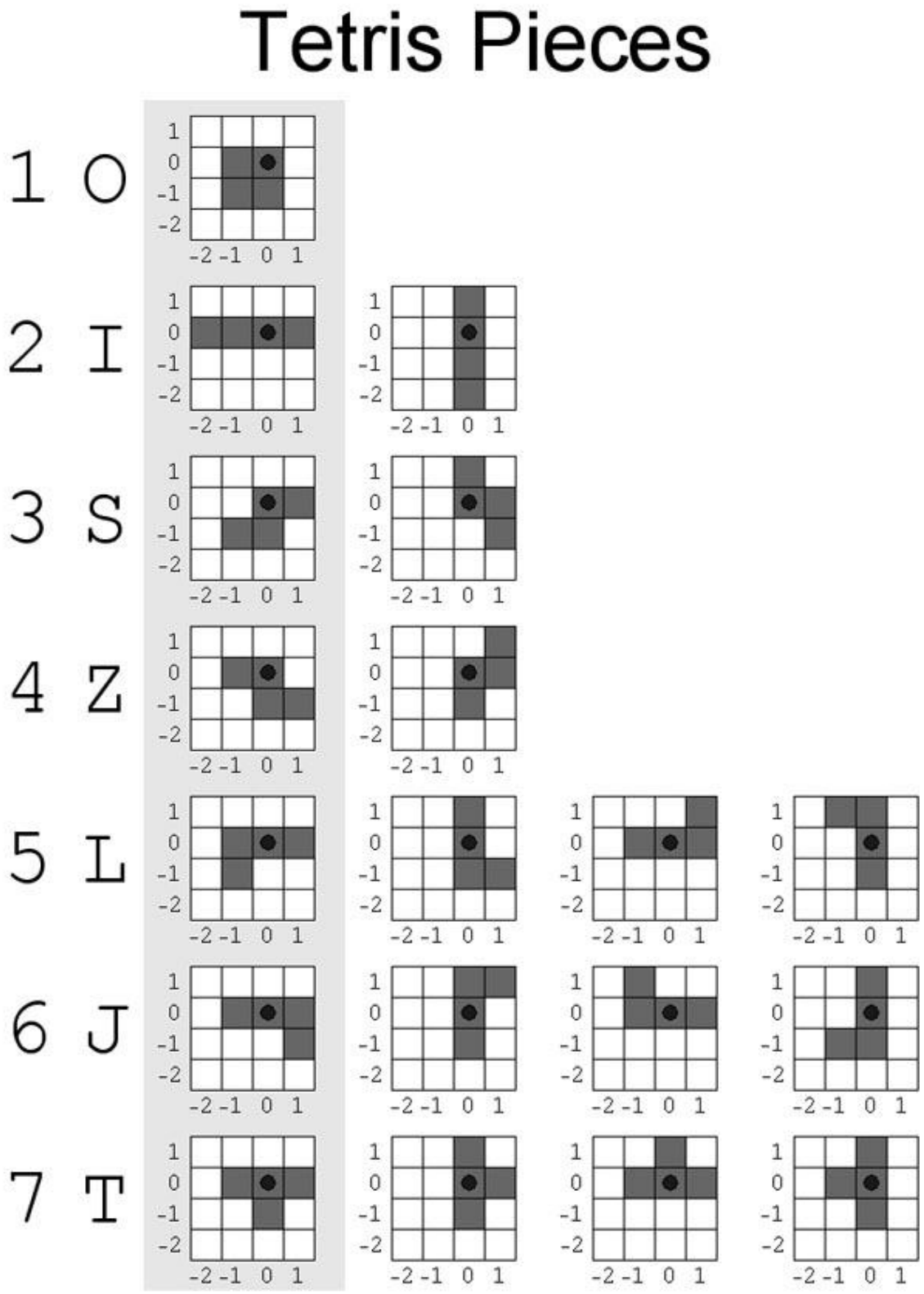
3. Popis algoritmu řešení

Hrací deska
Deska je tabulka o 10 sloupcích a 20 řádcích. Celkem tedy 200 buněk.

Tetrominy
Existuje 7 základních tetromin. Jejich jména jsou specifikovány dle připomínajícího tvaru písmen. Jmenovitě pak O, I, S, Z, L, J, T.

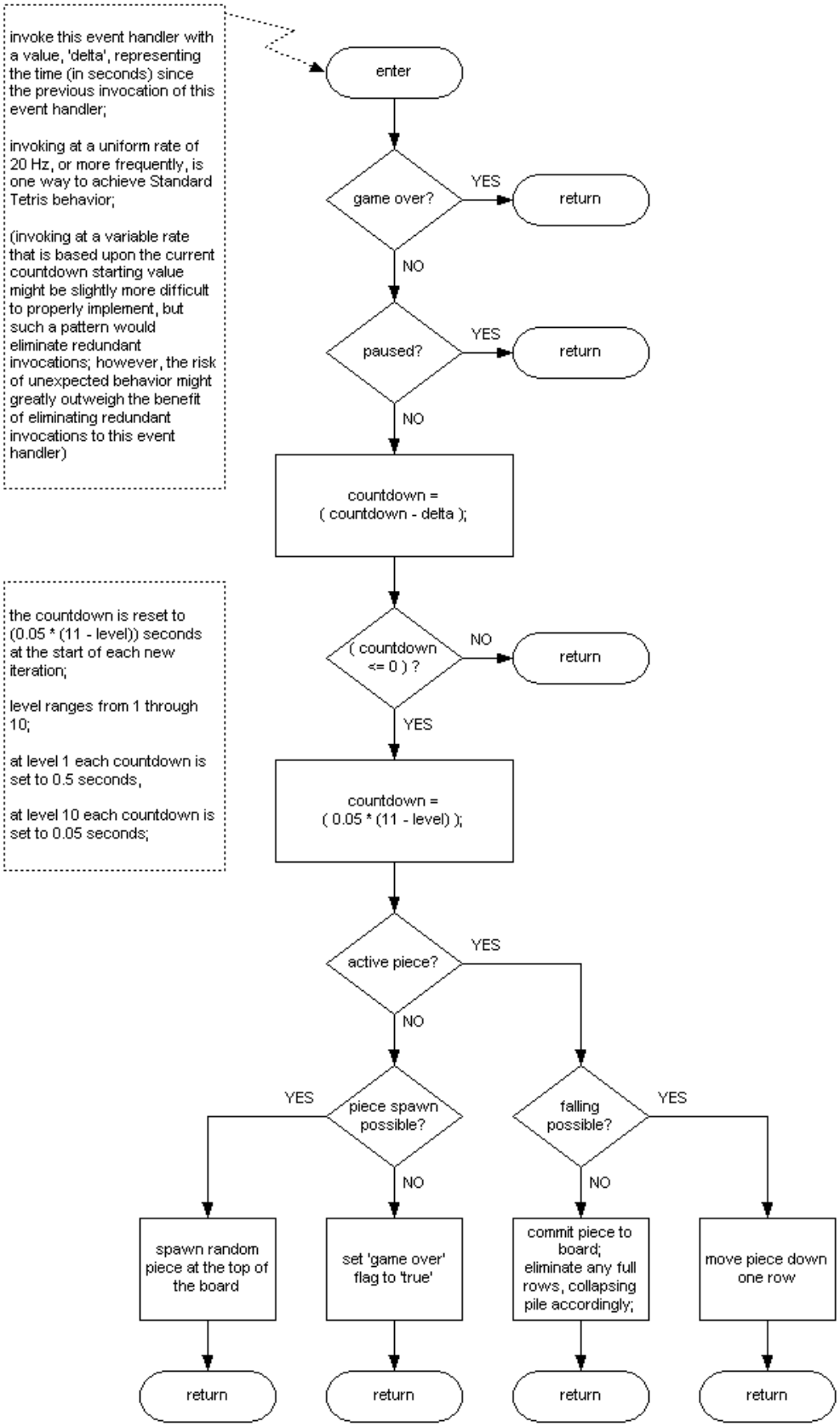
Mina „O“
Tato mina vypadá jako „čtvereček“. Je to jediný typ, který nelze rotovat.
Mina „I“
Tato mina připomíná „čáru“. Lze ji rotovat vertikálně a horizontálně. Tedy dvě rotace.
Mina „S“ a „Z“
Tyto dvě miny jsou zrcadlově převráceny. Mají opět dvě rotace.
Mina „L“ a J“
Tyto dvě miny jsou také zrcadlově převráceny. Mají čtyři možné směry rotace.
Mina „T“
Tato mina je jedinečná a obsahuje 4 možnosti rotace.

Viz `Obrázek 1- Rotace tetramin`.



Obrázek 1- Rotace tetramin

Algoritmus hry tetris



Tady jsem si hru trochu zjednodušil. Hra nelze pauzovat a nepočítám kola. Důležitější je pro mne umělá inteligence, která miny skládá než zrychlování hry.

Algoritmus umělé inteligence

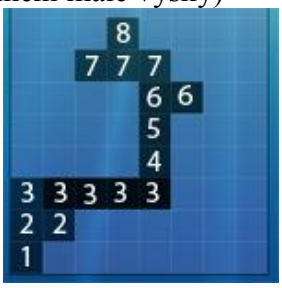
Po zapnutí hry tetris vygeneruje novou náhodnou minu. Tato událost vyvolá kontrolu, jestli je zapnutá umělá inteligence a pokud ano, tak se aktivuje vlákno čekající 200 ms a spustí se ohodnocovací algoritmus umělé inteligence. Uspání vlákna je zde jen proto, aby člověk mohl sledovat dění algoritmu.

Princip je ten, posunout aktivní minu do všech možných pozic a najít nejlépe ohodnocenou pozici. Tato pozice bude následně vykonána. Pro zjednodušení algoritmus umí jen jednoduché pohyby tedy posunout se kamkoliv vlevo či vpravo s jakkoliv otočenou minou. Neumí složitější manévry typu těsně před „zafixováním“ miny posunout se doleva. Tedy pokud nelze dosáhnout dané pozice pomocí „dropu“, tak jej algoritmus není schopen zpracovat.

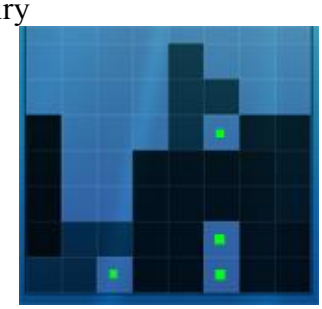
Cílem úspěšného algoritmu je co nejlépe ohodnotit situaci tak, aby umělá inteligence byla schopna simulovat reálného hráče.

<u>Jak na ohodnocení?</u>

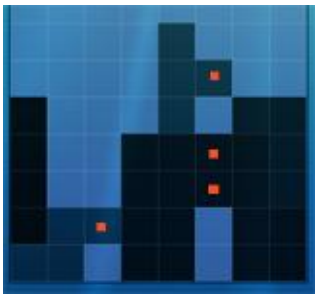
1. Penalizace za výšku (zvýhodnění malé výšky)



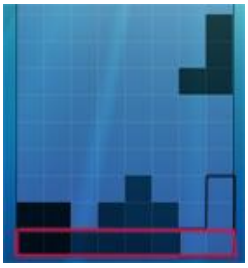
2. Penalizace za vytvořené díry



3. Penalizace za blokování děr



4. Zvýhodnění odstranění řádky



Protože toto ohodnocení (resp. nastavení vah) je samotnou podstatou tohoto projektu, tak jsem zvolil takovou technologii, aby si kdokoliv mohl vytvořit svůj algoritmus bez nutnosti zasáhnout do již existující implementace.

4. Popis programu (programová dokumentace)

Zvolené technologie

- Programovací jazyk C#
- Vývojové prostředí Visual Studio 2010
- .net framework 3.5 sp1
- Windows Presentation Foundation - jakožto technologii pro psaní tlustého klienta
- Managed Extensibility Framework – pro explicitní načítání knihovat

TetrisGame.cs

Veškerá herní logika je implementována ve třídě PaloTetris.TetrisGame. Tato třída implementuje rozhraní PaloTetris.ITetrisGame. Pokud tedy kdokoliv v budoucnosti bude chtít použít již vytvořené uživatelské prostředí popř. umělou inteligenci, tak si může zkusit implementovat vlastní Tetris hru.

Tato třída byla napsaná tak, aby byla nezávislá na volbě „zobrazení“. Lze tedy vytvořit např. konzolové zobrazení místo současného okenního a nebude to mít vliv na hru samotnou.

Hra se ovládá pomocí těchto veřejných metod:

- void Reset()
  - Restart celé hry.
- void Start()
  - Start hry. Zapnutí hraní.
- void Stop()
  - Pozastavení či opětovné spustění hry.
- bool Left()
  - Posuv aktivní miny doleva. Vrací false, pokud nelze.
- bool Right()
  - Posuv aktivní miny doprava. Vrací false, pokud nelze.
- bool Rotate()
  - Rotace aktivní miny. Vrací false, pokud nelze.
- bool Drop()
  - „Upuštění“ miny na nejnižší možný bod v poli. Vrací false, pokud nelze.
- bool IsSpaceFor(int x, int y, int[]piece)
  - Vrací true, pokud se aktivní prvek „vejde“ na danou pozici.
  - Tato metoda je veřejná právě kvůli umělé inteligenci.
- bool CanParkPiece(int x, int y, int[]piece)
  - Vrací true, pokud lze aktivní prvek posunout o jednu řádku níže.
  - Tato metoda je veřejná právě kvůli umělé inteligenci.

Dále hra poskytuje přehled o dění pomocí veřejných vlastností (Gettery):

- Guid UniqueID { get; }
  - Unikátní číslo hry.
- string DisplayName { get; }
  - Jméno hry.
- IField ActivePiece { get; }
  - Odkaz na aktivní minu.
- int ActiveX { get; }
  - Aktuální pozice X aktivní miny.
- int ActiveY { get; }
  - Aktuální pozice Y aktivní miny.
- int[,] Board { get; }
  - Hrací pole.
- int Width { get; set; }
  - Šířka hracího pole.
- int Height { get; set; }
  - Výška hracího pole.
- bool IsRunning { get; }
  - Je hra právě běžící?

Aby ale bylo možné na změny reagovat. Zapracoval jsem do hry tři události, na které se můžeme zaregistrovat a adekvátně na ně reagovat (např. překreslením okna):

- Event EventHandler GameEnd
  - Událost, která nastane na konci hry.
- Event EventHandler Repaint
  - Událost, která nastane při změně aktivní miny či pole a nutí k překreslení okna.
- Event EventHandler NextPieceGenerated
  - Událost oznamující, že byla právě vytvořena nová aktivní mina

Pomocí výše uvedených událostí, metod a vlastností lze plnohodnotně hru ovládat.

ITetrisAI.cs

Jak jsem ale již zmínil výše, nejdůležitějším prvkem umělé inteligence je zejména ohodnocení stavu. Aby si každý mohl vyzkoušet svůj způsob, tak jsem využil vlastnosti Managed Extensibility Frameworku, což je modularita a automatické prohledávání assemblies při startu aplikace.

V praxi to vypadá takto:

- Založíme si projekt (Class Library)
- Přidáme hru „PaloTetris.exe“ do referencí
  - Pravé tlačítko na Project -> Add Reference -> Browse
  - Exe soubor nalezneme a potvrdíme „Ok“.
- Vytvoříme novou třídu a implementujeme rozhraní „ITetrisAI“.
- Napíšeme si vlastní algoritmus do metody Run
- Zkompilujeme projekt

- Zkompilovanou knihovnu přikopírujeme do složky s hrou „PaloTetris.exe“
- Spustíme a v nastavení vybereme náš algoritmus

Pro ještě snazší práci jsem vytvořil abstraktní třídu „BaseAI“, kde je metoda run implementovaná, takže programátor nemusí vymýšlet procházení hracího pole a pouze implementovat ohodnocení.

Pro snazší pochopení jsem vytvořil v Solutionu knihovnu se jménem „PaloTetrisLib“, která obsahuje HeightAI a CleverAI.

Výhodou je, že aplikace běží bez přiložené knihovny a pokud ji sem přikopírujeme, tak automaticky načte všechny implementace umělé inteligence.

CleverAI.cs

Pro snazší pochopení ukazuji zdrojové kódy jednoho typu umělé inteligence.

```
/// <summary>
/// Ohodnoceni pozice dle vysky, prekrytych kosticek a kompletnich der.
/// </summary>
/// <param name="Tetris">instance hry</param>
/// <returns>ohodnoceni</returns>
protected override int Evaluate(PaloTetris.ITetrisGame game)
{
    int ret = 0;

    ModifiedFloodFill holeCounter = new ModifiedFloodFill(game, px, py);
    holeCounter.ColorField(0, 0, 0, 0, 9);

    // vyska
    int height = py * py;
    ret += height;
    // diry
    int holes = (holeCounter.GetCoveredHoles() * 25);
    ret -= holes;
    // radky
    int lines = (holeCounter.GetNumberOfFullLines() * 100);
    ret += lines;

    Console.WriteLine("[Height,Holes,Lines]:[{0},{1},{2}]:{3}",
            height, holes, lines, ret);

    return ret;
}
```

Pozor: Zde uvažujeme výšku aktivní miny, která je 0 pro horní pozici a 20 pro dolní pozici. Nejde tedy o nejvyšší bod v hracím poli!

5. Popis obsluhy programu (uživatelská dokumentace)

Požadavky

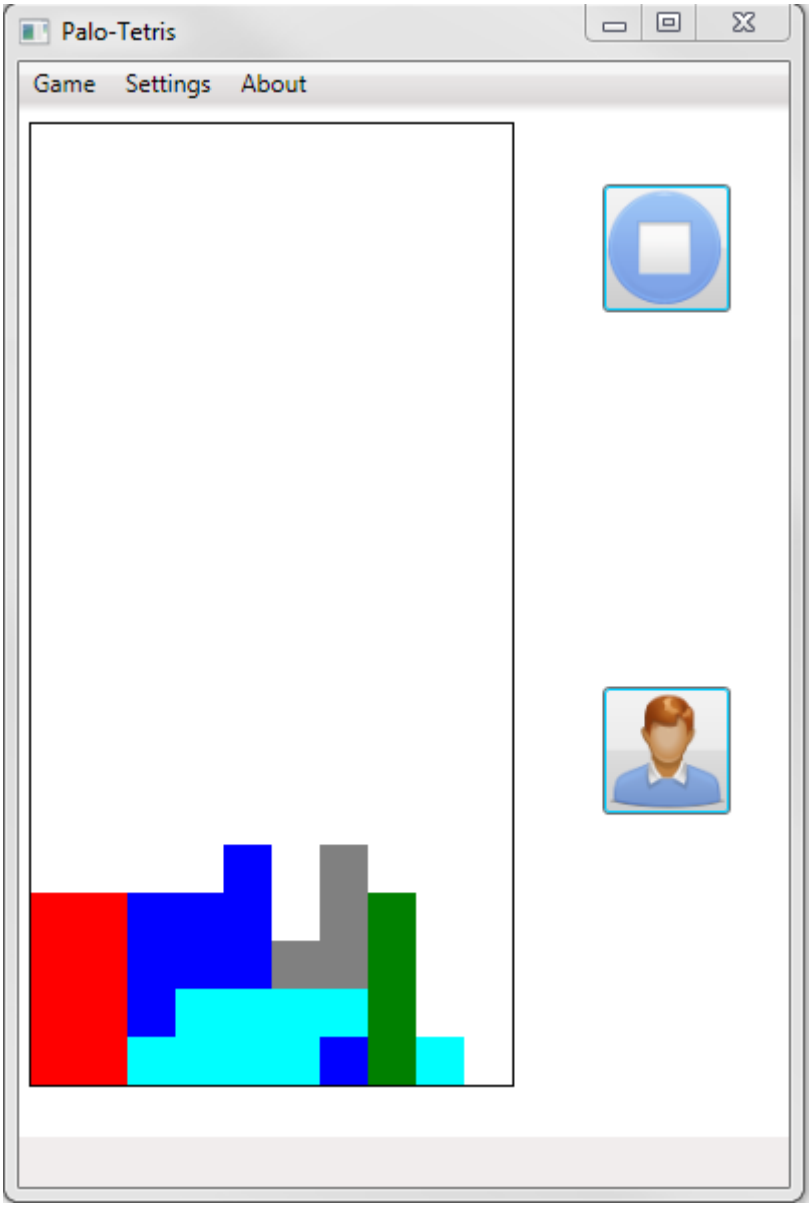
Pro spuštění aplikace je potřeba:

- Windows XP a vyšší
- .net framework 3.5 sp1

Návod ke spuštění

- Rozbalíme přílohu „UIR_Tetris_ A10B0061K.zip“
    - Pravá tlačítko -> „Extract all“ (popř. extrahovat vše)
- Otevřeme složku „Bin“ v rozbalené adresářové struktuře
- Poklepeme na soubor „PaloTetris.exe“

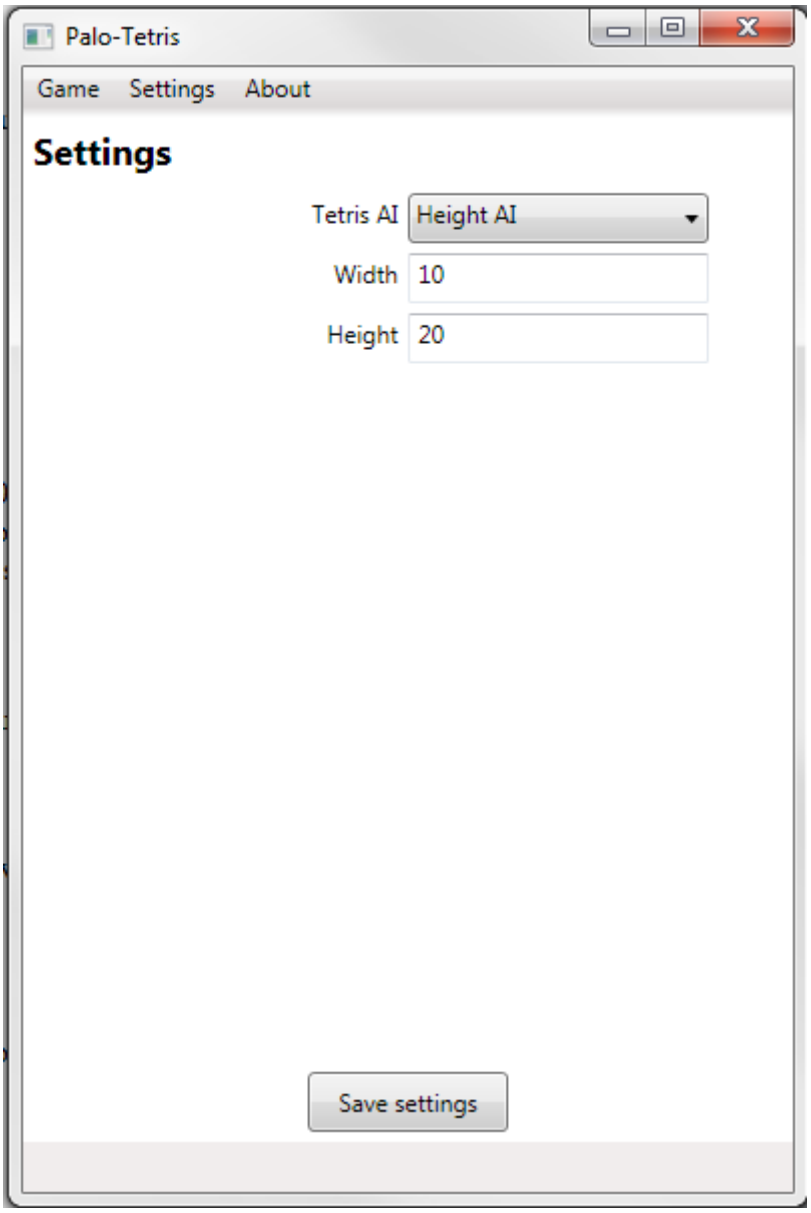
Návod k obsluze



Obrázek 2 - Game screen

Obrazovka „Game“

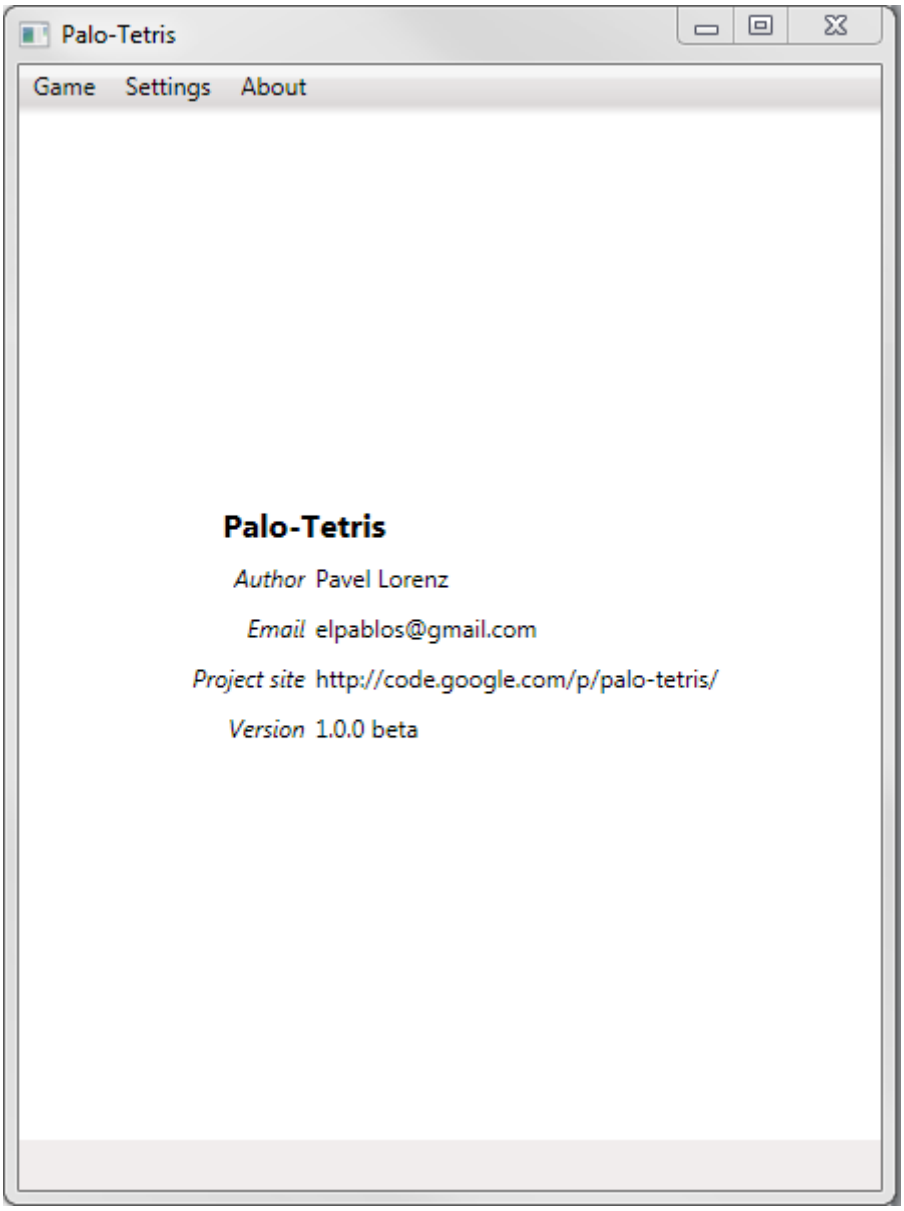
- Hru lze ovládat pomocí šipek na klávesnici
  - Šipka vlevo – posuv tetraminy doleva
  - Šipka vpravo – posuv tetraminy doprava
  - Šipka nahoru – rotace tetraminy
  - Šipka dolu – upuštění tetraminy dolu
- Hru lze zapnout či vypnout pomocí tlačítka Start Game/Stop Game
- Umělou inteligenci lze aktivovat pomocí Activate AI/Deactivate AI



Obrázek 3 - Settings screen

Obrazovka „Settings“

- Šířku hracího pole lze upravit pomocí textBoxu Width
- Výšku hracího pole lze upravit pomocí textBoxu Height
- Algoritmus UI lze zaměnit pomocí výklopníku „Tetris AI“
- POZOR! Změny je nutné nejprve uložit pomocí tlačítka „Save settings“



Obrázek 4 - About screen

Obrazovka „About“

- Tato obrazovka zobrazuje obecné informace o aplikaci
- Jméno autora
- Emailovou adresu
- Adresu projektu
- Verzi aplikace

6. Rozbor výsledků, zhodnocení

Obecně se nedá říct, který z algoritmů je lepší. Protože miny padají náhodně, tak každá hra je jiná a každý algoritmus se může zachovat jinak.

Pokud bychom chtěli doopravdy porovnat dva algoritmy, bylo by třeba mít předem vytvořenou množinu tetramin (resp. jejich posloupnost) a nad touto množinou porovnat, který z algoritmů se chová „inteligentně“.

Již teď ale mohu říci, že např. algoritmus postavený pouze na výšce je nedostatečný, což si lze prohlédnout v aplikaci. U složitějších algoritmů záleží zejména na nastavení vah.

U této aplikace bych pak zejména vyzdvihnul modularitu, která případnému následníkovi velice usnadní práci.

7. Závěr

S implementací v rámci možností jsem spokojen. Mým cílem bylo vyzkoušet si umělou inteligenci v praxi a připravit projekt tak, aby na něj šlo jednoduše navázat.
V aplikaci jsem objevil jednu chybu, a to tu, že nevaliduji nastavení rozměrů. Případný následovník by toto mohl opravit.

Možnosti rozvoje:

- Porovnání algoritmů
- Implementace nových algoritmů
- Implementace skóre
- Implementace levelů a zrychlování pádů tetramin
- Ukládání nastavení

Přílohy a zdroje

- UIR_Tetris_ A10B0061K.zip
  - Kompletní projekt s dokumentací, zdrojovými kódy a zbuildovanou aplikací
- http://code.google.com/p/palo-tetris/
  - Odkaz na projekt (SVN repository)
- http://www.microsoft.com/en-us/download/details.aspx?id=25150
  - Odkaz na .net Framework 3.5 sp1
  - Nutné ke spuštění
- http://mef.codeplex.com/
  - Odkaz na knihovnu pro podporu modularity
- http://www.wpftutorial.net/
  - Odkaz na stránku o WPF technologii
- http://colinfahey.com/tetris/tetris.html
  - Specifikace hry tetris
- http://luckytoilet.wordpress.com/2011/05/27/coding-a-tetris-ai-using-a-genetic-algorithm/
  - Popis generického algoritmu
- http://www.atagar.com/applets/tetris3D/tetrisAI.php
  - Jiný popis algoritmu
- http://www.cs.cornell.edu/boom/1999sp/projects/tetris/