Tetris – Dokumentace

*Zápočtový program, Tetris, je zábavný program, který lze využít buď pro zabití času hraním legendárních her, nebo pro edukativní cíle. Program obsahuje klasickou hru tetris, s typickým bodovým systémem a pravidly, která jsou drtivé většině lidí na planetě Zemi známy. Mimo to program obsahuje 2 různé umělé inteligence, které (za vhodných podmínek) dokážou hrát tetris i hodiny. Nejenom herní pamětníci zde najdou další hru, kterou nejspíš ve svých mladých létech pařili na gameboyích.*

# Programátorská sekce

## Úvod

Program je napsán v jazyce C# .NET 5.0 a jedná se o Windows Forms App. Program obsahuje 2 různé hry, které se ovládají pomocí klávesnice a 2 umělé inteligence, které dokážou hrát tetris.

Tento program obsahuje nemalé množství různých programovacích prvků, o kterých se zvídavý zájemce může dozvědět na přednáškách NPRG031. Mezi zmíněné prvky patří například: abstraktní třída s abstraktními metodami, statická třída, dědičnost, přetěžování funkcí, předávání parametrů pomocí reference, přepisování funkci ve třídách potomstva, přetypování proměnných, dosazování synovského objektu do mateřské proměnné, různé datové struktury (2 typy front, zásobník), globální proměnné, pestrá škála klíčových slov pro modifikaci přístupu k metodám či atributům. Formulářová aplikace pracuje s událostmi jako je Button\_Click, PictureBox\_Paint, Timer\_Tick. Za zmínku stojí i možnost si, při hraní nebo pozorování AI, pustit tetrisovou hudbu.

## Část I – Hra Tetris

Kapitola 1 – hrací deska, místo pro radost i rozruch

Hrací deska pro tetris a obě umělé inteligence v daném projektu je reprezentována pomocí objektu třídy GameBoard, který je v hlavní třídě Form1.cs uložen pod názvem GameBoard gb. Při zapnutí (nebo i přepnutí) kteréhokoliv režimu se vždy zinicializuje nová hrací deska – gb = new GameBoard(); –. Objekt této třídy obsahuje dvojrozměrné pole public char[,] Board, které je v konstruktoru defaultně naplněno nulovým charem ‘\0‘, což je chápáno jako prázdné pole, místo, kam se může TetroBlock posunout nebo rotovat. Při hraní her uživatelem nebo v případě AI režimu se zaplňuje toto pole různými chary, které symbolizují jednotlivé barvy v konkrétním místě ve hře. Tyto informace jsou pak použity při zobrazování a vykreslování hrací desky a hracích figurek. Hrací deska má rozměry 10x20, ale zobrazováno je jenom dolních 18 řad. Horní 2 sice nejsou zobrazovány, ale i přesto může hráč (například při rotaci) do nich zavítat. Tato třída kromě hracího pole obsahuje celočíselné hodnoty level, lines a score. Nejpodstatnější z této trojice čísel je level, tato hodnota upravuje rychlost hry. Dochází ke zrychlení hry až do 10. levelu (kdy je rychlost hry 140ms), pak už je zrychlení nulové.

Třída GameBoard je mimo jiné používána pro generaci TetroBlocků. Generace není přímo náhodná. Existuje 7 druhů TetroBlocků, se kterými můžeme hrát tetris. Generaci si můžeme představit následovně. Do měšce vložíme všech 7 existujících TetroBlocků a postupně je vytahujeme po jednom. V okamžiku, kdy je měšec prázdny, navrátíme všechny TetroBlocky zpět a začínáme od znova. Tím pádem je záruka, že každých 13 generací (v nejhorším případě) dostanu chtěný TetroBlock. Tuto úlohu řeší statická metoda static public Shape GeneratePiece(), která ukládá potřebné informace do statických proměnných static int numOfPieces a static bool[] piecesDistribution. Návratová hodnota této funkce je Shape, což je abstraktní mateřská třída všech 7 různých TetroBlocků, proto můžeme jednoduše dosadit a přetypovat proměnnou Shape na potřebný objekt. Další podstatné metody této třídy jsou:

* public bool AddToBoard(Shape shp)
* public int[] FindFullLines()/static public int[] FindFullLines(ref char[,] deska)
* static public void MoveMap(ref char[,] deska, int[] lines)
* static public void ClearLines(ref GameBoard gb, int[] lines)

První zmíněná metoda vrátí true, pokud se nám podaří vložit aktivní hrací TetroBlock do hrací desky, tím pádem to patří ke stěžejním metodám, která nám prozradí, kdy je třeba hru ukončit, protože už není místo pro další TetroBlock. Jedná se o klasický cyklus přes pole int[,] Pozice, který otestuje, zda každá pozice v poli char[,] Board má nulový char jako hodnotu. Druhá metoda patří k nejdůležitějším a je vyvolána následně po funkci AddToBoard. Tato metoda zkontroluje, jestli náhodou nedošlo k nějakému naplnění řádku a vrátí pole s čísly řádků, které je třeba vymazat. Při mazaní řádků dochází k úpravě hodnot level, score, lines, které jsou zobrazovány ve formuláři.

kapitola 2 – malujeme kostičky

Formulářová aplikace obsahuje celkově 20 prvků: 3x PictureBox, 5x Button, 5x Timer, 6x Label a 1x CheckBox. Všech 6 labelů složí pro zobrazování aktuálních údajů běžící hry. CheckBox se používá pro zapínání a vypínání Tetris Theme Song. Funkce každého tlačítka je zřejmá z textu, který je napsán na jeho tlačítku. Při přepnutí do kteréhokoliv režimu se nejprve vypnout všechny timery, vynulují se globální proměnné pro chod programu, vytvoří se nové objekty s „čistým štítem“, nastaví se správně hodnoty score a vygenerují se nové TetroBlocky, které jsou uloženy v proměnných Shape activePiece a Shape nextPiece. Jak už název napovídá, tak tyto proměnné definují figurku, se kterou hráč (popř. umělá inteligence) aktuálně hraje a manipuluje na hracím poli. Figurka uchovává svoji barvu a svoje 4 pozice, na kterých se „virtuálně“ nachází v hracím poli. Je zde použito slovo virtuálně, protože dokud je figurka aktivní a je s ní možnost zacházet a manipulovat, tak není zaznamenána v hracím poli. K této změně dojde po zavolání metody třídy GameBoard: public bool AddToBoard(Shape shp).

Nachází se zde 5 timerů, 2 jsou uživatelské pro 2 herní módy, 2 jsou pro 2 umělé inteligence a poslední je pro „střílení“ ve hře Wall Breaker, což je kromě pohybu doleva a doprava, jediný akční prvek. Zbývají 3 pictureboxy. Všechny hry a režimy se odehrávají v prvním z nich. 2. je čistě dekorační, pakliže obsahuje jenom tetris logo a ve 3. se zobrazuje následující TetroBlock, který je uložen v proměnné Shape nextPiece.

Tvorba a design jsou uskutečněny pomocí statické třídy Visual, která odpovídá za správné namalování hrací desky a figurek. Nutnost přemalovat hrací plochy zařídí Paint Event neboli událost, která je vyvolána pomocí funkce PictureBox[13].Invalidate(). Tato funkce na oba živé pictureboxy je volána při každém zmáčknutí každého tlačítka (kromě EXIT tlačítka) a ve všech událostech timer[12345]\_Tick. Příslušný Paint Event nejprve ověří, že hrací deska i aktivní figurka již byly inicializovány a následně zavolá metodu Visual.DrawGame(ref GameBoard gb, Shape shp, Graphics grafika, Pen tuzka). Tato funkce se skládá z dalších 2 funkcí pro zobrazení hrací desky a aktivní figurky. Pro obě funkce je stěžejní bratrská funkce void DrawRect(Graphics grafika, Pen tuzka, char color, int height, int width). Tato funkce namaluje krásný jeden čtvereček potřebné barvy a umístí ho podle maticových souřadnic height & width do správné polohy v herním poli.

Kapitola 3 – třída shape a jejich 7 dětí

Jak už bylo zmíněno, tak existuje zde abstraktní třída Shape, která obsahuje jak abstraktní metody nebo třídní metody, tak i statické metody. Tato třída obsahuje 2 veřejné atributy: char Color a int[4,2] Pozice. V tetrisu je možnost hrát se 7 různými tvary TetroBlocků, tedy se všemi kombinacemi, ke kterým dojdeme přehazováním 4 čtverečků (zanedbáme-li násobnost). Proto existuje 7 tříd, pro každý tvar zvlášť. Každá z těchto tříd je potomkem třídy Shape a musí obsahovat override metody pro pohyb doleva, doprava, dolů, oboustranné rotace a pohyb nahoru (pro účely ladění a nepovedený pokus o implementaci tetrisDFS). Protože logika rotací se u některých tvarů liší, existují pomocné soukromé funkce a atributy, které pomáhají chodu hry. Před každým pohybem nebo rotací se provede buď mateřská funkce pro kontrolu, zdali je možné se do tohoto místa pohnout, nebo kontrolní funkce příslušící danému tvaru. Pohyb nějakým směrem nebo kterákoliv rotace je možná v případě, že po dokončení akce bude možné TetroBlock zobrazit na hrací desce bez toho, aby byl překryt jakýkoliv čtvereček nebo aby figurka „vyletěla za hranice hrací desky“. Mateřská třída má ještě jednu podstatnou metodu a tou je: public int HardDrop(ref GameBoard gb). Vyvolání této metody je možné v uživatelském režimu Tetrisu zmáčknutím tlačítka SPACE. Jednoduše bude v cyklu probíhat funkce public bool MoveDown(ref GameBoard gb) dokud to bude možné a ve výsledku to vrátí počet úspěšných provedení této funkce. To číslo se pak přičte ke score uživatele. Je to praktické, v případě, že hráč už ví, kam chce figurku umístit a nechce několikanásobně mačkat tlačítko pro pohyb dolů. Zároveň získá body navíc, což je fajn bonus.

Třída Shape rovněž obsahuje metody, které jsou následně používány umělou inteligenci. Mezi tyto metody patří:

* static public bool check[Left/Right/Down]Side(ref GameBoard gb, int[,] Pozice)
* public static bool MoveDownNotPossible(ref GameBoard gb, int[,] Pozice)

Druhá funkce je hojně používána při prohledávání stavového prostoru hrací desky, pro nalezení všech možných umístění TetroBlocku a zároveň i navigací k daným lokacím.

Kapitola 4 – form1.cs aneb třída, co vládne všem a všem káže

Form1 je nejdůležitější třída, protože se nacházíme ve formulářové aplikaci a všechny akce vychází z interakcí ve formuláři. Můžeme zde uvidět různé množství globálních proměnných pro obecný chod hry nebo i specifičtější boolovské proměnné, které signalizují, který konkrétní režim právě běží a co se má anebo nemá vykreslovat na formulář. Mezi nejzákladnější obecné proměnné patří:

* Pen tuzka
* Shape activePiece
* Shape nextPiece
* int[] clearLines – označuje, které řádky jsou plně naplněny a můžou být vymazány
* bool gameover
* int moveSpeed

Jelikož bylo nemožné spolupracovat s událostmi jako Key\_Down Event nebo Key\_Press Event (nejspíš proto, že byl fokus vždy na nějakém tlačítku, a ne na PictureBoxu), musela se přepsat defaultní virtuální metoda, pro zpracovaní tlačítek ProcessCmdKey(ref Message msg, Keys keyData), která odpovídá za reakce při zmáčknutí tlačítka. V této funkci byl použit switch, který stisknuté akční tlačítko převedl na pohyb nebo jinou akci figurky.

Průběh života hrací figurky má několik fází. Nejprve se figurka vygeneruje a uloží do příslušné proměnné, na kterou vidí malovací události, které pak podstatné informace předávají funkcím na grafické znázornění. Hráč může pohybovat figurku všemi směry, nebo po uplynutí časové doby, v závislosti na obtížnosti, je figurka posunuta o jednu úroveň níž v události spojené s časovačem. V případě, že už není možné se posunout dolů, je figurka vložena správnou funkcí do hrací desky a následně přichází na řadu různé kontrolní funkce. Je zkontrolováno, zda nebyla naplněna nějaká řada a v pozitivním případě se řada vymaže, upraví se číselné hodnoty score, level, lines a moveSpeed, a nakonec se posune hrací deska a vygeneruje se nová hrací figurka.

## Část II – Umělá inteligence

Zápočtový program obsahuje umělou inteligenci, která je více či méně schopná hrát hru. Avšak není zde jenom jedna AI, ale dokonce 2. A nejsou si vůbec podobné! Jedna se jmenuje HardDropAI a jak už název napovídá, tak od této „inteligence“ nemůžeme čekat nějaké epické rotace a triky při hře. Vygenerovaná figurka se nejprve dostane do správné rotace a správné svislé polohy a následně jen klesá, dokud je to možné. Proto je název HardDropAI. Druhá má kratší název – AI, zato je mnohem efektivnější a „moudřejší“. Vygenerovaná figurka má možnost se dostat i POD nebo DO jiné figurky. Při průběhu obou AI můžeme nalézt 3 různé fáze, které se cyklí stále dokola. Fáze jsou následující:

1. Nalezení všech pozic, mezi kterými se budeme následně rozhodovat.
2. Bodový systém rozhodne, která pozice je pro dosavadní hrací desku optimální.
3. Máme už rozhodnuto a zbývá jenom „najít cestu od startovní pozice do cíle“.

Druhý krok je v obou AI téměř, až na nějaké drobnosti, identický. Ostatní kroky nemají skoro nic společného. Postupy se liší dramaticky, avšak i přes to používá druhá AI mnohé metody z třídy HardDropAI.

kapitola 1 – všechny cesty vedou na hrací desku

**HardDropAI**, jak bylo v předešlé části zmíněno, je sice „hloupější a slabší“, ale na druhou stranu je časově méně náročnější. V hlavní třídě Form1.cs se nachází globální proměnná int[5,2] placeToDropFrom, která je výsledkem komplikovaného hledání nejlepší pozice, ze které po HardDropu získáme optimální výsledek. Optimální pozici získáme zavoláním statické metody ze statické třídy HardDropAI – placeToDropFrom = HardDropAI.FindBestPlaceForDrop(ref GameBoard gb, Shape shp);. Tato metoda zavolá ve svém průběhu 13 dalších metod. Všechny však nebudeme probírat do puntíku. Nyní nás zajímá jenom první řádek této funkce– int[,,] drops = findAllHardDrops(ref GameBoard gb, Shape shp); –, zbytek je popsán v dalších kapitolách. Tato metoda posune čerstvě vygenerovanou aktivní figurku nejvíc doleva a postupně zapisuje souřadnice TetroBlocku po HardDropu. Takhle se to opakuje, dokud nejsou vyzkoušeny všechny pozice ve všech rotacích. Hodnota drops[i, 0, 2] označuje kolikrát musí TetroBlock rotovat doprava, než se dostane do správného rozpoložení. Tato informace usnadňuje v budoucnu práci.

**AI** má tento krok mnohem složitější a propracovanější. V tomto případě používáme upravenou verzi BFS algoritmu pro prohledávání stavového prostoru. Stavový prostor je aktuální hrací deska. Při prohledávání využíváme frontu, která kromě informací o souřadnicích nese i navigace k danému místu od startu. Výsledkem prohledávání je uložení všech pozic do fronty (recyklujeme datovou strukturu), která kromě konečných pozic uchovává i „navigaci“ k dané pozici. To je velmi praktické. Prohledávání probíhá následovně:

1. Vybereme prvek z fronty (potřebujeme, aby metoda fronta.Pop() kromě souřadnic vrátila i navigaci, tedy klasický string, kde jednotlivé chary znamenají pohyb některým směrem nebo rotaci, proto je návratová hodnota objekt InfoBlock, což lze přirovnat k obyčejné struktuře o 2 proměnných).
2. Pokud už pohyb dolů není možný (jsme na kraji desky nebo se pod TetroBlockem již něco nachází), tak pozici (včetně navigace) uložíme do výsledné fronty.
3. Zkusíme naše souřadnice posunout všemi směry a úspěšně posunutí vložíme do fronty s modifikovanou navigací. Vložené místo „označkujeme“, aby už nebylo možné do této pozice vstoupit.

Tento tetris BFS algoritmus provedeme ve všech rotacích. Nyní máme frontu naplněnou všemi pozicemi ve všech rotacích, ke kterým můžeme dojít. Můžeme se pustit do vyhodnocení pozic.

kapitola 2 – dobrý tah, nebo zlý, to je, oč tu běží

kapitola 3 – když jde tetroblock na výlet

## Část III – Bonus

kapitola 1 – drtič zdí

# Uživatelská sekce