

Gymnázium, Praha 6, Arabská 14

Obor programování



Ročníkový projekt

TETRIS

Martin Voplakal, 1.E

duben 2021

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 30.4.2021

Martin Voplakal

Název práce: TETRIS

Autor: Martin Voplakal

Anotace a zadání projektu

Zadáním projektu bylo naprogramovat hru Tetris v programovacím jazyce Java. Cílem této hry je pomocí šipek navigovat padající obrazce složené z kostiček a pokládat je na místa tak, aby do sebe zapadaly. Po zaplnění celého řádku v hracím poli se řádek kostiček odmaže, dílky nad ním se posunou směrem dolů a hráč obdrží odměnu v podobě bodů. Pokud dojde k situaci, kdy hráč zaplní celé hrací pole až k hornímu okraji, a tedy není možné již vygenerovat další dílek, nastane konec hry.

Obsah

Anotace a zadání projektu	1
1. Úvod.....	3
2. Objektový návrh.....	3
2.1 Tetris.....	3
2.2 Piece.....	3
2.3 Faces	3
3. Zvolené technologie	3
3.1 Grafická knihovna	3
3.2 Nástroje.....	4
4. Logika hry	4
4.1 Rotace dílku.....	4
4.1.1 Základní algoritmus	4
4.1.2 Chyby při přemazání pole	5
4.1.3 Ošetření rotace mimo herní pole.....	5
4.1.4 Rotace do místa s jiným dílkem (kolize)	6
4.2 Časovač posunu dolu	6
4.3 Zpracování uživatelského vstupu	6
5 Skóre.....	7
5.1 Bodovací systém.....	7
5.1.1 Soft Drop.....	7

5.1.2 Hard Drop	7
5.1.3 Multiple Line Clear.....	7
5.2 Ukládání skóre do souboru	7
6. Závěr.....	7
Seznam obrázků.....	8

1. Úvod

Hra Tetris byla takovým prvním milníkem, který jsem si chtěl splnit, a tak jsem využil příležitost a pojmul jsem to jako ročníkový projekt. Tušil jsem, že mě čeká nelehký úkol, což následně potvrdilo množství dílčích problémů a situací, které je třeba ošetřit. Některé, které považuji za zásadní jsou podrobněji popsány níže.

2. Objektový návrh

Hra byla naprogramována objektově a bylo zvoleno následující rozložení do tříd:

2.1 Tetris

Třída Tetris spravuje základní logiku hry. Zpracovává vstup od uživatele, obsahuje časovač pro pravidelné padání dílků a volá metody dalších tříd.

2.2 Piece

Instance této třídy představuje padající dílek. Obsahuje konstruktor, který při zavolání zvolí náhodný tvar. Třída také obsahuje všechny metody pro pohyb s aktuálně padajícím dílkem (pohyby dolů a do stran, rotaci a další podpůrné metody). Udržuje si informaci o aktuální pozici dílku, jeho barvě a podobně.

2.3 Faces

Tato třída byla vytvořena pouze za účelem zpřehlednění kódu. Je inicializována pouze jednou na začátku běhu programu a udržuje pouze „grafické“ reprezentace tvaru dílků v podobě dvourozměrných polí booleanů.

3. Zvolené technologie

3.1 Grafická knihovna

Pro tvorbu byla zvolena knihovna JavaFX, která obsahuje všechny potřebné funkce a oproti ostatním knihovnám, jako je např. Swing, je možné použít k přizpůsobení grafiky kaskádové styly CSS. Konkrétně bylo CSS použito ke stylování textů.

3.2 Nástroje

K vývoji bylo použito IDE Netbeans 8.2 a JDK 8.



Obrázek 1 NetBeans

Obrázek 2 Java JDK

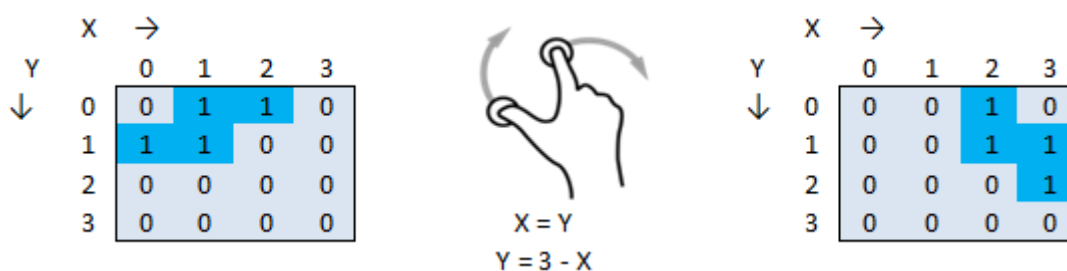
4. Logika hry

V této kapitole jsou vysvětleny zajímavé algoritmy a složitější konstrukce. Triviálním metodám pro pohyb do stran nebo dalším podpůrným metodám se zde věnovat nebudu.

4.1 Rotace dílku

4.1.1 Základní algoritmus

Zjednodušený algoritmus použitý v mé práci ilustruje obr. 3. Implementovaný algoritmus je ale vylepšen, je mnohem složitější a přizpůsobuje se různě velkým polím. To má za následek, že rotace dílku ve finále působí přirozeněji.



Obrázek 3 Rotace dílku

Základní logika rotace je následující: (viz. obr. 4)

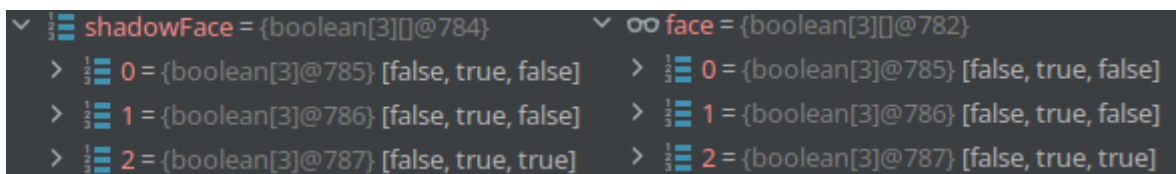
- 1) Aktuální orientace dílku se uloží do proměnné shadowFace
- 2) Proměnná face se přemaže (problém tohoto kroku je popsán v kapitole 4.1.2)
- 3) Je aplikován výše popsáný algoritmus

```
boolean[][] shadowFace = face;
this.face = new boolean[face.length][face[0].length];
for (int y = 0; y < face[0].length; y++) {
    for (int x = 0; x < face.length; x++) {
        face[x][y] = shadowFace[y][(face.length - 1) - x];
    }
}
```

Obrázek 4 Rotace (zkráceno)

4.1.2 Chyby při přemazání pole

Během vývoje jsem narazil na podstatný problém, který způsoboval chyby při pokusu o rotaci. Dvourozměrné pole bylo totiž kopírováno pomocí `array.clone()`, což byla zásadní chyba, protože došlo ke zkopírování pouze prvního pole, které obsahovalo reprezentace vnořených polí, která nebyla fakticky zkopírována, což dokládá obr. 5, kde jsou vypsané číselné ukazatele na hodnoty v paměti. Řešením by tedy bylo buď vytvořit metodu pro kopírování pole s vnořeným cyklem a metodou `clone()` aplikovanou na každé z vnořených polí, nebo jako v mém příkladě pokaždé pole přepsat konstruktorem a staré nahrazené pole přenechat Garbige Colectoru.



shadowFace = {boolean[3][]@784}	face = {boolean[3][]@782}
> 0 = {boolean[3]@785} [false, true, false]	> 0 = {boolean[3]@785} [false, true, false]
> 1 = {boolean[3]@786} [false, true, false]	> 1 = {boolean[3]@786} [false, true, false]
> 2 = {boolean[3]@787} [false, true, true]	> 2 = {boolean[3]@787} [false, true, true]

Obrázek 5 Reprezentace pole v paměti

4.1.3 Ošetření rotace mimo herní pole

Ošetření problému, kdy se dílek při rotaci dostává mimo hrací pole byl vyřešen do-while cyklem, ve kterém jsou kontrolovány souřadnice jednotlivých čtverců a případně cyklus je opakován do chvíle, kdy je již vše v pořádku.

4.1.4 Rotace do místa s jiným dílkem (kolize)

Na závěr metody pro rotaci je kontrolována kolize s jiným dílkem. Pokud k tomu dojde, je obnoven stav dílku i pozice před začátkem celého procesu rotace pomocí předem uložených souřadnic a orientace.

4.2 Časovač posunu dolu

Pro časování vyvolávání metody `moveDown()` jsem vytvořil následující časovač (viz. obr. 6) složený z objektů `Timer` a `TimerTask`. Jedinou nevýhodou této konstrukce je, že nejde po zastavení znovu vyvolat jiným způsobem, než novým zkonstruováním.

```
task = new TimerTask() {
    public void run() {
        Platform.runLater(new Runnable() {
            public void run() {
                piece.moveDown();
            }
        });
    }
};
fall.schedule(task, 0, delay);
```

Obrázek 6 Časovač

4.3 Zpracování uživatelského vstupu

Uživatelský vstup je načítán a pomocí konstrukce `switch` je podle stisklé klávesy volaná příslušná metoda. Ve chvíli, kdy nastává konec hry nebo uživatel hru pozastaví stiskem klávesy `Esc` se zavolá metoda `scene.setOnKeyPressed()` s prázdným lambda konstruktorem pro deaktivaci vstupu od uživatele.

```
scene.setOnKeyPressed((KeyEvent event) -> {
    switch (event.getCode()) {
        case RIGHT:
            piece.moveRight();
            break;
        case DOWN:
            ...
    }
});
```

Obrázek 7 Zpracování vstupu

5 Skóre

5.1 Bodovací systém

Pro různé verze Tetrisu existují různé bodovací systémy. Pro můj projekt jsem použil lehce zjednodušený bodovací systém podle online Tetrisu (tetris.com/play-tetris). Viz obr. 8



Soft Drop	1 x Distance
Hard Drop	2 x Distance
Single Line Clear	100
Double Line Clear	300
Triple Line Clear	500

Obrázek 8 Tabulka pro výpočet skóre

5.1.1 Soft Drop

Tento bonus je započítán při každém kroku dílku dolů, který je vyvolán stiskem šipky ↓. Jako odměnu obdrží hráč po každém kroku jeden bod.

5.1.2 Hard Drop

Hard Drop je situace, kterou vyvolá uživatel stiskem mezerníku a dílek se okamžitě usadí dolů na místo, kde se právě na ose X nachází. Za tento krok získá okamžitě bonus odpovídající dvojnásobné vzdálenosti posuvu.

5.1.3 Multiple Line Clear

Tato situace nastane, pokud se podaří odmazat jednu a více řádek. Bonus se přidělí podle počtu řádek odmazaných v jednou tahu, jak popisuje tabulka na obr. 8.

5.2 Ukládání skóre do souboru

Pro udržování nejvyššího skóre je ve složce s projektem soubor *score.txt*. Tento soubor je vždy načten, data z něj porovnána s aktuálním skóre, následně případně zobrazen nápis **High score** a data v souboru aktualizována. Celý kód je v try-catch a počítá i s případy, kdy je soubor poškozen nebo dokonce chybí úplně. V tom případě je vytvořen a do něj uloženo aktuální skóre.

6. Závěr

Z celé odvedené práce mám dobrý dojem. Myslím, že hra má všechny potřebné funkce, a jako další vylepšení by mohlo následovat např. vložení celé Pane do většího okna, které by se přizpůsobovalo obrazovce a udržovalo hrací pole na středu, o což jsem se pokoušel, ale nakonec jsem toto kvůli některým problémům do odevzdané práce nezahrnul.

Seznam obrázků

Obr. 1 NetBeans (zdroj: https://netbeans.apache.org).....	4
Obr. 2 Java JDK (zdroj: www.java.com)	4
Obr. 3 Rotace dílku (zdroj: https://www.itnetwork.cz/csharp/monogame/csharp-tvorba-her-monogame-tetris/xna-tutorial-kostka-k-tetrisu)	4
Obr. 4 Rotace (zkráceno).....	5
Obr. 5 Reprezentace pole v paměti	5
Obr. 6 Časovač	6
Obr. 7 Zpracování vstupu.....	6
Obr. 8 Tabulka pro výpočet skóre (zdroj: https://tetris.com/play-tetris - upraveno)	7