STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO, PURKYŇOVA, PŘÍSPĚVKOVÁ ORGANIZACE

Tetris

TOMÁŠ JUKELSON

V4A

Profilová část maturitní zkoušky
MATURITNÍ PRÁCE

BRNO 2021

Prohlášení

Prohlašuji, že jsem maturitní práci Tetris vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že:

Beru na vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu SPŠ Brno, Purkyňova, příspěvková organizace.

Beru na vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, příspěvková organizace, dostupná k prezenčnímu nahlédnutí. Škola zajistí, že nebude pro nikoho možné pořizovat kopie jakékoliv části práce.

Beru na vědomí, že SPŠ Brno, Purkyňova, příspěvková organizace, má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.

Beru na vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

Tomáš Jukelson

Brechtova 1

V Brně dne 17. 4. 2021 .....

Vedoucí práce: RNDr. Lenka Hrušková

Odborný konzultant:

Anotace

Tato maturitní práce se zabývá vytvořením 2D hry Tetris jako desktopovou aplikaci. V této aplikaci ihned po spuštění se dostane uživatel do hlavního menu, jež dává uživateli možnost zadat svoji přezdívku, pod kterou bude jeho skóre zaznamenané během hraní uloženo, spustit hru, podívat se na SB, změnit hotkeys pro ovládání padajících obrazců, ale také i možnost pozměnit pozadí a hudbu aplikace. Se zahájením hry se spouští i soundtrack Tetrisu a obrazce již začínají padat. Hráč má možnost obrazce posouvat vlevo i vpravo, otáčet je a urychlit jejich pád (Pokud tak učiní, je odměněn v podobě přidání bodů do jeho skóre). Pokud hráč naskládá obrazce tak, aby vyplnili celý řádek hrací plochy, tento řádek se smaže a hráči je připočten velký počet bodů. Hra končí v momentě, kdy se nové obrazce už nemůžou ihned po vytvoření posunout níže. Během této hry se může hráč kdykoliv rozhodnout pozastavit hru a následně buďto pokračovat, nebo se může vrátit do hlavního menu, a tak předčasně ukončit hru. Pokud se takto rozhodne, jeho progres je uložen a může si jej prohlédnout ve SB, který ukazuje nejlepší výkony hráčů zaznamenané na tomto počítači.

Obsah

Teoretický úvod

Tetris je jedna z prvních 2D počítačových her. Hra spočívá v tom, že se na horní straně hrací plochy objevují obraze o objemu čtyř kostek, které postupně padají na dno hrací plochy. Hráč má možnost během pádu obrazce ovládat posunováním těchto objektů vlevo, vpravo, otáčením či zrychlením pádu. Pokaždé, kdy obrazec dopadne až na dno hrací plochy, či na obrazec, který leží v cestě, ztratí hráč kontrolu nad tímto obrazcem a nahoře se objevuje další obrazec, kterým může uživatel hýbat.

V budoucnosti plánuji programovat více počítačových her, a tak jsem si chtěl ozkoušet na vlastní pěst, co vše je třeba udělat a znát ke zhotovení hry.

Existuje více možných řešení, jak vytvořit Tetris. Mezi možné užité jazyky patří Python, C++ nebo program Unity či popřípadě HTML a PHP, pokud bych si zvolil, že aplikace má být hratelná na internetové stránce.

Mým cílem bylo úplně od základu jen za pomocí Javy vytvořit aplikaci co nejpodobnější hře Tetris s přívětivým interfacem a během toho si zapamatovat svůj postup, abych své zkušenosti mohl použít v budoucnosti.

Seznam použitých zkratek

SB – score board

HTML – Hyper Text Markup Language

PHP – PHP: Hypertext Prerocessor

GUI – Graphical User Interface

UI – User Interface

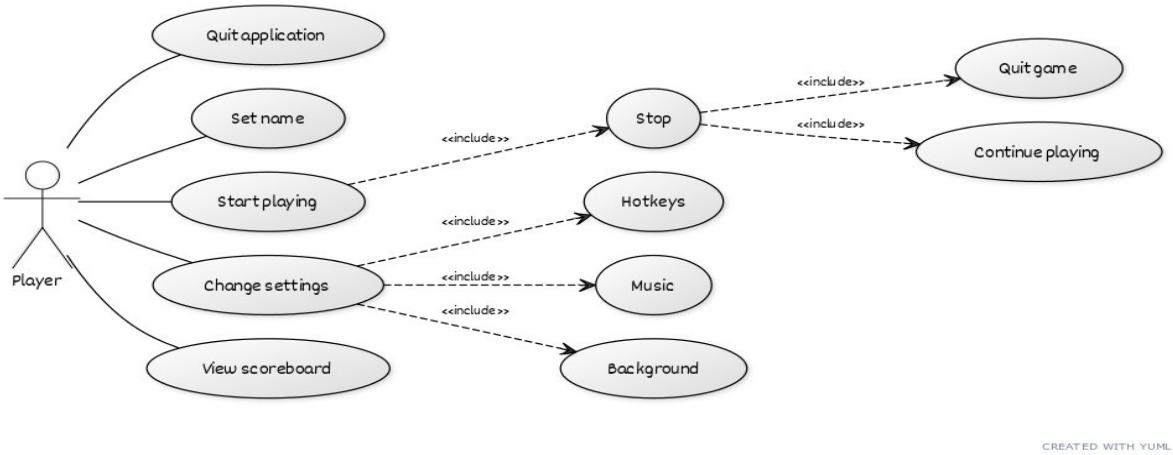
CSS – Cascading Style Sheets

UML – Unified Modeling Language

1 Rozbor řešení

Je nutné vytvořit uživatelsky přívětivé a intuitivní GUI. Stejný gameplay Tetris, možnost změny nastavení pro ovládání obrazců a možnost se následně podívat na své nejlepší dosažené skóre.

1 Návrh použití



Ilustrace 1: UML diagram

1.1 Hlavní menu

První, co uživatel uvidí po spuštění jakékoliv hry, by vždy mělo být hlavní menu, kde si může uživatel nastavit hotkeys a nickname před spuštěním hry. Zároveň by mu neměla být odepřena možnost bez jakýchkoliv prodlev hru spustit. Pokud uživatel spustí hru bez zadání přezdívky, bude jeho skóre zaznamenané pod „Guest“.

1.1.1 Tlačítko 'Play'

Velké nepřehlédnutelné tlačítko Play by zpravidla mělo být umístěno tam, kam se oči uživatele upnou hned po zapnutí programu. Proto jsem ho zvýraznil tyrkysovým ohraničením, umístil ho doprostřed obrazovky a nastavil jeho velikost na dvojnásobek velikosti ostatních tlačítek. Toto tlačítko přenese uživatele rovnou do hry Tetris.

1.1.2 Tlačítko 'Change settings'

Změna nastavení je druhá nejdůležitější funkce (hned po tlačítku 'Play'), proto jsem jej umístil hned pod tlačítko 'Play'. Pokud jej bude uživatel hledat, najde ho téměř ihned. Toto tlačítko umožní uživateli změnu hudby, barvy pozadí a hotkeys pro ovládání obrazců.

1.1.3 Tlačítko 'View scoreboard'

Pohled na své skóre není klíčové pro hru, proto je umístěno jako nejnižší z tlačítek. Toto tlačítko umožní uživateli se podívat na 5 nejvyšších dosažených bodů na tomto počítači.

1.1.4 Pole pro vepsání jména

Textové vstupní pole s průhledným textem „Your name:“, který zmizí po započnutí psaní do tohoto pole má být intuitivní způsob, jak si uživatel zadá svoji přezdívku, pod kterou si přeje mít uložené skóre po skončení jeho hry. Pro přehlednost jsem toto textové vstupní pole umístil právě nad tlačítko 'Play'

1.1.5 Quit application

Tlačítko umožňující uživateli odejít z aplikace bývá obvykle nejmenší a zpravidla umístěné v jednom z horních rohů aplikace. Učinil jsem taktéž.

1.2 Gameplayová část

Hlavní část je hrací pole 12 na 30. Vpravo od hracího pole může být čárou oddělená část, kde se zobrazuje, jaké má zrovna hráč skóre. Zároveň je třeba umístit Stop button tak, aby neblokoval pohled na hrací plochu a zbytečně neztěžoval hraní.

Po ukončení hry způsobem prohry mi přišlo vhodné zajistit, aby se přes velkou část aplikace zobrazilo velký text „Game Over“.

Nově vytvořené obrazce se vždy objeví uprostřed nejvyššího řádku hrací plochy. Tyto obrazce padají na dno hrací plochy tak, že každý časový interval, se posunjí vždy o vzdálenost jednoho čtverečku hrací plochy. Časový interval může buďto zůstat v průběhu celé hry stejný nebo jak to bývá u většiny verzí Tetrisu – čas v každém intervalu se bude postupně zmenšovat, aby se tak hra stávala těžší s každou sekundou hraní. Další velmi důležité pravidlo Tetrisu je, že pokud hráč naskládá obrazce tak, aby utvořily jeden nebo více plných řad bez mezer, dané řady zmizí. Všechny čtverečky, které ležely nad těmito řadami budou posunuty níže přesně o tolik políček, kolik se odstranilo řad. Hráč může obrazce posouvat vlevo a vpravo, otáčet je a urychlit jejich pád (Pokud takto hráč urychlí pád, je za to odměněn v podobě 1 bodu za každé posunuté políčko). Kontrolu nad obrazcem ztrácí hráč v momentě, kdy se obrazec dotkne dna hrací plochy nebo obrazce, jenž mu leží v cestě. Zároveň dostává hráč kontrolu nad nově vytvořeným obrazcem na vrcholu hrací plochy. Během celého průběhu gameplaye by měl hrát Tetris soundtrack.

1.3 Změna nastavení

Podstatná funkce, kterou by měla mít každá hra a ne jenom Tetris, je změna nastavení ovládání. V tomto případě se jedná změnu hotkeys pro pohyb obrazců. Další možné funkce, které toto submenu může nabízet, je výběr hudby, která bude v průběhu gameplaye hrát nebo změna pozadí.

1.4 Scoreboard

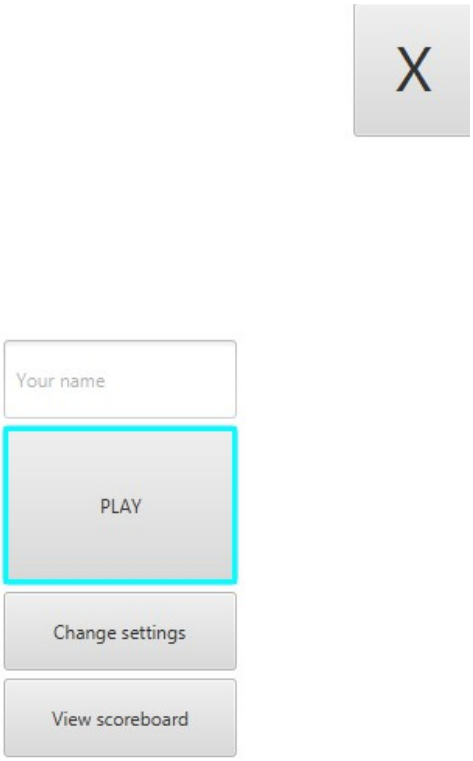
Scoreboard je tabulka, kde jsou vyobrazeny dosavadní úspěchy hráčů v této hře. Informace na SB bývají často zapsané ve formátu pořadí/nickname/skóre. Je možné dát uživateli možnost si prohlédnout všechny zapsané průběhy jeho her za pomoci například scroll baru, ale mě přišlo rozumnější zobrazit jenom **prvních pět** nejlepších průběhů, neboť si myslím, že málokoho budou zajímat úspěchy sotva o pět bodů větší než 0. Zároveň mi to takto přijde přehlednější.

2 Realizace

Program jsem se rozhodl vytvářit v JavaFX - Javou implementovaný builder GUI.

2.1 Hlavní menu

Ihned po spuštění programu se vygeneruje hlavní menu. To zahrnuje tlačítka 'Play', 'Change settings', 'View scoreboard', vstupní pole s textem „Your name“ a tlačítko ukončující program. Jak jsem již zmínil, tlačítko 'Play' je zvýrazněné jeho velikostí a ohraničením.

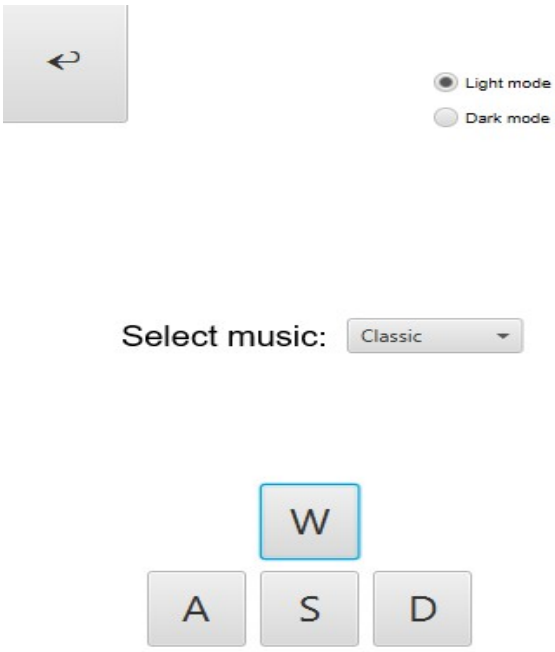


Ilustrace 2: Main menu

2.2 Changing settings

Po stisknutí tlačítka 'Change settings' umístěného v hlavním menu se z obrazovky smažou nynější textová pole a tlačítka. Na místo toho se vygeneruje submenu pro změnu nastavení obsahující tlačítko na návrat do hlavního menu', které opět smaže vše co toto submenu obsahuje a znovu vygeneruje hlavní menu. Toto submenu obsahuje 4 klávesy, kterýma

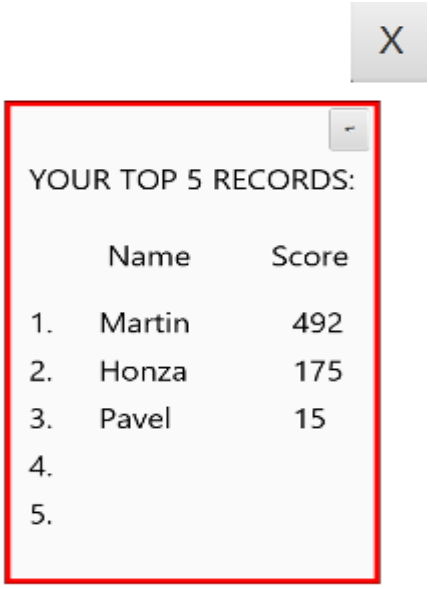
může hráč ovládat padající obrazce. Po stisknutí těchto kláves na obrazovce dostane uživatel možnost změnu daného hotkey. Uživatel si zde může také zvolit barvu pozadí a typ Tetris soundtracku.



Ilustrace 3: Changing settings

2.3 View scoreboard

Po stisknutí tlačítka 'View scoreboard' umístěném v hlavním menu se vygeneruje rámeček s rudým ohraničením obsahující dosavadní záznamy her hráčů a tlačítko pro opětovné schování SB. Jak jsem již zmiňoval, SB zobrazí pouze **nejlepších pět** záznamů.



Ilustrace 4: Scoreboard

2.3.1 Způsob uložení a čtení skóre

Po ukončení hry (ať už vlastnoručním navrácením do hlavního manu nebo prohrou) se do jednoho arraye uloží dosažené skóre a do druhého arraye se uloží nickname, pod kterým hráč hrál. Array se skórem se ihned začne řadit. Existuje nespočet řadících algoritmů. Mě přišlo nejlogičtější použít Bubble sort, neboť potřebuji správně seřadit pouze prvních **pět** míst a Bubble sort řadí tak, že jeho první seřazená čísla jsou právě ta nejvyšší (nebo nejnižší – podle toho jakým směrem se řadí). Z tohoto důvodu můžu cyklus řazení přerušit kdykoliv potřebuji a ta důležitá místa budou mít správnou hodnotu.

```
private static void Sort() {
    System.out.println("Sorting started");
    int helper = 0;
    int bottom = 0;
    for (int i=0; i<5; i++) { //only top 5 needed
        for (int j=Main.users; j > 0+bottom; j--) {
            System.out.println("one mini cycle started");
            if (Main.savedScore_numbers[j] > Main.savedScore_numbers[j - 1]) {
                helper = Main.savedScore_numbers[j];
                Main.savedScore_numbers[j] = Main.savedScore_numbers[j - 1];
                Main.savedScore_numbers[j - 1] = helper;
                nameSwap(j);
            }
        }
        bottom++;
        if (i+1 > Main.users) {
            break;
        }
    }
}

private static void nameSwap(int index){
    String temp;

    temp = Main.savedScore_names[index];
    Main.savedScore_names[index] = Main.savedScore_names[index-1];
    Main.savedScore_names[index-1] = temp;
}
```

Ilustrace 5: Bubble sort

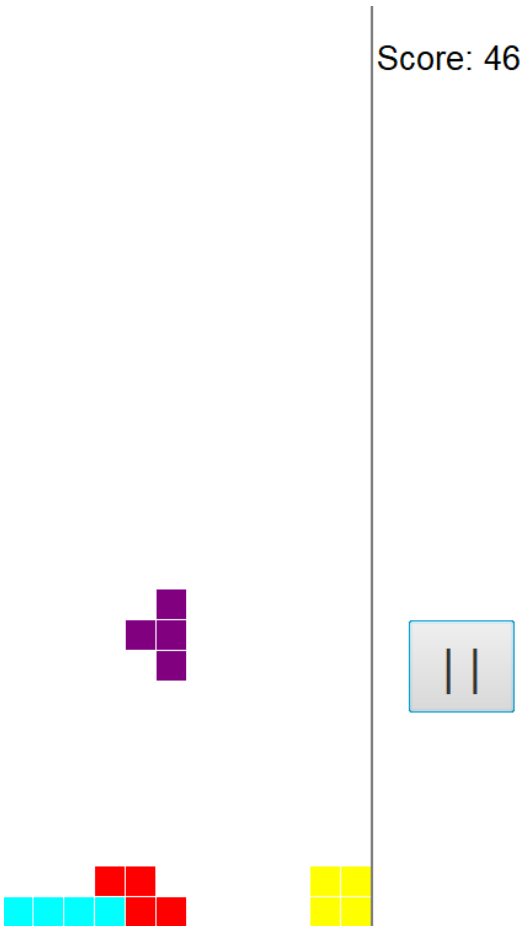
Pokaždé když prohodím místy hodnoty arraye skóre, prohodím stejná místa i v arrayi jmen. Po seřazení arrayí, přepisuji informaci do textových souborů. Tímto způsobem docílím zachování informací i po vypnutí programu.

Teď můžu při každém následném spuštění programu načíst starší záznamy her a pracovat s nimi.

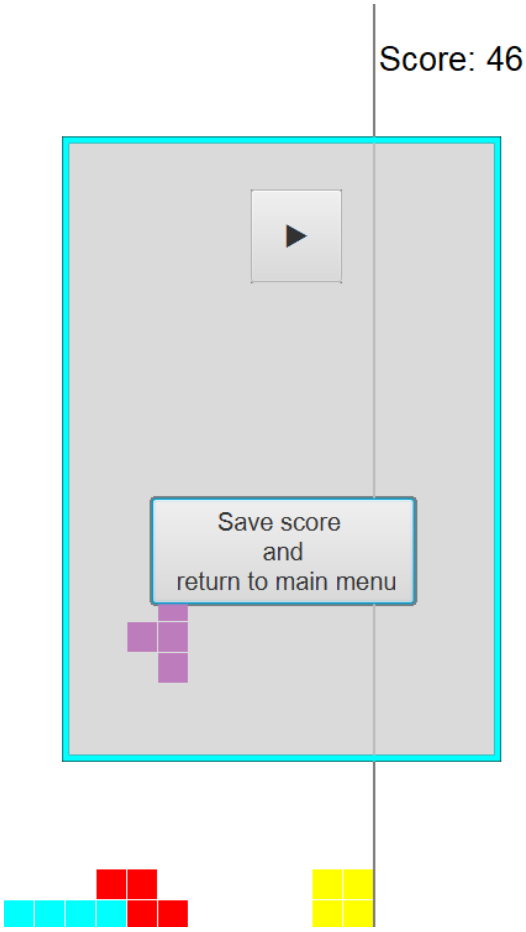
```
public static void recordScore() {

    //dont sort only 1 score
    if (Main.users != 0) {
        try {
            Sort();
        }
        catch (Exception e) {
            System.out.println(e + "\n Skipping the problem");
        }
    }

    try {
        FileWriter myWriter = new FileWriter( fileName: "scoredatabase.txt");
        for (int i = 0; i<Main.savedScore_numbers.length; i++) {
            if (Main.savedScore_names[i] != null) {
                myWriter.write(String.valueOf(Main.savedScore_numbers[i]));
                myWriter.write( str: "\n");
            }
        }
        myWriter.close();
        System.out.println("Recording of your score was successful.");
    } catch (IOException e) {
        System.out.println("Couldn't record your score.");
        e.printStackTrace();
    }

    try {
        FileWriter myWriter = new FileWriter( fileName: "namedatabase.txt");
        for (int i = 0; i<Main.savedScore_names.length; i++) {
            if (Main.savedScore_names[i] != null) {
                myWriter.write(Main.savedScore_names[i]);
                myWriter.write( str: "\n");
            }
        }
        myWriter.close();
        System.out.println("Recording of your name was successful.");
    } catch (IOException e) {
        System.out.println("Couldn't record your name.");
        e.printStackTrace();
    }
}
```

Ilustrace 6: Zapisování do souborů

2.4 Gameplay



Ilustrace 8: Gameplay



Ilustrace 7: Gameplay stopped

Po stisknutí klávesy 'Play' v hlavním menu začíná gameplayová část Tetrisu. Vygeneruje se hrací pole 12 na 30 a začíná hrát hudba. Na pravé části aplikace se zobrazí nynější

2.4.1 Způsob uložení a čtení skóre

Po ukončení hry (ať už vlastnoručnímskóre a tlačítko k pozastavení hry. Pokud hráč tlačítko stiskne, hra bude pozastavena a hráč si může následně zvolit, zda-li chce pokračovat ve hraní, nebo jestli chce hru ukončit.Při každém vytvořením obrazce se v pozadí náhodně generuje číslo. Podle hodnoty tohoto čísla se určuje jaký obrazec bude vytvořen. Obrazec vždy obsahuje 4 čtverečky a je zbarven podle tvaru, ve kterém jsou čtverečky sestaveny. Vždy, když obrazec někam dopadne, kontroluje se zda-li se jedna z řad nenaplnila a je generován další obrazec. Průběžně se také kontroluje, které klávesy hráč drží. Pokud nějaká z těchto kláves odpovídá hotkey pro ovládání obrazců, provede se.

```
playing = true;

//repeated refreshing of game status
TimerTask task = () -> {
        Platform.runLater(new Runnable() {
            public void run() {

                if (object.a.getY() == 0 || object.b.getY() == 0 || object.c.getY() == 0 || object.d.getY() == 0)
                    top++;
                else
                    top = 0;

                if (top == 2) {
                    // GAME OVER
                    Text over = new Text("GAME OVER");
                    over.setFill(Color.RED);
                    over.setStyle("-fx-font: 70 arial;");
                    over.setY(250);
                    over.setX(10);
                    group.getChildren().add(over);
                    playing = false;
                    cancel();
                }

                if (playing) {
                    group.getChildren().removeAll(scoretext);
                    group.getChildren().addAll(scoretext);
                    Moving.MoveDown(object);
                    scoretext.setText("Score: " + Integer.toString(score));
                }
            }
        });
};
//repeating task in a set period
if (!threadIsRunning) {
    threadIsRunning = true;
    Timer fall = new Timer();
    fall.schedule(task, delay: 0, period: 300);
}
```

Ilustrace 9: Thread

Zde je ukázáno, jak vypadá vlákno, které neustále kontroluje, zda-li se nějaký obrazec nedotýká vrcholu hrací plochy, které což by vedlo k vynucenému ukončení hry. Zároveň s každým uběhlým intervalem, se refreshuje ukazatel skóre.

3 Testy a bugfixy

Během své práce jsem průběžně testoval samostatné kusy programu odděleně od zbytku a pak jsem jenom na konci vyzkoušel jestli se tyto části vzájemně jakýmkoliv způsob neruší.

3.1 Mazání řádků a konce hry

První část kterou jsem vytvořil a testoval byla gameplayová část. Zajímalo mě jestli se správně mažou řádky. Po tom co jsem si ověřil, že se může mazat i více řádků zaráz, zavedl jsem i kontrolu konce hry způsobem prohry.

3.2 Z menu do hry

Po vytvoření jednoduchého menu s jediným funkčním tlačítkem 'Play' jsem ověřoval, jestli po opakovaném zapínání a vypínání hry nenastane nechtěná prohra. Protože jsem nastavil aby konec hry nastal, když se obrazec nachází na vrchu hrací plochy dva časové průběhy po sobě, mohlo nastat k předčasnému ukončení hry pokud bych vypnul hru právě v momentě, kdy se obrazec nacházel nahoře. Tomuto možnému problému lze předejít pouhým vynulováním proměnné, která počítá, jak dlouho se nachází obrazec na vrchu obrazovky.

Další problém bylo časování. Pokud se ze hry opakovaně odcházelo a zase přicházelo, vytvářelo se i více vláken určující intervaly padání obrazců. Nechal jsem proto začít pouze jedno vlákno, které bude celou dobu pracovat na pozadí i během menu. Přestože vlákno běží, není vygenerovaná hrací plocha ani žádné obrazce, které by mohly padat, a proto nemůže dojít ke konci hry, přestože se uživatel nachází pouze v hlavním menu.

```
if (!threadIsRunning) {
    threadIsRunning = true;
    Timer fall = new Timer();
    fall.schedule(task, delay: 0, period: 300);
}
```

Ilustrace 10: No multithreading

3.3 Ovládání

Zkoušel jsem hru spustit s přednastaveným ovládáním obrazců pomocí šipek. Bohužel program můj vstup neregistroval. Registroval vše ostatní jako byly obyčejné znaky či ctrl nebo alt. Nepodařilo se mi zjistit, proč jsou šipky jediný vstup uživatele, který program ani nepřijme. Proto jsem nechal jako defaultní nastavení klávesy A, W, S, D

```
public static void moveOnKeyPress(Form form) {
    scene.setOnKeyPressed(new EventHandler<KeyEvent>() {
        @Override
        public void handle(KeyEvent event) {
            String input = event.getCode().toString();
            System.out.println(input); //what key is pressed

            if (input.equals(UPcase)) {
                Moving.MoveTurn(form);
            }
            else if (input.equals(RIGHTcase)) {
                Moving.MoveRight(form);
            }
            else if (input.equals(LEFTcase)) {
                Moving.MoveLeft(form);
            }
            else if (input.equals(DOWNcase)) {
                Moving.MoveDown(form);
                score++;
            }
            else {
                System.out.println("Your input did not get recognised");
            }
        }
    });
}
```

Ilustrace 11: Keyboard inputs

3.4 Řazení a zápis skóre

Moje původní řešení bylo, zapisovat i nickname i skóre do stejného arraylistu a následně zapisovat vše do jednoho textového dokumentu. Vše fungovalo dokud nebylo třeba, aby se arraylist řadil potřetí. Kontroloval jsem aby pořadí skóre a jména se nespletla a opravdu se řadilo pouze podle skóre, nikoliv podle jména. I přes přetypování různých prvků v různých momentech řazení docházelo k neschopnosti programu rozlišit, zda se jedná o String nebo Integer. Problémům jsem předešel zapisováním skóre do jiného arraye a textového souboru, než do kterého byl zapisován nickname.

Programování zbytku aplikace procházelo bez dalších větších komplikací.

Závěr

Jediný nedostatek, který má mnou vytvořený Tetris oproti jiným verzím je, že se mi nepodařilo vymyslet instantní pád obrazce (obvykle pomocí mezerníku). Kromě toho se mi podařilo dodržet zásadní funkce jako je například pohyb obrazců, mazání plných řad, výpočet skóre a změnu nastavení pro dle preferencí uživatele.

Během své práce jsem si rozšířil znalosti ohledně práce s JavaFX a zároveň jsem využil většinu svých dosavadních zkušeností jako jsou způsoby řazení, vlákna a užívání externích souborů.

Zároveň jsem si dobře uvědomil postup pro vytváření her. To zahrnuje například: kterou část hry programovat prvně, časté možné chyby a nejlepší způsoby řešení těchto chyb a v neposlední řadě také přehlednost a intuitivitu programu.

Seznam ilustrací

Zdroje

[1] *JavaFX 2D Shapes* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[2] *JavaFX Text* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[3] *JavaFX Transformation* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[4] *JavaFX UI* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[5] *JavaFX CSS* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[6] *Media with JavaFX* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[7] *JavaFX Event Handling* [online]. Dostupné z: https://www.javatpoint.com/javafx-tutorial

[8] *Java Multithreading* [online]. Dostupné z: https://www.tutorialspoint.com/java/

[9] *Java Arrays* [online]. Dostupné z: https://www.tutorialspoint.com/java/

Přílohy

Uživatelský manuál

Odkaz na github: https://github.com/NejakyJmeno/Maturitni-prace/