Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÁ PRÁCE

2023

Jan Ševčík

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÁ PRÁCE

Předmět: Programování

Téma: Hra 2048

Autor: Jan Ševčík

Třída: 2. E

Školní rok: 2022/2023

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlášení

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 …….. dne …… ……

Jan Ševčík

Anotace

Toto je dokumentace mé ročníkové práce. V této práci popisuji jak kód funguje, jaká je grafická stránka a jak funguje aplikace jako celek. Práci jsem si zadal na začátku roku (viz. kapitola zadání). Zadání je naprogramovat známou hru 2048, ve které spojujete čísla a cílem je ve většině verzí těchto her dostat se k číslu 2048 nebo 8192.

Annotation

This is the documentation of my year project. In this documentation I explain how the code works, how the graphical side works and how the application works as a whole. I picked what I had to do at the start of this year (via. chapter “zadání”). The goal is to program a well known game called 2048, in which you connect numbers. In most of these games the goal is to get to the number 2048 or 8192.

Zadání

Jedná se o hru ve které máte pole 4*4 a v něm spojujete čtverečky se stejnými čísly. Čtverečky se v poli objevují náhodně po každém tahu jeden (na začátku dva (aby bylo co spojit)). Cílem je spojit čtverečky do čísla 2048 (nebo více). Prohrajete když už není čtverečky kam pohnout.

Obsah

1 Úvod

Toto je dokumentace mého programu vytvořeného podle zadání (viz. kapitola zadání), které jsem si vymyslel na začátku roku. Je zde zdokumentováno všechno od kódu a jeho logiky až po veškeré grafické části programu, které zahrnují FXML soubory a obrázky. Věnuji se i tomu jak celý program funguje a co mi dělalo největší problém. Na vyvinutí programu jsem využil programy InteliJ, SceneBuilder a Gimp.

Toto téma jsem si vybral, protože jsem zadanou hru hrál a stále hraji a chtěl jsem porozumět tomu jak hra funguje a díky tomu se ve hře možná i zlepšit.

2 KÓD

2.1 Třída “HelloApplication”

Třída HelloApplication se v programu zabývá otevřením FXML souboru a obsahuje také část kódu, který se stará o to, aby byl program ovladatelný pomocí kláves. Další části kódu už jsou pouze nastavení názvu stage a přidání ikony, která se objevuje ve windows na spodní liště.

FXML soubor je otevřen standardním způsobem, který mi vygenerovalo rovnou IDE při vytváření projektu, nastavení názvu stage je také jednoduchá jedna řádka kódu a nastavení ikony je také pouze jedna řádka kódu.

obr.1: Ikonka hry

Výrazně složitější částí této třídy je kód, který se stará o to, aby byl program ovladatelný pomocí kláves. Jak napsat tuto část kódu jsem zjistil ve videu (Bro Code, 2021). V kódu se vytvoří instance třídy “Controller” následně program kontroluje, zda byla nějaká klávesa stisknuta a pokud ano zjistí, zda to bylo “w”, “s”, “a” nebo “d” (tyto klávesy jsou v dnešní době standartně používány pro hraní videoher). Pokud to byla jedna z těchto kláves zavolá program ve třídě “Controller” metodu “up()”, “down()”, “left()” nebo “right()”.

```
Controller controller = loader.getController();
scene.setOnKeyPressed(new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
        switch(event.getCode()){
            case W:
                try {
                    controller.Up();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
            case S:
                try {
                    controller.Down();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
            case A:
                try {
                    controller.Left();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
            case D:
                try {
                    controller.Right();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
        }
    }
});
```

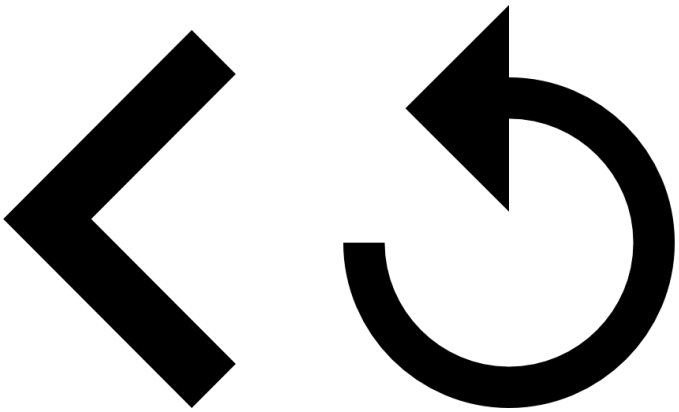
obr.2: Kód, který kontroluje klávesy

2.2 Třída “Controller”

Třída “Controller” implementuje interface “Initializable”, který obsahuje abstraktní metodu “initialize()”. Obsahuje i veškerou logiku za mým programem v různých metodách, které níže popíši. Také jsou vytvořeny různé proměnné a inicializované různé objekty z výše zmíněných FXML souborů.

2.2.1 metoda “initialize()”

Tato metoda se spustí při prvním použité třídy “Controller”, která nastane hned při otevírání prvního FXML souboru. Stará se o nastavení dvou ikon (viz. kapitola 3.5.2) do dvou buttonů, které obsahuje FXML “2048.fxml”. Dále kontroluje, zda existuje soubor “pane.dat” (viz. kapitola 1.4). Pokud ano volá metodu “fileload()”, která se souborem dále pracuje a pokud ne volá metodu Random(), která vytváří úplně nové dvě “číslovky” se kterými se následovně hraje. O této metodě jsem se dozvěděl ve videu (Bro Code, 2021).



obr. 3: Obrázky využité v programu

2.2.2 metoda “fileload()”

Metoda “fileload()” má v mém programu jen jeden úkol, a to sice pokud je volána nahrát to co je uloženého v souboru “pane.dat” (viz. kapitola 1.4) a to co je uloženého v souboru “score.dat” (viz. kapitola 1.5) na obrazovku. To znamená zobrazit uložené “číslovky” a “skóre”. Nahrává tedy to, co již “hráč” někdy nahrál.

2.2.3 metoda “Random()”

“Random()” je metoda volána v případě, že je potřeba vytvořit další novou “startovní” pozici. Tato metoda je tedy jednoduchý kus kódu, který náhodně vybere dvě možná místa v poli kam umístit začáteční “číslice”. Metoda samozřejmě dbá na to, aby se číslice nepřekrývaly (nebyli na stejných souřadnicích) a občas vygeneruje místo dvou dvojek jednu čtyřku a jednu dvojku. (v originální hře funguje začátek stejně).

```
int x = r.nextInt( origin: 0,  bound: 4);
int y = r.nextInt( origin: 0,  bound: 4);
board.add(x, y,  v: 2);

int x1 = r.nextInt( origin: 0,  bound: 4);
int y1 = r.nextInt( origin: 0,  bound: 4);
while (x1 == x && y1 == y) {
    x1 = r.nextInt( origin: 0,  bound: 4);
    y1 = r.nextInt( origin: 0,  bound: 4);
}
int r4 = r.nextInt( origin: 0,  bound: 6);
if (r4 == 5) {
    board.add(x1, y1,  v: 4);
} else {
    board.add(x1, y1,  v: 2);
}

board.printo(Pole);
```

obr. 4: Kód metody “Random()”

2.2.4 metoda “spawn()”

Tato metoda je podobná metodě “Random()” v tom, že se také zabývá generováním nových “číslic”. Tato je však volána po každém tahu hráče, aby nalezla volné místo v poli a přidala na něj “číslovku”. Podobně jako metoda “Random()” je zde šance na vygenerování místo číslice dva číslici čtyři.

Tato metoda obsahuje pro své účely poměrně jednoduchý algoritmus. Nejprve vygeneruje dvě náhodné číslice, které reprezentují budoucí pozici, kde by se mohla nacházet další “číslice”. Následovně program projede celé pole již existujících čísel a pokud jsou vygenerované souřadnice prázdné skončí a vloží zde “číslici”. Pokud nejsou souřadnice prázdné celý proces opakuje.

```
int x = 0;
int y = 0;
boolean yes = true;

while (yes) {
    x = r.nextInt( origin: 0,  bound: 4);
    y = r.nextInt( origin: 0,  bound: 4);
    for (int c = 0; c < 4; c++) {
        for (int k = 0; k < 4; k++) {
            if (board.arr[x][y] == 0) {
                yes = false;
                break;
            }
        }
    }
}
```

obr. 5: Kód metody”spawn()”

2.2.5 metoda “autosave()”

Metoda “autosave()” je jedna z nejjednodušších metod v celém programu, protože jediné, co dělá, je že do souboru “pane.dat” (viz. kapitola 1.4) a “score.dat” (viz. kapitola 1.5) ukládá aktuální stav celého hracího pole a “skóre”.

2.2.6 metoda “Restart()”

“Restart()” je metoda volána buttonem, který je obsáhnut v FXML souboru “2048.fxml” (viz. kapitola 1.1). Jak už název napovídá tato metoda je zodpovědná za “restartování” hry. Metoda tedy vymaže celé aktuální hrací pole a zavolá metodu “Random()” (viz. kapitola 2.2.3), která vygeneruje dvě začáteční “číslice”. Dále také vyresetuje “skóre” a zavolá metodu “autosave()” (viz. kapitola 2.2.5), která nové změny uloží.

2.2.7 metoda “undosave()”

Tato metoda je využívána na uložení předchozího stavu pole a “skóre”. Tyto hodnoty ukládá do pole pod názvem “temp” a “skóre” do proměnné “tempscore”. Je volána před každým tahem metodami “up()”, “dowm()”, “left()” a “right()”.

2.2.8 metoda “Undo()”

Metoda “Undo()” je volána Buttonem, který je obsáhnut v souboru “2048.fxml” (viz. kapitola 1.1). Kód v této metodě “vrací pole o krok dozadu”. Tato metoda po zavolání nastaví hodnotu “skóre” na hodnotu, která je uložena v proměnné “tempscore” (viz. kapitola 2.2.7), a do hracího pole nahraje pole, které je uložené v poli “temp” (viz. kapitola 2.2.7). Nakonec metoda zavolá metodu “autosave()”, aby byla změna uložena.

2.2.9 metoda “Up()”, “Down()”, “Left()” a “Right()”

Tyto čtyři metody se zabývají s počítáním pozice, kam se mají posunout všechny “číslice”. Všechny metody jsou volány za pomocí zmáčknutí kláves “W” (metoda “Up()”), “S” (metoda “Down()”), “A” (metoda “Left()”) a “D” (metoda “Right()”). Jak už z názvů metod vypovídá každá se zabývá jiným směrem. Metoda “Up()” posouvá “číslice” nahoru, metoda “Down()” dolů, “Left()” doleva a metoda “Right()” doprava.

Všechny tři metody obsahují podobné algoritmy liší se pouze v tom, zda posouvají “číslice” diagonálně nebo horizontálně. Algoritmy v těchto metodách pouze kontrolují od souřadnic, na které mají být “číslice” posunuté a kontrolují všechny možné situace a podle nich upravují pole pomocí metod “add()” a “remove()” (viz. kapitola 2.4.1). Nejprve tedy projede algoritmus druhou řadu, protože na první nelze nijak “číslicemi” posouvat.

```
if (board.arr[g][1] > 0 && board.arr[g][0] == 0) {
    board.add(g, Y: 0, board.arr[g][1]);
    board.remove(g, Y: 1);
    moved++;
} else if (board.arr[g][1] > 0 && board.arr[g][0] == board.arr[g][1]) {
    scorenumber = scorenumber + board.arr[g][1] * 2;
    board.add(g, Y: 0, v: board.arr[g][1] * 2);
    board.remove(g, Y: 1);
    moved++;
}
```

obr. 6: Příklad části kódu metody “Left()”, která ošetřuje druhou řadu

Tato část algoritmu, pokud je nutno přepíše pole a při spojení dvou číslic do jedné počítá “skóre”. Při jakémkoliv pohybu v poli se ještě k proměnné “moved” přičte 1. Tato proměnná na konci pomáhá s rozhodnutím, zda má být přidána nová “číslice” (pokud je větší než 1 je

volána metoda “spawn()” (viz. kapitola 2.2.4)). Po průchodu druhé řady logicky přijde i průchod třetí řady.

```
if (board.arr[g][2] > 0 && board.arr[g][0] == 0 && board.arr[g][1] == 0) {
    board.add(g, Y: 0, board.arr[g][2]);
    board.remove(g, Y: 2);
    moved++;
} else if (board.arr[g][2] > 0 && board.arr[g][0] > 0 && board.arr[g][2] != board.arr[g][0] && board.arr[g][1] == 0) {
    board.add(g, Y: 1, board.arr[g][2]);
    board.remove(g, Y: 2);
    moved++;
} else if (board.arr[g][2] > 0 && board.arr[g][0] > 0 && board.arr[g][2] == board.arr[g][0] && board.arr[g][1] == 0) {
    scorenumber = scorenumber + board.arr[g][2] * 2;
    board.add(g, Y: 0, v: board.arr[g][2] * 2);
    board.remove(g, Y: 2);
    moved++;
} else if (board.arr[g][2] > 0 && board.arr[g][1] > 0 && board.arr[g][2] == board.arr[g][1]) {
    scorenumber = scorenumber + board.arr[g][2] * 2;
    board.add(g, Y: 1, v: board.arr[g][2] * 2);
    board.remove(g, Y: 2);
    moved++;
}
```

obr. 7: Příklad části kódu metody “Left()”, která ošetřuje třetí řadu

Průchod druhé, třetí a čtvrté řady jsou v podstatě stejné až na to že s každou řadou přichází více možných pozic kde mohou “číslice” být.

```
if (board.arr[g][3] > 0 && board.arr[g][0] == 0 && board.arr[g][1] == 0 && board.arr[g][2] == 0) {
    board.add(g, Y: 0, board.arr[g][3]);
    board.remove(g, Y: 3);
    moved++;
} else if (board.arr[g][3] > 0 && board.arr[g][0] > 0 && board.arr[g][1] == 0 && board.arr[g][2] == 0 && board.arr[g][3] != board.arr[g][0]) {
    board.add(g, Y: 1, board.arr[g][3]);
    board.remove(g, Y: 3);
    moved++;
} else if (board.arr[g][3] > 0 && board.arr[g][0] > 0 && board.arr[g][1] > 0 && board.arr[g][2] == 0 && board.arr[g][3] != board.arr[g][1]) {
    board.add(g, Y: 2, board.arr[g][3]);
    board.remove(g, Y: 3);
    moved++;
} else if (board.arr[g][3] > 0 && board.arr[g][0] > 0 && board.arr[g][1] == 0 && board.arr[g][2] == 0 && board.arr[g][3] == board.arr[g][0]) {
    scorenumber = scorenumber + board.arr[g][3] * 2;
    board.add(g, Y: 0, v: board.arr[g][3] * 2);
    board.remove(g, Y: 3);
    moved++;
} else if (board.arr[g][3] > 0 && board.arr[g][0] > 0 && board.arr[g][1] > 0 && board.arr[g][2] == 0 && board.arr[g][3] == board.arr[g][1]) {
    scorenumber = scorenumber + board.arr[g][3] * 2;
    board.add(g, Y: 1, v: board.arr[g][3] * 2);
    board.remove(g, Y: 3);
    moved++;
} else if (board.arr[g][3] > 0 && board.arr[g][0] > 0 && board.arr[g][1] > 0 && board.arr[g][2] > 0 && board.arr[g][3] == board.arr[g][2]) {
    scorenumber = scorenumber + board.arr[g][3] * 2;
    board.add(g, Y: 2, v: board.arr[g][3] * 2);
    board.remove(g, Y: 3);
    moved++;
}
```

obr. 8: Příklad části kódu metody “Left()”, která ošetřuje čtvrtou řadu

Na konci metody jsou pouze volány metody “checkWin()” (viz. kapitola 2.2.10), “checkLost()” (viz. kapitola 2.2.11) a případně jak už bylo řečeno metoda “spawn()” (viz. kapitola 2.2.4). V poslední řadě je celé pole přetisknuto do nově přepočítané podoby pomocí metody “printo()”, které je metodou třídy “charts”, a je přepsáno nově spočítané “skóre”.

2.2.10 metoda “checkWin()”

“checkWin()” je velice jednoduchá metoda s úkolem zjistit zda “hráč” vyhrál. Jediné, co dělá je, že projde celé pole a pokud nalezne jedinou “číslici”, která má větší číslo než 8192, zavolá metodu “won()” (viz. kapitola 2.2.12).

2.2.11 metoda “checkLost()”

Metoda “checkLost()” má jediný úkol, a to sice zjistit, zda hráč prohrál. Činí tak za pomocí metod “checkUp()”, “checkDown()”, “checkRight()” a “checkLeft()”. Tyto metody zjišťují, jestli je možné pohnout jakoukoli “číslicí”. Dělají tak pomocí podobného algoritmu, jaký je používán pro pohybování “číslic”.

```
for (int g = 3; g >= 0; g--) {

    if (board.arr[2][g] > 0 && board.arr[3][g] == 0) {
        return false;
    } else if (board.arr[2][g] > 0 && board.arr[3][g] == board.arr[2][g]) {
        return false;
    }

    if (board.arr[1][g] > 0 && board.arr[3][g] == 0 && board.arr[2][g] == 0) {
        return false;
    } else if (board.arr[1][g] > 0 && board.arr[3][g] > 0 && board.arr[1][g] != board.arr[3][g] && board.arr[2][g] == 0) {
        return false;
    } else if (board.arr[1][g] > 0 && board.arr[3][g] > 0 && board.arr[1][g] == board.arr[3][g] && board.arr[2][g] == 0) {
        return false;
    } else if (board.arr[1][g] > 0 && board.arr[2][g] > 0 && board.arr[1][g] == board.arr[2][g]) {
        return false;
    }

    if (board.arr[0][g] > 0 && board.arr[3][g] == 0 && board.arr[2][g] == 0 && board.arr[1][g] == 0) {
        return false;
    } else if (board.arr[0][g] > 0 && board.arr[3][g] > 0 && board.arr[2][g] == 0 && board.arr[1][g] == 0 && board.arr[0][g] != board.arr[3][g]) {
        return false;
    } else if (board.arr[0][g] > 0 && board.arr[3][g] > 0 && board.arr[2][g] > 0 && board.arr[1][g] == 0 && board.arr[0][g] != board.arr[2][g]) {
        return false;
    } else if (board.arr[0][g] > 0 && board.arr[3][g] > 0 && board.arr[2][g] == 0 && board.arr[1][g] == 0 && board.arr[0][g] == board.arr[3][g]) {
        return false;
    } else if (board.arr[0][g] > 0 && board.arr[3][g] > 0 && board.arr[2][g] > 0 && board.arr[1][g] == 0 && board.arr[0][g] == board.arr[2][g]) {
        return false;
    } else if (board.arr[0][g] > 0 && board.arr[3][g] > 0 && board.arr[2][g] > 0 && board.arr[1][g] > 0 && board.arr[0][g] == board.arr[1][g]) {
        return false;
    }
}
return true;
```

obr. 9: Kód metody “checkRight()”

Pokud všechny tyto čtyři metody vrátí “true” metoda “checkLost()” zavolá metodu “lost()”.

2.2.12 metoda “win()” a “lost()”

Tyto dvě metody pouze volají metodu “Restart()” a mění soubor FXML na stage. Metoda “win()” mění FXML soubor z “2048.fxml” na “Win.fxml” a metoda “lost()” mění z “2048.fxml” na “Lost.fxml”.

2.3 Třída “Tools”

Třída “tools” obsahuje pouze jednu metodu a to sice “decode()”. Tato metoda se zabývá převedením souřadnice “číslic” (souřadnice je číslo od 0 do 3) na číslo, které následovně dosadí “číslici” v AnchorPanu (na tomto AnchorPanu se nachází hrací pole) na správné místo.

```
if (h.equals("x")) {
    if (x == 0) {
        return 7;
    } else if (x == 1) {
        return 81;
    } else if (x == 2) {
        return 154.5;
    } else if (x == 3) {
        return 227.5;
    }
} else {
    if (x == 0) {
        return 7;
    } else if (x == 1) {
        return 80;
    } else if (x == 2) {
        return 154.5;
    } else if (x == 3) {
        return 227.5;
    }
}
return 0;
```

obr. 10: Kód metody “decode()”

2.4 Třída “Charts”

Třída “charts” má na začátku vytvořen parametr “arr”, který je dvourozměrné pole reprezentující hrací pole, a instanci třídy tools, která je v některých částech této třídy využívána. Tato třída slouží jako reprezentace pole, na kterém jsou “číslice”.

2.4.1 metody “add()” a “remove()”

Tyto metody slouží k odebírání a přidávání čísel do pole “arr” (viz. kapitola 2.4). Jak vyplývá z názvů metod metoda “add()” je využívána k přidávání a metoda “remove()” k odebírání. Odebírání funguje tak, že metoda dostane dvě čísla, které reprezentují místo, kde má být číslice odebrána a zde dosadí do pole 0. U přidávání metoda dostane opět souřadnice a k tomu

ještě číslo, které má přidat, a následovně číslo do pole dosadí. Obě tyto metody jsou využívány velice často naskrz mým programem.

2.4.2 metoda “printo()”

“printo()” je jedna z nejdůležitějších metod v celém programu. Zabývá se “vytisknutím” pole “arr” (viz. kapitola 2.4) do AnchorPanu (viz. kapitola 1.1), na kterém leží celé hrací pole. Zároveň také převádí pouhá čísla v “arr” (viz. kapitola 2.4) do obrázků a přidává samotné pozadí (mřížka ve které se pohybují “číslice”).

Metoda nejprve přidá na AnchorPane obrázek pozadí (*2048-Grid*, 2016). Následovně projde program každé jedny souřadnice pole “arr” (viz. kapitola 2.4), “přemění” je na obrázek a přidá je na AnchorPane. Na AnchorPane přidá program obrázek na správné místo pomocí metody “decode()” (viz. kapitola 2.3.).

obr. 11: Hrací pole

2.5 Třída “win_lost”

Tato třída má čtyři parametry, a to sice instanci třídy Stage se jménem “stage” a dvě instance třídy AnchorPane “won_pane” a “lost_pane”. Dva AnchorPany jsou ve dvou FXML souborech, a to sice v “Lost.fxml” (viz. kapitola 1.3.) a “Win.fxml” (viz. kapitola 1.2.). Třída obsahuje i dvě metody a to “re_lost()”, “re_won()”, “ex_lost()” a “ex_won()”.

2.5.1 metody “re_lost()” a “re_won()”

Metody “re_lost()” a “re_won()”̈jsou stejné až na jednu část, protože každá dělá to stejné akorát pro jiný FXML soubor (pro soubory “Win.fxml” (viz. kapitola 1.2.) a “Lost.fxml” (viz.

kapitola 1.3.)). Metody jsou také volány Buttony, které jsou obsáhnuty v FXML souborech. Jediné, co metody dělají je, že vytvoří novou instanci třídy “HelloApplication” (viz. kapitola 2.1), vytvoří novou stage a spustí metodu “start()” (viz. kapitola 2.1) s parametrem stage. Poslední věcí, co metoda udělá je, že uzavře aktuální stage.

2.5.2 metody “ex_lost()” a “ex_won()”

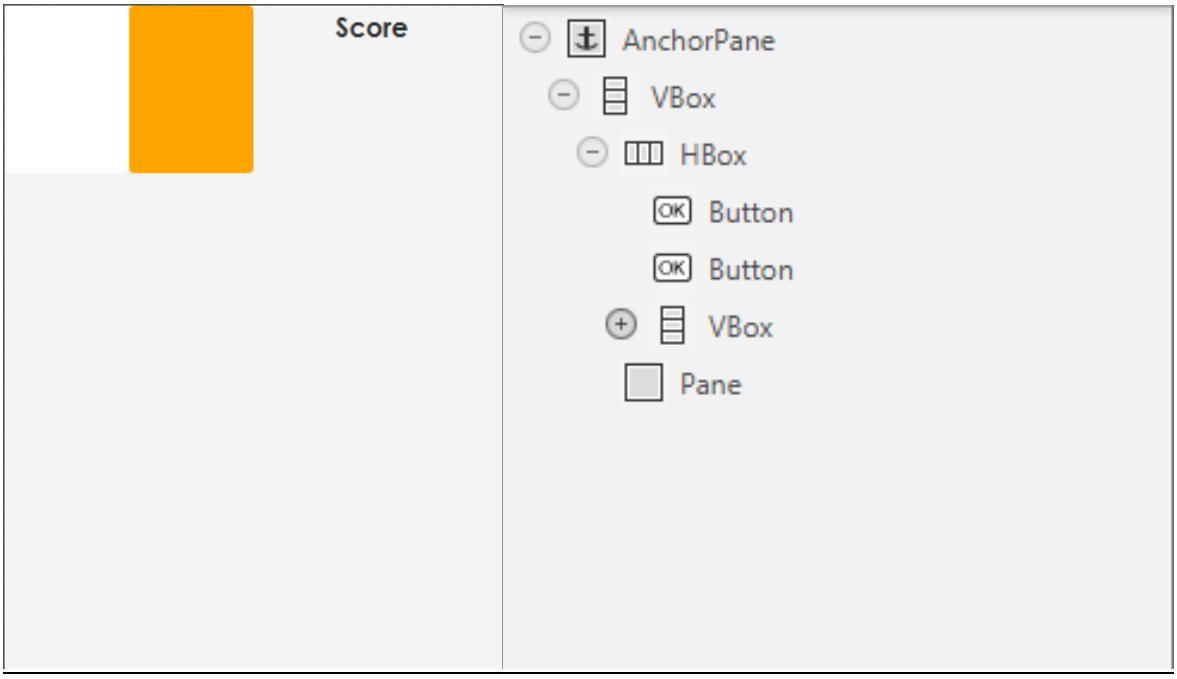
Tyto metody jsou jako metody “re_lost()” a “re_won()” hodně podobné až na to pro jaký soubor FXML fungují. Jsou volány Buttony umístěnými na FXML souborech “Win.fxml” (viz. kapitola 1.2.) a “Lost.fxml” (viz. kapitola 1.3.). Jediné, co tyto metody dělají je, že zavírají FXML soubor “Win.fxml” nebo “Lost.fxml”.

3 RECOURSES

3.1 “2048.fxml”

Tento FXML soubor obsahuje dva Buttony bez nápisů jeden bílý a jeden oranžový. Bílý button má označení “Undo” (slouží k použití kroku zpátky) a oranžový je označen “Restart” (slouží k restartování hry). Dva Labely jeden s nápisem “score” a jeden prázdný. Prázdný slouží pouze jako počítadlo a má označení “score”. V poslední řadě obsahuje FXML soubor Pane označený “pole” (slouží jako “hrací pole” ve hře)

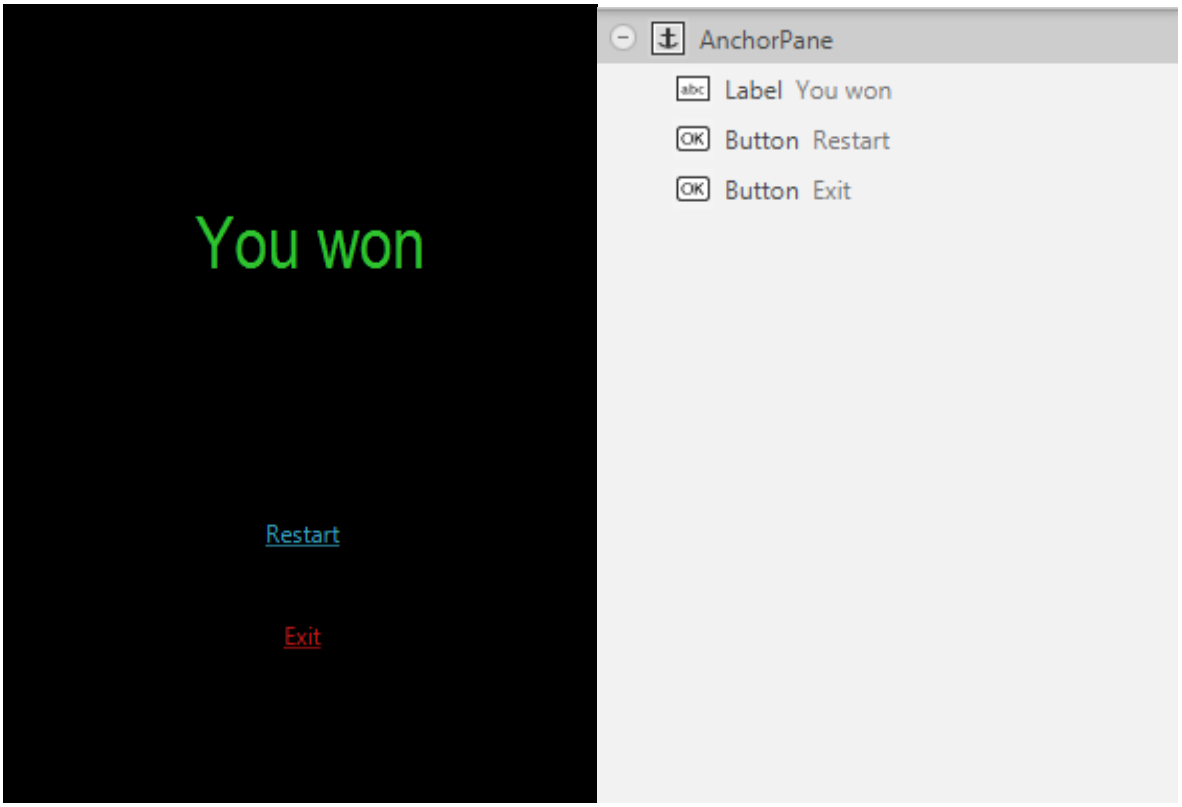
Všechny Buttony a Labely jsou v souboru rozmístěny v několika Hboxech a Vboxech tak aby utvářeli aplikaci.



obr. 12: Vzhled “2048.fxml” souboru a jeho struktura

3.2 “Win.fxml”

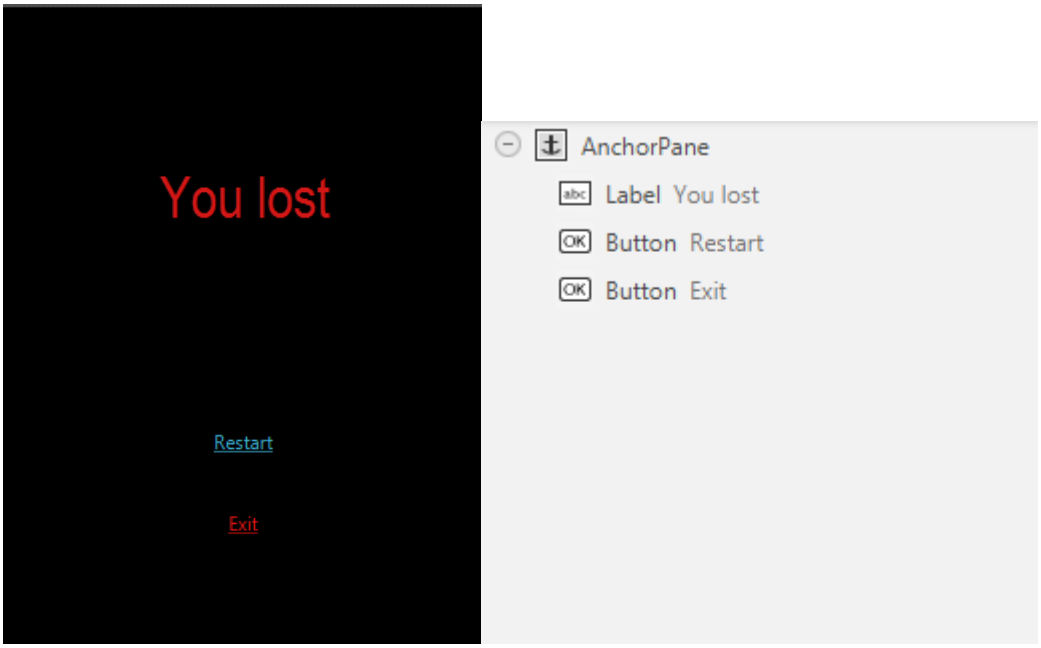
Tento FXML soubor je soubor, který se zobrazí, pokud vyhrajete hru (spojíte dvě “číslice” 8192). Tento soubor se skládá z jednoho AnchorPanu (označením “won_pane”), na kterém je všechno položeno. Konkrétně je na něm dvakrát Button jeden s nápisem Restart (modře obarvený) a jeden s nápisem Exit (červeně obarvený). Oba dva buttony jsou obarveny černě. Poslední částí je Label s nápisem “You won” a s textem obarveným na zeleno.



obr. 13: Vzhled “Win.fxml” souboru a jeho struktura

3.3 “Lost.fxml”

“Lost.fxml” je velice podobný souboru “Win.fxml” (viz. kapitola 3.2). Jediným rozdílem je Label. Na něm je místo “You won” napsáno “You lost” a je obarven červeně.



obr. 14: Vzhled “Lost.fxml” souboru a jeho struktura

3.4 “pane.dat” a “score.dat”

Tyto dva soubory jsou ve formátu “.dat” a slouží k uložení pole již hráč “uhrál” a aktuálního skóre. S těmito dvěma soubory pracují metody “initialize()” (viz. kapitola 2.2.1), “fileload()” (viz. kapitola 2.2.2) a “autosave()” (viz. kapitola 2.2.5). Jsou zakládány programem a program tudíž funguje i bez nich.

3.5 Obrázky

Obrázky ve funkčnosti tohoto programu nehrají důležitou roli, ale v grafické stránce ano. V celém programu je použito celkem 16 obrázků. 13 z těchto obrázků je použito jako “číslice” a zbylých 3 v různých částech programu.

3.5.1 “číslice”

Tyto obrázky jsem stáhl z internetu (nazywam, 2014) ve formátu “.svg”. JavaFX neumí zobrazovat soubory ve formátu “.svg”, a proto jsem všechny “číslice” předělal do formátu “.jpg”. Tohoto jsem docílil pomocí aplikace gimp.

3.5.2 ostatní obrázky

Ostatní obrázky v programu jsou:

- “2048-grid.png” (*2048-Grid,* 2016)
- “restart.png” (Google, 2010 - 2023)
- “undo.png” (Google, 2010 - 2023)

Obrázek “2048-grid.png” slouží jako pozadí, na kterém se hýbou “číslice” (viz. kapitola 3.5.1). Obrázek “2048-grid.png” jsem navíc musel oříznout pomocí programu gimp. “restart.png” a “undo.png” slouží zase jako značky vkládané do dvou Buttonů v FXML souboru “2048.fxml”.

4 JAK PROGRAM FUNGUJE?

Při zapnutí programu se uživateli otevře FXML soubor “2048.fxml” (viz. kapitola 3.1). V tento moment již může uživatel začít hru hrát pomocí kláves “W”, “S”, “A” a “D”. Pokud měl již uživatel rozehranou hru načte se mu samozřejmě ze souborů “pane.dat” (viz. kapitola 3.4) a “score.dat” (viz. kapitola 3.4).

V levém horním rohu FXML souboru se nachází dvě tlačítka. Tlačítko nalevo (tlačítko s bílým pozadím) slouží k použití funkce krok zpátky a tlačítko více vpravo slouží k restartování hry. Napravo od tlačítek je text obsahující aktuální skóre, které hráč má, a toto skóre se aktualizuje s každým tahem.

V případě že hráč spojí dvě číslice 8192 vyhraje a je otevřen FXML soubor “Win.fxml” (viz. kapitola 3.2). Na Této scéně je pouze nápis “You win” a dvě tlačítka. Jedno z tlačítek s nápisem “Exit” a jedno s nápisem “Restart”. Tyto tlačítka dělají přesně to, co je na nich napsané. Tlačítko “Exit” uzavře program a tlačítko “Restart” spustí celý program od začátku.

Pokud se stane že hráč zaplní celé pole “číslicem” a už není možné jimi pohnout program otevře FXML soubor “Lost.fxml” (viz. kapitola 3.3). Na této scéně jsou stejné tlačítka jako na scéně z “Win.fxml”. Jediným rozdílem je že na scéně je nápis “You lost”.

5 ZÁVĚR

Dle mého názoru jsem tuto ročníkovou práci zpracoval poměrně dobře a narozdíl od minulého roku dokonce celou podle zadání. Odnesl jsem si nové zkušenosti s implementací klávesnice do ovládání a další jiné zkušenosti s JavouFX. Bohužel jsem práci nemohl věnovat tolik času kolik jsem si představoval z důvodu psaní i druhé ročníkové práce z dějepisu, ale i přes toto jsem spokojen s finálním výsledkem.

Do budoucna bych možná hru trochu optimalizoval, zkrátil některé kusy kódu, zlepšil grafickou stránku jak scény na které se hraje a scén, které se zobrazí když vyhrajete nebo prohrajete. Dále bych přidal animace pohybu "číslic" (viz. kapitola 3.5.1) a možná i nějaké speciální výzvy pro ozvláštnění hry.

1. POUŽITÁ LITERATURA

Bro Code. (2021, March 15). *JavaFX KeyEvent* [video]. Youtube. Retrieved May 23, 2023, from https://www.youtube.com/watch?v=tq_0im9qc6E

Bro Code. (2021, March 21). *JavaFX animations* [video]. Youtube. Retrieved May 23, 2023, from https://www.youtube.com/watch?v=UdGiuDDi7Rg

Google. (2010 - 2023). *Left Arrow Key free icon* [Obrázek]. flaticon. Retrieved may 23, 2023, from https://www.flaticon.com/packs/material-design/4

Google. (2010 - 2023). *Undo Button free icon* [Obrázek]. flaticon. Retrieved May 23, 2023, from https://www.flaticon.com/free-icon/undo-button_60690

nazywam. (2014, November 15). *2048 tiles* [Obrázky]. OpenGameArt.Org. Retrieved May 23, 2023, from https://opengameart.org/content/2048-tiles

2048-grid [Obrázek]. (2016, March). smhow. Retrieved May 23, 2023, from https://www.smhow.com/wp-content/uploads/2016/03/2048-grid.png

2. SEZNAM OBRÁZKŮ