Gymnázium, Praha 6, Arabská 14
Obor programování

ROČNÍKOVÝ PROJEKT

Vladimír Samojlov, 2.E

Fotopuzzle

Duben 2023

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, Arabská14 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 ..... Vladimír Samojlov .....

Název práce: Fotopuzzle

Autoři: Vladimír Samojlov

Abstrakt: Tato práce se zabývá dvourozměrnou hrou, jejímž cílem je seskládat výsledný obrázek pomocí jednotlivých dílků, které do sebe zapadají pouze jediným správným způsobem. V nastavení aplikace má uživatel možnost zvolit fotografii a také obtížnost. Podle zvolené obtížnosti se vytvoří počet dílků z vybrané fotografie, které se poté zobrazí v aplikaci. Při zahájení hry bude hráči odstartován čas a odpočítáváno skóre, které se po skončení hry zobrazí uživateli pouze v případě správného řešení. Při složení fotografie se uživateli zobrazí výsledek hry s pomocí efektu.

Klíčová slova: Puzzle, Fotografie, JavaFX, CSS, Data, Drag and drop, Dílčí obrázky

Title: Fotopuzzle

Authors: Vladimír Samojlov

Abstract: This work deals with a two-dimensional game, the goal of this project is to assemble the resulting image using individual pieces that fit together in only one correct way. In the application settings, the user has the option to choose a photo and also the difficulty. Based on the selected difficulty, the number of pieces from the selected photo will be created, which will then be displayed in the application. When the game starts, the time will be started for the player and the score will be counted down. The results will be displayed to the user only if the solution is correct after the game is over. When the photo is composed, the user is shown the game result with the help of an effect.

Keywords: Puzzle, Picture, JavaFX, CSS, Data, Drag and drop, Sub-images

Obsah

1. Úvod

Tento dokument se zabývá hrou Fotopuzzle. Začátek práce pojednává o základních principech hry, které byly využívány během tvorby tohoto projektu. V průběhu projektu jsou uvedeny a popsány základní funkce a scény hry, jež program využívá.

Cílem projektu bylo vytvořit plně hratelnou hru, jejímž cílem je co nejrychleji správně vyřešit hlavolam vybraný uživatelem. Před zahájením hry má uživatel následující možnost si zvolit:

- obtížnost, jež určuje počet dílků.
- fotografii z nabídky nebo z vlastní složky počítače.

V průběhu hry může hráč také sledovat průběžný čas nebo využít nápovědu (zobrazení obrázku). Hratelnost hry není pro hráče nijak náročná, jelikož hra nabízí i stručně vysvětlený návod.

Zadání

Aplikace Fotopuzzle má mít níže uvedenou funkcionalitu:

1. Nastavení – výběr obtížnosti hlavolamu a fotografie z nabídky nebo z vlastního počítače.
2. Hrací scéna – sestavení celého obrázku s pomocí dílčích vygenerovaných částic.
    a) Spuštění času při dotyku libovolné částice.
    b) Nápověda – zobrazí výsledný obrázek na určitou dobu.
    c) Návod – stručný popis a ovládání hry.
3. Zobrazení výsledku hry – při správném složení hlavolamu bude zobrazeno skóre.

2. Herní scény a použité technologie

V této velké kapitole jsou popsány herní scény aplikace, které byly vytvořeny s pomocí technologií. V samotné aplikaci nyní zmíníme použité technologie. Následně se v následujících podkapitolách podrobněji zaměříme na jednotlivé scény aplikace, mezi kterými si zároveň vysvětlíme přechody.

2.1 Technologie

2.1.1 Grafické uživatelské rozhraní (GUI)

Pro vytváření grafického uživatelského rozhraní byl zvolen framework *JavaFX*, který je zároveň součástí *open source*, tedy systému s volně přístupným zdrojovým kódem. [1] Tato platforma v dnešní době nabízí spoustu možností, jak vytvářet okenní aplikace. Poskytuje tak podporu pro obrázky, videa, hudbu, grafy, kaskádové styly (*CSS*) a jiné technologie. [2] Tento framework lze také využít i pro vývoj mobilních i webových aplikací.

2.1.2 Vzhled a grafická podoba

V souladu aplikační platformy *JavaFX* byl využit již zmíněný stylový jazyk *CSS*. Přestože kaskádové styly byly primárně navrženy pro úpravu grafického vzhledu webových stránek, je možné je využít i v rámci *JavaFX* aplikací. Rozšíření jazyka *CSS* s *JavaFX* bylo však upraveno tak, aby všechny názvy vlastností obsahovaly předponu "*-fx-*". Standardní kombinace jazyků *HTML* a *CSS* tuto předponu neobsahuje, jelikož aplikace *JavaFX* používá vlastní sémantiku hodnot. [3] Na obrázku 2.1 je uvedena ukázka syntaxe stylového jazyka *CSS* při použití s jazykem *JavaFX*.

S pomocí *CSS* lze v okenní aplikaci přidávat motivy k jednotlivým ovládacím prvkům *JavaFX* (například k *Button, Label*, atd.). Typickým příkladem motivu je zbarvení textu, změna fontu a velikosti písma nebo také použití obecných pseudotříd (nejznámější – *:hover*, *:focus*).

```
.label {
    -fx-text-fill: linear-gradient(to right, #b1227a 0%, #429aee 100%);
    -fx-font-size: 80;
    -fx-background-color: #000000;
    -fx-font-weight: bold;
    -fx-font-style: italic;
}
```

Výpis kódu 2.1: Ukázka syntaxe kaskádových stylů s frameworkem *JavaFX*.

2.1.3 Ukládání dat

Jelikož autor neměl zkušenosti s ukládáním dat do databáze, rozhodl se využít tzv. *flat files* neboli textové soubory s plochou strukturou (s příponou *.txt*). V tomto typu souboru se data jednoduše ukládají na jednotlivých řádcích. [4] V případě oddělení řádků textu je nutno v platformě *JavaFX* použít pouze jednořádkový příkaz – "\n". Celý jednoduchý program ukládání dat s oddělením textu do textového souboru *.txt* je zobrazen v jazyce *Java* na výpisu kódu 2.2. V informatice však existuje i několik dalších způsobů, jak si uživatel může data uložit. Každý způsob pro ukládání dat se liší svým zápisem kódu.

```
try {
    FileWriter napisDoSouboru = new FileWriter("nazevSouboru.txt");
    napisDoSouboru.write("Oddělení textu " + "\n");
    napisDoSouboru.close();
    System.out.println("Věta byla úspěšně napsaná do textového souboru.");

} catch (IOException e) {

    System.out.println("Došlo k chybě.");
    e.printStackTrace();
}
```

Výpis kódu 2.2: Program v jazyce *Java* ukládající slova do textového souboru *.txt*

2.2 Scény aplikace

Tato podkapitola popisuje jednotlivé druhy scén vytvořené aplikace (úvodní, nastavení, herní). Ke každému typu scény jsou popsány a vysvětleny dílčí použité prvky v *JavaFX*. Na konci této kapitoly jsou ukázány přechody mezi scény.

2.2.1 Úvodní scéna

Při spuštění aplikace se zobrazí úvodní či startovní scéna. Tento typ scény má velice důležitou roli pro uživatele, jelikož mu poskytuje informace a možnosti dalšího výběru (přesunutí do ostatních scén hry). Na obrázku 2.3 je vyobrazena hlavní scéna zahrnující dva hlavní komponenty *Button* – tlačítka „*Play*“ a „*Settings*“, ale také i některé další významné prvky. S pomocí těchto tlačítek má hráč této hry na výběr přejít přímo do hracího režimu nebo režimu nastavení. S využitím tlačítek budou v podkapitole 2.2.4 prováděny již zmíněné přechody mezi scény.

Obrázek 2.3: Úvodní herní scéna aplikace.

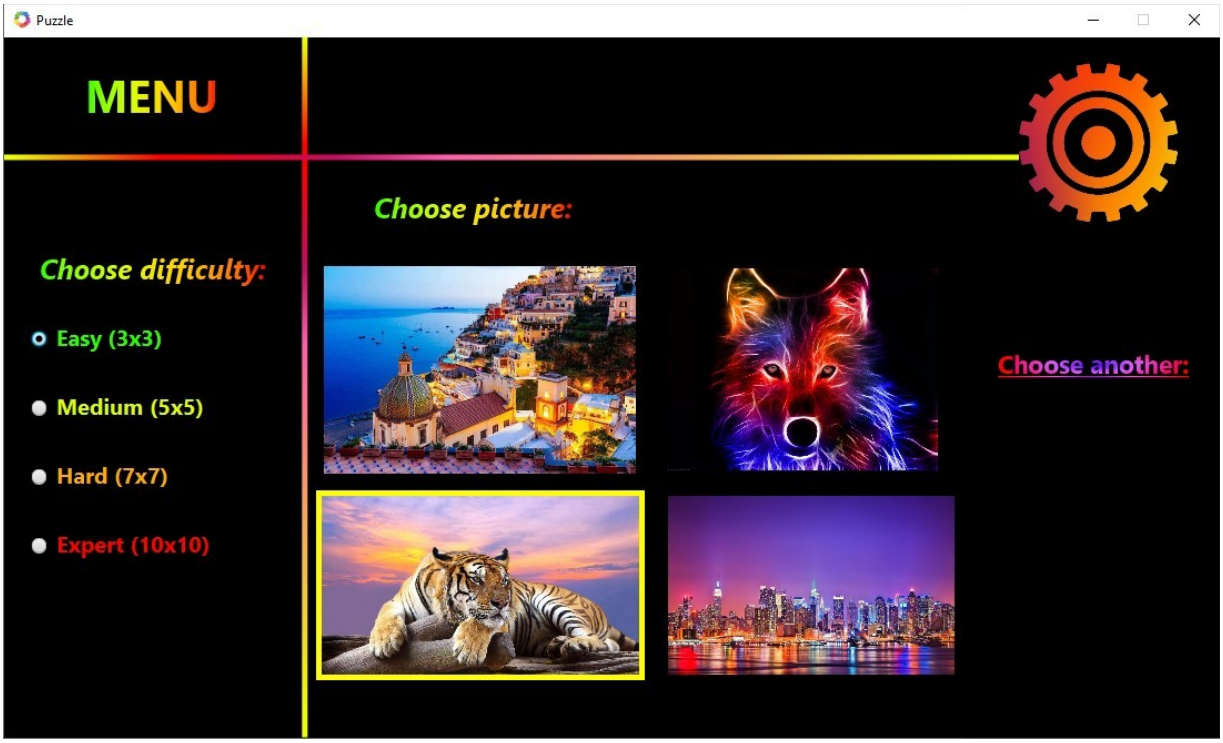
Jedním z dalších podstatných elementů je ovládací prvek *ImageView*. Společnost Oracle Corporation popisuje tento element takto: „*The ImageView is a Node used for painting images loaded with Image class*.“ [5] Lehčeji řečeno, jedná se o ovládající prvek, jenž má „schopnost“ zobrazit obrázek načtený s pomocí třídy *Image*. Prvek *ImageView* v tomto případě prezentuje tvary dílků puzzle.

Nadpis celé úvodní scény tvoří prvek *Label* – s názvem „*Puzzle*“. Tento prvek se zejména používá pro zobrazení textu, který po spuštění celé aplikace nelze upravovat. V souvislosti s kaskádovými styly byl pro barvu písma v *Label* použit barevný přechod *linear-gradient* (viz. výpis kódu 2.1). Kaskádové styly byly rovněž použity pro komponentu *Button* v rámci pseudotřídy *:hover*, která zobrazuje stav, kdy se kurzor myši nachází nad tlačítkem. Tato pseudotřída byla aplikována tak, aby se navzájem vyměnily barvy pozadí a písma.

2.2.2 Nastavení

Scéna nastavení primárně slouží k přizpůsobení parametrů hry dle preferencí uživatele. Mezi konfigurace této hry patří zvolení obtížnosti (*Easy*, *Medium*, *Hard*, *Expert*) a obrázku (z nabídky nebo vlastního z počítače). Pokud se však uživatel přímo rozhodne hru spustit bez nastavování, aplikace automaticky spustí úroveň *Easy* a základní obrázek *Tygr*. Jednotlivé obtížnosti se liší počtem vygenerovaných dílků z vybraného obrázku. Možnost při výběru vlastní fotografie je omezena na soubory s příponami *.bmp*, *.png* nebo *.jpg*. Po nahrání obrázku se uživateli pod nápisem „*Choose another:*“ zobrazí vybraný obrázek a následně se přehraje krátká animace potvrzující nahrání obrázku z počítače (viz obr. 2.5).

Oproti úvodní scéně zmíněné v podkapitole 2.2.1, byl ve scéně nastavení aplikován element *ToggleButton*. Jedná se zpravidla o tlačítko, s jehož pomocí lze nastavit aktuální stav na zapnutý nebo vypnutý. Pro zvětšení výběru počtu možností je zapotřebí vytvořit tzv. *ToggleGroup*. Tato skupina komponentů *ToggleButton* zajišťuje, aby v rámci celé skupiny byl zapnutý pouze jeden prvek. Příklad užití *ToggleGroup* pro výběr obtížností je uveden na obrázku 2.4, ve kterém je znázorněna aktivní zvolená možnost *Easy*. Pro ukončení voleb v nastavení je třeba stisknout tlačítko "*MENU*". V tento okamžik se uloží vybrané možnosti do textových souborů a zároveň se uživateli zobrazí počáteční obrazovka hry.



Obrázek 2.4: Nastavení hry podle vybraných uživatelských preferencí.

Obrázek 2.5: Animace potvrzující nahrání obrázku z počítače.

2.2.3 Herní prostředí

Jako hlavní scénu aplikace lze označit celkové prostředí hry. Celé hlavní prostředí aplikace se skládá z herní plochy a navigačního panelu. Tyto dvě části byly rozdílně barevně odlišeny pro zvýraznění navigačního panelu.

V rámci herní plochy jsou náhodně rozmístěny vygenerované obrázky z předem vybrané fotografie. Počet obrázků se tak může lišit v závislosti na vybrané obtížnosti. Jednotlivé částice jsou na ploše rozdělené do dvou skupin a lze je libovolně přesouvat jak na, tak v rámci hrací desky. Jedná se o tzv. *Drag and drop* funkci, která se využívá pro přetahování prvků s pomocí myši. Během provádění operace této funkce je vidět průsvitná reprezentace přetahovatelných prvků. [6] Cílem samotné hry je poskládat destičky puzzle správně do prázdných míst hrací desky. Do každého z míst je povoleno vkládat pouze jednu částici puzzle. V případě, kdy je celá hrací deska zaplněná, je uživateli ukázán výsledek hry – výhra nebo prohra. Pro hrací desku byl využit komponent rozvržení *GridPane*, jenž v této situaci představuje prázdně mřížky a sloupce, do kterých lze vkládat jednotlivé částice puzzle. Součástí plochy je i stručné vysvětlení hry skryté pod znaménkem otazníku. Tato funkce se automaticky aplikuje při již zmíněné pseudotřídě *:hover* (vysvětlené v podkapitole 2.2.2).

Druhou důležitou částí prostředí je navigační panel, jehož součástí jsou další významné objekty: stopky (průběžný čas skládání), nápověda (zobrazení výsledné fotografie) a tlačítko (návrat do úvodní scény). Při správném řešení se uživateli zobrazí výsledná tabulka, ve které je ukázán finální čas, skóre a další jiné informace týkající se hry. Hráč hry má také povoleno po skončení jedné hry zahájit další. Na obrázku 2.8 je ukázán samotný začátek hry v aplikaci.

V herním prostředí hraje velkou roli již uvedené vytváření vedlejších obrázků z hlavní fotografie, které se při spuštění herní scény zobrazí. Vzápětí bude vysvětlen hlavní algoritmus pro generování dílčích obrázků z fotografie a následně pro jejich zobrazení. Algoritmy jsou znázorněny a vysvětleny na obrázcích 2.6 a 2.7.

```
// Počet fotek dle obtížnosti
BufferedImage images[] = new BufferedImage[PocetFotek];

// Počáteční fotografie vybraná uživatelem
fotografie = ImageIO.read(getClass().getResourceAsStream(celeJmenoFotografie));

sirka = fotografie.getWidth() / sloupec;
vyska = fotografie.getHeight() / radek;

int VybranyObrazek = 0;

// Algoritmus pro generování obrázků z fotografie
for (int radky = 0; radky < radek; radky++) {

    for (int sloupce = 0; sloupce < sloupec; sloupce++) {

        images[VybranyObrazek] = (new BufferedImage(sirka, vyska, fotografie.getType()));

        Graphics2D grafika2D = images[VybranyObrazek++].createGraphics();

        int vybranaFotkaX = sirka * sloupce;
        int vybranaFotkaY = vyska * radky;

        int PuzzlePieceX = sirka * sloupce + sirka;
        int PuzzlePieceY = vyska * radky + vyska;

        // Vykreslení jednotlivých obrázků (částic puzzle)
        grafika2D.drawImage(fotografie,  dx1: 0,  dy1: 0, sirka, vyska, vybranaFotkaX, vybranaFotkaY,
                PuzzlePieceX, PuzzlePieceY,  observer: null);
        grafika2D.dispose();

    }
}
```

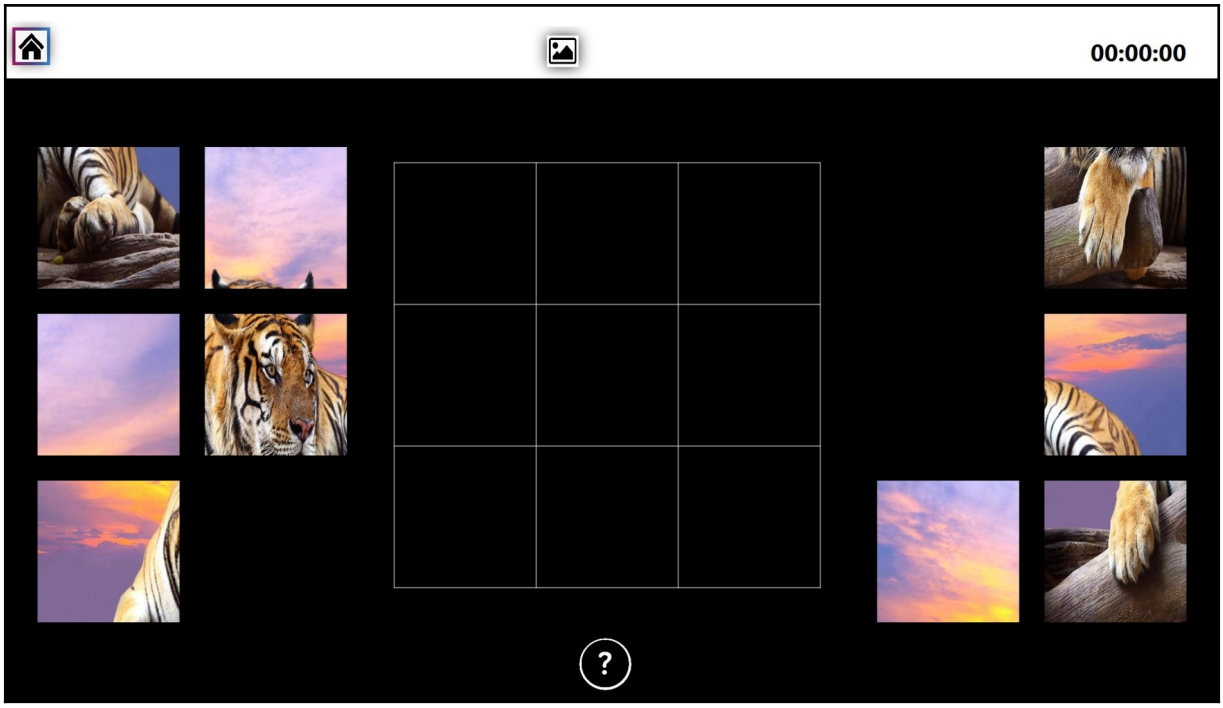
Obrázek 2.6: Algoritmus pro generování částic puzzle v herním prostředí.

Pro generování částic puzzle byla použita API *Java AWT* z důvodu speciálních komponentů, s jejichž pomocí lze vykreslit obrázky v jazyce *Java*. Částice se vytvářejí díky třídě *Graphics2D*. Každý objekt *Graphics2D* je propojen s určeným místem, které určuje, kde bude vykreslování probíhat. Zároveň u tohoto objektu je možné definovat vlastnosti vykreslování místa, jako například formát pixelů a rozlišení. [7] Zde je objekt *Graphics2D* využit pro generování obrázků z vybrané fotografie se stejnými rozměry – šířka a výška. Celkový počet částic se určí podle zvolené úrovně uživatelem (např. úroveň *Easy* – 9 částic puzzle). Ve výsledku tento objekt vyobrazí obrázky s pomocí metody *.drawImage*.

```
// Zobrazení jednotlivých obrázků z vybrané fotografie
for (int m = 0; m < PocetFotek; m++) {

    ImageView[] imageView = new ImageView[PocetFotek];
    imageView[m] = new ImageView();
    Fotky = imageView[m];
    Fotky.setImage(fotky.get(m).image);
    Fotky.setFitWidth(Piece);
    Fotky.setFitHeight(Piece);
    Fotky.setCursor(Cursor.cursor( identifier: "OPEN_HAND"));

    // Získávání souřadnic pro jednotlivé částice do rozvržení GridPane
    int x = m / sloupec;
    int y = m % radek;

    // Funkce pro přetahování částic puzzle
    setOnDragDetected(Fotky);
    setOnDragDone(Fotky);

    // Rozdělení částic do dvou skupin
    if (m <= halfGrid) {

        grid1.add(Fotky, x, y);

    } else {

        grid2.add(Fotky, x, y);
    }
}
```

Obrázek 2.7: Algoritmus pro zobrazení částic puzzle v herním prostředí.

Výše uvedený program s pomocí *for* cyklu postupně prochází celým polem – *ArrayList*. Pro jednotlivé hodnoty *m* se tak vytvoří nový prvek *ImageView*, jenž zároveň nastaví příslušnou část obrázku. Dále se pro element *ImageView* nastaví stejná výška a šířka, ale také se změní kurzor myši. Pro přetahování tohoto typu elementu byly vytvořeny metody, ve kterých je nezbytné uplatnit některé podstatné funkce pro užití *Drag and drop*. Na konci celého programu probíhá rozdělení prvků do dvou skupin. Každá skupina je uspořádána do rozvržení *GridPane* – počet sloupců a řádků se liší v závislosti na vybrané obtížnosti. Obě dvě skupiny obsahují vygenerované částice puzzle, které lze vidět spolu s hrací plochou na obrázku 2.8.



Obrázek 2.8: Herní prostředí před zahájením hry.

Jedním ze způsobů, jak aplikovat celou operaci *Drag and drop* je implementování metod pro jednotlivé typy událostí *DragEvent*, které se od sebe navzájem liší svojí funkcionalitou. Pro lepší názornost jsou nyní u každé události stručně vysvětleny její funkce pomocí bodového seznamu:

- *setOnDragDetected* – Začátek přetahování prvku.
- *setOnDragOver* – Přetahování prvku nad platný cíl přetažení po dobu několik set *ms*.
- *setOnDragEntered* – Přetažený prvek vstoupí do platného cíle přetažení.
- *setOnDragExited* – Ukončení operace přetažení prvku.
- *setOnDragDone* – Přetažený prvek zanechá platný cíl přetažení.
- *setOnDragDropped* – Prvek je úspěšně vypuštěn na platný cíl upuštění. [8]

S pomocí těchto událostí lze ve hře přetahovat jednotlivé obrázky do komponenty *GridPane*.

Při operaci *Drag and drop* může nastat výjimečná situace, kdy uživatel vloží obrázek do jednoho z volných míst v hrací ploše a zároveň událost *setOnDragOver* umožní uživateli umístit další částice na již obsazené místo. Pro vyřešení tohoto problému je potřeba během události *setOnDragOver* zkontrolovat, zda *GridPane* již na daném prostoru obsahuje nějaký element. V obrázku 2.9 je znázorněn program, který během události *setOnDragOver* pomocí funkce *consume* nastavuje zamezení skládání obrázků na sebe vrstvami pro všechny prvky na hrací ploše – *dragboard*. Pro úplné zabránění překrývání obrázků je však stále nutno použít funkce *consume* a *setDropCompleted* i pro událost *setOnDragDropped*. Pokud v funkci *setDropCompleted* nastavíme výraz *boolean* na hodnotu *false*, bude akce při upuštění prvku do hrací plochy nedokončena a prvek se tak vrátí na svou původní pozici ve hře.

```
for (Node node : dragboard.getChildren()) {

    node.setOnDragOver(onDragOver -> {
        onDragOver.consume();
    });

    node.setOnDragDropped(onDragDropped -> {

        onDragDropped.consume();
        onDragDropped.setDropCompleted(false);
    });
}
```

Obrázek 2.9: Omezení pokládat obrázky na sebe během celé hry.

Pohyb částic puzzle uvnitř hrací plochy má v rámci hry další významnou funkci. Pro přemisťování částic v *dragboard* stačí pouze v době upuštění (*drop*) zavolat metodu znovu. Tedy pro opětovné přetáhnutí prvků je důležité do metody *setOnDragDropped* zavolat dvě podstatné metody *setOnDragDetected()* a *setOnDragDone()*. Uživatel má tak další možnosti jak postupně upravovat puzzle v *dragboard*. Pokud však uživatel celý prostor hrací plochy naplní, jedná se automaticky o výsledek hry – výhra nebo prohra, který bude podrobněji popsán v další kapitole 3.

2.2.4 Přechody mezi scénami

Přechody z počáteční scény (vytvořené s pomocí *JavaFX FXML*) do dalších scén se nastavují přímo ve třídě, která slouží jako kontroler pro daný soubor *FXML* (obsahující atribut *onAction*). Pro propojení tohoto kontroleru se souborem *FXML* je nezbytné pro každé tlačítko v úvodní scéně („*Play*“ a „*Settings*“) definovat metodu s anotací *FXML* typu *void*, jež nastaví událost *ActionEvent* jako argument. Dále musí být název metody ve třídě stejný jako pojmenování atributu v souboru *FXML*. Posledním krokem je zapsání programu do jednotlivých metod, které nahradí aktuální scénu v aplikaci. Nicméně program pro přechody mezi scénami se liší v závislosti na cílové scéně. Autor v tomto případě využil pouze framework *JavaFX* pro přechod do scény s herním prostředím, nikoliv *FXML*. Zatímco pro přechod do scény nastavení byla využita kombinace *JavaFX FXML*.

```
@FXML
public void Play(ActionEvent event) throws Exception {

    MainFX newScene = new MainFX();
    root = newScene.getRoot();

    scene = new Scene(root);

    scene.getStylesheets().add(getClass().getResource( name: "/css/MainScene.css").toExternalForm());

    stage = (Stage) (((Node) event.getSource()).getScene().getWindow());
    stage.setScene(scene);
    stage.setFullScreenExitHint("");
    stage.setFullScreen(true);
    stage.show();
}
```

Obrázek 2.10: Metoda umožňující přechod z počáteční scény do scény herního prostředí.

Na začátku metody zobrazené na obrázku 2.10 je vytvořena instance třídy *MainFX*, která má funkci nové scény. V této třídě je zároveň aplikována metoda *getRoot*, která vrací uzel (*Node)* typu *Parent*. Do proměnné *root* se tedy uloží uzel třídy *MainFX*, který je typu *Parent*. Následně je proměnná *root* použita ve scéně, jež je součástí kontejneru *stage*. Kontejner *stage* tak vyobrazí výslednou scénu na celý monitor počítače (*full-screen)*, přičemž původní scéna je odebrána. Do scény byl také importován soubor se zkratkou *.css* – kaskádové styly.

Program umožňující přechod do scény nastavení je podobný předchozímu kódu na obrázku 2.10, jelikož se liší však pouze na úplném začátku, kdy se do proměnné *root* ukládá uzel scény, který byl načten z *FXML* souboru *SettingsMenu.fxml*. Na obrázku 2.11 je ukázán celý program pro změnu scény po kliknutí na tlačítko *Settings*.

```
@FXML
public void settingsButtonClicked(ActionEvent event) throws Exception {

    Parent root = FXMLLoader.load(getClass().getResource( name: "/fxml/SettingsMenu.fxml"));
    Scene scene = new Scene(root);

    Stage stage = (Stage) (((Node) event.getSource()).getScene().getWindow());

    stage.setScene(scene);
    stage.show();
}
```

Obrázek 2.11: Metoda umožňující přechod z počáteční scény do scény nastavení.

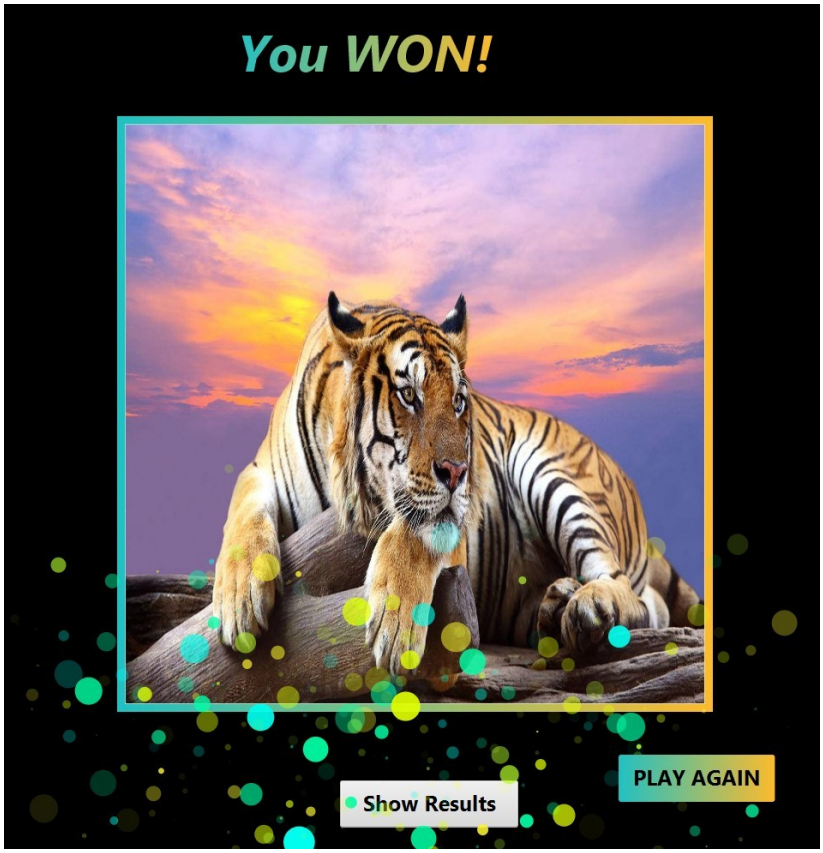
Oba zmíněné přechody mezi scénami usnadňují uživateli lépe pochopit strukturu celé aplikace, ale zároveň také nabízejí úplný přehled o aplikaci.

3. Výsledek hry

V této kapitole je popsán stav, kdy je hrací plocha zaplněná obrázky. V tomto momentě je uživateli ukázán výsledek hry – výhra nebo prohra. Při výsledku hry se zobrazí animace s barvami charakteristickými pro daný výsledek. Po skončení každé hry je možno si hru zopakovat kliknutím na tlačítko „*PLAY AGAIN*“.

3.1 Výhra

Při správném složení fotografie se uživateli v hracím prostředí objeví název „*You WIN*“. U celé složené fotografie se mění barva okrajů. V případě výhry má uživatel možnost si zobrazit informace o průběhu hry – název fotografie, obtížnost, čas, počet použití funkce nápovědy, finální skóre. Všechny tyto informace jsou uspořádány do jedné tabulky (viz obrázek 3.2). Na obrázku 3.1 je zobrazena správná kompozice fotografie *Tygr* v herním prostředí.



Obrázek 3.1: Správné poskládání fotografie *Tygr*.

Obrázek	Level	Čas	Použití nápovědy	Skóre
Tygr	Easy	00:00:22	3	9975
Město č.2	Medium	00:04:08	22	9426
Liška	Easy	00:00:29	3	9940

Obrázek 3.2: Informace o průběhu hry se správným řešením.

3.2 Prohra

Při nesprávném řešení se uživateli v hracím prostředí zobrazí text „*You LOST*“. V případě prohry nejsou uživateli poskytnuty informace o průběhu hry, ale animace stále probíhá. Na obrázku 3.3 je zobrazena nesprávné řešení fotografie *Tygr* v herním prostředí.

Obrázek 3.3: Nesprávné poskládání fotografie *Tygr*.

Závěr

Výsledná aplikace byla úspěšně dokončena a hra funguje bez problémů. Vytváření celého programu se obešlo bez velkých obtíží. Při průběhu projektu bylo ve vývojovém prostředí uplatněno spousta způsobů, podle kterých byla hra vylepšena. Tyto úpravy zlepšily nejen vzhled, ale také funkčnost samotné hry. Během tvorby tohoto projektu si autor výrazně zlepšil své dovednosti a získal tak spoustu nových poznatků ve frameworku *JavaFX*. Projekt by šlo rozšířit o několik dalších funkcí ve hře, mezi které patří například vytvoření útvaru částice puzzle.

Splnění zadání

Výsledná aplikace splňuje zadání, které bylo uvedeno v úvodu. Při projektu bylo nejtěžší zkontrolovat, zda vygenerované částice puzzle jsou správně umístěny na jednotlivých pozicích v hrací ploše. Během řešení této úlohy autor narazil na několik dalších problémů, které byly následně opraveny a výsledná aplikace se tak vyvinula nad očekávání autora.

Bibliografie

[1] *Open Source* https://www.mioweb.cz/slovnicek/open-source/ [cit. 2023-04-10].

[2] ŠTECHMÜLLER, Petr. *JavaFX* https://www.itnetwork.cz/java/javafx/uvod-do-javafx [cit. 2023-04-10].

[3] Oracle Corporation. *JavaFX CSS Reference Guide* https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html [cit. 2023-04-10].

[4] ČÁPKA, David. *Práce s textovými soubory* https://www.itnetwork.cz/csharp/soubory-a-sit/c-sharp-tutorial-prace-se-soubory-txt [cit. 2023-04-10].

[5] Oracle Corporation. *Class ImageView* https://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/ImageView.html [cit. 2023-04-10].

[6] *HTML Drag and Drop API* https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API [cit. 2023-04-18].

[7] Oracle Corporation. *Class Graphics2D* https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html [cit. 2023-04-20].

[8] *Drag Events* https://developer.mozilla.org/en-US/docs/Web/API/DragEvent [cit. 2023-04-20].

Seznam obrázků