Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníkový projekt

Předmět: Programování

Téma: Interaktivní atlas sluneční soustavy

Autor: Sabina Javůrková

Třída: 2.E

Školní rok: 2022/2023

Vyučující: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Čestné 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 veřejnost (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 27.4. 2023 .....

Sabina Javůrková

Anotace:

Jako téma mé ročníkové práce z předmětu programování jsem si vybrala interaktivní atlas sluneční soustavy. Kód zahrnuje vytvoření jednotlivých vesmírných těles formou objektů a jejich animace. V mém programu si uživatel může procházet 3D sluneční soustavu za pomocí kurzoru myši a zároveň se o ní něco přiučit.

Abstract:

As the topic for my programming project i chose an interactive atlas of the solar system. The code includes the creation of individual space bodies in the form of an object and their animation. In my program, the user can browse the 3D solar system with the help of a mouse cursor and at the same time learn something about it.

Zadání ročníkového projektu

Vytvořit program, který umožní uživateli procházet a komunikovat se 3D sluneční soustavou.

Upřesnění

Při spuštění programu se zobrazí 8 planet sluneční soustavy a Slunce ve správných velikostních a rotačně rychlostních poměrech. Při přejetí kurzoru myši na jednotlivou planetu se rozsvítí modré označení a její anglický název. Po kliknutí na ni se přiblíží a vedle ní se zobrazí box s informacemi o zvolené planetě.

V pravém horním rohu okna aplikace je po celou dobu běhu programu viditelné tlačítko s názvem „Info“, na které když uživatel klikne, zobrazí okno s obecnými informacemi o sluneční soustavě. V pravém dolním rohu svítí tlačítko s názvem „Exit“, které zavře aplikaci po kliknutí.

Platformy

- Intellij IDEA Community
- Java
- JavaFX

Obsah

1. Úvod 1

2. Popis kódu 2

2.1. Třída Sphere 2

2.2. Třída Planet 2

2.2.1. Metoda rotate() 3

2.3. Třída Main 3

2.3.1. Metoda start() 3

2.3.2. Metoda bigBang() 5

2.3.3. Metoda moon() 5

2.3.4. Metoda sunglight() 5

2.3.5. Metoda background() 5

2.3.6. Metoda highlights() a titles() 6

2.3.7. Metoda saturnRings() 6

2.3.8. Metoda buttons() 7

2.3.9. Metoda animation() 8

2.3.10. Metoda planetInteractions() 8

2.3.11. Metoda action() 9

2.3.12. Metoda prepareInfoBox() 11

2.3.13. Metoda planetsInfoBox() 12

2.3.14. Metoda initMouseControl() 12

3. Nezávislé komponenty programu 13

4. Použité nástroje 14

5. Komplikace při psaní programu 15

6. Potenciální vylepšení do budoucna 16

7. Závěr 17

Reference 18

Seznam použitých webových zdrojů 19

Seznam obrázků 20

1. Úvod

V tomto dokumentu budu především rozebírat a vysvětlovat kód, který jsem sepsala v rámci ročníkové práce z předmětu programovaní. Zároveň budu popisovat průběžný proces jeho psaní jako jsou obtíže, které při sestavování nastaly, nebo potenciální zlepšení jeho efektivnosti a vzhledu v budoucnosti.

Pro svoji ročníkovou práci jsem se rozhodla zvolit téma interaktivní atlas sluneční soustavy. Jeden z důvodů, proč jsem si vybrala zpracovat tento námět, je můj osobního zájem o sluneční soustavu a obecně o vesmír, přijde mi fascinující a ráda využiji každou možnou příležitost se o něm přiučit něco nového. U tohoto tématu si můžu vyhrát s grafickou stránkou aplikace a vyladit spoustu detailů, což mi, jako kreativnímu člověku, přijde velmi sympatické.

Po spuštění aplikace se v okně zobrazí 10 sfér, které představují 8 planet sluneční soustavy, náš měsíc a Slunce. Při přejetí kurzoru myši na jednotlivé sféry se kolem nich rozsvítí barevné označení společně s jejich názvem. Kliknutím na jednu zvolenou planetu nebo Slunce se objekt přiblíží na levou stranu okna a vedle něj se objeví box s informacemi o určitém vesmírném tělesu. Pro vrácení sluneční soustavy do původního stavu stačí znovu pokliknout na zvolenou sféru, a ta se přiblíží zpátky na své místo do řady a informační okno mizí. V pravém horním rohu po celý běh programu svítí tlačítko s nápisem „Info“, které po kliknutí změní nápis na „Back“ a zobrazí box s obecnými informacemi o sluneční soustavě jako celku. V pravém dolním rohu se nachází tlačítko s textem „Exit“, kterým může uživatel aplikaci zavřít. Pozadí vyplňuje fotografie hvězd a Mléčné dráhy. Se scénou aplikace může uživatel po celý běh programu komunikovat za pomocí tlačítek myši, tak, že ji může posouvat ze strany na stranu kurzorem nebo oddalovat či přiblizovat kolečkem myši.

Program jsem vytvářela ve vývojovém prostředí IntelliJ IDEA Community Edition a je sepsán pomocí jazyku Java a JavaFX. Pro běh aplikace je nutné stáhnout programovací knihovnu JavaFX.

2. Popis kódu

V této kapitole rozeberu kód, díky jemuž můj program funguje tak, jak má. Je rozdělen do dvou tříd Planet a Main, přes které postupně vytvořím veškerá vesmírná tělesa, tlačítka a informační boxy viditelné po spuštění aplikace. Program má celkem 867 řádků, z čehož 640 je čistý kód – tedy bez JavaDoc komentářů a prázdných řádků. Jeho sepsání mi trvalo kolem 30- ti hodin.

2.1. Třída Sphere

Třída Sphere je v JavaFX 3D tvar, který představuje dokonalou kouli. Je součástí balíčku javafx.scene.shape, tím pádem od třídy Shape dědí několik vlastností jako radius (poloměr koule) a material (textura planet). Kromě nastavení pozice sféry a materiálu ji můžu také přidat do grafu scény pomocí metody getChildren().add() objektu Group nebo Pane. *(1)* Třída Planet, která reprezentuje Měsíc, Slunce a všechny planety vyobrazené na scéně aplikace, je potomek třídy Sphere, tedy dědí její vlastnosti.

2.2. Třída Planet

Třída Planet ztělesňuje všechny kulaté objekty mého programu, tedy Slunce, Měsíc a 8 planet sluneční soustavy. Je potomkem třídy Sphere, což znamená, že dědí její vlastnosti, a tím dělá můj kód efektivnějším. Díky této třídě je vytvoření jednotlivých již výše zmíněných objektů rychlejší a méně chaotické, než kdyby byly tvořeny jako objekty typu Sphere.

Jako první ve třídě Planet vytvořím proměnné, které představují název, poloměr, rotační rychlost a směr, a cestu k obrázku s texturou vesmírného objektu. Poté vytvořím konstruktor pro dané proměnné a definuji objekt.

2.2.1. Metoda rotate()

Jedinou metodou ve třídě Planet je metoda rotate(). Jedná se o metodu typu public void, tedy je k ní přístup i z ostatních tříd a nevrací žádnou proměnnou, objekt, nebo pole s nimi. Tato metoda se stará o to, aby vesmírné těleso rotovalo kolem osy Y a svojí správnou rychlostí i směrem. Tato metoda je později použita ve třídě Main jako součást animace, tedy je konstantně volána a objekt typu Planet se točí nepřetržitě.

2.3. Třída Main

Třída Main představuje hlavní část mého kódu, je nejrozsáhlejší, má nejvíce metod, a samotný program také spouští. Je potomkem třídy Application, což znamená, že obsahuje metodu start(), která je nutná pro každou JavaFX aplikaci. *(2)* V této třídě se všechny vesmírné i nevesmírné objekty, jako box s informacemi a tlačítka, vytvářejí za pomocí mnou sepsaných 14- ti metod a jsou poté vyobrazeny v okně při běhu aplikace.

Jako první si ve třídě vytvořím velké množství proměnných a některé i deklaruji. Jsou to proměnné, ke kterým potřebuji přístup z několika metod, nebo by se jejich definování uvnitř metody opakovalo stále dokola se stejnými parametry. Jsou typu private, protože k nim nepotřebuji přístup z ostatních tříd, a ty co jsou definované jsou k tomu i typu final, tedy hodnota jim přiřazená se už nemění (je finální).

2.3.1. Metoda start()

Metoda start je první metoda volána po spuštění aplikace. Je to již vývojovým prostředím před-vytvořená metoda a je typu public void s atributem stage typu Stage. V této metodě vytvořím a definuju nejvíce potřebné proměnné a objekty jako jsou scene, její kořen a stage.

V metodě start() pracuji se dvěma proměnnými typu Group a to z důvodu interakce kurzoru myši. Do proměnné universe typu Group jsou přiřazeny veškeré objekty viditelné při běhu aplikace. Ale jako kořen scény scene definuji proměnnou root typu Group, do které přiřadím objekty z universe, a objekty vrácené metodou buttons(). *(Obrázek 1)* Toto dělám, protože na proměnnou universe následně volám metodu initMouseControl(). Proměnná root obsahuje objekty, které při spuštění programu představují tlačítka „Info“ a „Exit“, a informační box s obecnými informacemi o sluneční soustavě jako celku. Na tyto objekty nechci volat initMouseControl(), protože ta se stará o přibližování a jiné přizpůsobování scény a tlačítka s boxem chci mít stále ukotvena na jednom místě.

```
universe.getChildren().addAll(bigBang());
universe.getChildren().add(moon());
universe.getChildren().add(sunLight());
universe.getChildren().add(background());
universe.getChildren().addAll(highlights());
universe.getChildren().addAll(titles());
universe.getChildren().addAll(saturnRings());

Group root = new Group();
root.getChildren().add(universe);
root.getChildren().addAll(buttons());
```

Obrázek 1 – Ukázka kódu z metody start()

Kromě již zmíněné definice scény scene a okna stage programu tato metoda volá většinu ostatních metod, designuje kurzor myši a definuje proměnnou camera typu Camera, která je následně přiřazena ke scéně.

2.3.2. Metoda bigBang()

Tato metoda, přestože velmi krátká, je nesmírně důležitá, protože vytváří všech 8 planet sluneční soustavy a Slunce. Je to první metoda volaná po metodě start() a vrací pole planets typu Planet. Jediné, co obsahuje, je for cyklus, ve kterém po jednom vytvoří 9 vesmírných těles a podle jejich indexu jim přiřazuje z již deklarovaných polí správné hodnoty.

2.3.3. Metoda moon()

Metoda moon() manuálně vytváří a vrací objekt moon typu Sphere. Obsahuje pouze pár řádku, ve kterých specifikuji parametry sféry. Tato metoda mi přijde lehce zbytečná, protože jsem mohla proměnnou moon vytvořit zároveň s objekty Planet v metodě bigBang(), ale bohužel jsem toto kvůli časové omezenosti nestihla.

2.3.4. Metoda sunLight()

Opět velmi krátká metoda, vrací proměnnou pl typu Pointlight, která představuje svit ze Slunce. Obsahuje upřesnění souřadnic, ze kterých pl svítí.

2.3.5 Metoda background()

Metoda background() vrací proměnnou imageView typu ImageView. Tato proměnná se poté zobrazuje jako pozadí celé sluneční soustavy, tedy hvězdy. V metodě vytvořím proměnnou image typu Image, který následně přiřadím ke vracející proměnné imageView.

2.3.6. Metoda highlights() a titles()

Metoda highlights() vrací pole planetsHighlights typu Circle, které při spuštění představuje barevné označení planet. Za pomocí for cyklu každé proměnné z pole přiřadím její atribut, který získám univerzálním algoritmem, do kterého zakomponuji poloměr proměnné z pole planets se stejným indexem *i*. Toto určení použiji u všech proměnných z pole, kromě té na místě s indexem 0, ta představuje označení Slunce. Slunce má obrovský poloměr, proto na něj univerzální určení neplatí a hodnotu zadávám manuálně. Po upřesnění atributů u každé proměnné z pole, mu určím zápornou viditelnost, tedy není viditelný. To udělám, protože pole s barevným označením planet/Slunce bude viditelné pouze po interakci s kurzorem, tedy výchozí viditelnost je záporná.

Metoda titles() pracuje na stejném principu akorát s proměnnými typu Text, a vrací pole planetsTitles typu Text.

2.3.7. Metoda saturnRings()

Metoda saturnRings() vrací pole saturnRings typu Circle. Toto pole je po spuštění aplikace vyobrazeno jako 4 kruhy kolem Saturnu, tedy prstence. Nejdříve manuálně každé proměnné z pole přiřadím její poloměr a poté ve for cyklu každé přidám stejné atributy. Každá z proměnných se orientuje podle souřadnic proměnné z pole planets s indexem 6 – tedy Saturnu. Jako poslední si do pole ringRadiuses typu double na místo s indexem *i* uložím poloměr prstence s se stejným indexem. Toto dělám, protože až budu prstence animovat, potřebuji vědět jejich výchozí poloměr.

2.3.8. Metoda buttons()

Již zmíněná metoda buttons() vrací pole objektů typu Button a StackPane. Vrácené objekty po spuštění představují tlačítko „Info“ a „Exit“, a informační okno s obecnými informacemi o sluneční soustavě jako celku.

V této metodě načtu text z textového souboru a následně ho uložím do proměnné systemInfo typu Label, který představuje samotný text. Proměnná systemInfoBox typu Rectangle slouží jako box, ve kterém se zmíněný text nachází. Poté vytvořím objekt popUp typu StackPane, do kterého přidám systemInfo a systemInfoBox, abych s nimi mohla manipulovat najednou.

Pro vytvoření tlačítek vytvořím pole buttons typu Button, ve for cyklu jim přiřadím souřadnice, rozměry a další detaily. Pomocí metody setOnMouseClicked() zajistím, aby tlačítka reagovala na kliknutí. V případě kliknutí na tlačítko z pole buttons s indexem 0, tedy „Info“, se zobrazí informační box a text ukázaný na tlačítku se změní na „Back“. Po opětovném kliknutí na toto tlačítko, nyní s titulkem „Back“, informační okno zmizí a titulek se přepíše zpátky na „Info“. Při kliknutí na tlačítko „Exit“ se aplikace zavře. *(Obrázek 2)*

```
int finalI = i;

buttons[i].setOnMouseClicked(MouseEvent -> {

    if(finalI==0){

        if(!infoPopUp.isVisible()){
            infoPopUp.setVisible(true);
            buttons[finalI].setText("Back");
        }
        else{
            infoPopUp.setVisible(false);
            buttons[finalI].setText("Info");
        }

    }
    else{
        Platform.exit();
    }

});
```

Obrázek 2 - Ukázka kódu z metody buttons()

2.3.9. Metoda animation()

Metoda animation() je první nevracící metoda a stará se o rotaci vesmírných objektů pomocí třídy AnimationTimer. Tato třída se v JavaFX využívá pro vytvoření plynulých animací a poskytuje způsob, jak opakovat blok kódu s pevným intervalem. Její součástí je metoda handle(), která je konstantně volána pro každý snímek animace. Tato metoda třídy AnimationTimer obsahuje atribut long l, který představuje interval, podle kterého bude metoda volána. *(3)*

Uvnitř zmíněné metody handle() vytvořím for cyklus, ve kterém na každý objekt pole planets zavolám, se správnými atributy, metodu rotate(), která je definována ve třídě Planet. Mimo tento for cyklus se v metodě handle() nachází řádek kódu, který se stará o rotaci proměnné moon typu Sphere za pomocí dvou Sphere metod setRotate() a getRotate(). Díky těmto do sebe vnořeným metodám se vesmírné objekty nepřetržitě točí.

2.3.10. Metoda planetInteractions()

Metoda planetInteraction() se stará o animaci vrácených objektů z metod highlights() a titles(). Ve for cyklu všem proměnným z pole planets přiřadí stejné funkce, tedy setOnMouseEntered() a setOnMouseExited(). Funkce setOnMouseEntered() probíhá jen v případě, kdy je planeta oddálena – toto určíme proměnnou zoomedIn typu boolean. V případě, že se na vesmírný objekt přejede kurzorem zobrazí se jeho barevné označení a název, oba se stejným indexem *i* jako proměnná z pole planets. To samé platí pro odstranění kurzoru z vesmírného objektu, tedy zavolá se metoda setOnMouseExited(), a barevné označení s textem zmizí.

Kromě toho zajišťuje i animaci při kliknutí na vesmírný objekt pomocí metody setOnMouseClicked(). V tomto případě volá metodu action() s atributem finalI typu int, což je index aktuální planety v poli planets.

2.3.11. Metoda action()

Metoda action() se zaslouží o animaci planet/Slunce a informačního okna ke každé z nich při kliknutí. Je to první metoda nevolaná metodou start(), je poprvé použita již zmíněnou metodou planetInteractions(). Pomocí atributu index typu int zná, na jaký vesmírný objekt bylo kliknuto a ten poté zanimuje.

V této metodě používám 2 animovací třídy TranslateTransition a TimeLine. Třída TranslateTransition dokáže v libovolném čase přesunout objekt z místa na místo ve X, Y a Z souřadnicích. Obsahuje spoustu metod jako setCycleCount(), setAutoReverse(), setToX(), setFromX(), play() a další. *(4)*

Třída TimeLine je o něco komplikovanější a skládá se z proměnných typu KeyFrame. Objekt TimeLine je vytvořen určením jeho délky trvání, hodnot, které animované objekty získají po dokončení, a přidáním komponentů, které bude animovat – takzvané KeyValue. Každá KeyFrame má v sobě tedy uloženou časovou hodnotu, objekt nebo objekty, které se budou ve zmíněném čase animovat, a jejich finální hodnoty po dokončení animace. Tato třída opět obsahuje několik metod jako zmíněná třída TranslateTransition, ale ve svém kódu využiji pouze getKeyFrames(), add() a play(). *(5)*

S již zmíněnými třídami pracuji formou devíti objektů zoomInX, zoomOutX (TranslateTransition) a zoomInSize, zoomOutSize, zomInPl, zoomInPlSun, zoomOutPl, tlIn, tlOut (TimeLine). Je to z toho důvodu, že vesmírné objekty se mi přibližují/oddalují nikoliv pomocí změny souřadnice Z, jako je to například u informačních boxů, ale díky změnám jejich poloměrům. Možnost animovat změnu poloměru má třída TimeLine. Když jsem využila třídu TimeLine i na upravení X souřadnic objektu, neustále se mi měnil takzvaný pivotX, tedy středový bod tělesa, proto využívám i objekty TranslateTransition. Jsem si jistá, že je této animace možno dosáhnout jednodušším způsobem, ale to je zatím mimo moje schopnosti.

Jako první si v této metodě definuji objekty, které se zaslouží o animaci při prvním kliknutím (přiblížení), tak i přitom druhém (oddálení). Jako objekt, který budou animovat, jim přiřadím proměnnou z pole planets s indexem proměnné index, který metoda získala při jejím volání. *(Obrázek 3)* Díky proměnné zoomIn typu boolean znám stav přiblížení vesmírného tělesa, a taky podle toho volám další animace. *(Obrázek 4)*

```
private void action(int index) throws IOException {

    zoomInX.setNode(planets[index]);
    zoomInX.setToX(HEIGHT /2);
    zoomOutX.setNode(planets[index]);
    zoomOutX.setToX(xProperties[index]);

    zoomInSize.getKeyFrames().add(
            new KeyFrame(Duration.seconds( v: 1.5),
                    new KeyValue(planets[index].radiusProperty(), t: 220)
            ));

    zoomOutSize.getKeyFrames().add(
            new KeyFrame(Duration.seconds( v: 1.5),
                    new KeyValue(planets[index].radiusProperty(), sizes[index])
            ));
```

Obrázek 3 - Ukázka kódu z první části metody action()

```
if(!zoomedIn) {

    currentInfoBox = prepareInfoBox(index);
    planetsInfoBox(zoomedIn, currentInfoBox);

    zoomedIn = true;

    zoomInSize.play();
    zoomInX.play();

    if(index==0){
        zoomInPlSun.play();
    }
    else{
        zoomInPl.play();
    }

    Timeline tlIn = new Timeline();
    for (int i = 0; i < planets.length; i++) {

        if(i != index){
            tlIn.getKeyFrames().add(
                    new KeyFrame(Duration.seconds( v: 1.5),
                            new KeyValue(planets[i].radiusProperty(), t: 0
                    ));
        }

    }

    for (Circle r : saturnRings) {

        tlIn.getKeyFrames().add(
                new KeyFrame(Duration.seconds( v: 1.5),
                        new KeyValue(r.radiusProperty(), t: -1)
                ));

    }

    tlIn.play();
}
```

```
else{

    zoomOutSize.play();
    zoomOutX.play();

    if(index != 0){
        zoomOutPl.play();
    }

    planetsInfoBox(zoomedIn, currentInfoBox);
    zoomedIn=false;

    Timeline tlOut = new Timeline();
    for (int i = 0; i < planets.length; i++) {

        if(i != index){
            tlOut.getKeyFrames().add(
                    new KeyFrame(Duration.seconds( v: 1.5),
                            new KeyValue(planets[i].radiusProperty(), sizes[i])
                    ));
        }

    }

    for (int i = 0; i < saturnRings.length; i++) {

        tlOut.getKeyFrames().add(
                new KeyFrame(Duration.seconds( v: 1.5),
                        new KeyValue(saturnRings[i].radiusProperty(), ringRadiuses[i])
                ));

    }

    tlOut.play();
}
```

Obrázek 4 - Ukázka kódu z druhé časti metody action()

U této metody jsem měla problém s proměnnou pl typu Pointlight, tedy slunečního svitu. Jelikož objekt s indexem 0 v poli planets, Slunce, má jako svůj „kabátek“ (Material) nastavenou osvětlující mapu (SelfIlluminationMap). V normálním případě jsem proměnnou pl, představující světlo ze Slunce, při přiblížení planety přiblížila a centrovala, aby na vybranou planetu bylo hezky vidět, ale se Sluncem to tak nešlo. Již zmíněná SelfIlluminationMap vypadá velmi vybledle, když se střetne s jiným zdrojem světla. To mě tedy donutilo k tomu, aby Slunce mělo svou vlastní animaci proměnné pl, proto jsou vytvořené 2 objekty typu TimeLine zoomInPl a zoomInPlSun.

Jak je vidět na *Obrázku 4*, metoda během svého běhu volá další dvě metody, metodu prepareInfoBox() s atributem index typu int a metodu planetsInfoBox() s atributem zoomedIn typu boolean a currentInfoBox typu StackPane.

2.3.12. Metoda prepareInfoBox()

Metoda prepareInfoBox() je volána metodou action() s atributem index typu int, který určuje na jakou planetu/Slunce uživatel klikl (její místo v poli planets), vrací proměnnou infoBox typu StackPane. Metoda se postará o „připravení“ informačního okna, které bude viditelné po kliknutí na planetu/Slunce.

V této metodě načítám do proměnné planetInfo typu StringBuilder informační text, ke kterému získám cestu z proměnné umístěné v poli infoUrls se stejným indexem jako je planeta/Slunce. Text následně uložím do proměnné text typu Label, vytvořím a definuju proměnnou box typu Rectangle, která představuje box, do kterého text náleží. Tyto oba objekty poté přiřadím objektu infoBox typu StackPane, abych s nimi mohla manipulovat najednou. Nejdůležitějším řádkem této metody je řádek *universe.getChildren().add(infoBox)*, který se postará o to, aby bylo informační okno zakomponováno do scény.

2.3.13. Metoda planetsInfoBox()

Nevracející metoda planetsInfoBox() je volána metodou action() a má atributy zoom typu boolean a infoBox typu StackPane. Tato metoda se stará o animaci objektu infoBox, představující informační okno uživatelem zvolené planety/Slunce. To, zda se se má informační okno přibližovat nebo oddalovat, určuje známý atribut zoomIn typu boolean – když je false, okno není přiblíženo, tedy se volá animace, která ho přiblíží a naopak.

V metodě používám pro animaci objekty třídy ScaleTransition scaleIn a scaleOut. Jako první určím dobu trvání u každé z nich a přiřadím jim aktuální informační okno. Poté definuji přiblížení u objektu scaleIn a oddálení u scaleOut. Animace volám podle výše zmíněné proměnné zoom typu boolean.

2.3.14. Metoda initMouseControl()

Metoda initMouseControl() je nevracející, a je to poslední mnou vytvořená metoda volána metodou start(). Tato metoda se zaslouží o komunikaci scény jako celku s myší. Je tedy zodpovědná za pohyb scény do stran při stisknutí kurzoru myši a přiblížení a oddálení při interakci s kolečkem myši. Ke své plné funkčnosti potřebuje atributy scene typu Scene a stage typu Stage.

Jako první proměnné universe typu Group, ke které mám přístup ze všech metod, přiřadím souřadnice 0 v X, Y i Z, a ty následně uložím do proměnných tX a tY typu Translate. V této metodě volám 2 metody spojené s mouse events a metodu spojenou se scroll event. Prvně volám metodu setOnMousePressed(), ve které získám aktuální polohu kurzoru. Poté metodu setOnMouseDragged(), kde k souřadnicím scény přiřazuji hodnoty, ze kterých plyne pohyb kurzoru myši. Jako poslední vytvářím takzvaný event handler pro scroll events. Ten detekuje pohyb kolečka myši a následně přidává a odebírá proměnné universe souřadnici Z – oddaluje ji a přibližuje. *(Obrázek 5)*

```
scene.setOnMousePressed(mouseEvent -> {

    startingX = mouseEvent.getSceneX();
    startingY = mouseEvent.getSceneY();

    currentX = newX.get();
    currentY = newY.get();

});

scene.setOnMouseDragged(mouseEvent -> {

    newX.set(currentX + (mouseEvent.getSceneX()-startingX));
    newY.set(currentY + (mouseEvent.getSceneY()- startingY));

});

stage.addEventHandler(ScrollEvent.SCROLL, scrollEvent -> {

    double movement = scrollEvent.getDeltaY();
    universe.translateZProperty().set(universe.getTranslateZ() - 1.5*movement);

});
```

Obrázek 5 - Ukázka kódu z metody initMouseControl()

3. Nezávislé komponenty programu

Ke správné funkčnosti mého programu je potřeba dohromady 14 obrázků a 10 textových souborů. 9 ze zmíněných obrázků obsahuje obyčejnou texturu všech planet sluneční soustavy a texturu Slunce. Na objekt, který představuje Zemi, jsem přidala 2 vrstvy obrázků navíc, aby vypadal více realisticky. Součástí programu je i vesmírné pozadí, na kterém jsou vidět hvězdy a Mléčná dráha, a 14. obrázek činí obrázek kurzoru myši. Všechny přiložené obrázky mají koncovku buď .png nebo .jpg, a všechny jsou velmi vysokého rozlišení. Mezi 10 potřebných textových souborů patří 9 popisujících jednotlivé planety a Slunce, a jeden s textem o sluneční soustavě jako celku. Všechny tyto soubory se nacházejí přímo ve složce resources nebo případně v podsložce této složky.

4. Použité nástroje

Jako vývojové prostředí pro moji ročníkovou jsem si vybrala IntelliJ IDEA Community Edition a psala ji za pomocí programovacího jazyku Java a knihovny JavaFX. Všechny tyto technologie jsou bezplatné a přístupné ke stažení na internetu.

IntelliJ IDEA je integrované vývojové prostředí (IDE) pro Javu, které poskytuje komplexní sadu nástrojů a funkcí pro vývoj softwaru. Nabízí inteligentní pomoc s kódem, automatizované refaktorování, nástroje pro ladění a mnoho dalších funkcí, které zvyšují produktivitu a dělají ho oblíbenou volbou mezi vývojáři v Javě. *(6)*

Java je vysokoúrovňový, objektově orientovaný programovací jazyk používaný pro vývoj široké škály aplikací, od mobilních aplikací po podnikový software. Je známý pro svou nezávislost na platformě, bezpečnostní funkce a robustnost, což ho dělá populární volbou pro budování komplexních a škálovatelných softwarových systémů. *(7)*

JavaFX je knihovna, která poskytuje moderní, intuitivní a lehký toolkit uživatelského rozhraní pro vývoj bohatých desktopových a mobilních aplikací v Javě. Nabízí výkonnou sadu grafických a mediálních API, animační funkce a ovládací prvky uživatelského rozhraní, které ho dělají vynikající volbou pro vytváření moderních a dynamických aplikací. *(8)*

5. Komplikace při psaní programu

Během psaní ročníkové práce jsem narazila na spoustu problémů, které mi zkomplikovaly a pozdržely psaní. Větší část z těchto nepříjemností jsem po čase vyřešila a ponaučila se z nich, ale na některé se mi nepodařilo přijít doteď, což v krajním případě může i kazit vzhled mého finálního výsledku.

Jeden z mých prvních vážnějších problémů bylo programování funkce initMouseControl(). Scéna mi na stisknutí kurzoru myši a jeho tažení reagovala zvláštně. Nedokázala jsem přijít na jednotný algoritmus, který by se postaral o přiřazení správných souřadnic scéně po tažení kurzoru. *(Obrázek 5)* Nakonec jsem našla video na YouTube, kde tvůrce videa tento problém rozebíral a zvládla jsem na to přijít (2. odkaz v kapitole Seznam zdrojů). V tomto případě jsem nad problémem strávila přibližně dva celé dny, což není tak dlouhá doba, ale možnost ovládat scénu myší byla pro mě jedna z nejdůležitějších součástí mé ročníkové práce.

Další komplikace nastala při vytváření prstenců planety Saturn. Nejdříve jsem se několik hodin snažila vytvořit neobvyklý 3D objekt, který by napodoboval prstence, poté průhlednou sféru, jejíž textura by byl jenom proužek, ale nic mi nefungovalo, nebo to vypadalo zvláštně. U tohoto jsem byla velmi zoufalá, jelikož prstence Saturnu mi přišly jako nepostradatelný komponent sluneční soustavy. Naštěstí mě po pár dnech napadla myšlenka vytvořit pole několika objektů Circle s jinými poloměry a obtočit je kolem Saturnu. Nejdříve to vypadalo zvláštně, ale po přidání efektu Shadow se mi podařilo je hezky upravit a nyní vypadají lépe.

Můj největší problém nastal při následné animaci již zmíněných prstenců Saturnu. Doteď se mi tento problém nepovedlo vyřešit nějak efektivně, takže při spuštění aplikace a kliknutí na Saturn se jeho prstence nepřiblíží, ale oddálí společně s ostatními tělesy. Byla bych schopna je přiblížit, ale kód, kterým bych toho docílila, by byl velmi složitý a nehezký, takže jsem to radši přestala řešit a nechala mu animaci oddálení společně s ostatními planetami. Tuším proč, tento problém nastal, jsou to 2D objekty zakomponované ve 3D soustavě objektů, tím pádem manipulace s nimi nebude stejná. Tohoto je mi velmi líto, ale naštěstí to není to nejdůležitější na celém programu.

Mojí asi poslední větší komplikací bylo při odkazování na potřebné soubory. V průběhu psaní jsem odkazy na všechny soubory měla v takzvané Absolute Path, tedy velmi dlouhém odkazu, který odkazoval na soubor až od disku mého počítače. Nějak jsem si s tím nelámala hlavu, ale ke konci psaní mi došlo, že takto si ostatní můj program nepustí, tím pádem jsem musela přijít na to, jak na ně odkázat nějak univerzálně jenom od složky, kde se můj program nachází. Inspirovala jsem se z kódu mých spolužáků na webové stránce GitHub, kde všichni používali metody *getClass().getResource()*, ale z nějakého důvodu mi toto nefungovalo. Nakonec jsem přišla na to, že mi celou dobu chybělo jednoduché lomítko, což je nyní k pousmání. Než mi toto došlo, trvalo to kolem hodiny, což není vůbec moc, ale byla jsem z toho hrozně nervózní, protože jsem si myslela, že můj už skoro hotový program nebude nakonec funkční.

6. Potenciální vylepšení do budoucnosti

Ať už je to vzhled kódu nebo aplikace, mám na mysli velké množství věcí, které bych moc ráda vylepšila. V budoucnosti se mám v plánu k programu pravděpodobně vrátit a vylepšit ho, ale potřebuji si dá na chvíli od programování pauzu.

Za prvé, co se týče kódu, během psaní tohoto dokumentu a při rozebírání jednotlivých metod mi došlo, že spousta z nich by se dala napsat mnohem jednodušeji nebo zkombinovat jednu s druhou. Za druhé, kód mám rozdělený pouze do dvou tříd a vím, že jsou tam objekty a pole, které by se určitě daly definovat jednodušeji, kdybych vytvořila třídu přímo pro ně.

Na vizuální stránce programu by se dalo vylepšit spoustu věcí, ale některé jsou jednoduše mimo moje znalosti a na některé jsem neměla čas. Jako první bych rozhodně vylepšila animaci již zmíněných prstenců Saturnu, které se společně s Měsícem nepřibližují při kliknutí. Mám spoustu dalších nápadů jako měsíce ostatních planet, nejen Země, interaktivní vesmírné satelity, 3D povrchy planet, animované pozadí, rotační osa planet, ale to už jsou takové extrémy, na které bohužel zatím nemám dostatek znalostí.

Co mi na vzhledu programu ale velmi vadí, je rozměr jeho okna. Po celý průběh psaní ročníkové práce jsem měla Stage určenou na FullScreen a program vypadal velmi pěkně. Bohužel pak jsem zjistila, že při běhu aplikace na počítači s jinými rozměry obrazovky, než jaký mám já, se aplikace sice do FullScreenu dá, ale všechny do scény přidané objekty nejsou uprostřed okna, a to vypadalo zvláštně. Musela jsem tedy na poslední chvíli měnit rozměry okna.

Jeden nápad, na který bych měla dostatečně rozšířené obzory a mrzí mě nejvíc je, že jsem nevytvořila nějaké úvodní menu a popřípadě test znalostí sluneční soustavy.

7. Závěr

Ve své ročníkové práci z programování jsem se snažila naprogramovat aplikaci, která zobrazí sluneční soustavu ve 3D a uživatel bude mít možnost s ní kurzorem myši komunikovat a zobrazovat informační okna.

Psaní programu mě obohatilo o spoustu nových vědomostí programovacích i astronomických. Během jejího zpracování jsem se naučila lépe si rozvrhnout čas a jak být více a déle produktivní bez potřeby se zabývat například mobilním telefonem. S finálním výsledkem jsem hodně spokojena, i když tam vidím spoustu chyb. Jsem na sebe velmi pyšná, protože jsem se ze skoro kompletní neznalosti JavaFX dokázala dostat až na naprogramování interaktivní 3D sluneční soustavy.

Podařilo se mi vytvořit, definovat a do scény zakomponovat devět 3D sfér, které představují jednotlivé planety sluneční soustavy a Slunce. Dále jsem dokázala vytvořit sféru, která ztělesňuje Měsíc a obíhá planetu Zemi. Všechna moje vesmírná tělesa se konstantě točí kolem své osy a jsou osvícena svitem ze Sluncem. Tlačítky je možno aplikaci zavřít nebo zobrazovat informační okna, a kurzorem myši s jednotlivými objekty komunikovat a přizpůsobovat si vzhled a polohu scény.

Reference

1) Informace dostupná z:
   https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Sphere.html
2) Informace dostupná z:
   https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html
3) Informace dostupná z:
   https://docs.oracle.com/javase/8/javafx/api/javafx/animation/AnimationTimer.html
4) Informace dostupná z:
   https://docs.oracle.com/javase/8/javafx/api/javafx/animation/AnimationTimer.html
5) Informace dostupná z:
   https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Timeline.html
6) JetBrains. (2023). IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains.
   Navštíveno: April 27, 2023, Dostupné z: https://www.jetbrains.com/idea/ (přeloženo)
7) Wikipedia. (2023, April 24). Java (programming language).
   In Wikipedia, The Free Encyclopedia. Navštíveno: April 27, 2023,
   Dostupné z: https://en.wikipedia.org/wiki/Java_(programming_language) (přeloženo)
8) OpenJFX. (n.d.). OpenJFX: Rich client application platform for Java.
   Navštíveno: April 27, 2023, Dostupné z: https://openjfx.io/ (přeloženo)

Seznam použitých webových zdrojů

1) https://www.youtube.com/@GenuineCoder
2) https://genuinecoder.com/
3) https://www.javatpoint.com/
4) https://docs.oracle.com/
5) https://jenkov.com/
6) https://www.javaguides.net/
7) https://www.itnetwork.cz/
8) http://www.java2s.com/

Seznam obrázků

Obrázek 1 – Ukázka kódu z metody start() 4

Obrázek 2 – Ukázka kódu z metody buttons() 7

Obrázek 3 – Ukázka kódu z první části metody action() 10

Obrázek 4 – Ukázka kódu z druhé části metody action() 10

Obrázek 5 – Ukázka kódu z metody initMouseControl() 13