Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Využití JavaFX při tvorbě grafických aplikací

Ondřej Beneš

Bakalářská práce

2015

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej Beneš**
Osobní číslo:	**I12102**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Využití JavaFX při tvorbě grafických aplikací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Práce se bude po teoretické i praktické stránce věnovat programování grafických aplikací s využitím knihovny JavaFX 2.

V teoretické části budou popsány možnosti vytváření aplikací pomocí JavaFX, zejména ve srovnání s vytvářením aplikací, postavených na jiných technologiích a knihovnách. V praktické části budou implementovány ukázkové příklady, vhodně doplňující teoretickou část.

Volba příkladů bude volena s ohledem na možné využití při výuce počítačové grafiky. V rámci praktické části bude vytvořen multimediální výukový materiál, seznamující uživatele se základy využívání knihovny JavaFX 2.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

WEAVER, James L. Pro JavaFX 2: a definitive guide to rich clients with java technology. Lexington: Apress, 2012, 620 s. The expert's voice in Java. ISBN 978-1-4302-6872-7.

DEA, Carl. JavaFX 2.0: Introduction by Example. New Edition. New York: Apress, 2011, 200 s. ISBN 978-143-0242-574.

DIMARZIO, J. JavaFX: a beginner's guide. New Edition. New York: McGraw-Hill, 2011, 300 s. ISBN 00-717-4241-7.

Vedoucí bakalářské práce: **Ing. Petr Veselý**
Katedra softwarových technologií

Datum zadání bakalářské práce: **20. prosince 2014**
Termín odevzdání bakalářské práce: **11. května 2015**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2015

PROHLÁŠENÍ AUTORA

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.
Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.
Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 5. 2015.

Ondřej Beneš

PODĚKOVÁNÍ

Mé díky patří vedoucímu práce, Ing. Petru Veselému, za jeho vždy profesionální a zároveň také lidský přístup. Děkuji také mým blízkým za poskytnutou zpětnou vazbu.

ANOTACE

Práce se po teoretické i praktické stránce zabývá softwarovou platformou JavaFX. Práci lze využít jako studijní materiál pro předmět Počítačová grafika. Prvních sedm kapitol tvoří teoretickou část. Stručné představení platformy JavaFX je následováno srovnáním se softwarovými knihovnami Swing a WPF. V dalších kapitolách je popsáno vytváření grafického uživatelského rozhraní, použití animací a práce s mediálními soubory. Pozornost je také věnována tvorbě 3D obsahu a umístění aplikace na web. V praktické části jsou představeny ukázkové JavaFX aplikace a webová aplikace. Popsán je také proces tvorby video tutoriálů, ve kterých jsou probrány základy tvorby JavaFX aplikací.

KLÍČOVÁ SLOVA

JavaFX, Swing, FXML, grafické uživatelské rozhraní, animace, grafika

TITLE

Developing graphical applications using JavaFX

ANNOTATION

This thesis covers the JavaFX software platform from both a theoretical and practical standpoint. It is well suited to be used as a study material for the Computer Graphics course. The first seven chapters comprise the theoretical part. A brief introduction to JavaFX is followed by a comparison with Swing and WPF libraries. The following chapters describe the creation of Graphical User Interfaces, the usage of animations and working with media files. Due attention is also paid to developing 3D content and deploying the application on Web. Example applications as well as a Web application are described in the practical part of the thesis. The processes of making video tutorials which provide an introduction to JavaFX are also described.

KEY WORDS

JavaFX, Swing, FXML, Graphical User Interface, animations, graphics

OBSAH

ÚVOD 10
1 PŘEDSTAVENÍ JAVAFX 12
1.1 Historie 12
1.2 Srovnání s knihovnou Swing 12
1.2.1 Výhody JavaFX 12
1.2.2 Výhody knihovny Swing 13
1.2.3 Různé přístupy 13
1.2.4 Graphics2D a GraphicsContext 14
1.3 Srovnání s WPF 15
1.4 Alternativní JVM programovací jazyky 15
2 TVORBA GRAFICKÉHO UŽIVATELSKÉHO ROZHRANÍ (GUI) 16
2.1 Vytvoření projektu v NetBeans 16
2.2 Třídy GUI 16
2.2.1 Stage – Jeviště 17
2.2.2 Scene – Scéna 18
2.2.3 Node – Uzel 19
2.2.4 Kontejnery – Region 19
2.2.5 Ovladače – Control 20
2.2.6 Canvas – Plátno 20
2.2.7 Shape – Tvar 21
2.2.8 Chart – Diagramy 23
2.3 Vytváření GUI pomocí jazyka FXML 24
2.4 Úprava vzhledu aplikace použitím jazyka Cascading Styles Sheet (CSS) 26
2.4.1 Selektory 26
2.4.2 Použití CSS 26
2.4.6 Unikátní vlastnosti 28
2.4.4 Pseudotřídy 28
2.4.5 Stínování a barevné přechody 28
2.4.7 Substruktury 29
2.5 Scene Builder 30
3 ANIMACE 31
3.1 Timeline 31
3.2 Transition 32
3.2.1 Fade 32
3.2.2 Fill 32
3.2.3 Parallel 33
3.2.4 Path 33
3.2.5 Pause 33
3.2.6 Rotate 33
3.2.7 Scale 34
3.2.8 Sequential 34
3.2.9 Stroke 34
3.2.10 Translate 34

4 DATA BINDING, KOLEKCE ... 35
4.1 Data binding ... 35
4.1.1 Vysokoúrovňový přístup ... 35
4.1.1 Nízkoúrovňový přístup ... 36
4.2 Balíček FXCollections ... 37
5 OBRÁZKY, AUDIO A VIDEO ... 39
5.1 Obrázky ... 39
5.1.1 Image ... 39
5.1.2 ImageView ... 39
5.2 Audio a video ... 41
5.2.1 Media ... 41
5.2.2 MediaPlayer ... 41
5.2.3 MediaView ... 42
5.2.4 AudioClip ... 43
6 3D SCÉNY ... 44
6.1 Tvorba a použití 3D těles ... 44
6.2 Práce s kamerou ... 45
6.3 Práce se světlem ... 46
7 UMÍSTĚNÍ APLIKACE NA WEB ... 47
7.1 Deployment Toolkit ... 47
7.2 Proces umístění aplikace na web ... 47
7.3 Certifikáty ... 48
8 PRAKTICKÁ ČÁST ... 49
8.1 Ukázkové aplikace ... 49
8.1.1 ListViewTableViewDemo ... 49
8.1.2 PerformanceComparison ... 49
8.1.3 PanesDemo ... 51
8.1.4 CanvasDemo ... 52
8.1.5 PieChartDemo ... 52
8.1.6 XYChartDemo ... 53
8.1.7 AnimationsDemo ... 53
8.1.8 Triangle ... 53
8.1.9 MediaDemo ... 54
8.1.10 3DDemo ... 55
8.2 Webová aplikace ... 55
8.3 Video tutoriály ... 56
8.4 Srovnání JavaFX a Swing po stránce výkonu ... 56
9 ZÁVĚR ... 64
10 POUŽITÁ LITERATURA ... 66
11 PŘÍLOHY ... 70

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1: Znázornění grafu scény v programu Scene Builder 16
Obrázek 2: Umístění Jeviště v hierarchii prvků GUI 17
Obrázek 3: Styl Jeviště Utility a styl Undecorated 18
Obrázek 4: Geometrické tvary 22
Obrázek 5: Tvar vzniklý sjednocením obdélníku a kruhu 22
Obrázek 6: LineChart zobrazující hustotu normálního rozdělení 24
Obrázek 7: Nástroj Scene Builder 30
Obrázek 8: Ořez obrázku 40
Obrázek 9: Zorný úhel kamery 30° a 120° 45
Obrázek 10: Část Scény snímaná kamerou 45
Obrázek 11: Scéna osvětlená bodovým zdrojem a ambientním světlem 46
Obrázek 12: Zpráva o zablokování nedůvěryhodné aplikace 48
Obrázek 13: Aplikace ListViewTableViewDemo 49
Obrázek 14: JavaFX varianta aplikace ScreenSaver 50
Obrázek 15: JavaFX varianta aplikace ImageEditor 50
Obrázek 16: JavaFX varianta aplikace Drawing 51
Obrázek 17: Aplikace PanesDemo 51
Obrázek 18: Aplikace CanvasDemo 52
Obrázek 19: Aplikace PieChartDemo 52
Obrázek 20: Aplikace AnimationsDemo 53
Obrázek 21: Aplikace Triangle, varianta bez FXML a varianta s Data binding 54
Obrázek 22: Aplikace MediaDemo 54

Graf 1: Průměr FPS na PC1 58
Graf 2: Průměr FPS na PC2 58
Graf 3: Průběh FPS pro 10 tis. kruhů na PC1 59
Graf 4: Čas nutný k převedení obrázku běžné velikostí do odstínů šedi, PC1 60
Graf 5: Čas nutný k převedení obrázku běžné velikostí do odstínů šedi, PC2 61
Graf 6: Čas nutný k převedení velkých obrázků do odstínů šedi, PC1 61
Graf 7: Čas nutný k vykreslení Bézierových křivek, PC1 63
Graf 8: Čas nutný k vykreslení Beziérových křivek, PC2 63

Tabulka 1: Průměr FPS v aplikaci Screensaver 57
Tabulka 2: Čas nutný k převedení obrázku do odstínů šedi (PC1) 60
Tabulka 3: Vykreslování Bézierových křivek 62

SEZNAM ZKRATEK A ZNAČEK

AIFF	Audio Interchange File Format
API	Application Programming Interface
CSS	Cascading Style Sheets
DSL	Domain Specific Language
FLV	Flash Video
FPS	Frames Per Second
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
HTML	HyperText Markup Language
HUD	Head-Up Display
HWA	Hardware Acceleration
IDE	Integrated Development Environment
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MP3	MPEG-1 or MPEG-2 Audio Layer III
MPEG	Moving Pictures Experts Group
MVC	Model-View-Controller
PHP	PHP: Hypertext Preprocessor
RIA	Rich Internet Application
RGB	Red, Green, Blue
URL	Uniform Resource Locator
VPN	Virtual Private Network
WAV	Waveform Audio File Format

TERMINOLOGIE

FXML	Značkovací jazyk založený na XML (Extensible Markup Language), používá se pro definici stromové struktury prvků GUI.

ÚVOD

V současné době je běžné, že počítačové aplikace disponují bohatým grafickým uživatelským rozhraním. Výjimku tvoří konzolové aplikace, s těmi se ale běžný uživatel setká minimálně. Uživatelské rozhraní hraje klíčovou roli, protože na uživatele vytváří první dojem. Je tedy důležité, aby práce s ním byla intuitivní a aby bylo působivé po vzhledové stránce.

V aplikacích vyvíjených v jazyce Java se pro tvorbu grafického uživatelského rozhraní používá knihovna Swing. Ta ale postupně zastarává. Jako její náhrada vznikla softwarová platforma JavaFX. Tato práce si klade za cíl popsat základy této technologie, provést srovnání s knihovnou Swing a odpovědět na otázku, zda je JavaFX dostatečně vyspělá na to, aby knihovnu Swing nahradila.

Popis kapitol

V první kapitole je představena knihovna JavaFX, je popsáno její zaměření a ve stručnosti také její historie. Dále se kapitola věnuje srovnání s knihovnou Swing. Jsou popsány výhody jak JavaFX, tak Swing. Charakterizovány jsou také různé přístupy, které knihovny používají k obsluze událostí, k práci s tabulkami aj. Obecné srovnání je také provedeno s knihovnou WPF. V této kapitole jsou rovněž zmíněny alternativní JVM jazyky, které je možné k vývoji JavaFX aplikací použít.

Druhá kapitola se věnuje tvorbě grafického uživatelského rozhraní. Nejdříve je zmíněna podpora vývoje JavaFX aplikací v prostředí NetBeans. Dále jsou uvedeny třídy, které se za účelem tvorby GUI používají. Představen je také značkovací jazyk FXML. Následně je popsáno použití jazyka CSS k úpravě vzhledu aplikace. Pozornost je také věnována nástroji Scene Builder, který tvorbu GUI značně ulehčuje.

Třetí kapitola se zabývá animacemi. Je zde popsána abstraktní třída `Animation` a třídy od ní odvozené. Jedná se o třídy `Timeline` a `Transition`, které nabízejí různý přístup k tvorbě animací. Pro jednotlivé třídy jsou v přehledné formě uvedeny klíčové metody.

Ve čtvrté kapitole je popsán mechanismus Data binding, neboli svazování dat. V kapitole jsou uvedeny příklady, kdy je vhodné tento mechanismus použít a také jsou popsány různé způsoby jeho implementace. Tato kapitola se také zabývá balíčkem javafx.collections, který rozšiřuje standardní framework kolekcí jazyka Java.

Pátá kapitola je věnována médiím. Jsou v ní popsány třídy pro práci s obrázky, s audio soubory a s video soubory.

Šestá kapitola se zabývá tvorbou 3D scén. Jsou zde představena základní tělesa, která je možné v aplikacích použít. Zmíněna je také možnost tvorby vlastních těles, případně importu ze specializovaných programů. Dále se kapitola věnuje vytvoření a použití kamery, která scénu snímá. Zmíněno je také použití zdrojů světla.

V sedmé kapitole je ve stručnosti popsána knihovna *Deployment Toolkit*, pomocí které je možné umisťovat JavaFX aplikace na web. Pozornost je také věnována procesu umístění

aplikace na web a certifikátům, které snižují bezpečnostní riziko uživatelů webových JavaFX aplikací.

Poslední kapitola se věnuje praktické části bakalářské práce. Předkládá popis ukázkových aplikací, včetně odkazu na webovou verzi aplikace a ilustračního obrázku. Dále je popsána webová aplikace, ve které jsou umístěny důležité pasáže z textové části, ukázkové aplikace a video tutoriály. Video tutoriálům se věnuje poslední část této kapitoly. Rozebrán je zejména proces jejich tvorby. V této kapitole je také provedeno srovnání JavaFX a knihovny Swing po stránce výkonu.

Předpokládané znalosti čtenáře

Očekává se, že čtenář má základní znalosti z oblasti objektově orientovaného programování (pojmy třída, metoda apod.). Pro snadnější porozumění některým kapitolám je vhodná znalost knihovny Swing.

Použité konvence

Názvy tříd, rozhraní, metod apod. jsou psány písmem `Courier New`. Tento font byl použit také pro příklady. V příkladech jsou modrou barvou značená `klíčová slova`, oranžovou `řetězce` a zelenou `identifikátory proměnných` a *`HODNOTY VÝČTOVÝCH TYPŮ`*. Toto barevné značení odpovídá značení, které používá vývojové prostředí NetBeans.

1 PŘEDSTAVENÍ JAVAFX

JavaFX je softwarovou knihovnou (nebo také platformou), napsanou v jazyce Java a určenou k vývoji grafických aplikací. Dá se využít jako alternativa ke knihovně Swing, kterou by měla v budoucnu nahradit [1.1]. Vyvíjí ji společnost Oracle a je součásti JRE a JDK.

JavaFX odděluje vzhled aplikace od logiky, a to použitím jazyků CSS a FXML, které budou popsány v samostatných kapitolách. Díky tomuto separování mohou na aplikaci paralelně pracovat vývojář logiky aplikace, designér GUI a grafický designér. Oddělení vzhledu od logiky odpovídá návrhovému vzoru MVC, který tak lze při vývoji JavaFX aplikací přirozeně použít.

JavaFX je zaměřena zejména na vývoj RIA, nicméně vhodně slouží i k vývoji klasických desktopových aplikací. Podporovány jsou platformy Windows od verze XP, Mac OS X a Linux [1.2]. Distribucí JavaFX aplikací na mobilní zařízení a tablety se zabývá projekt JavaFXPorts[1]. RIA jsou „webové aplikace, které nabízejí vlastnosti a funkce běžně spojované s desktopovými aplikacemi“ [1.3]. Typickými zástupci technologií pro vývoj těchto aplikací jsou Adobe AIR[2], Microsoft Silverlight[3] a právě JavaFX [1.4].

1.1 Historie

JavaFX vychází z jazyka JavaFX Script, který vznikl za účelem tvorby propracovaných grafických uživatelských rozhraní (GUI) webových aplikací. Ve verzi 2.0, vydané v říjnu 2011, došlo k vypuštění JavaFX Script a nahrazení jazykem Java. Vývoj JavaFX Script pokračuje ve formě jazyka Visage, který je více rozebrán v kapitole 1.4 [1.5] [1.6].

Ve verzích 2.x byly zaváděny funkcionality jako jazyk FXML, podpora spolupráce s knihovnou Swing, kontejnery umožňující zobrazení webových stránek psaných v HTML / JavaScript kódu, a mnohé další. Od verze JRE 8 se číslování verzí JRE a JavaFX sjednotilo, v době vzniku této práce je aktuální verze JavaFX 8. Mezi hlavní novinky této verze patří rozšířená podpora 3D grafiky [1.7] [1.8].

1.2 Srovnání s knihovnou Swing

1.2.1 Výhody JavaFX

JavaFX přináší ve srovnání s knihovnou Swing mnoho novinek a vylepšení. Pomocí JavaFX je možné produkovat obsah, k jehož vytvoření by za použití knihovny Swing bylo nutné použít externí knihovny. Jedná se zejména o diagramy, 3D grafiku a audio/video obsah. JavaFX také umožňuje jednoduché vytváření animací. Za tímto účelem je v knihovně Swing možné použít třídu `Timer`. Knihovna zjednodušuje práci s obrázky (viz kapitola 5.1).

[1] http://javafxports.org/page/home
[2] http://www.adobe.com/cz/products/air.html
[3] http://www.microsoft.com/silverlight/

Pomocí tříd JavaFX lze vytvářet vazbu mezi daty (Data binding). Swing toto implicitně neumožňuje, nicméně použitím návrháře GUI v IDE NetBeans lze vazbu vytvořit pomocí tříd, nepatřících do knihovny Swing.

Díky tomu, že logika, vzhled a rozvržení GUI JavaFX aplikace může být rozděleno mezi Java třídy, CSS a FXML, je kód přehlednější. Je-li aplikace vyvíjena v týmu, tak jsou na sobě vývojáři méně závislí. Použitím CSS lze snadno upravovat vzhled aplikace (barva pozadí, textu, styl ohraničení, průhlednost Uzlu apod.).

V JavaFX aplikaci je možné nastavit obsluhu událostí vyvolaných dotykem na obrazovku (`GestureEvent`). Reagovat lze na gesto otočení, potažení, dotyk a zoom. Swing na tento typ událostí implicitně reagovat neumožňuje.

1.2.2 Výhody knihovny Swing

Vzhledem k tomu, že knihovna Swing je součásti JDK více než 15 let [1.9], existují mnohé knihovny třetích stran, které Swing používají, ale s JavaFX kompatibilní být nemusí [1.10]. Problém s kompatibilitou je možné řešit použitím SwingNode, což je JavaFX kontejner, do kterého je možné vkládat swingové komponenty (viz kapitola 2.2.3).

Knihovna Swing na sebe navázala rozsáhlou komunitu. V době vzniku této práce bylo na stránce Stack Overflow[4] přibližně padesát tisíc otázek se štítkem Swing oproti necelým osmi tisícům otázek se štítkem JavaFX. Lze tedy předpokládat, že při potýkání se s problémem během vývoje swingové aplikace bude snadnější najít (na Stack Overflow) řešení, než kdyby aplikace používala JavaFX. Nicméně pro JavaFX existují oficiální tutoriály, dostupné na webu Oracle, které pokrývají různé fáze vývoje JavaFX aplikace a které lze považovat za spolehlivý zdroj.

Při vývoji jednoduchých formulářových aplikací může být vhodnější použít knihovnu Swing. Současný vývoj FXML souboru a Řadiče může být časově náročnější než přepínání se mezi návrhářem a editorem kódu. Stejně tak rozšíření konzolové aplikace o prvek GUI může být rychleji provedeno pomocí knihovny Swing. V těchto případech se vývojář rozhodne spíše podle vlastní preference.

JavaFX do verze JDK 8u40 neobsahovala třídy pro vytváření dialogových oken, známých z knihovny Swing (dialogy pro potvrzení, vstup od uživatele apod.). Od této verze je možné k vytváření dialogů používat třídu `Dialog` a třídy od ní odvozené (`Alert`, `ChoiceDialog`, `TextInputDialog`) [1.15].

1.2.3 Různé přístupy

Ve Swingových aplikacích se vytváří třída odvozená od `JFrame`, která obvykle obsahuje statickou metodu `main()` a slouží tedy jako hlavní (spustitelná) třída. V JavaFX je hlavní třídou třída odvozená od `Application`. Přetížením metody `start()` této třídy se provádí prvotní nastavení aplikace (zejm. vytvoření grafu scény).

[4] http://www.stackoverflow.com/, Internetové fórum zabývající se problémy spojenými s vývojem aplikací

Swing pro nastavení rozložení komponentů uvnitř kontejneru Panel používá třídy implementující rozhraní `LayoutManager`. Takovou třídou je například `BorderLayout`. V JavaFX jsou pro různé typy rozložení definovány kontejnery, např. `BorderPane` odpovídá `BorderLayout`.

Knihovny přistupují různými způsoby k uchovávání dat zobrazených v tabulce (`JTable`, `TableView`) a seznamu (`JList`, `ListView`). V knihovně Swing se používají třídy `DefaultTableModel`, resp. `DefaultListModel`. V JavaFX jsou data uložena v kolekci `ObservableList`. Vytváří se třída, reprezentující řádek tabulky. Tato třída má JavaFX Vlastnosti namísto primitivních datových typů. Sloupce tabulky se s těmito vlastnostmi propojují, viz ukázková aplikace ListViewTableViewDemo (kap. 8.1.1). Tento postup je komplikovanější a časově náročnější, zato ale nabízí větší flexibilitu.

V JavaFX je možné používat různé přístupy pro obsluhu událostí. Mezi tyto události patří kliknutí na tlačítko, výběr položky ze seznamu apod. Pokud je GUI vytvářeno pomocí jazyka FXML, je možné použít atributy, které odkazují na metody, ve kterých je implementována obsluha události. Tento přístup je ukázán v příkladu 7.

Dále je také možné použít metody `setOn…()`. V příkladu 1 je vytvořena obsluha události „kliknutí myší na plochu Scény“. Provede se výpis souřadnic bodu, na který uživatel kliknul.

Příklad 1: EventHandler Scény

```
scene.setOnMouseClicked((MouseEvent event) -> {
    System.out.println("Kliknuto na souřadnice" +
        event.getX() + "; " + event.getY());
});
```

Pokud je aplikace vytvářena na platformě JDK verze 1.8 nebo novější, je možné použít lambda výraz. Lambda výrazy zkracují a zpřehledňují kód. V [1.8] je o lambda výrazech uvedeno: „Umožňují použít kód jako argument metody“. Lze je použít v místě, kde by se v dřívějších verzích JDK použila anonymní třída s jedinou metodou. V příkladu 1 je anonymní třída odvozená od třídy `EventHandler<MouseEvent>` s metodou `handle(MouseEvent event)` nahrazena lambda výrazem.

Posledním přístupem je použití metody `addEventHandler()`. Její použití může omezit duplicitu v kódu (jedna instance pro více prvků GUI). Tento přístup nejvíce odpovídá přístupu z knihovny Swing, ve které se pro prvky GUI registrují Listenery (`ActionListener` apod.) pomocí metod `add…Listener()`.

1.2.4 Graphics2D a GraphicsContext

Zatímco v knihovně Swing je možné kreslit na všechny komponenty (pomocí instance třídy `Graphics` daného komponentu), v JavaFX lze kreslit pouze na Plátno (`Canvas`). Swing ke kreslení používá třídu `Graphics`, resp. `Graphics2D`, JavaFX třídu `GraphicsContext`. Metody, poskytované těmito třídami, jsou obdobné.

`Graphics2D` má metody `draw()` a `fill()` pro kreslení obrysu, resp. výplně tvaru. Tento tvar je odvozený od třídy `java.awt.Shape`. `GraphicsContext` má pro každý tvar definovánu metodu (např. `strokeRect()` a `fillRect()` pro obdélník). Kreslení tvarů uložených v kolekci bude jednodušší při použití knihovny Swing.

1.3 Srovnání s WPF

WPF (Windows Presentation Foundation) je „nástroj firmy Microsoft, určený k vývoji webových aplikací a uživatelsky bohatých aplikací“ [1.11]. Jak vyplývá z názvu, je cílený na platformu Windows, oproti JavaFX, která je multiplatformní.

Stejně jako JavaFX aplikace, i WPF aplikace může být vložena na webovou stránku. Obdobně jako JavaFX, i WPF odděluje vzhled aplikace od logiky, a to použitím značkovacího jazyka XAML, který je založen na XML a který svým určením přibližně odpovídá jazyku FXML, používaným v JavaFX. Pro snadné vytváření uživatelského rozhraní slouží Microsoft Blend, obdobně jako Scene Builder pro JavaFX [1.12] [1.13].

1.4 Alternativní JVM programovací jazyky

Díky tomu, že je JavaFX psána v jazyku Java, je možné pro tvorbu JavaFX aplikací použít jazyky, které (stejně jako Java) běží na Java Virtual Machine (JVM). Motivací k využití alternativního JVM jazyka může být například lepší čitelnost kódu, kratší kód či osobní preference vývojáře [1.14].

Při použití alternativního JVM jazyka je třeba použít DSL (Domain Specific Language – doménově specifický jazyk). Je možné si jej představit jako obal či adaptér okolo JavaFX, vytvořený pro daný jazyk.

Pro jazyk Groovy byl vytvořen DSL GroovyFX[5]. Mezi výhody Groovy oproti Java (uvedené v [1.14]) patří mj. jednodušší tvorba anonymních tříd či pojmenovávání parametrů volaných metod, což usnadňuje čitelnost kódu. Navíc si vývojář při psaní kódu nemusí pamatovat pořadí parametrů.

Dalším DSL je ScalaFX[6], vytvořený pro jazyk Scala. Tento jazyk poskytuje stejné výhody jako Groovy, viz výše. Navíc nabízí funkcionality jako typová bezpečnost nebo přetěžování operátorů.

Visage[7] je jazyk, který vznikl specificky pro tvorbu GUI, a proto se při implementaci aplikace v tomto jazyce nepoužívá DSL. Kromě výše uvedených jazyků se dají využít i další, např. pro jazyk Clojure je vyvíjen DSL ClojureFX[8].

[5] http://groovyfx.org/
[6] http://www.scalafx.org/
[7] https://code.google.com/p/visage/
[8] https://github.com/zilti/clojurefx

2 TVORBA GRAFICKÉHO UŽIVATELSKÉHO ROZHRANÍ (GUI)

2.1 Vytvoření projektu v NetBeans

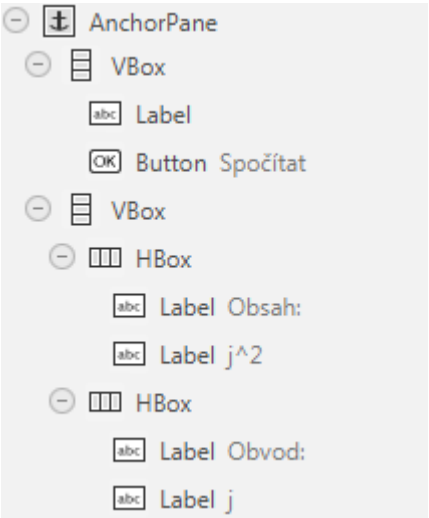
Vývojové prostředí NetBeans verze 8.0 nabízí v kategorii JavaFX vytvoření jednoho z pěti typů projektů. Především se jedná o JavaFX Application a JavaFX FXML Application. V JavaFX aplikaci jsou prvky uživatelského rozhraní definovány v kódu, zatímco JavaFX FXML aplikace používá pro tento účel jazyk FXML. V obou případech je po založení projektu vytvořena jednoduchá, ihned spustitelná aplikace typu „Hello World“, která je dále vyvíjena do zamýšlené podoby.

Postup založení projektu je ukázán v prvním a druhém video tutoriálu.

2.2 Třídy GUI

JavaFX obsahuje množství tříd pro tvorbu uživatelského rozhraní. Kromě tříd známých z knihovny Swing (tlačítka, textová pole atd.) se v balíčku javafx.scene nacházejí třídy takových prvků, jako jsou kontejnery pro média, dialogová okna umožňující výběr dne z kalendáře a mnoho dalších.

Jednotlivé prvky GUI (kontejnery, ovládací prvky, apod.) jsou umístěny ve stromové datové struktuře, nazývané *Scene graph* (graf scény) [2.1]. Při vytváření grafu scény se používá metoda `getChildren()` třídy `Parent`, která vrací seznam potomků. Tito potomci jsou instance tříd odvozených od třídy `Node` (Uzel). Přidáváním Uzlů do tohoto seznamu (metodou `add()` či `addAll()`) je vytvářena stromová struktura, viz obrázek 1. Tuto strukturu je možné vytvářet dvěma způsoby. Prvním z nich je „ruční“ psaní kódu (obvykle do spustitelné třídy), druhým je použití jazyka FXML.

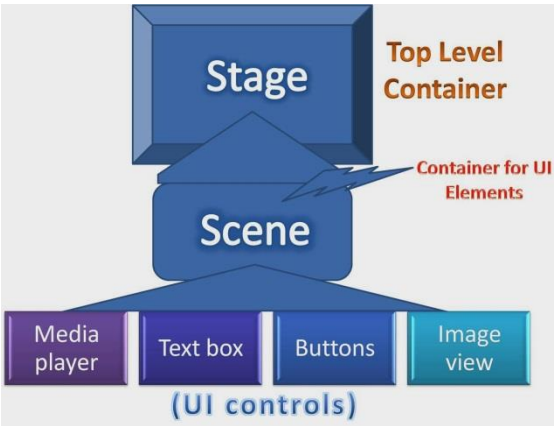


Obrázek 1: Znázornění grafu scény v programu Scene Builder
zdroj: vlastní

Následující podkapitoly jsou věnovány představení základních tříd.

2.2.1 Stage – Jeviště

`Stage` (Jeviště) je kontejnerem nejvyšší úrovně, viz obrázek 2. Na platformě Windows představuje okno aplikace. Platforma, na které aplikace běží, vytváří jednu instanci třídy `Stage`. Parametr `stage` metody `Application.start()` je referencí na tuto instanci. V kódu mohou být vytvářeny další instance, například při vytváření dialogových oken. Třída `Stage` je odvozena od třídy `Window` [2.2].



Obrázek 2: Umístění Jeviště v hierarchii prvků GUI
zdroj: http://www.javafxapps.in/tutorials/javafx_stage_scene/Picture1.jpg

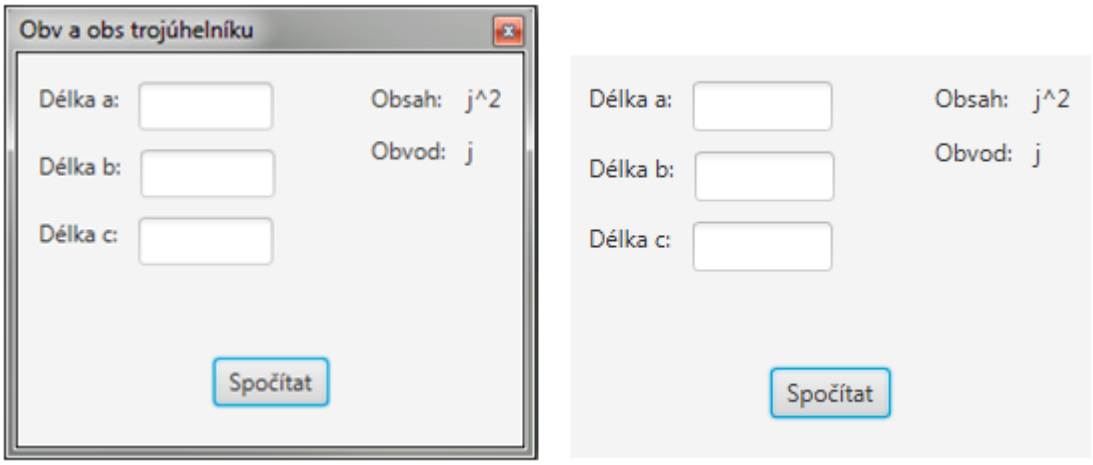
Scénu (`Scene`), která bude na `Stage` zobrazena, se určí pomocí metody `setScene()`, která jako parametr očekává právě referenci na Scénu. Jeviště tak může v jednu chvíli zobrazovat jen jednu Scénu. Pomocí metody `setScene()` je možné se mezi Scénami přepínat. Stage se zobrazí metodou `show()` a zavře metodou `close()`.

U Jeviště je možné definovat maximální a minimální rozměry (`setMinWidth()`, `setMaxWidth()`, `setMinHeight()`, `setMaxHeight()`) a pomocí metody `setResizable()` také to, zda je povoleno rozměry měnit. `Stage` je možné zobrazit na celou obrazovku použitím metody `setFullScreen()`, dále je možné `Stage` maximalizovat (`setMaximized()`) a minimalizovat (`setIconified()`). Titulek, který se zobrazuje v záhlaví aplikace a na hlavní liště operačního systému, lze nastavit metodou `setTitle()`.

Příklad 2: Přizpůsobení stage v metodě `start()`

```
public void start(Stage stage) throws Exception {
    Parent root = FXMLLoader.load(
      getClass().getResource("Layout.fxml"));
    Scene scene = new Scene(root, 300, 200);
    scene.getStylesheets().add(
      getClass().getResource("Style.css").toString());
    stage.setScene(scene);
    stage.setTitle("Obvod a obsah trojúhelníku");
    stage.setResizable(false);
    stage.initStyle(StageStyle.UTILITY);
    stage.show();
}
```

Vizuální styl Jeviště je možné nastavit metodou `initStyle()`, která jako parametr očekává hodnotu výčtového typu `StageStyle` a která musí být volána před tím, než je Jeviště zobrazeno. Výchozím stylem je `StageStyle.DECORATED`. Při použití tohoto stylu je pozadí `Stage` bílé a jsou použity implicitní dekorace platformy (zejména záhlaví aplikace). Dalšími styly jsou mj. `UNDECORATED` a `UTILITY`, jejichž vliv na styl Jeviště je demonstrován na obrázku 3.



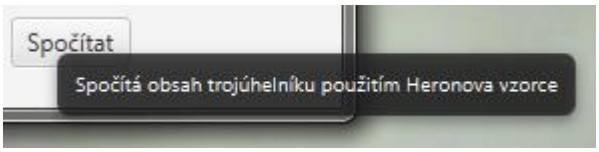
Obrázek 3: Styl Jeviště Utility (vlevo) a styl Undecorated (vpravo)
zdroj: vlastní

Mezi další třídy odvozené od třídy `Window` patří `Tooltip`, sloužící k zobrazení nápovědy či jiného textu, který se zobrazí ve chvíli, kdy uživatel umístí kurzor na Uzel, na kterém je `Tooltip` nainstalován [2.3].

Příklad 3: Tooltip

```
Tooltip tt = new Tooltip("Spočítá obsah trojúhelníku"
            + " použitím Heronova vzorce");
Tooltip.install(btnSpocitat, tt);
```

Výsledek:



2.2.2 Scene – Scéna

Všechny prvky stromové hierarchie Uzlů (grafu scény) jsou uloženy v kontejneru Scéna (`Scene`). V konstruktoru Scény se definuje kořenový Uzel této hierarchie. Obecně se jedná o Předka (`Parent`), konkrétně jsou nejčastěji používány Oblasti (`Region`, tedy např. `AnchorPane`). Také může být použita Skupina (`Group`). Pomocí přetížených konstruktorů třídy `Scene` je možné definovat velikost scény, barvu pozadí a další vlastnosti [2.4].

2.2.3 Node – Uzel

Abstraktní třída `Node` (Uzel) je předkem všech tříd, které jsou součástí stromové hierarchie GUI, tedy zejména kontejnerů a ovládacích prvků. Poskytuje metody pro transformace, které zahrnují [2.5]:

- Posun ve směru os x, y a z, a to pomocí metod `setTranslateX()`, `setTranslateY()` a `setTranslateZ()`.
- Otočení okolo středu uzlu za použití metody `setRotate()`
- Změna měřítka ve směru os x, y a z (metody `setScaleX()`, `setScaleY()` a `setScaleZ()`).

Finální umístění Uzlu se určí jako součet hodnot Vlastností (Property) `layoutX` a `translateX` pro osu x, pro osu Y obdobně. K nastavení hodnot `layoutX` a `layoutY` slouží metody `setLayoutX()`, resp. `setLayoutY()`.

To, zda bude Uzel viditelný či ne, je možné nastavit metodou `setVisible()`. Metoda `setDisabled()` slouží k uzamčení Uzlu. S uzamčeným Uzlem není možné provádět interakci (např. zamčené tlačítko není možné stisknout).

Ve třídě Uzel jsou také definovány metody `setId()`, `setStyle()` a `getStyleClass()`, umožňující stylování pomocí CSS. Tyto metody jsou rozebrány v kapitolách 2.4.1 a 2.4.2.

Snímek Uzlu je možné zachytit přetíženou metodou `snapshot()`. Jinými slovy je pomocí této metody možné vytvořit a uložit fotografii Uzlu. Snímek je uložen jako instance třídy `WritableImage`. Obdobné metody poskytuje také třída `Scene`.

V JavaFX verze 8 byl přidán Uzel `SwingNode`, který v JavaFX aplikaci umožňuje používat komponenty z knihovny Swing. Obsah, který se do `SwingNode` vloží, se určí pomocí metody `setContent()`, která jako parametr očekává třídu odvozenou od Swingové třídy `JComponent`. Mezi tyto třídy patří například `JPanel`, `JButton` a další [2.6].

Díky `SwingNode` je v JavaFX mimo jiné možné použít „komplexní Swingové komponenty třetích stran“ [2.7]. Swing a JavaFX mohou kooperovat i v obráceném vztahu, tedy do Swingové aplikace je možné vkládat JavaFX obsah, a to pomocí třídy `JFXPanel` [2.8].

2.2.4 Kontejnery – Region

Kontejnery slouží k uchovávání dalších Uzlů, kterými mohou být další kontejnery nebo listy (tlačítko, tvar, atp.). Kontejnery jsou odvozeny od třídy `Region` (Oblast, ta je odvozena od `Parent`). V závislosti na tom, jakým způsobem mají být Uzly do kontejneru přidávány a jak mají reagovat na změnu rozměrů, se volí jeden z následujících kontejnerů[2.9]:

- `Pane` je bázovou třídou pro všechny následující kontejnery. Při změně velikosti `Pane` implicitně nedochází k přemisťování Uzlů.

- `AnchorPane` „zakotví“ uzly ve specifikované vzdálenosti, která zůstává stejná i při změně rozměrů.
- `BorderPane` vkládá své potomky do jedné z pěti oblastí (top, left, center, right, bottom – nahoru, vlevo, doprostřed, vpravo a dolů).
- Uzly uložené ve `FlowPane` tvoří dynamický řádek či sloupec.
- `GridPane` vytváří mřížku uzlů. Jednotlivá políčka jsou určena indexem řádku a sloupce.
- `StackPane` vrství Uzly přes sebe (první vložený je nejníže).
- `TilePane` vytváří sloupec či řádek uzlů obdobně jako `BorderPane`, s tím rozdílem, že mění velikost Uzlu tak, aby vyplnil celé „políčko“, je-li to možné.

`ScrollPane` přidává na okraj oblasti posuvníky (horizontální a vertikální) v případě, že obsah v něm vnořený nelze zobrazit celý. Zvláštním druhem kontejnerů jsou `HBox` a `VBox`, které sdružují Uzly do řádku, resp. sloupce. Je-li to možné, Uzly umístěné v těchto kontejnerech si mezi sebou udržují proporcionálně stejnou vzdálenost. Používají se podobně jako Swingový `Box`.

K vizuálnímu rozčlenění GUI je dále možné použít následující kontejnery:

- Kontejner `Acordion` (Harmonika) obsahuje rozbalovací kontejnery typu `TiltedPane`.
- `SplitPane` dělí svoji plochu na dvě části (horizontálně či vertikálně), a to pomocí oddělovače (Separator), který může být pohyblivý.
- `TabPane` sdružuje záložky (`Tab`), mezi kterými je možné se přepínat.

Jednotlivé kontejnery jsou demonstrovány v aplikaci PanesDemo (kap. 8.1.3).

2.2.5 Ovladače – Control

Ovladače umožňují uživateli provádět interakci s aplikací. Jedná se o tlačítka, posuvníky, lišty menu apod. V JavaFX jsou ovladače odvozeny od abstraktní třídy `Control`, a to jak přímo, tak nepřímo. V příloze B se nachází výčet ovladačů se stručným popisem a uvedením odpovídající třídy z knihovny Swing [2.10]. Výčet vychází z tutoriálu [2.11], ve kterém je podrobně popsáno použití ovladačů.

2.2.6 Canvas – Plátno

`Canvas`, neboli Plátno, je Uzel, na jehož plochu je možné kreslit. Kreslení, spolu s dalšími, níže popsanými funkcemi, zajišťuje třída `GraphicsContext2D` (dále GC). Každé Plátno má jeden GC a referenci na něj je možné získat metodou `getGraphicsContext2D()` [2.12]. Metody nabízené třídou `Canvas`, resp. GC, jsou podobné JavaScriptovým funkcím používaným pro kreslení na element Canvas jazyka HTML5, což poskytuje částečnou znovu použitelnost kódu a usnadňuje zaučování vývojářů [2.13].

Pomocí GC je možné kreslit grafická primitiva, konkrétně úsečku - `strokeLine()`, lomenou čáru – `strokePolyline()`, oblouk - `strokeArc()`, elipsu a kružnici - `strokeOval()`, n-úhelník - `strokePolygon()`, obdélník – `strokeRect()`, obdélník se zaoblenými rohy – `strokeRoundRect()` a zobrazit je možné také text – `strokeText()`. S výjimkou úsečky je možné tato primitiva vyplnit barvou, a to pomocí metod `fill...()`. Na plátno je také možné nakreslit obrázek (či jeho část), a to přetíženou metodou `drawImage()` [2.14].

Barvu čáry je možné změnit metodou `setStroke()` a barvu výplně metodou `setFill()`. Obdélníkovou plochu je možné vyčistit metodou `clear()`. K uložení atributů jako jsou aktuální barva čáry, barva výplně apod. slouží metoda `save()`. Zpětné načtení je možné provést pomocí `restore()`.

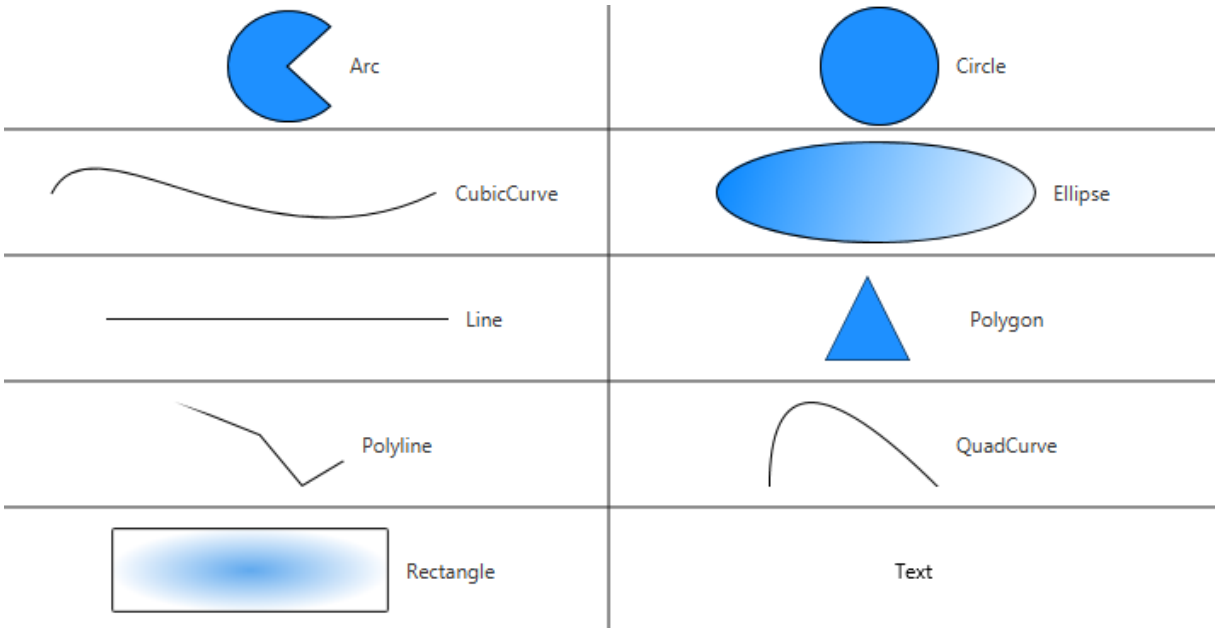
GC umožňuje vytvářet Cesty (`Path`). Cestu je možné vykreslit na Plátno (`stroke()`), vyplnit barvou (`fill()`) či podle ní provést ořez (`clip()`). Cesta může být tvořena úsečkami (`lineTo()`), oblouky (`arc()`, `arcTo()`), obdélníky (`rect()`), kvadratickou křivkou (`quadraticCurveTo()`) či kubickou Bézierovou křivkou (`bezierCurveTo()`). K zahájení cesty slouží metoda `beginPath()` a k uzavření metoda `closePath()`. Posun „pera“ je možné provést metodou `moveTo()`.

S Plátnem je také možné pracovat nízko úrovňově, tedy s jednotlivými pixely. Pro práci s pixely slouží třída `PixelWriter`, jejíž instanci je možné získat metodou `getPixelWriter()` třídy GC. Možné je upravovat každý pixel zvlášť (metoda `setPixel()`), nebo je možné upravit obdélníkovou oblast (přetížená metoda `setPixels()`).

Ukázková aplikace CanvasDemo demonstruje možnosti použití třídy `Canvas`.

2.2.7 Shape – Tvar

JavaFX poskytuje třídy pro reprezentaci geometrických tvarů. Tyto třídy jsou odvozeny od abstraktní třídy `Shape` (Tvar). Tvary jsou Uzly, lze je tedy vkládat do kontejnerů, provádět s nimi animace atd. Na obrázku 4 jsou představeny Tvary spolu s třídou, která slouží k jejich reprezentaci.



Obrázek 4: Geometrické tvary
zdroj: vlastní

Obrys a výplň Tvarů je možné upravit metodami `setStroke()` a `setFill()`. Obrys i výplň mohou být tvořeny jednou barvou či barevným přechodem (lineárním nebo radiálním). U obrysu je rovněž možné pomocí metody `setStrokeType()` nastavit, zda bude zasahovat do plochy tvaru, mimo plochu tvaru či vyváženě do i mimo plochu tvaru. Šířka obrysu se nastaví metodou `setStrokeWidth()` [2.15].

S Tvary je možné provádět množinové operace. Třída Shape poskytuje statické metody pro průnik – `intersect()`, rozdíl – `substract()` a sjednocení tvarů – `union()`. Návratovým typem těchto metod je `Shape`. Na obrázku 5 je zachycen tvar vzniklý sjednocením obdélníku a kruhu.

Obrázek 5: Tvar vzniklý sjednocením obdélníku a kruhu
zdroj: vlastní

Tvarem je také Cesta (`Path`). Entity, tvořící Cestu, byly popsány v předchozí kapitole. Cestu je možné vytvořit z řetězce, představujícího definici cesty podle SVG[9]. K vytvoření tohoto typu Cesty slouží třída `SVGPath`.

[9] http://www.w3schools.com/svg/svg_path.asp

2.2.8 Chart – Diagramy

Pomocí JavaFX je možné vytvářet koláčové diagramy (třída `PieChart`) a diagramy zobrazující v kartézské soustavě souřadnic (`XYChart`). Abstraktní třída `Chart`, od které jsou `PieChart` a `XYChart` odvozeny, poskytuje metody pro nastavení názvu diagramu (`setTitle()`) a metodu pro zobrazení/skrytí legendy (`setLegendVisible()`). Umístění názvu a legendy vzhledem k diagramu je možné změnit metodami `setTitleSide()`, resp. `setLegendSide()`. Změny dat zachycených v diagramech je možné animovat. K zapnutí či vypnutí animace slouží metoda `setAnimated()` [2.16].

PieChart

Jednotlivé výseče koláčového grafu jsou reprezentovány třídou `PieChart.Data`. Parametry konstruktoru této třídy jsou jméno výseče (`String`) a četnost (`double`). Výseče jsou uchovávány v kolekci `ObservableList<PieChart.Data>`. Asociování diagramu s touto kolekcí je možné provést v konstruktoru třídy `PieChart` nebo pomocí metody `setData()`.

Pomocí metody `setLabelsVisible()` je možné nastavit, zda budou zobrazovány popisky jednotlivých výsečí. Vzdálenost popisku od grafu lze upravit metodou `setLableLineLength()`. Dále je možné metodou `setStartAngle()` nastavit počáteční úhel (implicitně 0°) a metodou `setClockwise()` směr vkládání výsečí (ve směru či proti směru hodinových ručiček).

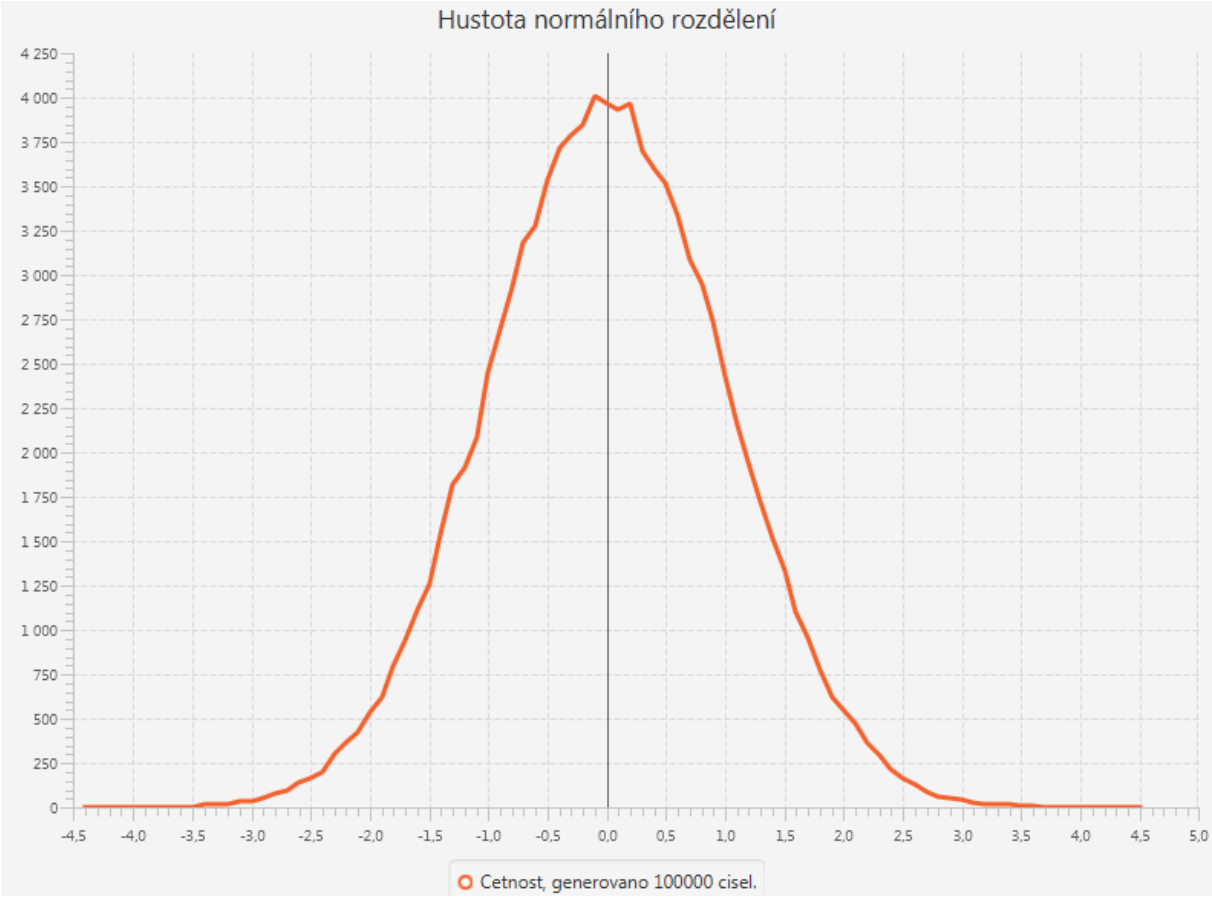
Koláčovým grafem se zabývá ukázková aplikace PieChartDemo (kap. 8.1.5).

XYChart

Od třídy `XYChart` jsou odvozeny další třídy, umožňující vytváření různých typů diagramů (grafů). Jedná se o plošný graf (`AreaChart`), sloupcový graf (`BarChart`), spojnicový graf (`LineChart`) a bodový graf (`ScatterChart`). Třídy poskytují metody specifické pro daný typ grafu, nicméně postup vytvoření všech typů je obdobný a vypadá následovně [2.17]:

1) Definice os (pomocí tříd `NumberAxis` a `ValueAxis`)
2) Definice grafu (osy jsou argumenty konstruktoru)
3) Definice datové řady (`XYChart.Series`) a bodů řady (`XYChart.Data`)
4) Vložení série (sérií) do grafu

V ukázkové aplikace XYChartDemo (kap. 8.1.6) je použit spojnicový graf k zobrazení hustoty normálního rozdělení pravděpodobnosti, viz obrázek 6.



Obrázek 6: LineChart zobrazující hustotu normálního rozdělení
zdroj: vlastní

2.3 Vytváření GUI pomocí jazyka FXML

Jak již bylo zmíněno v kapitole 1, jazyk FXML umožňuje separaci vzhledu a logiky aplikace (spolu s CSS). Jedná se o jazyk založený na XML, ve kterém jsou instance tříd reprezentovány elementy a jejich Vlastnosti atributy. Přestože výhody FXML oproti kódování do Java tříd vyniknou zejména u větších aplikací, je dobré FXML používat i u menších aplikací, vzhledem ke snadnější údržbě a testování. V souboru FXML je zapsán graf scény.

Při psaní této podkapitoly bylo čerpáno z tutoriálu [2.18] a ze stránky [2.19].

Příklad 4: Ukázka FXML kódu

```
<AnchorPane>
     <children>
          <Button layoutX="126" layoutY="90" text="klikni"/>
          <Label layoutX="126" layoutY="120"/>
     </children >
</AnchorPane>
```

Připojení FXML souboru se provádí pomocí třídy `FXMLLoader`, konkrétně pomocí její statické metody `load()`, a to obvykle v metodě `start()` hlavní třídy aplikace. Metoda `load()` vrací hierarchii objektů (strom), jehož kořen poslouží jako kořenový Uzel Scény.

Příklad 5: Běžné použití třídy FXMLLoader

```
@Override
public void start(Stage stage) throws Exception {
      Parent root = FXMLLoader.load(
            getClass().getResource("Layout.fxml"));

      Scene scene = new Scene(root);

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

FXML soubory pracují s Řadiči (`Controller`). Jedná se o třídu, ve které vývojář implementuje logiku GUI, obsahuje tedy zejména obsluhu událostí (jako je např. stisknutí tlačítka). Konkrétní Řadič se definuje jako atribut kořenového Uzlu v souboru FXML. Více souborů FXML může používat stejný řadič.

Příklad 6: Definování Řadiče

```
<BorderPane
      xmlns:fx="http://javafx.com/fxml/1"
      fx:controller="editorobrazku.LayoutController">
```

V Řadiči je obvykle potřeba odkazovat se na Uzel definovaný v souboru .fxml. Pro takovýto prvek je nutné v souboru .fxml definovat atribut fx:id. V Řadiči se poté deklaruje proměnná stejného typu, pojmenovaná stejně, jako je hodnota atributu fx:id. Této deklaraci předchází anotace @FXML. Alternativně je možné anotaci vypustit a použít přístupové právo public. Tato anotace se také využívá u metod, na které je v souboru .fxml odkazováno (obsluhy událostí).

Příklad 7: Anotace @FXML
FXML soubor

```
<children>
      <Button onAction="#handleButtonAction/>
      <Label fx:id="label"/>
</children >
```

Java třída (Řadič)

```
@FXML
private Label label;

@FXML
private void handleButtonAction(ActionEvent event) {
      label.setText("Hello World!");
}
```

Ačkoliv se v FXML kódu dá snadno orientovat (zejména díky hierarchické struktuře), může být pro vytváření rozvržení GUI vhodné použít program speciálně určený k úpravě FXML souborů – JavaFX Scene Builder (viz kapitola 2.5). Při ladění a testování se jako výhodnější jeví naopak editace FXML přímo (není třeba se přepínat mezi IDE a programem Scene Builder).

2.4 Úprava vzhledu aplikace použitím jazyka Cascading Styles Sheet (CSS)

V JavaFX lze pro definování vzhledu Uzlů použít stylový jazyk CSS. Podporována je verze CSS 2.1 spolu s některými identifikátory vlastností z CSS3. Identifikátory vlastností použitelné pro Uzly se vyznačují předponou -fx- (např. `-fx-background-color`). Přehled těch nejčastěji používaných je uveden v příloze C. Tato kapitola vychází z příručky [2.20].

2.4.1 Selektory

Stejně jako v HTML, tak i JavaFX je možné Uzlům nastavit třídu stylu (příp. třídy) nebo id. Id by mělo být unikátní, ale není to vyžadováno. Třídy uzlu jsou uchovávány v kolekci `styleClass`. Pro přidání třídy slouží metoda `add()`, viz příklad 8. Id je možné definovat metodou `setId()`. Třídy a Id se dají také definovat v souboru .fxml (atributy `styleClass` a `id`).

Příklad 8: Definování třídy a id v Java kódu

```
Button btn1 = new Button("Spustit");
Button btn2 = new Button("Konec");
btn1.setId("tlSpustit");
btn2.getStyleClass().add("tlacitko");
```

Příklad 9: Definování třídy a id v FXML

```
<Button id="tlSpustit" styleClass="tlactiko" />
```

Pro určité Uzly byly definovány výchozí stylové třídy, které je možné využívat jako selektory. Například Tlačítko (`Button`) má třídu button. Přehled tříd je uveden v příloze D.

2.4.2 Použití CSS

Pro prvky GUI je možné definovat styly přímo v kódu aplikace, v souboru .fxml nebo v souboru .css. První způsob spočívá ve využití metody `setStyle()` třídy `Node`. Tento způsob může způsobit znepřehlednění kódu a zkomplikovat jeho údržbu (obdobně jako použití atributu `style` v HTML kódu).

Příklad 10: Definice stylu tlačítka v Java kódu
`Button tlacitko = new Button("Klikni");` `tlacitko.setStyle("-fx-background-color: red");`
Výsledek:

Styl se dá také definovat v souboru .fxml, a to pomocí atributu `style`. Pro dosažení stejného výsledku jako v příkladu výše by kód vypadal následovně:

Příklad 11: Definice stylu tlačítka v FXML
`<Button style="-fx-background-color: red;" text="Klikni"/>`

Příklad 12: Definice stylu tlačítka v CSS

`.button {`
`  -fx-background-color: red;`
`}`

Z pohledu přehlednosti kódu a případného rozdělení zodpovědnosti mezi kodéra a designéra je vhodné definovat styly v souboru .css. Tento soubor může také definovat styly pro webovou stránku, na které je aplikace umístěna, což dále zjednodušuje údržbu. Odkazy na soubory .css jsou uchovávány v kolekci `stylesheets`.

Příklad 13: Připojení souboru .css v Java kódu

`Scene scene = new Scene(root);`
`scene.getStylesheets().add(`
`  getClass().getResource("Style.css").toString());`

V příkladu výše byl soubor .css připojen na instanci třídy Scene. Přestože pro scénu nelze definovat styly, je vhodné k ní připojit soubor .css, protože ostatní uzly jsou potomky této instance a styly tak budou aplikovány na nejvyšší možný počet uzlů.

Soubor .css lze také připojit v FXML, a to pomocí elementu `stylesheets`. Tento element musí obsahovat element URL s atributem `value`, ve kterém je definována adresa souboru .css.

Příklad 14: Připojení souboru .css v FXML

`<AnchorPane>`
`  <stylesheets>`
`  <URL value="@Style.css" />`
`  </stylesheets>`
`</AnchorPane>`

Po připojení souboru .css se styly v něm definované aplikují na samotný uzel a na jeho potomky. Styly je možné překrývat, tedy potomek může mít definován vlastní odkaz na soubor .css.

2.4.6 Unikátní vlastnosti

Pro některé uzly byly vytvořeny unikátní CSS vlastnosti, které vycházejí z funkce daného uzlu. Např. pro Posuvník (`Slider`) existuje identifikátor vlastnosti `-fx-show-tick-labels`, pomocí kterého se definuje, zda se zobrazí popisky pod posuvníkem.

2.4.4 Pseudotřídy

Styl uzlu v závislosti na jeho stavu může být definován pomocí pseudotříd. Např. pro odlišení zaškrtnutého `CheckBox`u lze použít pseudotřídu `selected`.

Příklad 15: Odlišení zaškrtnutého CheckBoxu

```
.check-box:selected {
    -fx-underline: true;
}
```

2.4.5 Stínování a barevné přechody

Uzlům lze pomocí vlastnosti `-fx-effect` nastavit stínování, a to jednak vnější (`DropShadow`) a jednak vnitřní (`InnerShadow`).

`DropShadow` má následující syntax:

`dropshadow(<blur-type>, <color>, <number>, <number>, <number>, <number>)`, kde blur-type je typ rozmazání, color je barva stínu a následující čísla určují poloměr jádra stínu, rozsah stínu, posun ve směru osy x a posun ve směru osy y (v tomto pořadí).

Příklad 16: Vnější stín

```
Circle c = new Circle(30);
c.setStyle("-fx-fill: blue; "
 + "-fx-effect: dropshadow(gaussian, black, 10, 0.5, 2, 3);");
```

Výsledek:

Syntax `InnerShadow` vypadá obdobně:

`innershadow(<blur-type>, <color>, <number>, <number>, <number>, <number>).`

Příklad 17: Vnitřní stín

```
Circle c = new Circle(30);
c.setStyle("-fx-fill: blue; "
 + "-fx-effect: innershadow(gaussian, black, 10, 0.5, 0, 0.5);");
```

Výsledek:

Datový typ `Paint`, který používají vlastnosti jako např. `-fx-background-color`, může představovat nejen barvu (definovanou ustáleným názvem, pomocí RGB(A) nebo HSV(A)), ale také barevný přechod, který může být lineární (Linear Gradient) nebo radiální (Radial Gradient). Pro vytvoření lineárního přechodu je nutné zadat dvě barvy, které tvoří přechod.

Příklad 18: Lineární přechod

```
Circle c = new Circle(30);
c.setStyle("-fx-fill: linear-gradient(black, blue);");
```

Výsledek:

Radiální přechod vyžaduje určení poloměru a dvou barev.

Příklad 19: Radiální přechod

```
Circle c = new Circle(30);
c.setStyle(
     "-fx-fill: radial-gradient(radius 75%, black, blue);");
```

Výsledek:

Pro lineární i radiální gradient je možné definovat řadu dalších parametrů. Kompletní syntax lze nalézt v [2.20].

2.4.7 Substruktury

Některé uzly, zejména ovládací prvky, mají substruktury, které se dají dále stylovat. Například substruktura již výše zmíněného posuvníku (Slider) se skládá z osy - axis (NumberAxis), dráhy posuvníku - track (Region) a ukazatele - thumb (Region).

Příklad 20: Definice stylu substruktury v souboru .css

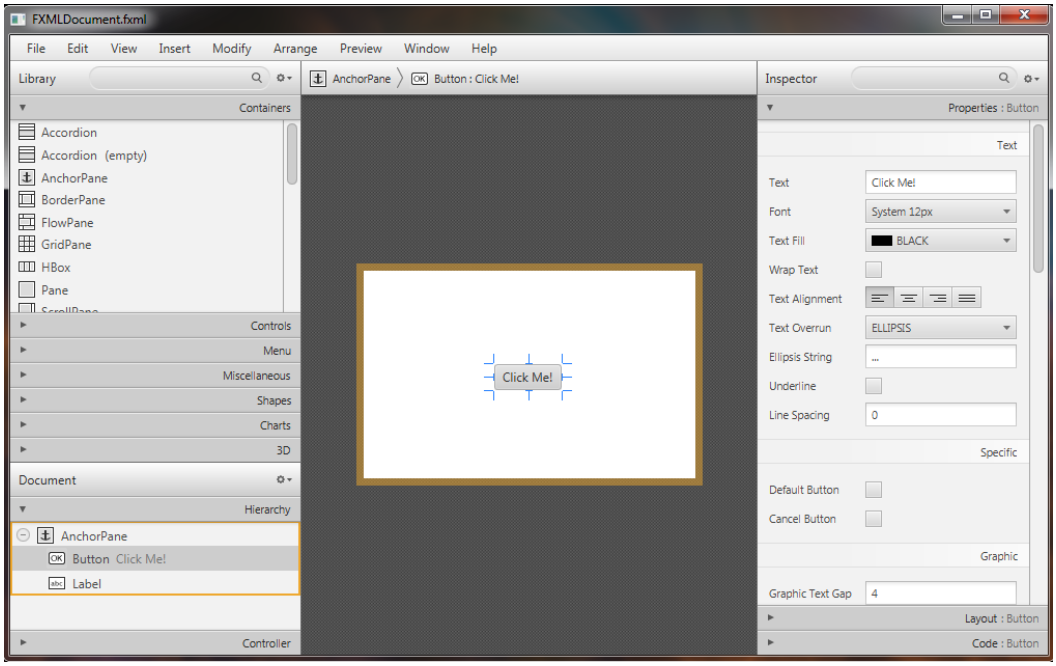
```
.slider .track {
    -fx-background-color: black;
    -fx-opacity: 0.75;
}
.slider .thumb {
    -fx-background-color: red;
    -fx-opacity: 0.7;
}
```

Výsledek:

2.5 Scene Builder

Scene Builder je oficiální nástroj společnosti Oracle pro intuitivní tvorbu GUI. Pracuje s FXML soubory a jednoduchým kliknutím a přetažením Uzlu ze seznamu umožňuje vytvářet komplexní uživatelská rozhraní (tedy obdobným způsobem jako např. Swing Designér v IDE NetBeans). Scene Builder umožňuje vygenerovat kostru Řadiče, což napomáhá odstranit chyby vzniklé kopírováním a také zkracuje čas nutný k vytvoření logiky Řadiče [2.21].

Scene Builder lze používat samostatně, je možné jej také propojit s IDE jako NetBeans nebo Eclipse. Postup propojení s NetBeans je ukázán ve video tutoriálu 2. Populární IDE InteliJ Idea má od verze 14 Scene Builder přímo zabudován [2.22].



Obrázek 7: Nástroj Scene Builder
zdroj: vlastní

Obrázek 7 zachycuje grafické uživatelské prostředí nástroje SceneBuilder. Nástroj je možné stáhnout na webu Oracle.

3 ANIMACE

JavaFX nabízí prostředky k jednoduchému vytváření komplexních animací. Třídy, které se animacemi zabývají, se nacházejí v balíčku javafx.animation. Při vytváření animace je možné využít dvou odlišných přístupů. Prvním z nich je vytvoření Časové osy (`Timeline`) a druhým je použití tříd odvozených od třídy `Transition` [3.1].

Ať už se využije první či druhý přístup, budou používány metody abstraktní třídy `Animation` (která je bázovou třídou jak pro `Timeline`, tak pro `Transition`). Přehled těch nejčastěji využívaných je uveden v následující tabulce.

Metoda	Popis
`play()`	Spustí animaci.
`setAutoReverse()`	Nastaví, zda po sobě následující cykly budou přehrány v opačném pořadí (např. cyklus 1 – přesun z A do B, cyklus 2 – z B do A).
`setCycleCount()`	Umožňuje nastavit počet cyklů (opakování) animace.
`setOnFinished()`	Nastavení události, která se vykoná po dokončení animace.

Praktické použití aplikací je ukázáno v aplikaci AnimationsDemo (kap. 8.1.7).

3.1 Timeline

Při vytváření animace za použití Časové osy (`Timeline`) jsou pro klíčové snímky (`KeyFrame`) definovány klíčové hodnoty (`KeyValue`), které představují hodnotu, kterou má určitá Vlastnost (např. x-ová souřadnice Uzlu) nabýt v definovaný čas (stanoveném v příslušném `KeyFrame`). Tento způsob je vhodný zejména pro situace, kdy je animováno více Uzlů, případně pokud se jedná o složitou animaci [3.2].

Příklad 21: Posun obdélníků pomocí Timeline

```
Timeline timeline = new Timeline(
      new KeyFrame(Duration.ZERO,
         new KeyValue(rect9.xProperty(), rect9.getX()),
         new KeyValue(rect10.xProperty(), rect10.getX())),
      new KeyFrame(Duration.millis(1000),
         new KeyValue(rect9.xProperty(), rect9.getX() + 50),
         new KeyValue(rect10.xProperty(), rect10.getX() - 50)),
      new KeyFrame(Duration.millis(1000),
         new KeyValue(rect9.xProperty(), rect9.getX() + 150),
         new KeyValue(rect10.xProperty(), rect10.getX() - 150)));
```

3.2 Transition

Třídy odvozené od abstraktní třídy `Transition` jsou vhodné ke tvorbě animací jednoho Uzlu (v některých případech Tvaru). Rozdělit by se daly jak na třídy, které pracující přímo s objektem animování (např. třída umožňující posun – `TranslateTransition` či třída umožňující rotaci – `RotateTransition`), tak na třídy pracujícími s již vytvořenými animacemi (např. `SequentialTransition`, která přehrává posloupnost animací) [3.3].

Třídy pracují na obdobném principu – v konstruktoru či pomocí setterů je nastavena délka přehrávání animace, cílový uzel, výchozí a konečná hodnota. Využití těchto tříd může být časově méně náročné než vytváření časové osy a má také mnohdy za následek kratší a přehlednější kód.

Příklad 22: Posun obdélníků pomocí TranslateTransition

```
TranslateTransition tt1 = new TranslateTransition(
     Duration.millis(1000), rect9);
tt1.setByX(50);
TranslateTransition tt2 = new TranslateTransition(
     Duration.millis(1000), rect10);
tt2.setByX(-50);
```

V následujících kapitolách budou pro každý druh animace popsány klíčové metody. Kompletní přehled metod je možné vyhledat v oficiální dokumentaci.

3.2.1 Fade

FadeTransition plynule mění průhlednost Uzlu [3.4].

Metoda	Popis
`FadeTransition()`	Konstruktor, parametry jsou délka trvání animace a Uzel, který má být animován.
`setFromValue()`	Počáteční hodnota (v případě nedefinování se použije aktuální průhlednost Uzlu).
`setToValue()`	Konečná hodnota.

3.2.2 Fill

FillTransition plynule mění barvu výplně tvaru (objekt třídy Shape) [3.5].

Metoda	Popis
`FillTransition()`	Konstruktor, parametry jsou délka trvání animace, tvar, který má být animován, počáteční barva (fromValue) a konečná barva (toValue).

3.2.3 Parallel

ParallelTransition slouží k souběžnému přehrávání více animací [3.6].

Metoda	Popis
`ParallelTransition()`	Konstruktor, paramtery jsou Uzel, který má být animován a seznam animací.

3.2.4 Path

PathTransition slouží k posunu Uzlu po určité cestě (křivce) [3.7].

Metoda	Popis
`PathTransition()`	Konstruktor, paramtery jsou délka trvání animace, cesta, po které bude Uzel posouván a Uzel, který má být animován.
`setOrientation()`	Nastavení orientace Uzlu v průběhu animace. Implicitně se orientace nemění, při použití statické proměnné OrientationType. ORTHOGONAL_TO_TANGENT bude Uzel kolmý na tangentu křivky.

3.2.5 Pause

PauseTransition je možné využít k pozastavení sekvence animací či k vykonání akce se zpožděním (využitím metody onFinished) [3.8].

Metoda	Popis
`PauseTransition()`	Konstruktor, paramterem je délka pauzy.

3.2.6 Rotate

RotateTransition provádí plynulou rotaci Uzlu [3.9].

Metoda	Popis
`RotateTransition()`	Konstruktor, paramtery jsou délka trvání animace a Uzel, který má být animován.
`setByAngle()`	Nastavení velikosti úhlu, ve kterém bude provedena rotace.
`setFromAngle()`	Nastavení počátečního úhlu.
`setToAngle()`	Nastavení konečného úhlu.
`setAxis()`	Nastavení osy, kolem které bude rotace probíhat. Osy X, Y a Z jsou definovány ve třídě Rotate.

3.2.7 Scale

ScaleTransition slouží k plynulé změně měřítka zobrazení Uzlu [3.10].

Metoda	Popis
`ScaleTransition()`	Konstruktor, paramtery jsou délka trvání animace a Uzel, který má být animován.
`setByX()`	Zvětšení/zmenšení měřítka (osa X). Pro osy Y a Z existují obdobné metody.
`setFromX()`	Počáteční měřítko (osa X). Pro osy Y a Z existují obdobné metody.
`setToX()`	Konečné měřítko (osa X). Pro osy Y a Z existují obdobné metody.

3.2.8 Sequential

SequentialTransition slouží k postupnému přehrávání více animací [3.11].

Metoda	Popis
`SequentialTransition ()`	Konstruktor, paramtery jsou Uzel, který má být animován a seznam animací.

3.2.9 Stroke

StrokeTransition plynule mění barvu ohraničení Tvaru (objekt třídy Shape) [3.12].

Metoda	Popis
`StrokeTransition()`	Konstruktor, paramtery jsou délka trvání animace, tvar, který má být animován, počáteční barva (fromValue) a konečná barva (toValue).

3.2.10 Translate

TranslateTransition provádí plynulý posun Uzlu [3.13].

Metoda	Popis
`TranslateTransition()`	Konstruktor, paramtery jsou délka trvání animace a Uzel, který má být animován.
`setByX()`	Nastavení vzdálenosti posunutí (osa X). Pro osy Y a Z existují obdobné metody.
`setFromX()`	Nastavení počátečního posunutí (osa X). Pro osy Y a Z existují obdobné metody.
`setToX()`	Nastavení konečného posunutí (osa X). Pro osy Y a Z existují obdobné metody.

4 DATA BINDING, KOLEKCE

4.1 Data binding

Data binding (svazování dat) je funkčnost, pomocí které je možné vytvářet vazbu mezi datovými složkami tříd. Pro lepší představu je uveden následující příklad. Mějme třídu Trojúhelník, s Vlastnostmi reprezentujícími strany trojúhelníku (viz příklady 23, 24 a 25). Pomocí třídy `NumberBinding` je možné vytvořit vazbu, představující obvod trojúhelníku jako součet jednotlivých stran, tedy jako součet hodnot Vlastností. Při změně délky jedné ze stran se automaticky přepočítá obvod. Obvod může být svázán s prvkem GUI, například vlastností `textProperty` třídy `TextField`. Tento příklad byl použit jako základ ukázkové aplikace Triangle (kap. 8.1.8).

Pro práci s Data binding jsou důležité JavaFX Vlastnosti (JavaFX Properties). Jedná se o obalové třídy primitivních datových typů (a také kolekcí), které mohou být svazovány s dalšími Vlastnostmi a které umožňují reagovat na změny obalené hodnoty (pomocí Listenerů). Například pro celá čísla (`int`) byla vytvořena třída `SimpleIntegerProperty`. Podle konvence se pro Vlastnost vytváří getter a setter pro obalenou hodnotu a metoda vracející samotnou Vlastnost. IDE NetBeans umožňuje generování kódu definujícího Vlastnost a tyto metody, což je časově výhodné oproti ručnímu psaní.

V JavaFX je možné využít dva přístupy při vytváření vazby, a to buď vysokoúrovňový či nízkoúrovňový.

4.1.1 Vysokoúrovňový přístup

Vysokoúrovňový přístup může být představován buďto použitím statických metod třídy `Bindings` , nebo použitím tzv. *Fluent API* či kombinací těchto technik [4.1].

V případě, že je vytvářena vazba mezi čísly, je vazba představována třídou implementující rozhraní `NumberBinding` (konkrétně se může jednat např. o `IntegerBinding`). K dispozici jsou také třídy pro vazbu řetězců (`StringBinding`), objektů (`ObjectBinding`) a kolekcí.

Pokud je použito *Fluent API*, tak se vztah mezi svazovanými hodnotami určí pomocí metody třídy implementující rozhraní `NumberExpression` (v případě, že vázané hodnouty jsou čísla – pokud by byly řetězce, jednalo by se o rozhraní `StringExpression`). Případně se vztah určí pomocí vhodné metody třídy `Bindings`. Pokud by vazbu představoval součet hodnot, jako v příkladech 23 a 24, použili bychom metodu `add()`.

Příklad 23: Fluent API
IntegerProperty strA = new SimpleIntegerProperty(3); IntegerProperty strB = new SimpleIntegerProperty(4); IntegerProperty strC = new SimpleIntegerProperty(5); NumberBinding obvod = strA.add(strB.add(strC));

Příklad 24: Třída Bindings
IntegerProperty strA = new SimpleIntegerProperty(3); IntegerProperty strB = new SimpleIntegerProperty(4); IntegerProperty strC = new SimpleIntegerProperty(5); NumberBinding obvod = Bindings.add(strA, strB).add(strC);

Pokud v příkladech výše provedeme změnu délky strany trojúhelníku, změna se projeví i v jeho obvodu, ale z důvodu, že je používáno tzv. líné vyhodnocování (lazy evaluation), dojde ke změně až ve chvíli, kdy bude program s hodnotou dále pracovat. Objektu představujícímu vazbu (v případě výše je jím `obvod`) je možné přidat `InvalidationListener`, v jehož metodě `invalidated()` je možné definovat chování programu v případě, že dojde ke změně vázané proměnné. Pokud má Vlastnost zaregistrován `ChangeListener`, tak se změna projeví hned (eager evaluation).

4.1.1 Nízkoúrovňový přístup

Nízkoúrovňový přístup nabízí větší flexibilitu a výkonnost na úkor složitějšího kódu. Přístup spočívá ve vytvoření potomka jedné ze tříd, implementujících rozhraní `Binding`, zavolání metody `bind()` této třídy (parametry jsou Vlastnosti, které tvoří vazbu) a přetížení její metody `computeValue()` [4.2]. V metodě `computeValue()` může být komplexní kód, což poskytuje výše zmíněnou flexibilitu. Tyto třídy jsou definovány jednak pro primitivní typy (`Boolean-`, `Double-`, `Float-`, `Integer-` a `LongBinding`), jednak pro kolekce (`List-`, `Map-`, `SetBinding`) a také pro objekty (`ObjectBinding`) a řetězce (`StringBinding`).

Příklad 25: Nízkoúrovňový Data binding (výpočet obsahu trojúhelníku)

```
IntegerProperty strA = new SimpleIntegerProperty(3);
IntegerProperty strB = new SimpleIntegerProperty(4);
IntegerProperty strC = new SimpleIntegerProperty(5);

DoubleBinding obsah = new DoubleBinding() {
      { super.bind(strA, strB, strC) ;}

      @Override
      protected double computeValue() {
            double obs;
            double s = obvod.doubleValue() / 2.;
            obs = Math.sqrt(
                  s * (s - strA.get()) * (s - strB.get()) *
                  (s - strC.get()));
            obs = Math.round(1000 * obs)/1000.;
            return obs;
      }
};
```

Vazbu je také možné vytvořit pomocí metody `bind()`, kterou poskytují třídy implementující rozhraní `Property<T>`. Takto je například možné svázat vlastnost kontejneru ImageView `rotateProperty` s vlastností posuvníku (`Slider`) `valueProperty`. Výsledkem je, že se při změně hodnoty posuvníku bude otáčet i ImageView a obrázek v něm vložený. Vazba může být i obousměrná, a to při použití metody `bindBidirectional()`.

4.2 Balíček FXCollections

JavaFX poskytuje rozšíření standardního frameworku kolekcí (Collection Framework) programovacího jazyka Java. Do tohoto frameworku patří často používané třídy, implementující rozhraní `List` (např. `ArrayList`) nebo `Map` (např. `HashMap`). Významnou je také třída `Collections`, která mj. poskytuje statické metody pro práci s kolekcemi (např. *`max()`* pro vyhledání prvku s nejvyšší hodnotou). Rozšiřující třídy a rozhraní se nacházejí v balíčku javafx.collections [4.3].

Třída `FXCollections` poskytuje obdobné prostředky pro práci s kolekcemi jako třída `java.util.Collections`, která již byla zmíněna výše. Navíc obsahuje definice obalových tříd, které implementují rozhraní `ObservableList`. Tyto třídy umožňují přidávání (a odebírání) Listenerů, pomocí kterých je možné reagovat na změny prováděné s obalenou kolekcí (kterou může být např. `ArrayList`). Obdobně rozhraní `ObservableMap` umožňuje reakci na změny asociativního pole (např. `HashMap`). Jako Listener pro `ObservableList` slouží rozhraní `ListChangeListener` a pro `ObservableMap` rozhraní `MapChangeListener`. V kódu vytváříme anonymní třídy, implemetnující tato rozhraní. Alternativně je možné použít lambda výraz.

K vytvoření instancí obalových tříd se používají statické metody třídy `FXCollections`. Jednou z nich je přetížená metoda `FXCollections.`*`observableArrayList()`*, viz příklad 26.

Příklad 26: Vytvoření ObservableList a nastavení Listeneru

```
ObservableList<String> olist
           = FXCollections.observableArrayList();
olist.addListener(
           (ListChangeListener.Change<? extends String> c) -> {
                while (c.next()) {
                     if (c.wasAdded()) {
                          System.out.println("Přidán prvek");
                     }
                     if (c.wasRemoved()) {
                          System.out.println("Odebrán prvek");
                     }
                }
           });
```

Provedená změna je reprezentována třídou `Change`. Pomocí metod této třídy je mj. možné zjistit, zda došlo k přidaní, odebrání či nahrazení prvku (metody `wasAdded()`, `wasRemoved()`, `wasReplaced()`). Metody třídy `Change` je nutné volat v cyklu, viz příklad výše.

Mnohé prvky GUI používají tyto obalové třídy. Např. řádky tabulky (`TableView`) nebo položky `ComboBoxu` jsou uchovány v kolekci `ObservableList`. Příkladem použití `ObservableMap` může být reprezentace metadat mediálního souboru ve třídě `Media`.

5 OBRÁZKY, AUDIO A VIDEO

5.1 Obrázky

Třídy pro práci s obrázky jsou definovány v balíčku javafx.scene.image. Jedná se zejména o třídy `Image` a `ImageView`, které jsou popsány v následujících podkapitolách.

5.1.1 Image

Třída `Image` slouží k načtení a reprezentaci obrázku. Obrázek může být načten jak z lokálního souboru, tak z webu. V konstruktoru je také možné definovat nové rozměry obrázku, zda bude zachován poměr stran, zda se použije algoritmus pro vyhlazení a zda má být načítání provedeno na pozadí [5.1].

Příklad 27: Načtení ze souboru, umístěného v balíčku imview.img

```
//načte obrázek, nastaví velikost 300x200px
//nebude zachován poměr stran a obr. nebude vyhlazen
//načítání neproběhne na pozadí

Image img = new Image(
     "/imview/img/Desert.jpg", 300, 200, false, false, false);
```

Příklad 28: Načtení z webu

```
Image img =
     new Image("http://www.upce.cz/vvr/odborna-komunita.jpg");
```

Třída také nabízí metodu `getPixelReader()`, která vrací objekt `PixelReader`, pomocí kterého je možné číst jednotlivé pixely obrázku. Pro úpravu pixelů je nutné místo `Image` použít odvozenou třídu `WritableImage` a její metodu `getPixelWriter()` [5.2].

Použitím obrázků se zabývá ukázková aplikace ImageEditor (součást projektu PerformanceComparison, viz kap. 8.1.2).

5.1.2 ImageView

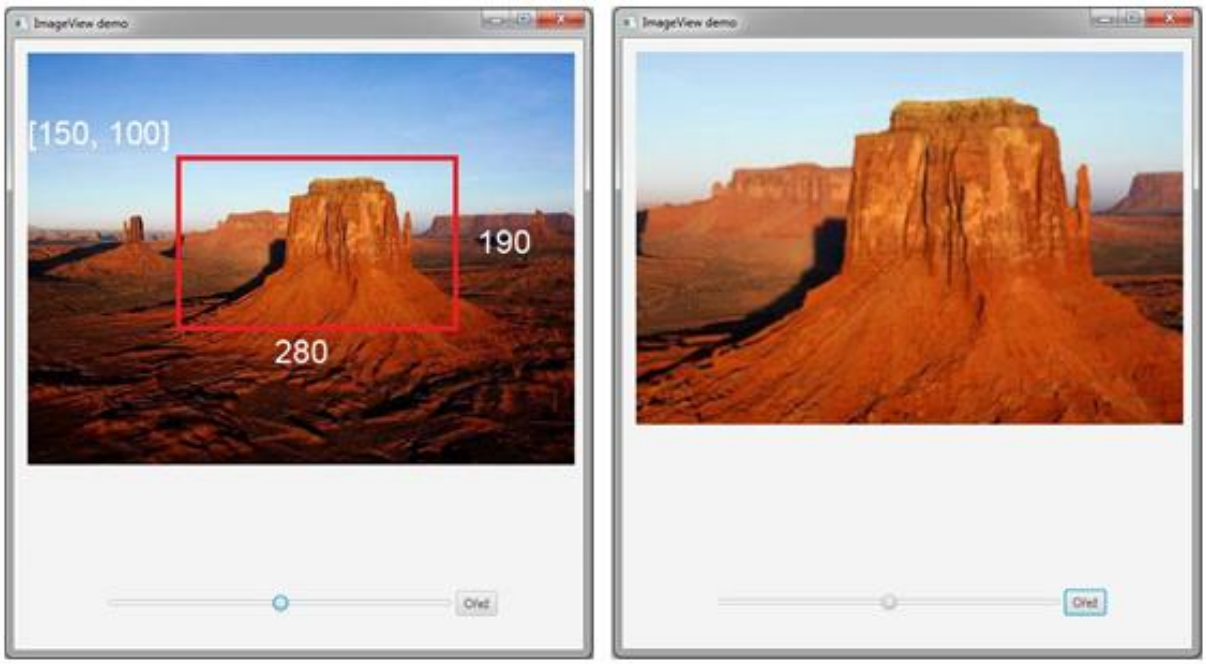
`ImageView` je třída odvozená od třídy `Node`, která slouží k zobrazování obrázků. Dále nabízí metody pro otáčení a ořez obrázku, které jsou popsány níže. Cílový obrázek, reprezentovaný třídou `Image`, je možné načíst v konstruktoru nebo pomocí metody `setImage()`.

Obrázky je možné otáčet, a to použitím metody `setRotate()`, která jako parametr očekává úhel otočení. Kladné hodnoty značí otočení ve směru hodinových ručiček. Metoda `setRotate()` je definována v třídě Uzel (`Node`) a lze ji tedy použít k otočení jakéhokoliv Uzlu.

K ořezání obrázku slouží metoda `setViewport()`, která jako parametr očekává oblast, podle které se bude ořezávat, definovanou pomocí třídy Rectangle2D. V konstruktoru této třídy určíme x-ovou a y-ovou souřadnici levého horního rohu a následně šířku a výšku oblasti v pixelech. Souřadnice se stejně jako délka a šířka vztahují na zdrojový obrázek. Zdrojový obrázek zůstává neořezán. Na obrázku 8 je obrázek v aplikaci zobrazen v měřítku 1:1.

Příklad 29: Oříznutí obrázku

```
Rectangle2D orez = new Rectangle2D(150, 100, 280, 190);
imView.setViewport(orez);
```



Obrázek 8: Ořez obrázku
zdroj: vlastní, editovaný obrázek je součástí OS Windows 7

Výšku a šířku obrázku je možné definovat pomocí metod `fitHeight()`, resp. `fitWidth()`. Pokud budou obě hodnoty nastaveny na 0 (nula), bude obrázek vykreslen v originální velikosti. To, zda bude zachován poměr výšky a šířky, je možné nastavit pomocí metody `setPreserveRatio()`. Argument true značí, že poměr bude zachován.

ImageView také umožňuje zvolit, zda má být k zobrazení obrázku použit rychlý nebo časově náročnější algoritmus. Slouží k tomu metoda `setSmooth()`. Po zadání argumentu true bude použit algoritmus vykreslování s vyššími časovými nároky, obrázek ale bude vykreslen ve vyšší kvalitě [5.3].

5.2 Audio a video

Třídy, zaměřené na práci s médii, se nacházejí v balíčku javafx.media. Jedná se zejména o třídy `Media`, `MediaPlayer`, `MediaView` a `AudioClip`.

5.2.1 Media

Obdobně jako třída `Image` slouží k reprezentaci obrázků, tak třída `Media` reprezentuje mediální soubor. Soubor může být jak lokální, tak umístěný na webu. Podporovány jsou soubory ve formátu MP3, AIFF a WAV pro audio a soubory FLV a MPEG-4 pro video. Na formát souboru jsou kladeny další požadavky, které je možné dohledat v tutoriálu [5.4].

Cesta k souboru (URL) je specifikována v parametru konstruktoru. URL je možné získat voláním metody `getResource()` třídy `Class`, jak je ukázáno na příkladu 30 [5.5].

Příklad 30: Načtení souboru kal.mp3, umístěného v balíčku media.sound

```
URL u = getClass().getResource("/media/sound/kal.mp3");
Media m = new Media(u.toString());
```

Třída poskytuje metodu k získání metadat `getMetadata()`. Metoda vrací `ObservableMap` metadat, v případě .mp3 se může jednat o autora, název alba, rok vydání apod. Délku souboru v sekundách je možné získat pomocí metody `getDuration()`. K zjištění rozlišení videosouboru slouží metody `getWidth()` - šířka a `getHeight()` – výška [5.6].

5.2.2 MediaPlayer

Pro přehrání mediálního souboru slouží třída `MediaPlayer`. Při vytváření instance třídy `MediaPlayer` je nutné zadat referenci na objekt třídy `Media`. Po vytvoření již není možné soubor změnit, to znamená, že pro každý soubor je vytvořena instance třídy `Media` a instance třídy `MediaPlayer` [5.7].

Třída poskytuje klíčové funkcionality, mezi které patří spuštění přehrávání (metoda `play()`), pozastavení (`pause()`) a zastavení (`stop()`). Převíjení vpřed a zpět se provádí metodou `seek()`.

Příklad 31: Převinutí o 5 s vpřed

```
@FXML
private void vpred5s(ActionEvent event.) {
      mp.seek(mp.getCurrentTime().add(new Duration(5000)));
}
```

Rovněž je možné upravovat hlasitost, a to metodou `setVolume()`, která očekává jako argument hodnotu od 0.0 (nulová hlasitost) po 1.0 (maximální hlasitost). Pro ztlumení hlasitosti, resp. nastavení hlasitosti na úroveň před ztlumením, slouží metoda `setMute()` – argument true hlasitost ztlumí, false nastaví na hlasitost před ztlumením.

Dále je možné definovat procedury, které se vykonají v případě, že nastane událost (tou může být konec přehrávaného souboru, pozastavení přehrávání, chyba při přehrávání a další). Procedura je definována v metodě `run()` třídy implementující rozhraní `Runnable`. Instance této třídy se použije jako argument metody `setOn...()` (např. `setOnEndOfMedia()`, viz příklad 32). Alternativně je možné použít lambda výraz.

Příklad 32: Definice procedury, provedené po dokončení přehrávání

```
mp.setOnEndOfMedia(() -> {
      System.out.println("Konec souboru");
});
```

Stav přehrávače je možné získat pomocí metody `getStatus()`. Stavy jsou definovány ve výčtovém typu `Status`, a jedná se například o stav `PLAYING`, `PAUSED` apod. V závislosti na stavu může program provádět různé reakce na stejnou událost. Je tak například možné použít jedno tlačítko pro spuštění a pozastavení přehrávání, což demonstruje příklad 33.

Příklad 33: Použití stavu přehrávače

```
@FXML
private void handleButtonAction(ActionEvent event) {
      if(mp.getStatus() == MediaPlayer.Status.PLAYING) {
            mp.pause();
            button.setText(">");
      } else {
            mp.play();
            button.setText("||");
      }
}
```

5.2.3 MediaView

Třída `MediaView` slouží pro zobrazení videosouboru (který je reprezentován třídou Media a jehož přehrávání zajišťuje `MediaPlayer`). Pokud aplikace přehrává pouze audio soubory, není nutné `MediaView` používat.

Klíčovou je metoda `setMediaPlayer()`, pomocí které se definuje přehrávač. Následující příklad demonstruje vytvoření instancí tříd `Media` a `MediaPlayer` a připojení `MediaPlayer` k `MediaView`.

Příklad 34: Inicializace Media, MediaPlayer a připojení k MediaView

```
Media m = new Media("http://www.mediacollege.com/" +
            "video-gallery/testclips/20051210-w50s.flv");
MediaPlayer mp = new MediaPlayer(m);
mediaView.setMediaPlayer(mp);
```

Zdroj videosouboru: http://www.mediacollege.com/

`MediaView` nabízí obdobné metody jako `ImageView`. Jedná se o `setViewport()` pro ořez, `setFitWidth()` a `setFitHeight()` pro definici šířky a výšky videa, `setPreserveRatio()` pro určení, zda bude zachován poměr stran a `setSmooth()` pro výběr algoritmu vykreslování (rychlý nebo pomalejší, vykreslující ve vyšší kvalitě) [5.8].

5.2.4 AudioClip

Audio soubory lze přehrávat také pomocí třídy `AudioClip`. Její použití vyžaduje méně režie (nepoužívá `MediaPlayer`) a je vhodné ji použít v případech, kdy se od přehrávání očekává minimální interakce s uživatelem (upozornění, efekty apod.). Specifikace zdrojového souboru se provádí obdobně jako u třídy `Media`, tedy zadáním zřetězené URL jako argumentu konstruktoru.

Příklad 35: Načtení souboru kala.mp3, umístěného v balíčku media.sound

```
URL resource = getClass().getResource("/media/sound/kala.mp3");
AudioClip clip = new AudioClip(resource.toString());
```

Přehrávaný soubor není možné pozastavit, stejně tak není možné posunout aktuální čas. Tentýž `AudioClip` může být v jednu chvíli spuštěn vícekrát. Pro přehrávání jsou poskytovány metody `play()` a `stop()`. Pro ovládání hlasitosti lze použít metody `setVolume()` a `setMute()`, které již byly popsány u třídy `MediaPlayer`. Přehrávaný soubor je načten celý (na rozdíl od `MediaPlayer`, který načítá po částech), což má za následek vyšší paměťové nároky [5.9].

Práce s médii je demonstrována v aplikaci MediaDemo (kap. 8.1.9).

6 3D SCÉNY

JavaFX aplikace mohou obsahovat 3D grafiku. Vzhledem k obsáhlosti tohoto tématu jsou v této kapitole pouze nastíněny postupy vytváření 3D scény. Problematice se více věnuje tutoriál [6.1], ze kterého bylo při tvorbě této kapitoly čerpáno. Vytvořením tělesa, prací s kamerou a se světlem se zabývá ukázková aplikace 3DDemo (kap. 8.1.10).

6.1 Tvorba a použití 3D těles

Při tvorbě 3D scény je možné použít předdefinovaná 3D tělesa, a to kvádr (třída `Box`), kouli (`Sphere`) a válec (`Cylinder`). Vlastní těleso je možné vytvořit pomocí třídy `TriangleMesh`. Materiál tělesa je reprezentován třídou `PhongMaterial`. Pomocí této třídy je možné nastavit difúzní a lesklou barvu, aplikovat na těleso texturu apod. [6.2]

Do aplikace je také možné importovat tělesa vytvořená v programech specializujících se na 3D modelování. Jedním z těchto programů je Autodesk Maya[10]. Problematikou importu těles se zabývá projekt InteractiveMash[11].

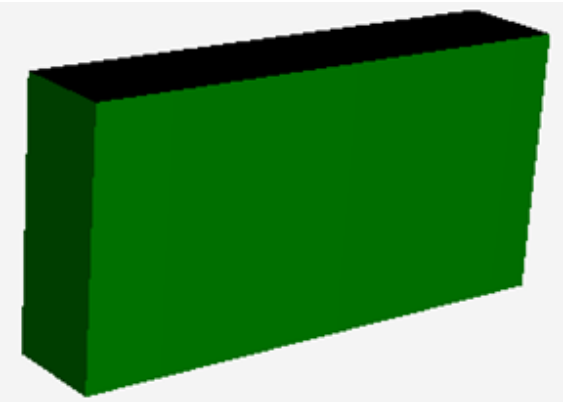
3D tělesa je vhodné umístit do kontejneru `SubScene`, u kterého je mj. možné upravit systém souřadnic. V konstruktoru SubScény je také možné zvolit, zda bude použito vyhlazování (antialiasing). SubScéna je snímána Kamerou (instance třídy `PerspectiveCamera`). V jednu chvíli může SubScénu snímat jen jedna Kamera. Změnit Kameru je možné metodou `setCamera()`. SubScénu je možné použít pro rozčlenění aplikace do částí, snímaných různými kamerami. Jednou z těchto částí může být průhledový displej (HUD).

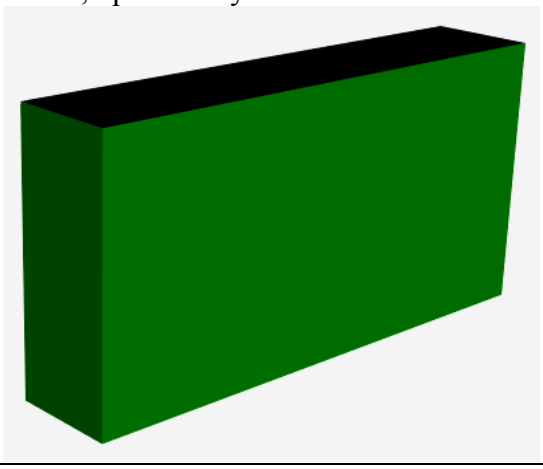
Příklad 36: Vytvoření kvádru

```
PhongMaterial boxMat = new PhongMaterial(Color.GREEN);
sphereMat.setSpecularColor(Color.DARKGREEN);

Box b = new Box(10, 5, 2);
b.setMaterial(boxMat);
```

Výsledek po přidání do SubScény, vlevo bez vyhlazování, vpravo s vyhlazováním:



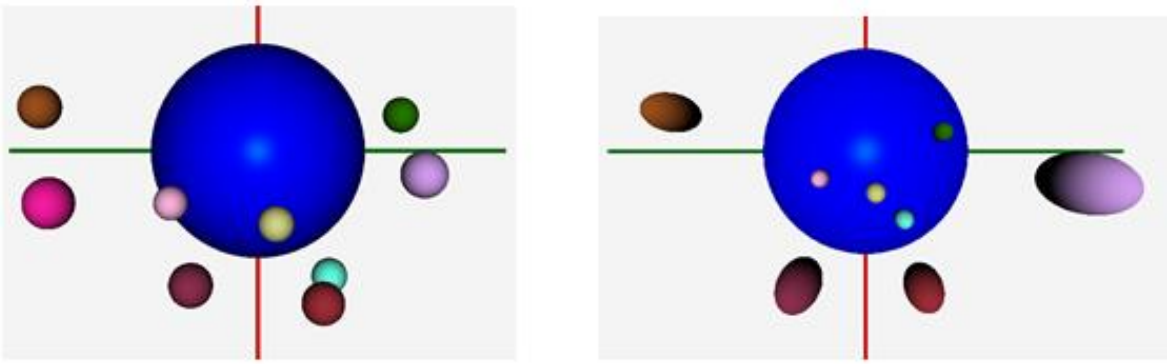


[10] http://www.autodesk.com/products/maya/overview
[11] http://www.interactivemesh.org/index.html

6.2 Práce s kamerou

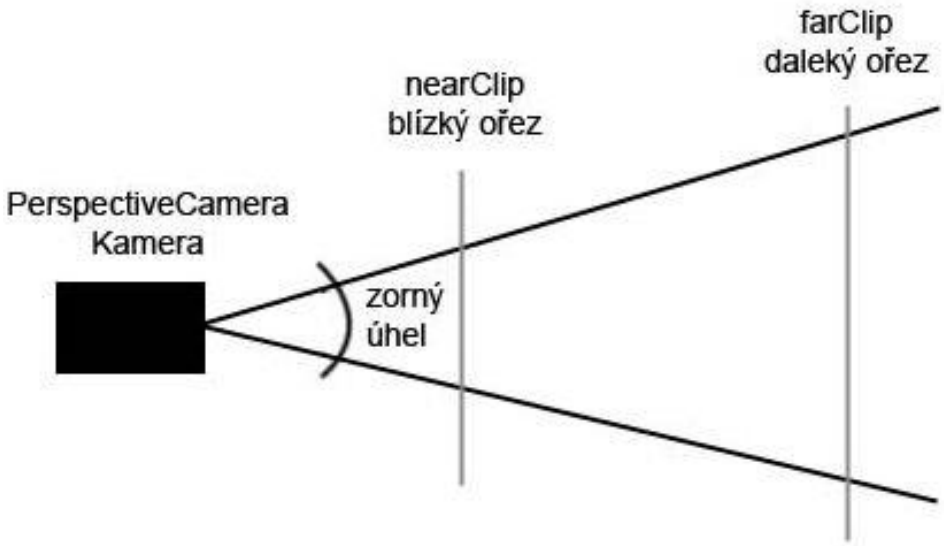
Kameru je možné posouvat a otáčet ve směru os x, y a z. Slouží k tomu metody třídy `Node`, od které je `PerspectiveCamera` (nepřímo) odvozená. Jedná se o metody známe z kapitoly 2.2.3, tedy `setTranslate...()` a `setRotate()`. Osu, podle které je otáčení prováděno, je možné definovat pomocí metody `setRotationAxis()`.

Dále je možné nastavit zorný úhel kamery (ve stupních), a to metodou `setFiledOfView()`. Vliv změny zorného úhlu je zachycen na obrázku 9. Defaultní zorný úhel je 30°.



Obrázek 9: Zorný úhel kamery 30° (vlevo) a 120° (vpravo)
zdroj: vlastní

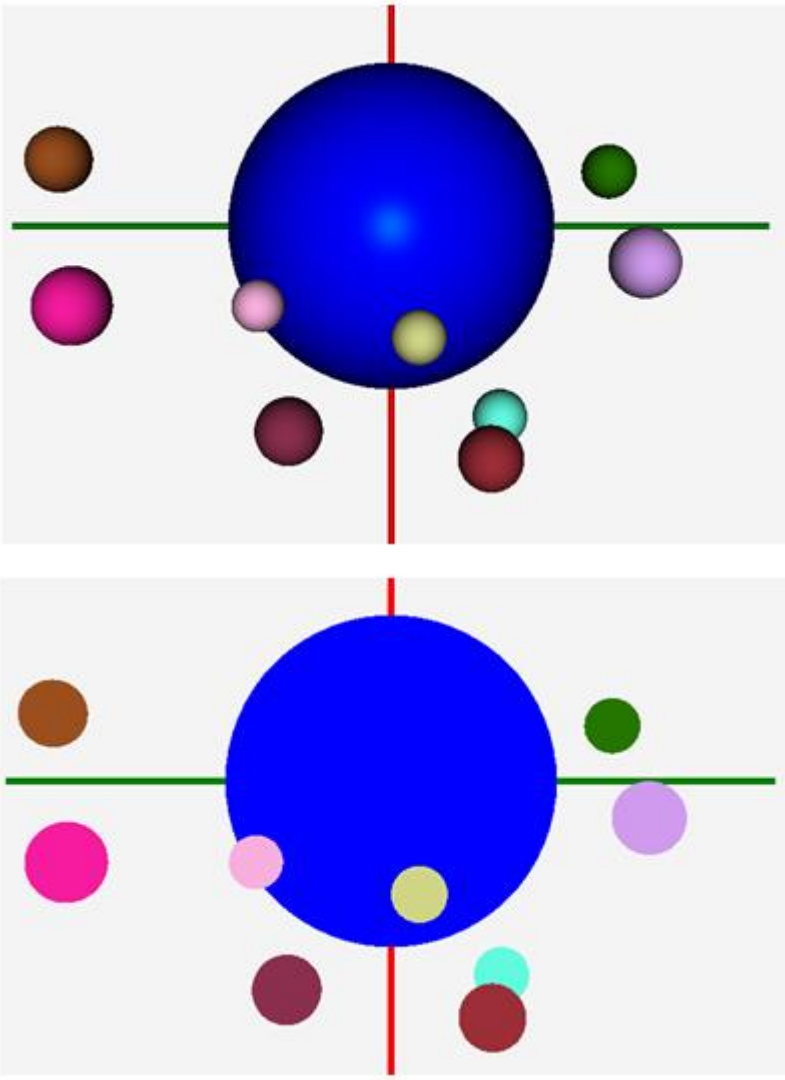
Kamerou jsou snímána tělesa ve vzdálenosti větší než je hodnota Vlastnosti `nearClip` a menší než je hodnota Vlastnosti `farClip`, viz obrázek 10. Tyto vzdálenosti je možné nastavovat metodami `setNearClip()` a `setFarClip()`.



Obrázek 10: Část Scény snímaná kamerou
zdroj: vlastní

6.3 Práce se světlem

Zdroje světla jsou reprezentovány Uzly `AmbientLight` (ambientní světlo) nebo `PointLight` (světlo vycházející z bodu). Světla se do Scény přidávají stejným způsobem jako 3D tělesa, tedy přidáním do seznamu potomků některého z Uzlů Scény, obvykle kořenového.



Obrázek 11: Scéna osvětlená bodovým zdrojem (nahoře) a ambientním světlem (dole)
zdroj: vlastní

Barvu světla je možné nastavit v konstruktoru nebo metodou `setColor()` a světlo je také možné vypnout nebo zapnout metodou `setLightOn()`. Zdroj bodového světla je možné posouvat jako všechny ostatní Uzly metodami `setTranslate()`. Osvětlení scény je modelováno pomocí Phongova osvětlovacího modelu[12].

[12] http://www.is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=6364

7 UMÍSTĚNÍ APLIKACE NA WEB

JavaFX aplikaci je možné umístit na web, a to bez nutnosti provádět do aplikace jakékoliv zásahy. Aplikace může být přímo vložena do webové stránky (embedded application), nebo může být ze stránky spuštěna (web start). Pokud je aplikace spuštěna z webu, tak je možné ji používat i po opuštění webové stránky.

7.1 Deployment Toolkit

Pro usnadnění umístění aplikace na web byla společností Oracle vytvořena knihovna *Deployment Toolkit*, jejíž API je napsáno v jazyce JavaScript. Knihovna mj. obstarává vložení potřebných HTML značek (tagů) a provádí kontrolu, zda uživatel splňuje požadavky pro spuštění aplikace, jako jsou podporovaná platforma, aktuální verze JRE atd. V případě zastaralé verze, nebo pokud nemá uživatel JRE vůbec nainstalováno, nabídne uživateli instalaci potřebných prostředků.

Pro programátory nabízí knihovna množství funkcí, pomocí kterých je možné konfigurovat umístění aplikace dle vlastních potřeb (specifikace požadavků na platformu, ošetření chyb apod.), nicméně pro základní použití je dostatečné použít soubory a kód generovaný vývojovým prostředím, viz níže [7.1].

7.2 Proces umístění aplikace na web

Vývojové prostředí NetBeans dále usnadňuje proces umístění aplikace na web. V kořenovém adresáři projektu se nachází složka dist, která obsahuje soubory pro distribuci aplikace. Nalézají se zde soubory .jar a .jnlp, dále také ukázková webová stránka, psaná v jazyce HTML, obsahující odkaz na spuštění aplikace z webu a také samotnou vloženou aplikaci. V kódu jsou volány funkce knihovny *Deployment Toolkit*. Ve složce dist se také nachází složka web-files, obsahující podpůrné soubory, a to zejména soubor dtjava.js. Tento soubor je také možné získat z webu Oracle (java.com/js/dtjava.js), kde se vždy nachází jeho nejnovější verze.

Při použití NetBeans tedy postačí překopírovat obsah složky dist na webový server. Ukázkový .html soubor jako takový nemá smysl kopírovat. Části kódu v něm zapsané se zkopírují do kódu stránky, na které aplikace poběží. Je-li aplikace spouštěna z webu, je postup následující:

1) Vložení odkazu na soubor dtjava.js
2) Překopírování funkce `launchApplication()`
3) Kopie odkazu na soubor .jnlp

V případě, že se aplikace vkládá do webové stránky, vypadá postup následovně:

1) Vložení odkazu na soubor dtjava.js
2) Překopírování funkce `javafxEmbed()` a volání metody `addOnloadCallback()`, zajišťující vložení aplikace až po načtení stránky

3) Umístění elementu div s id javafx-app-placeholder na místo, na které bude aplikace vložena

Kód vygenerovaný IDE poskytuje pouze základní funkčnost, jeho další úpravou je možné nastavit rozlišení aplikace, ošetřit chyby apod., viz [7.1]. Na stránku je samozřejmě možné umístit více aplikací (přidáním dalšího volání funkce dtjava.embed()) a také je možné kombinovat spouštění z webu a vkládání aplikace přímo na web.

7.3 Certifikáty

Z bezpečnostních důvodů se u aplikací umístěných na web používají certifikáty. Pokud aplikace nemá certifikát, nebo je podepsána pro uživatele neznámou autoritou, bude při jejím spuštění zobrazeno varování (viz obrázek 12). Takovéto aplikace jsou od verze Javy 7u51 implicitně blokovány [7.2]. Aplikacím je možné udělit výjimku, postup je popsán na webu Oracle. Uživatel musí přijmout bezpečnostní riziko, pokud chce aplikaci spustit.

Obrázek 12: Zpráva o zablokování nedůvěryhodné aplikace
zdroj: vlastní

Pokud bude aplikace používána ve známé komunitě, tedy například na intranetových stránkách školy či firmy, je možné použít „self-signed“ certifikát, tedy certifikát podepsaný tvůrcem aplikace. Vytvoření a distribuce „self-signed“ certifikátu je popsáno v článku [7.3]. Certifikát pro veřejně dostupné aplikace je možné získat od certifikační autority, kterou je například První certifikační autorita, a.s [7.4].

IDE NetBeans umožňuje v dialogovém okně Project Properties nastavit podepisování certifikátu. Pokud aplikace požaduje neomezený přístup, bude pro ni vytvořen certifikát (implicitně „self-signed“). Aplikace s omezeným přístupem nemůže přistupovat k lokálním souborům uživatele a vztahují se na ni další omezení, uvedené v dokumentaci [7.5].

8 PRAKTICKÁ ČÁST

8.1 Ukázkové aplikace

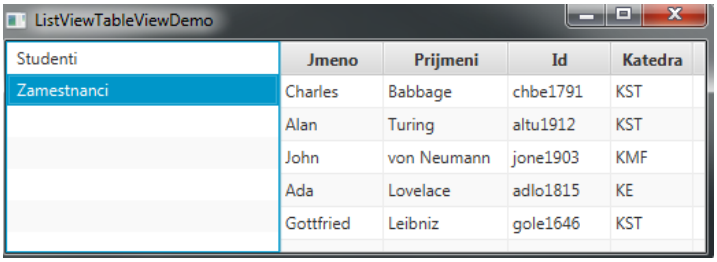
Praktickou část této bakalářské práce tvoří především ukázkové JavaFX aplikace. Po prostudování těchto aplikací bude čtenář schopen samostatně vyvíjet vlastní aplikace. Aplikace jsou dostupné na přiloženém DVD jako projekty vývojového prostředí NetBeans a byly také umístěny jako embedded aplikace do webové aplikace (viz kapitola 8.2). Před spuštěním aplikací je doporučeno se ujistit, že nainstalovaná verze JRE a JDK je aktuální.

V tomto přehledu je uveden odkaz na web, kde je aplikace vystavena. Aplikace jsou také stručně představeny. Způsob implementace je blíže popsán ve zdrojovém kódu formou komentářů. Aplikace jsou uvedeny v tom pořadí, v jakém na ně bylo odkazováno v předchozích kapitolách.

8.1.1 ListViewTableViewDemo

Odkaz na web

V této aplikaci je ukázán způsob vytváření a zaplnění Ovladačů Tabulka (`TableView`) a Seznam (`ListView`). Aplikace je ilustrována na obrázku 13. V aplikaci je vytvořeno spojení mezi sloupcem Tabulky a JavaFX Vlastností třídy, která reprezentuje řádky Tabulky. Také je použit Listener, pomocí kterého je obsloužena událost „změna vybrané položky Seznamu“.



Obrázek 13: Aplikace ListViewTableViewDemo
zdroj: vlastní

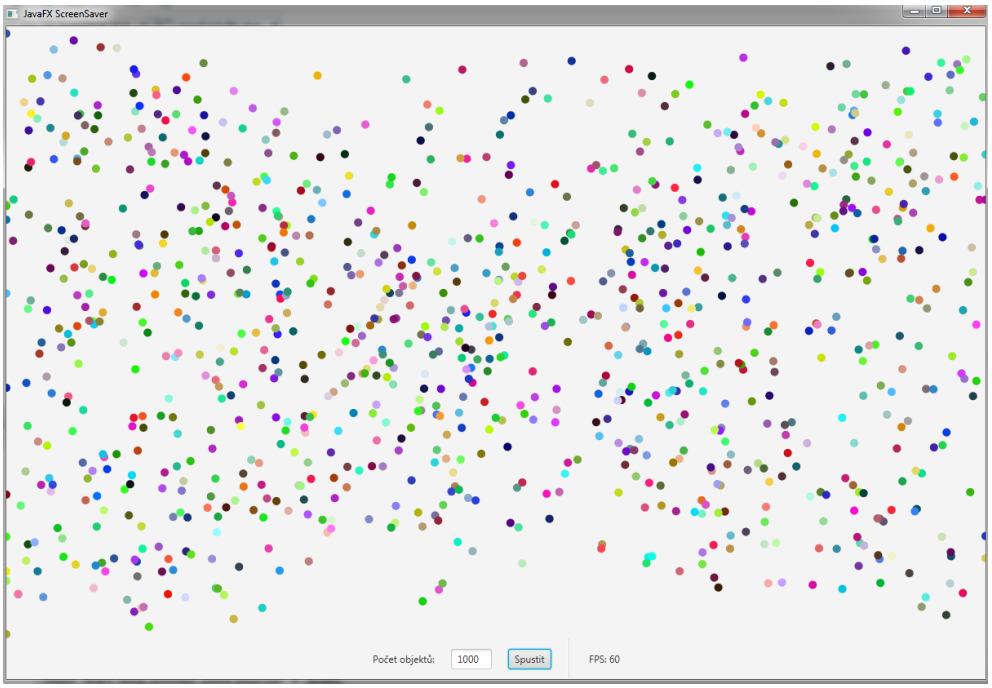
8.1.2 PerformanceComparison

Odkaz na web (JavaFX varianty)

Projekty PerformanceComparisonJavaFX a PerformanceComparisonSwing se skládají z šesti aplikací, které byly použity pro provedení srovnání JavaFX a Swing po stránce výkonu. Aplikace jsou členěny do balíčků.

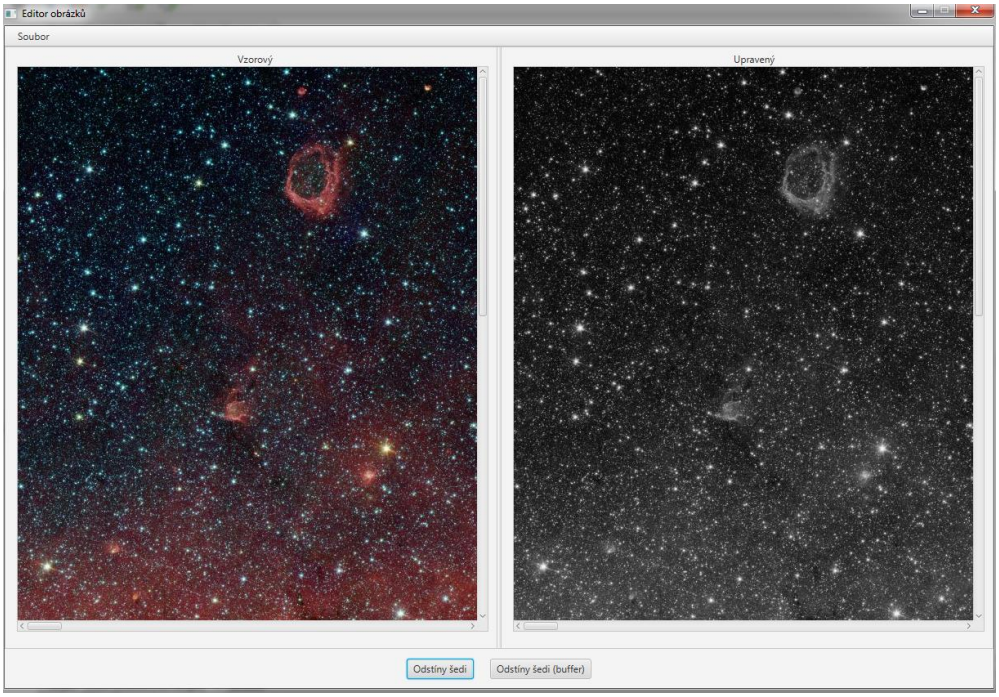
V aplikacích ScreenSaver je vytvořen uživatelem zadaný počet kruhů. Tyto kruhy jsou poté přesouvány od jednoho kraje aplikace ke druhému. V JavaFX aplikaci (obrázek 14) byla pro vytvoření animace použita třída `TranslateTransition`. Ve Swingové aplikaci byla vytvořena třída `Circle` s jednoduchým fyzikálním modelem. Swingová aplikace byla inspirována aplikací BouncingBall[13].

[13] https://github.com/I82Much/BouncingBall



Obrázek 14: JavaFX varianta aplikace ScreenSaver
zdroj: vlastní

Aplikace ImageEditor umožňují uživateli načíst obrázek a převést jej do odstínů šedi. Převedení může být provedeno s nebo bez použití bufferu. Za použití bufferu trvá převedení kratší čas, zvýší se ale paměťová náročnost aplikace. V JavaFX aplikaci (obrázek 15) je obrázek reprezentován třídou `Image`, resp. `WritableImage`. Ve Swingové aplikaci je použita třída `BufferedImage`. Obrázek je vykreslován na `JPanel`.



Obrázek 15: JavaFX varianta aplikace ImageEditor
zdroj: vlastní

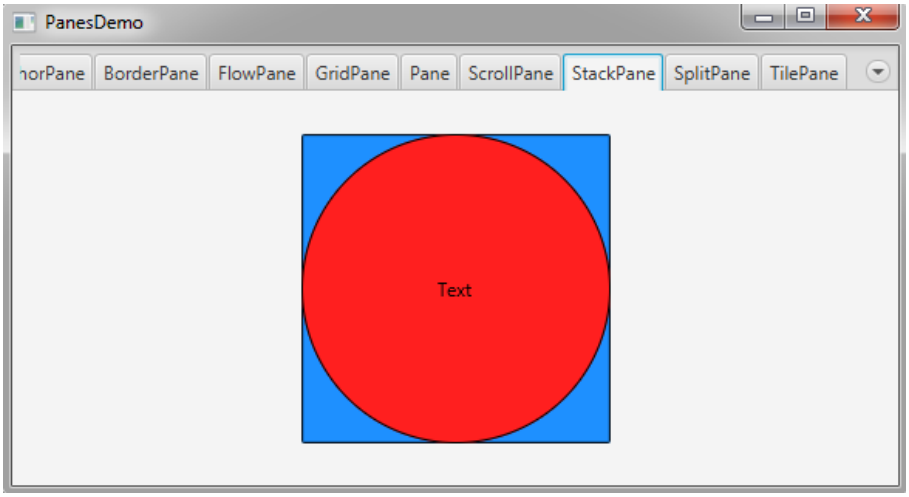
Pomocí aplikací Drawing (obrázek 16) byl měřen čas nutný k nakreslení různého počtu Bézierových křivek. Počet vykreslovaných křivek volí uživatel. V případě JavaFX aplikace bylo kresleno na `Canvas`, ve Swingové aplikace na `JPanel`. V obou aplikacích byl použit stejný algoritmus, přestože jak v JavaFX, tak v knihovně Swing existují metody pro kreslení těchto křivek. Důvodem bylo objektivnější srovnání. Po vykreslení aplikace zobrazí, kolik milisekund vykreslení trvalo.

Obrázek 16: JavaFX varianta aplikace Drawing
zdroj: vlastní

8.1.3 PanesDemo

Odkaz na web

V této aplikaci jsou demonstrovány všechny kontejnery, které je v JavaFX možné použít. Je možné sledovat, jak který kontejner reaguje na změnu velikosti okna aplikace. Do kontejnerů je také vložena řada Ovladačů a Tvarů, čímž aplikace pomáhá vytvářet přehled o Uzlech, které je možné při tvorbě GUI použít. GUI aplikace je zobrazeno na obrázku 17.

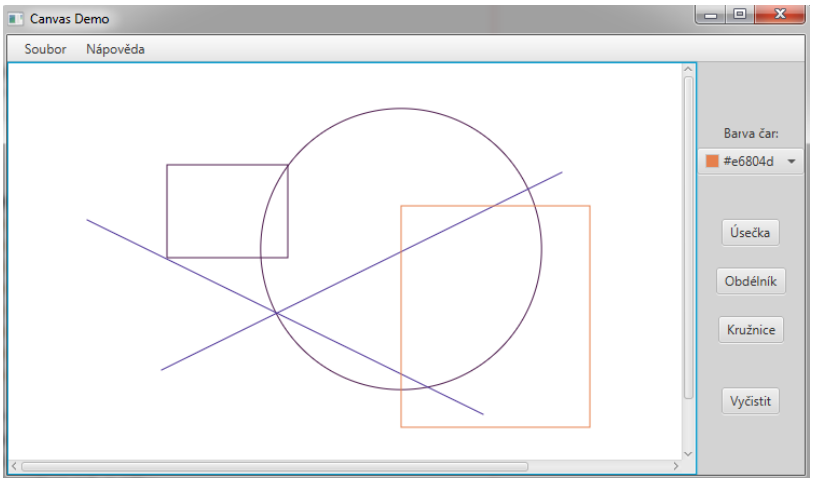


Obrázek 17: Aplikace PanesDemo
zdroj: vlastní

8.1.4 CanvasDemo

Odkaz na web

Aplikace CanvasDemo (obrázek 18) umožňuje uživateli kreslit grafická primitiva (úsečky, kružnice a obdélníky) na Plátno (`Canvas`). Rovněž je možné vykreslit Cestu (`Path`). Poté, co uživatel vybere počáteční bod, je vykreslován náhled. Uživatel také může měnit barvu čáry (pomocí Ovladače `ColorPicker`) a může nakreslená primitiva smazat. Aplikace obsahuje nápovědu ve formě dialogového okna. Dialogové okno bylo vytvořeno pomocí třídy `Alert`. Tato třída byla přidána do JDK ve verzi 1.8u40. Pro spuštění aplikace je tedy nutné mít dostatečně aktuální JRE.

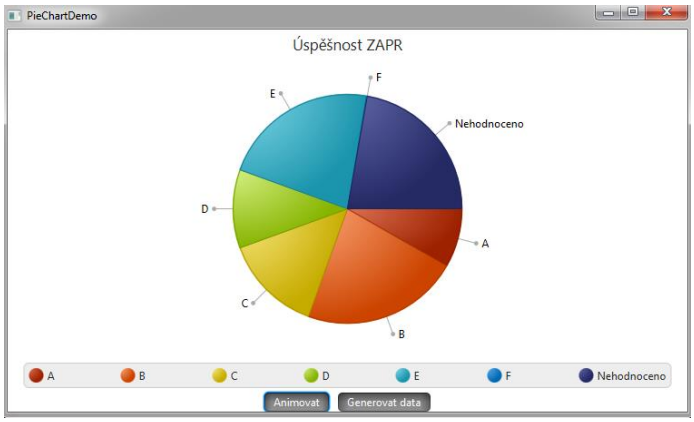


Obrázek 18: Aplikace CanvasDemo
zdroj: vlastní

8.1.5 PieChartDemo

Odkaz na web

V aplikaci PieChartDemo je ukázáno použití koláčového diagramu (`PieChart`). Uživatel může zvolit, zda budou periodicky generovány velikosti výsečí diagramu a zda budou změny animovány. GUI aplikace je zobrazeno na obrázku 19.



Obrázek 19: Aplikace PieChartDemo
zdroj: vlastní

8.1.6 XYChartDemo

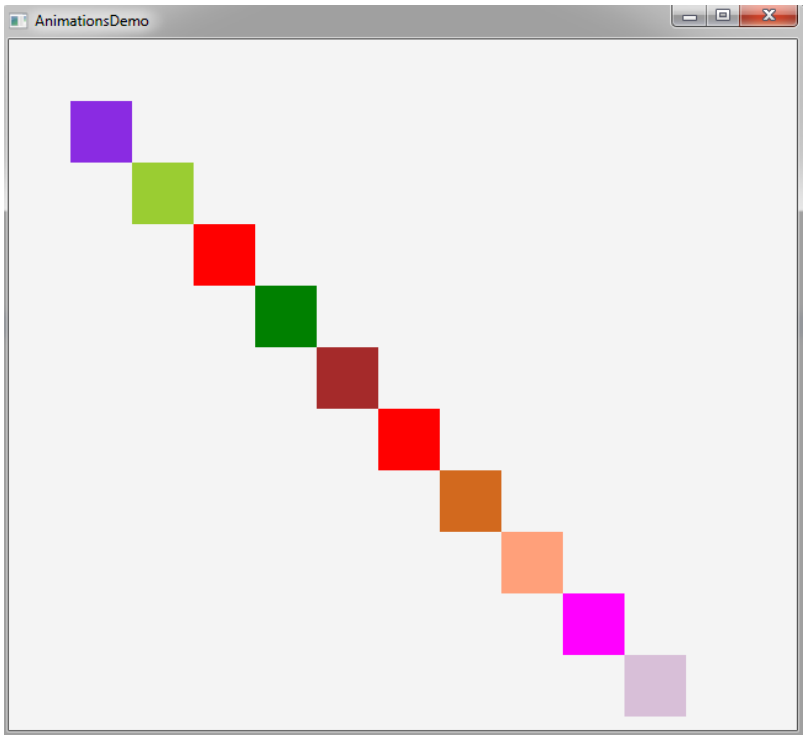
Odkaz na web

V této aplikaci byl použit diagram s kartézským systémem souřadnic (`XYChart`) pro zobrazení hustoty normálního rozdělení pravděpodobnosti. Konkrétně byl použit spojnicový diagram (`LineChart`). Aplikace byla zachycena na obrázku 6 v kapitole 2.2.8.

8.1.7 AnimationsDemo

Odkaz na web

Pomocí aplikace AnimationsDemo (viz obrázek 20) je možné se obeznámit se všemi druhy animací, které jsou v JavaFX k dispozici. V aplikaci byly vytvořeny čtverce reprezentované třídou `Rectangle`, pro které byly definovány animace. Tyto animace se spouští kliknutím na daný čtverec levým tlačítkem myši. Na čtverce byly také nainstalovány Tooltipy, vypisující název animace. Ty se zobrazí po najetí kurzorem na čtverec.



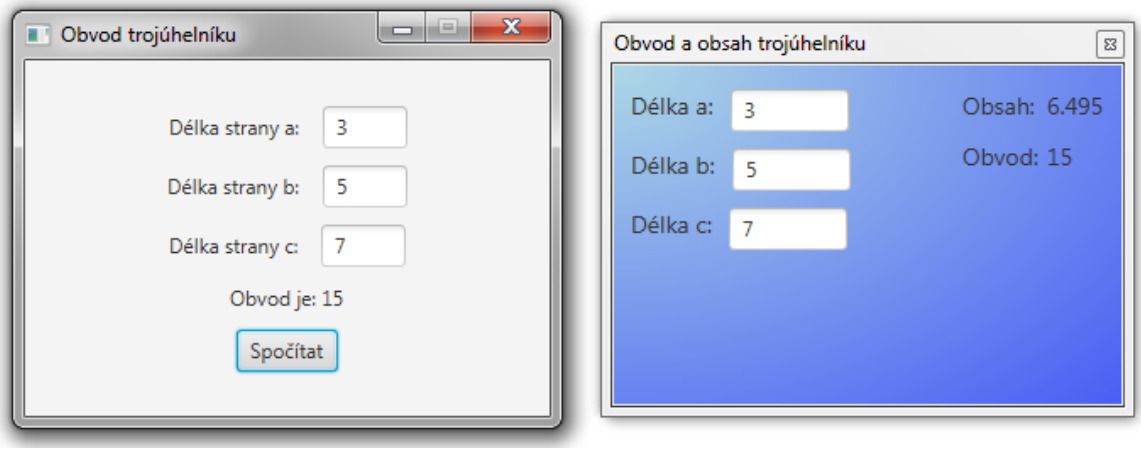
Obrázek 20: Aplikace AnimationsDemo
zdroj: vlastní

8.1.8 Triangle

Odkaz na web

Projekt Triangle tvoří tři aplikace, demonstrující základy tvorby JavaFX aplikací. Aplikace umožňují uživateli zadat délky stran trojúhelníku. Pokud se jedná o validní hodnoty (délka je celé kladné číslo a trojúhelník je sestrojitelný), tak aplikace vypíše obvod a obsah trojúhelníku. V opačném případě se zobrazí chybová hláška.

V první aplikaci, umístěné v balíčku `code`, byl graf scény vytvářen v kódu. Ve druhé aplikace (balíček `fxml`) byly použity jazyky FXML a CSS. Třetí aplikace je rozšířením aplikace druhé. Byla vytvořena třída `Triangle` s JavaFX Vlastnostmi reprezentujícími délky stran, obvod a obsah. Aplikace využívá Data binding, což umožňuje dynamicky reagovat na změnu délek stran. GUI aplikace je zachyceno na obrázku 21.



Obrázek 21: Aplikace Triangle, varianta bez FXML (vlevo) a varianta s Data binding (vpravo)
zdroj: vlastní

8.1.9 MediaDemo

Odkaz na web

Tato aplikace slouží jako velice jednoduchý přehrávač mediálních souborů. Aplikace umožňuje uživateli načíst a přehrát soubor ve formátu MP3. Soubory jsou reprezentovány třídou `AudioClip`. Aplikace také demonstruje přehrávání videosouboru dostupného z Internetu, viz obrázek 22. Za tímto účelem jsou použity třídy `Media`, `MediaPlayer` a `MediaView`.

Obrázek 22: Aplikace MediaDemo
zdroj: vlastní

8.1.10 3DDemo

Odkaz na web

V ukázkové aplikaci 3DDemo je demonstrována práce s 3D tělesy, s kamerou a se světlem. V aplikaci je vytvořena 3D scéna, kterou tvoří různobarevné koule (`Sphere`). Součástí scény jsou také válce (`Cylinder`), reprezentující osy x, y a z. Uživatel může posouvat kamerou tím způsobem, že posouvá myš a současně drží její levé tlačítko. Obdobně je kameru také možné otáčet, místo levého tlačítka je ale třeba držet prostřední tlačítko. Přibližovat a oddalovat kameru je možné otáčením prostředního tlačítka. Součástí aplikace jsou také posuvníky, pomocí kterých je možné měnit polohu bodového zdroje světla (`PointLight`). Aplikace je ilustrována na obrázcích 9 a 11 v kapitole 6. Aplikace vychází z ukázkového příkladu Sample3DBoxApp z tutoriálu [6.1].

8.2 Webová aplikace

V rámci praktické části byla vytvořena webová aplikace. Je dostupná z přiloženého DVD. Pro spuštění aplikace je třeba otevřít jeden z .html souborů ve složce BP_WEB, například soubor index.html. Dále je aplikace dostupná z internetu na adrese http://st39703.fei-hosting.upceucebny.cz/BP_WEB/. Pro spuštění aplikace z internetu je nutné připojení do VPN Univerzity Pardubice.

Po obsahové stránce je aplikace tvořena výňatky z textové části BP. Pro jednotlivé kapitoly byly vytvořeny HTML dokumenty, které je možné procházet pomocí menu. Vzhled stránky byl vytvořen pomocí jazyka CSS.

Původně aplikace obsahovala kód v jazyce PHP[14]. Pomocí PHP byl do hlavního souboru index.php vkládán obsah z další souborů. To umožnilo definování hlavičky a patičky stránky na jednom místě (v souboru index.php). Takto navržená aplikace je snadnější na údržbu. Toto řešení se ale pro potřeby práce ukázalo být nevhodné, protože by aplikace mohla být spouštěna pouze z webového serveru. Aby uživatel mohl spustit aplikaci offline (z přiloženého DVD), musel by mít na svém počítači nainstalován webový server (například Apache[15]) a zprovozněné PHP.

Součástí webové aplikace jsou také ukázkové JavaFX aplikace, popsané v předchozí kapitole. Aplikace je možné jednak z webu spustit (web start), nebo je používat přímo z webové stránky (embedded). Vzhledem k tomu, že aplikace nemají důvěryhodný certifikát, je nutné webu udělit výjimku, viz kapitola 7.3.

Do aplikace byly také přidány video tutoriály, kterým se více věnuje následující kapitola. Pro vložení tutoriálu do stránky byl použit HTML tag `<iframe>`. Kód pro vkládání videí do webových stránek poskytuje server YouTube ke každému videu, na tomto serveru umístěném. Nalézt jej lze v záložce Embed, která se nachází pod samotným videem.

[14] http://php.net/
[15] http://httpd.apache.org/

8.3 Video tutoriály

Součástí bakalářské práce jsou video tutoriály, určené pro JavaFX začátečníky. V tutoriálech je postupně vyvíjena aplikace, přibližně odpovídající ukázkové aplikaci Triangle. Před zhlédnutím je doporučeno získat základní přehled o JavaFX, například z úvodní části kapitoly 2. Tutoriály jsou dostupné na portálu YouTube a také na přiloženém DVD.

Tutoriály byly vytvářeny podle instruktážního videa uživatele Somuel Conlogue, dostupného na YouTube[16]. Obraz byl nahráván pomocí programu CamStudio[17] (verze 2.0), dostupného zdarma pod licencí GPL z oficiálních stránek. Jako kodek byl použit ffdshow[18].

Zejména z důvodu nestability programu CamStudio byly vytvářeny krátké (cca dvouminutové) segmenty. V případě, že program přestal odpovídat, vyprodukoval nespustitelné video či pokud nastal jiný problém během nahrávání, stačilo znovu nahrát pokažený segment. Později bylo zjištěno, že program produkuje nespustitelná videa v případě, že nebyl mezi nahráváním segmentů restartován.

Po nahrání byly segmenty umístěny na server YouTube, kde proběhl střih videa pomocí vestavěného editoru (*Video Editor*). Segmenty byly spojeny do jednoho celku. Byla vystřižena „hluchá“ místa apod. K videu byl také doplněn titulek, popis a klíčová slova.

Z důvodu, že se *Video Editor* ukázal být nestabilním, byl pro spojení segmentů u druhého a následných dílů použit program *VideoPad Video Editor*[19]. Původním záměrem bylo v tomto programu provést i střih videa, to ale řešil program tak neintuitivně, že byl opět použit Video Editor na YouTube.

Po spojení segmentů v programu *VideoPad* mělo po kompresi výsledné video mnohonásobně menší velikost. Např. u druhého tutoriálu bylo video zmenšeno z 2,5 GB na pouhých 50 MB bez výrazného snížení kvality. Menší velikost urychlila proces nahrávání videa na YouTube.

V prvním tutoriálu[20] je vytvořena jednoduchá JavaFX aplikace, umožňující uživateli zadat délky stran trojúhelníku a kliknutím na tlačítko vypsat obvod tohoto trojúhelníku. Ve druhém tutoriálu[21] je vytvořena obdobná aplikace s tím rozdílem, že graf scény je definován pomocí FXML. Ve třetím tutoriálu[22] je aplikace z předchozího dílu rozvinuta po vzhledové stránce použitím jazyka CSS.

8.4 Srovnání JavaFX a Swing po stránce výkonu

Součástí JavaFX je „vysoce výkonný grafický engine, zvaný Prism“ [1.19]. Pokud je zařízení na kterém je aplikace spuštěna dostatečně výkonné, je použita hardwarová akcelerace (HWA). Pokud není, je použito méně výkonné softwarové renderování. HWA je možné vypnout

[16] https://www.youtube.com/watch?v=aHYwG-DgI3E
[17] http://camstudio.org/
[18] http://ffdshow-tryout.sourceforge.net/
[19] http://www.nchsoftware.com/videopad/
[20] https://www.youtube.com/embed/kQZH3X3sTFw
[21] https://www.youtube.com/embed/xvurERWXQCs
[22] https://www.youtube.com/embed/kV4EyHOmiMI

pomocí JVM parametru *-Dprism.order=j2d* nebo *-Dprism.order=sw* (doporučeno pro JDK 1.8) [1.20]. Swing HWA nepodporuje.

Pro srovnání výkonu byly vytvořeny tři dvojice aplikací (JavaFX a Swing). Aplikace jsou blíže popsány v kapitole 8.1.2. Srovnávání bylo provedeno na dvou PC sestavách, dále značených jako PC1[23] a PC2[24].

8.4.1 Animace

První dvojice testovacích aplikací uživateli umožňuje zadat počet generovaných objektů (kruhů). Po spuštění jsou tyto kruhy přesouvány mezi levým a pravým okrajem aplikace. Pro různé počty kruhů byla sledována hodnota FPS. Hodnota FPS představuje počet snímků za sekundu a jedná se o běžně používaný ukazatel při provádění výkonnostních testů. Vyšší hodnota FPS značí vyšší výkon.

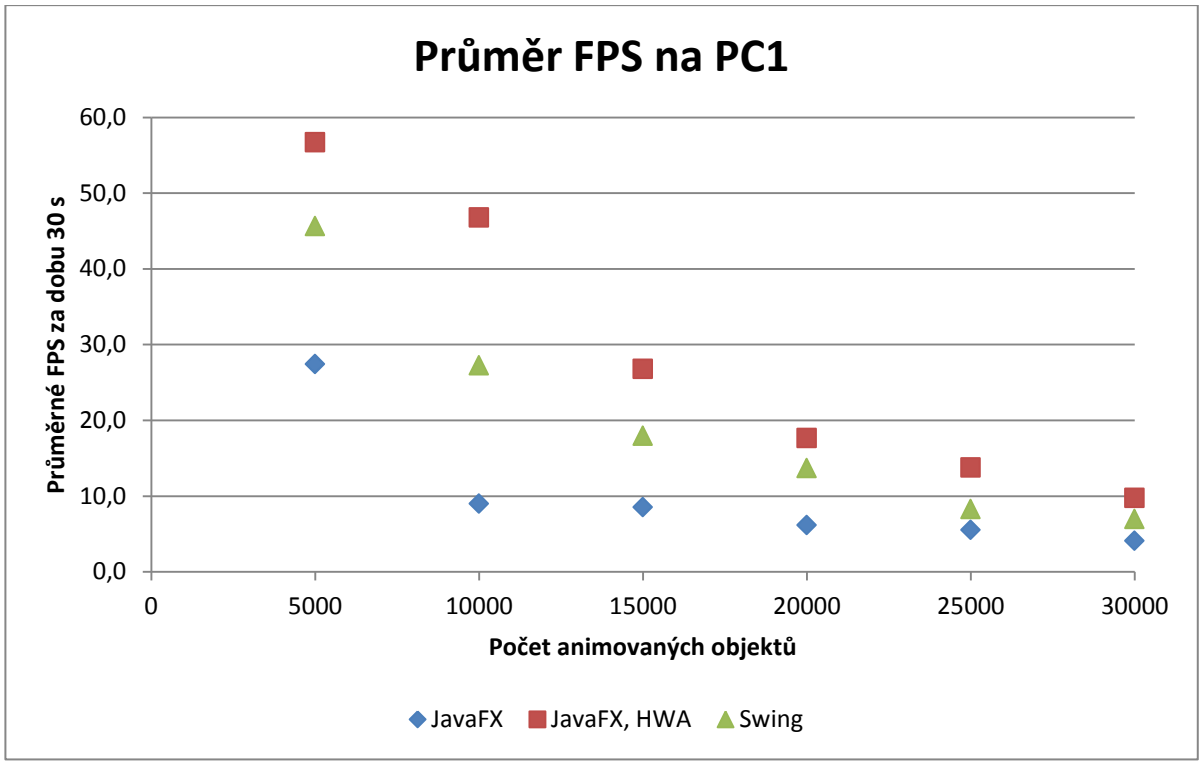
Pro měření FPS byla v JavaFX aplikaci použita třída `AnimationTimer`. Ve Swingové aplikaci byl zaznamenáván počet volání metody `paint()` za sekundu. Hodnoty byly vypisovány na konzoli a po ukončení aplikace byly zkopírovány do programu Microsoft Excel, ve kterém byly dále zpracovány. Aplikace byla spuštěna po dobu třiceti sekund. Získaná data jsou uvedena v tabulce 1.

Tabulka 1: Průměr FPS v aplikaci Screensaver

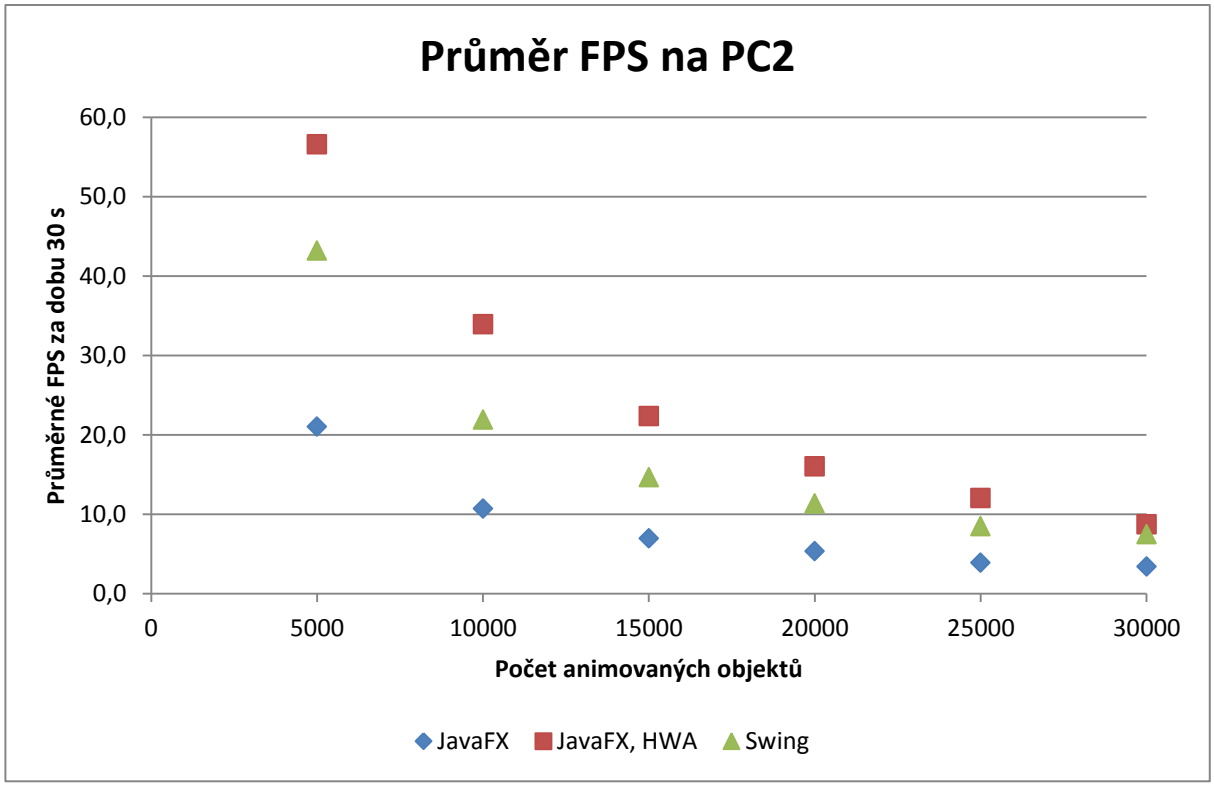
Počet objektů	PC 1			PC 2		
	JavaFX	JavaFX, HWA	Swing	JavaFX	JavaFX, HWA	Swing
5000	27,4	56,7	45,6	21,0	56,6	43,2
10000	9,0	46,8	27,2	10,7	33,9	21,9
15000	8,5	26,8	18,0	6,9	22,3	14,6
20000	6,2	17,7	13,7	5,3	16,0	11,3
25000	5,5	13,8	8,3	3,9	12,0	8,5
30000	4,1	9,7	7,0	3,4	8,7	7,5

Průměrné hodnoty FPS jsou zobrazeny v grafech 1 a 2. Na obou grafech je možné sledovat stejný trend. Se zvyšujícím se počtem animovaných objektů klesá hodnota FPS. Při použití HWA je výkon JavaFX aplikace vyšší, a to na obou testovaných PC sestavách. Pokud není použita HWA, je výkonnější Swing.

[23] Procesor: Intel Core i7-4820K, Grafická karta: NVIDIA GeForce GTX 770, RAM: 8GB
[24] Procesor: Intel Core i5-2450M, Grafická karta: AMD Radeon HD 6630M, RAM: 4GB

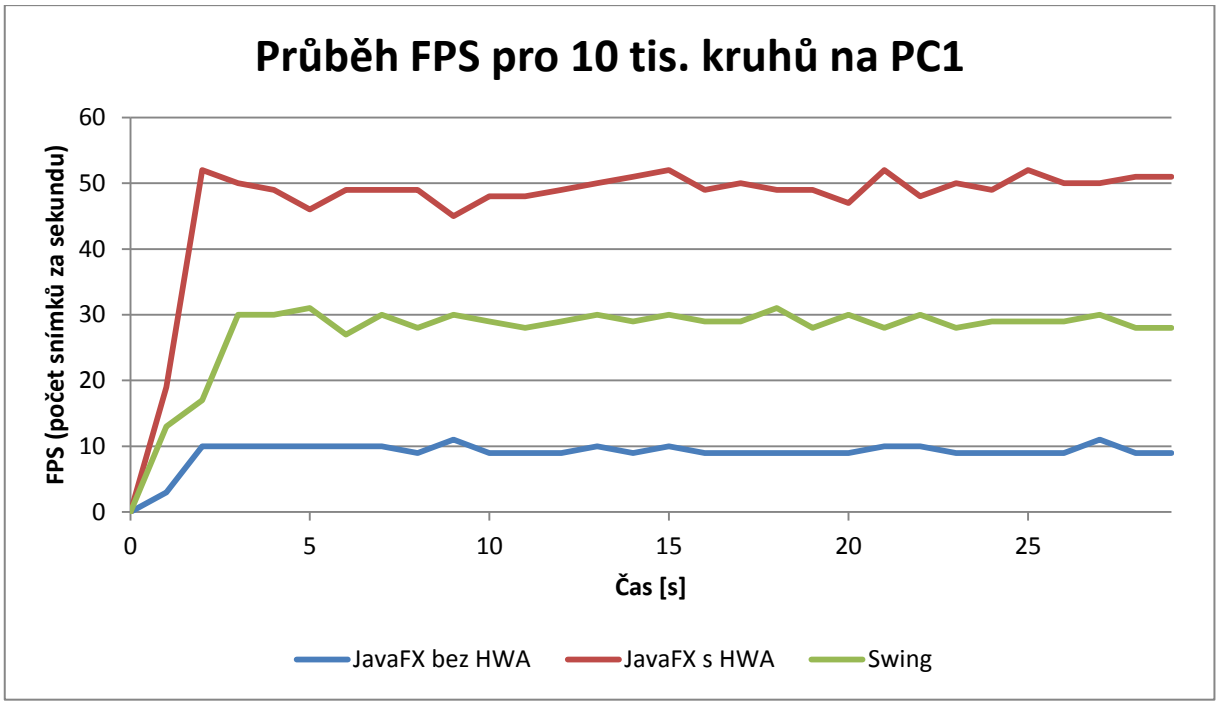


Graf 1



Graf 2

Na grafu 3 je zaznamenán průběh hodnoty FPS pro 10 tis. kruhů. Z tohoto grafu je možné sledovat ustálení FPS poté, co byl dokončen výkonnostně náročnější proces generování animovaných objektů.



Graf 3

Paměťová náročnost aplikací byla zjišťována pomocí profileru nástroje NetBeans. Nástroj vygeneroval graf, zobrazující alokovanou a využitou paměť. Z těchto grafů bylo zjištěno, že JavaFX aplikace s použitím HWA je obecně méně paměťově náročná, než Swingová aplikace. Bez použití HWA je JavaFX stále méně paměťově náročná než swingové aplikace, ale je náročnější, než s použitím HWA. Vzhledem k odlišným implementacím je ale toto srovnání spíše orientační. Grafy pro 10 tis. animovaných objektů jsou umístěny v příloze A.

JavaFX aplikace byla díky vysoko-úrovňovému přístupu k animacím implementována rychleji. Vývoj trval přibližně 30 minut, zatímco vývoj Swing aplikace trval cca 2 hodiny.

8.4.2 Úprava obrázku

Pomocí druhé dvojice testovacích aplikací byl měřen čas potřebný k převedení obrázku do odstínů šedi. Pro test byl použit obrázek o velikosti 27000 Px x 9000 Px (155 MPix), získaný z webu NASA[25]. Pomocí programu IrfanView[26] byly vytvořeny kopie obrázku o velikosti 38,6 MPix, 9,7 MPix, 2,4 MPix a 0,6 MPix.

Nejdříve byl použit algoritmus, kdy byl obrázek procházen po jednotlivých pixelech. Nová barva pixelu, definovaná ve formátu RGB, byla zjištěna pomocí vztahu šedá = 0.35 * (intenzita červeného kanálu) + 0.5 * (intenzita zeleného kanálu) + 0.15 * (intenzita modrého kanálu). Takto získaný odstín šedi je přizpůsoben vnímavosti lidského oka.

Změřený čas byl opět vypisován na konzoli a dále zpracováván v programu MS Excel. JavaFX aplikace vyžadovala výrazně více času pro převedení relativně velkého obrázku (155

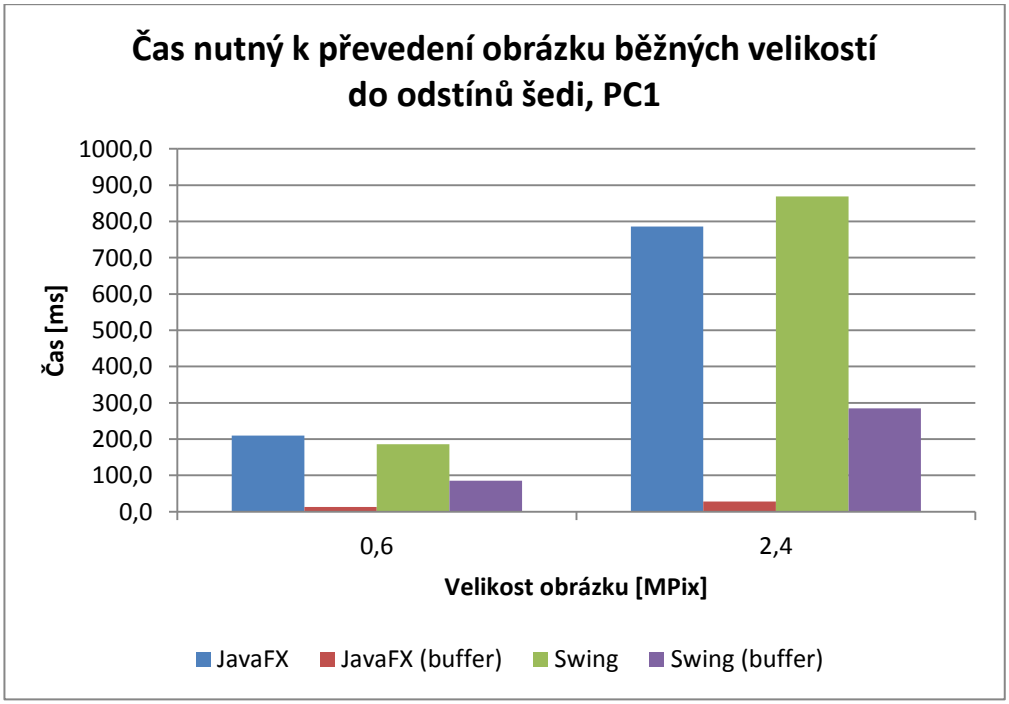
[25] http://photojournal.jpl.nasa.gov/jpeg/PIA03239.jpg
[26] http://www.irfanview.cz/

MPix). Na fóru Stack Overflow[27] bylo hledáno časově méně náročné řešení. Použití kódu, poskytnutého uživatelem James_D, vedlo k mnohonásobnému zkrácení času nutného na převod, viz tabulka 2. Algoritmus používá buffer a je tak náročný na paměť, že bylo nutné pomocí parametru JVM *Xms* zvýšit výchozí velikost paměti JVM (konkrétně na 2700 MB), jinak docházelo k výjimce `OutOfMemoryError` [1.16].

Buffer je možné použít i v knihovně Swing [1.17]. I zde jeho použití znamenalo zvýšení výkonu. Nároky na paměť jsou ale tak vysoké, že se na PC1 pro obrázek s rozlišením 155 MPix nepodařilo vytvořit instanci třídy `BufferedImage` ani po zvětšení maximální velikosti heap. Na PC2, který má méně paměti RAM, se nepodařilo načíst v JavaFX aplikaci obrázek s rozlišením 155 MPix. Ve Swingové aplikaci se nepodařilo načíst obrázek s rozlišením 155 MPix a 39 MPix.

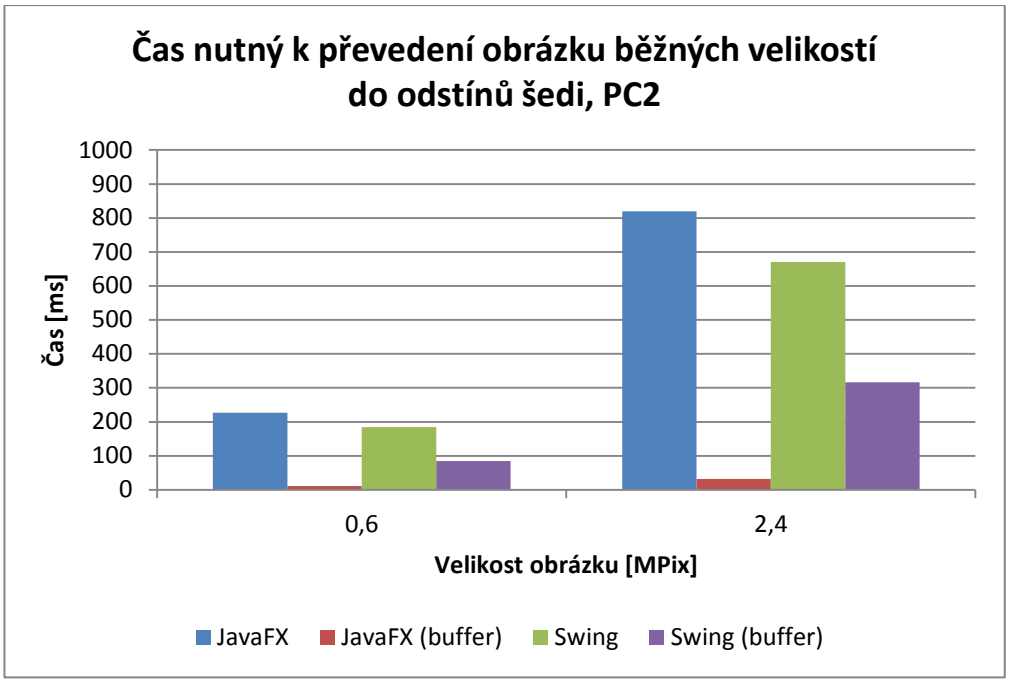
Tabulka 2: Čas (v milisekundách) nutný k převedení obrázku do odstínů šedi (PC1)

Velikost obr. [MPix]	0,6	2,4	9,7	38,6	154,5
JavaFX	209,7	785,5	3088,0	13206,4	51842,4
JavaFX (buffer)	12,9	27,9	88,1	358,4	1419,8
Swing	185,6	868,9	3787,8	12920,1	20821,8
Swing (buffer)	85,6	284,2	1003,1	4264,2	-

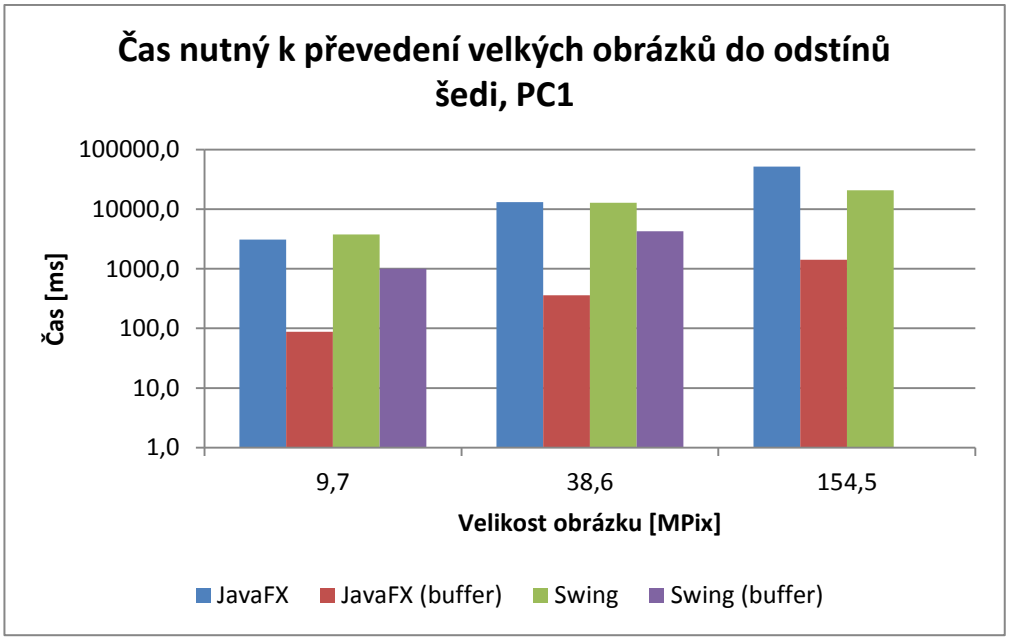


Graf 4

[27] http://stackoverflow.com/questions/29260147/javafx-pixelwriter-low-performance/29275154#29275154



Graf 5



Graf 6

Na grafech 4 a 5 je možné sledovat, že na obou PC sestavách je bez použití bufferu JavaFX přibližně stejně výkonná jako Swingová alternativa. Při práci s obrázkem ve velice vysokém rozlišení (viz graf 6) je výkonnější Swing. Použití bufferu v JavaFX způsobilo výrazné zvýšení výkonu. Převedení probíhá řádově desetkrát rychleji než u alternativních implementací.

Vypnutí hardwarové akcelerace u JavaFX způsobilo výjimku `NullPointerException` během načítání obrázku. Příčinou může být to, že HWA byla „násilně“ vypnuta pomocí parametru JVM. Vliv HWA se tedy nepodařilo zjistit.

8.4.3 Kreslení

Pomocí třetí dvojice aplikací byl měřen čas nutný k nakreslení různého počtu Bézierových křivek. V JavaFX aplikaci bylo kresleno na `Canvas`, ve Swingové na `JPanel`. Vlastnosti křivek (barva, pozice apod.) byly generovány pomocí třídy `Random`. Za účelem co nejobjektivnějšího srovnání byla při vytváření instance třídy `Random` v JavaFX i Swing aplikaci použita stejná násada.

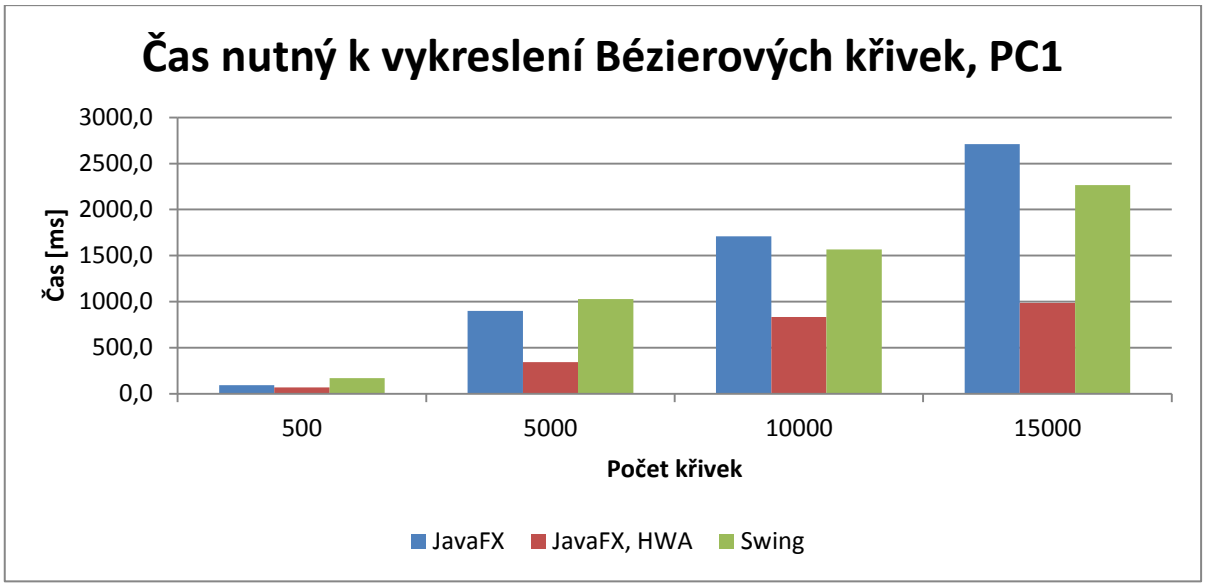
Při kreslení velkého množství objektů na `Canvas` trvalo první vykreslení výrazně déle (na PC1 přibližně desetkrát). Důvodem pravděpodobně byl lineární růst bufferu, který `Canvas` používá. V JDK verze 1.8u40 již buffer roste exponenciálně. První vykreslení stále trvá déle, ale rozdíl není nijak výrazný [1.18].

V JavaFX aplikaci bylo problém změřit, jak dlouho trvá vykreslení, protože neprobíhá v aplikačním vláknu (*JavaFX Application Thread*), ale v renderovací vláknu (*Quantum-Renderer*). Dobu vykreslení při použití HWA se podařilo změřit pomocí třídy `AnimationTimer`. Při vypnuté HWA byly získávány nepřesné hodnoty, a tak byla průměrná doba získána jako podíl doby, po kterou pracovalo vlákno *QuantumRederer* a počtu vykreslovaných objektů. Doba, po kterou renderovací vlákno pracovalo, byla získána pomocí profileru nástroje NetBeans.

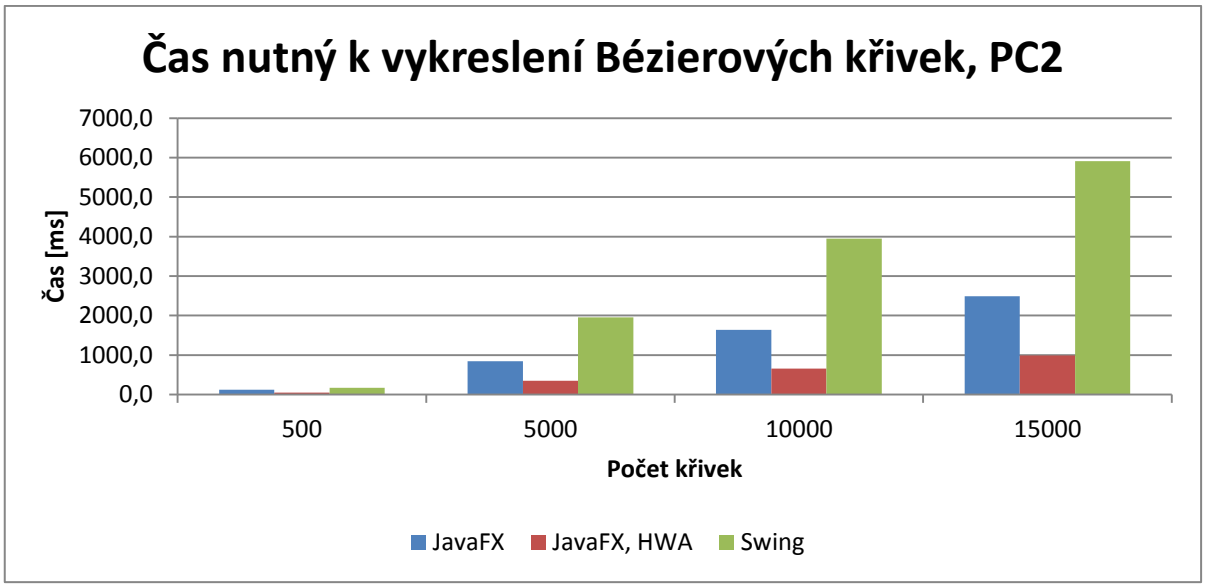
V tabulce 3 jsou uvedeny průměry z deseti vykreslení různého počtu Bézierových křivek.

Tabulka 3: Vykreslování Bézierových křivek, čas v milisekundách

Počet křivek	PC1			PC2		
	JavaFX	JavaFX HWA	Swing	JavaFX	JavaFX HWA	Swing
500	92,3	67,8	169,5	120	44,3	171,4
5000	900,0	343,1	1027,5	840	347	1955
10000	1709,1	834,3	1565,3	1640,4	656,9	3947,7
15000	2709,3	987,2	2265,9	2490,4	987,8	5912,7



Graf 7



Graf 8

Jak je možné posoudit z elektronické přílohy 1, JavaFX implicitně vykresluje ve vyšší kvalitě. Na PC1 je bez použití HWA výkon JavaFX a Swing aplikace srovnatelný. Při použití HWA je JavaFX více než dvojnásobně rychlejší, viz graf 7. Také na PC2 byla JavaFX s HWA nejvýkonnější (viz graf 8). Swingová aplikace ale byla méně výkonná než JavaFX bez HWA. Z jakého důvodu je Swing aplikace na PC2 výrazně méně výkonná se nepodařilo zjistit.

9 ZÁVĚR

Cíle práce se podařilo splnit. Platforma JavaFX byla popsána jak z teoretického, tak praktického hlediska. Webová aplikace spolu s video tutoriály může sloužit jako studijní materiál do předmětu Počítačová grafika. V práci byl kladen důraz na popsání základů JavaFX. Z tohoto důvodu nebyla zmíněna některá témata, jako například práce s vlákny.

Na základě srovnání s knihovnou Swing lze konstatovat, že se JavaFX jeví jako lepší volba. Podporuje tvorbu obsahu, který pomocí Swing tvořit nelze, případně který vyžaduje externí knihovny (např. 3D grafika a diagramy). Výhodou knihovny Swing je zejména to, že se kolem ní za patnáct let vytvořila rozsáhlá komunita, která vyprodukovala mnoho externích knihoven, tutoriálů apod.

V provedených výkonnostních testech vykazovaly JavaFX lepší výsledky za předpokladu, že byla povolena hardwarová akcelerace. Tu v dnešní době podporují i počítače nízké výkonnostní třídy. Testovací aplikace by bylo možné dále optimalizovat. Také by bylo dobré získat výsledky z více počítačových sestav.

Platforma je stále ve vývoji, a tak je možné očekávat ladění bugů a přidávání nových funkcí. Již v současném stavu se ale jedná o plnohodnotnou náhradu knihovny Swing. Výhody JavaFX však nejsou natolik výrazné, aby opodstatnily migraci aplikace z knihovny Swing na JavaFX. To platí zejména pro aplikaci využívající minimum grafického obsahu (např. formulářové aplikace).

Při tvorbě GUI je možné vybírat z velkého množství ovládacích prvků a kontejnerů. Mnohé z nich jsou známé z knihovny Swing, což ulehčuje přechod vývojáře z jedné technologie na druhou. Je možné zvolit mezi vytvářením GUI v Java kódu nebo v jazyce FXML. Jako vhodnější se jeví použití FXML, protože je kód přehlednější a díky vizualizaci v nástroji Scene Builder je vývoj rychlejší. Definování vzhledu aplikace pomocí CSS znamená, že vzhled může tvořit člen vývojářského týmu, který nemusí rozumět programování. Díky CSS je možné vytvářet aplikace, které na uživatele působí atraktivně.

Práce s animacemi je díky vysokoúrovňovému přístupu poměrně snadná. Na běžné účely (posun, otáčení atp.) poslouží třídy odvozené od `Transition`. Pro vytváření složitějších animací je vhodné použít třídu `Timeline`.

Velice užitečný je mechanismus Data binding. Při jeho použití je kód přehlednější a kompaktnější. Užitečné jsou také nově přidané kolekce, zejména možnost monitorovat změny jejich stavu a na tyto změny reagovat.

Co se týče práce s mediálními soubory, i zde došlo k usnadnění oproti knihovně Swing. Díky tomu, že kontejner pro obrázky `ImageView` je odvozený od třídy `Node`, stačí například pro posun či otočení obrázku zavolat metody této třídy. Zajímavá je také možnost načíst mediální soubor z internetu.

Pro tvorbu 3D obsahu jsou nabízeny spíše základní funkcionality. Není implementována například detekce kolizí 3D těles či zaměření kamery na zadaný bod. Přesto jsou pro tvorbu

jednoduchých 3D scén poskytované třídy dostatečné. Pro tvorbu komplexních scén existují vhodnější technologie, než JavaFX.

Možnost umístit aplikaci na web je velkou výhodou JavaFX aplikací. Proces umístění je velice jednoduchý a aplikaci není nutné nijak upravovat. Překážkou ale mohou být certifikáty. Defaultně jsou blokovány všechny aplikace bez důvěryhodného certifikátu, a to i takové, které pracují v omezeném režimu. Pokud má být aplikace volně dostupná z internetu, je nutné certifikát zakoupit, protože nelze očekávat, že uživatel umístí web do seznamu výjimek.

Ukázkové aplikace se zabývají platformou JavaFX z různých pohledů. Po jejich prostudování bude čtenář schopen samostatně vyvíjet vlastní JavaFX aplikace. Vývoj ukázkových aplikací se obešel bez výraznějších problémů. V budoucnu by bylo vhodné aplikace dále optimalizovat. Také by bylo vhodné vytvořit větší aplikaci, která by demonstrovala součinnost technik použitých v ukázkových aplikacích.

Bez problémů se také obešla tvorba webové aplikace. I tu by bylo možné dále vyvíjet, například přidáním diskuzního fóra. Vhodné by také bylo umístit aplikaci na veřejně dostupný server (momentálně je přístupná jen z univerzitní sítě).

Tvorba video tutoriálů se ukázala být značně časově náročná. Důvodem byl jednak nedostatek zkušeností s touto tématikou a také nestabilita programů, používaných při nahrávání a úpravě videa. Jedné minutě finálního videa odpovídá zhruba patnáct minut práce „za oponou“. Natočené video tutoriály objasňují úplné základy tvorby JavaFX aplikací. Autor doufá, že jejich zhlédnutí motivuje diváka k vyhledání dalších informací.

10 POUŽITÁ LITERATURA

[1.1] *JavaFX FAQ* [online]. Oracle, [2012] [cit. 2014-10-24]. Dostupné z: http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html

[1.2] POTTS, Jasper, Nancy HILDEBRANDT, Joni GORDON a Cindy CASTILLO. *JavaFX Getting Started with JavaFX, Release 8: JavaFX Overview* [online]. Oracle, August 2014 [cit. 2014-10-23]. Dostupné z: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm

[1.3] ROUSE, Margaret. *What is Rich Internet Application (RIA)? - Definition from WhatIs.com* [online]. TechTarget, © 2001 - 2015 [cit. 2014-10-24]. Dostupné z: http://searchsoa.techtarget.com/definition/Rich-Internet-Application-RIA

[1.4] SELTZER, Larry. *Rich Internet Applications: The Next Frontier of Corporate Development* [online]. 2010-08-25 [cit. 2014-10-24]. Dostupné z: http://www.eweek.com/c/a/Security/Rich-Internet-Applications-The-Next-Frontier-of-Corporate-Development-732651

[1.5] WEAVER, James L. *Pro JavaFX 2: a definitive guide to rich clients with java technology*. Lexington: Apress, 2012, s. 465. The expert´s voice in Java. ISBN 978-1-4302-6872-7.

[1.6] WEAVER, James L. *Pro JavaFX 2: a definitive guide to rich clients with java technology*. Lexington: Apress, 2012, s. 2-4. The expert´s voice in Java. ISBN 978-1-4302-6872-7.

[1.7] LEAHY, Paul. *What Is JavaFX?* [online]. About.com, © 2015 [cit. 2015-10-24]. Dostupné z: http://java.about.com/od/javafx/a/What-Is-Javafx.htm

[1.8] *What's New in JDK 8* [online]. Oracle, [2014] [cit. 2015-03-16]. Dostupné z: http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html

[1.9] *SUN DELIVERS NEXT VERSION OF THE JAVA PLATFORM: Java™ 2 Brand Unveiled* [online]. NEW YORK CITY: Sun Microsystems, December 8, 1998 [cit. 2015-02-27]. Dostupné z: http://web.archive.org/web/20070816170028/http:/www.sun.com/smi/Press/sunflash/1998-12/sunflash.981208.9.xml

[1.10] *Swing Depot: Component Suites* [online]. Sun Microsystems, © 1995-2004 [cit. 2015-02-27]. Dostupné z: http://www.javadesktop.org/rollups/components/

[1.11] ROUSE, Margaret. *What is Windows Presentation Foundation (WPF)? - Definition from WhatIs.com* [online]. TechTarget, © 2000 - 2015 [cit. 2015-02-28]. Dostupné z: http://searchwindevelopment.techtarget.com/definition/Windows-Presentation-Foundation

[1.12] *How to: Create a New WPF Browser Application Project* [online]. Microsoft, © 2015 [cit. 2015-02-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb628663%28v=vs.110%29.aspx

[1.13] HORSDAL, Christian. *XAML or JavaFx?* [online]. Exelixis Media, December 26th, 2012 [cit. 2015-02-28]. Dostupné z: http://www.javacodegeeks.com/2012/12/xaml-or-javafx.html

[1.14] WEAVER, James L. *Pro JavaFX 2: a definitive guide to rich clients with java technology*. Lexington: Apress, 2012, s. 431-439. The expert´s voice in Java. ISBN 978-1-4302-6872-7.

[1.15] JAKOB, Marco. *JavaFX Dialogs (official)* [online]. Oct 28, 2014 [cit. 2015-03-14]. Dostupné z: http://code.makery.ch/blog/javafx-dialogs-official/

[1.16] *-X Command-line Options* [online]. Oracle, 04/05/11 [cit. 2015-03-28]. Dostupné z: http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionX.html

[1.17] *Faster pixel method than BufferedImage's setRGB* [online]. Oracle, 2.7.2004 [cit. 2015-03-28]. Dostupné z: https://community.oracle.com/thread/1266869
[1.18] GRAHAM, Jim. *Canvas initial delay issue* [online]. Oracle, May 12 2014 [cit. 2015-03-29]. Dostupné z: http://mail.openjdk.java.net/pipermail/openjfx-dev/2014-May/013838.html
[1.19] CASTILLO, Cindy. *JavaFX Architecture* [online]. Oracle, April 2013 [cit. 2015-03-29]. Dostupné z: https://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm
[1.20] ZENBENI. *How to disable or bypass Hardware Graphics Acceleration(Prism) in JavaFX* [online]. Stack Exchange, 2013-09-12 [cit. 2015-03-29]. Dostupné z: http://stackoverflow.com/questions/18754803/how-to-disable-or-bypass-hardware-graphics-accelerationprism-in-javafx

[2.1] LEAHY, Paul. *JavaFX: The JavaFX Scene Graph* [online]. About.com, © 2015 [cit. 2015-02-18]. Dostupné z: http://java.about.com/od/javafx/a/The-Javafx-Scene-Graph.htm
[2.2] *Stage (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-18]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html
[2.3] *Tooltip (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-18]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Tooltip.html
[2.4] *Scene (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-18]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html
[2.5] *Node (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-19]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html
[2.6] *SwingNode (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-19]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/embed/swing/SwingNode.html
[2.7] FEDORTSOVA, Irina. *Embedding Swing Content in JavaFX Applications* [online]. Oracle, September 2013 [cit. 2015-02-19]. Dostupné z: http://docs.oracle.com/javafx/8/embed_swing/jfxpub-embed_swing.htm
[2.8] FEDORTSOVA, Irina, Nancy HILDERBRANDT a Steve NORTHOVER. *JavaFX: Interoperability: Integrating JavaFX into Swing Applications* [online]. Oracle, March 2014 [cit. 2015-02-19]. Dostupné z: http://docs.oracle.com/javase/8/javafx/interoperability-tutorial/swing-fx-interoperability.htm
[2.9] GORDON, Joni. *JavaFX: Working with Layouts in JavaFX: Using Built-in Layout Panes* [online]. Oracle, March 2014 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm
[2.10] *Control (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/scene/control/Control.html
[2.11] REDKO, Alla. *Using JavaFX UI Controls* [online]. Oracle, September 2013 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm
[2.12] *Canvas (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/scene/canvas/Canvas.html
[2.13] BEMBRICK, Felix. *JavaFX Canvas versus HTML5 Canvas* [online]. August 11, 2013 [cit. 2015-02-20]. Dostupné z: http://justmy2bits.com/2013/08/11/javafx-canvas-versus-html5-canvas/
[2.14] HOMMEL, Scott. *Working with Canvas* [online]. Oracle, April 2013 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm
[2.15] *Shape (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-21]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Shape.html

[2.16] *Chart (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2015-02-21]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/scene/chart/Chart.html
[2.17] REDKO, Alla. *Using JavaFX Charts* [online]. Oracle, January 2014 [cit. 2015-02-21]. Dostupné z: http://docs.oracle.com/javafx/2/charts/jfxpub-charts.htm
[2.18] FEDORTSOVA, Irina. *Mastering FXML* [online]. Oracle, January 2014 [cit. 2014-12-26]. Dostupné z: http://docs.oracle.com/javafx/2/fxml_get_started/jfxpub-fxml_get_started.htm
[2.19] *Introduction to FXML* [online]. Oracle, 6/21/2012 [cit. 2014-12-27]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html
[2.20] *JavaFX CSS Reference Guide* [online]. Oracle, (c) 2008, 2014 [cit. 2014-10-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html
[2.21] CASTILLO, Cindy a Yves JOAN. *JavaFX Scene Builder User Guide* [online]. Oracle, October 2013 [cit. 2014-10-22]. Dostupné z: https://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm
[2.22] CHEPTSOV, Andrey. *IntelliJ IDEA 14 is Released!* [online]. November 5, 2014 [cit. 2014-11-10]. Dostupné z: http://blog.jetbrains.com/idea/2014/11/intellij-idea-14-is-released/

[3. 1] *Animation (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-19]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/Animation.html
[3. 2] *Timeline (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-19]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/Timeline.html
[3. 3] *Transition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-19]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/Transition.html
[3. 4] *FadeTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/FadeTransition.html
[3. 5] *FillTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/FillTransition.html
[3. 6] *ParallelTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/ParallelTransition.html
[3. 7] *PathTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-20]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/PathTransition.html
[3. 8] *PauseTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-21]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/PauseTransition.html
[3. 9] *RotateTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-21]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/RotateTransition.html
[3. 10] *ScaleTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-21]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/ScaleTransition.html
[3. 11] *SequentialTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-22]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/SequentialTransition.html
[3. 12] *StrokeTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-22]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/StrokeTransition.html
[3. 13] *TranslateTransition (JavaFX 2.2)* [online]. Oracle, 2013-12-09 [cit. 2014-12-22]. Dostupné z: http://docs.oracle.com/javafx/2/api/javafx/animation/TranslateTransition.html

[4.1] HOMMEL, Scott. *Using JavaFX Properties and Binding* [online]. Oracle, June 2013 [cit. 2015-02-06]. Dostupné z: http://docs.oracle.com/javafx/2/binding/jfxpub-binding.htm
[4.2] BRUNO, Eric. *JavaFX 2.0 Binding* [online]. UBM Tech, November 17, 2011 [cit. 2015-02-07]. Dostupné z: http://www.drdobbs.com/jvm/javafx-20-binding/231903245
[4.3] HOMMEL, Scott. *Using JavaFX Collections* [online]. Oracle, April 2013 [cit. 2015-02-08]. Dostupné z: http://docs.oracle.com/javafx/2/collections/jfxpub-collections.htm

[5.1] *Image (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-10]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/Image.html
[5.2] HOMMEL, Scott. *Using the Image Ops API* [online]. Oracle, June 2013 [cit. 2015-02-10]. Dostupné z: http://docs.oracle.com/javafx/2/image_ops/jfxpub-image_ops.htm
[5.3] *ImageView (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-10]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/ImageView.html
[5.4] CASTILLO, Cindy. *Incorporating Media Assets Into JavaFX Applications* [online]. Oracle, April 2013 [cit. 2015-02-11]. Dostupné z: http://docs.oracle.com/javafx/2/media/jfxpub-media.htm
[5.5] *Class (Java Platform SE 7)* [online]. Oracle, 2014-09-26 [cit. 2015-02-11]. Dostupné z: http://docs.oracle.com/javase/7/docs/api/java/lang/Class.html
[5.6] *Media (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-11]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/media/Media.html
[5.7] *MediaPlayer (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-12]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/media/MediaPlayer.html
[5.8] *MediaView (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-12]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/media/MediaView.html
[5.9] *AudioClip (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-12]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/media/AudioClip.html

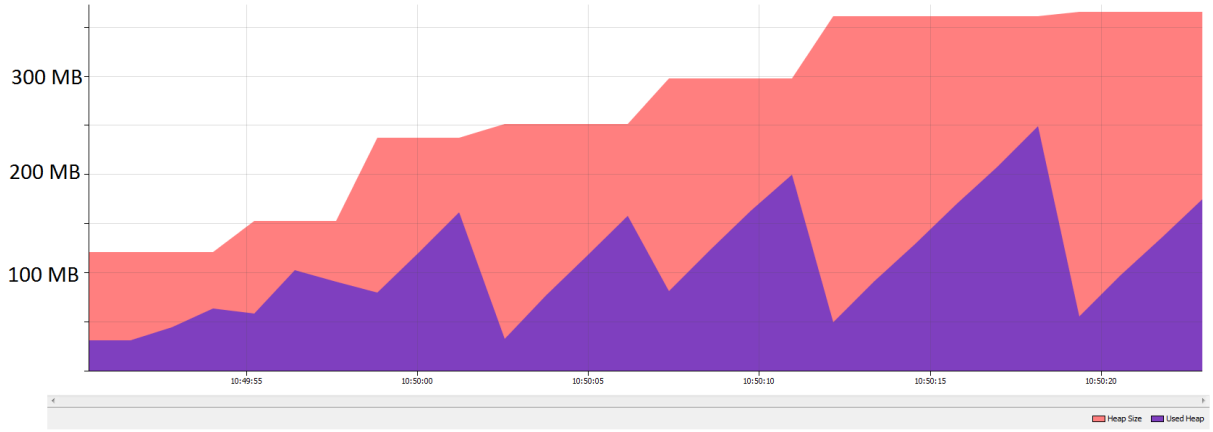
[6.1] CASTILLO, Cindy a Scott HOMMEL. *JavaFX: Working with JavaFX Graphics: Getting Started with JavaFX 3D Graphics* [online]. Oracle, March 2014 [cit. 2015-02-13]. Dostupné z: http://docs.oracle.com/javase/8/javafx/graphics-tutorial/javafx-3d-graphics.htm
[6.2] *PhongMaterial (JavaFX 8)* [online]. Oracle, 2015-02-10 [cit. 2015-02-14]. Dostupné z: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/phongmaterial.html

[7.1] KOSTOVAROV, Dmitry. *Deploying JavaFX Applications: Deployment in the Browser* [online]. Oracle, October 2013 [cit. 2015-02-15]. Dostupné z: http://docs.oracle.com/javafx/2/deployment/deployment_toolkit.htm
[7.2] *Why are Java applications blocked by your security settings with the latest Java?* [online]. Oracle, 2014-10-16 [cit. 2015-02-15]. Dostupné z: https://www.java.com/en/download/help/java_blocked.xml
[7.3] COSTLOW, Erik. *Self-signed certificates for a known community* [online]. Oracle, Nov 11, 2013 [cit. 2015-02-15]. Dostupné z: https://blogs.oracle.com/java-platform-group/entry/self_signed_certificates_for_a
[7.4] DOLEŽAL, Dušan. *Jak si vybrat certifikační autoritu* [online]. *Interval.cz*, Brno: ZONER software s. r. o., 14. 3. 2003 [cit. 2015-02-15]. ISSN 1212-8651. Dostupné z: https://www.interval.cz/clanky/jak-si-vybrat-certifikacni-autoritu/
[7.5] *The Java™ Tutorials: What Applets Can and Cannot Do* [online]. Oracle, © 1995, 2015 [cit. 2015-03-23]. Dostupné z: http://docs.oracle.com/javase/tutorial/deployment/applet/security.html

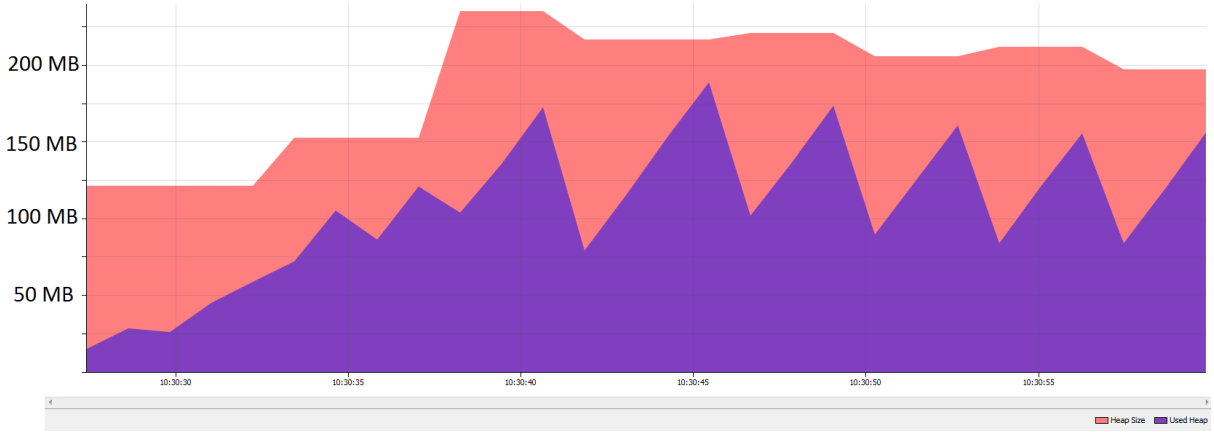
11 PŘÍLOHY

Příloha A *Paměťová náročnost jednotlivých variant aplikace ScreenSaver* 71
Příloha B *Přehled ovládacích prvků* 72
Příloha C *Výběr identifikátorů CSS vlastností* 74
Příloha D *Defaultní CSS třídy* 73

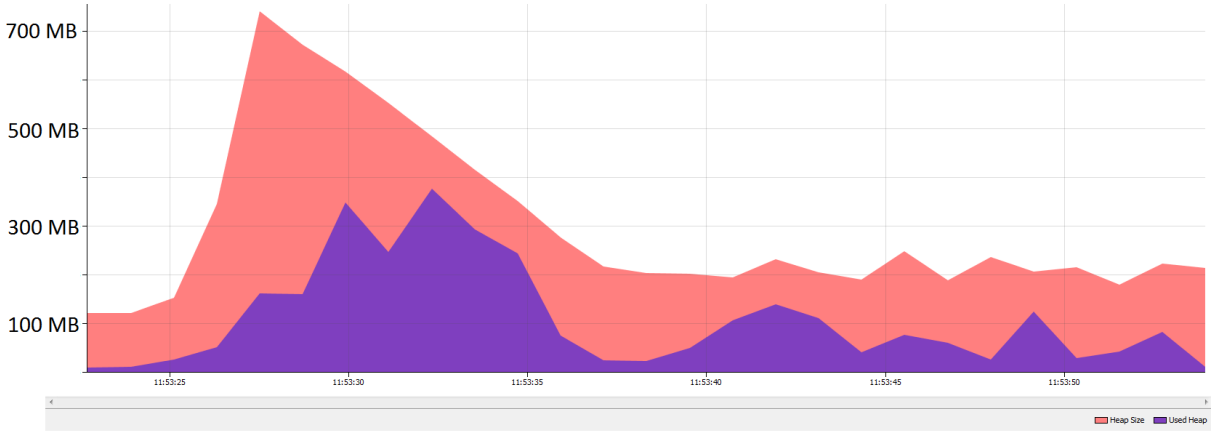
Příloha A *Paměťová náročnost jednotlivých variant aplikace ScreenSaver při animování deseti tisíc objektů. Oranžová barva znázorňuje maximální velikost heap, fialová využitou heap.*



JavaFX bez hardwarové akcelerace



JavaFX s hardwarovou akcelerací



Swing

Příloha B *Přehled ovládacích prvků*

JavaFX třída	Popis	Swing třída
Button	Klasické tlačítko	JButton
CheckBox	Zaškrtávací pole	JCheckBox
ChoiceBox	Výběrové pole	-
ColorPicker	Umožňuje uživateli vybrat barvu	JColorChooser*
ComboBox	Výběrové pole, volitelně s posuvníkem	JComboBox
DatePicker	Umožňuje uživateli vybrat datum	-
HTMLEditor	WYSIWYG HTML editor	-
Hyperlink	Hypertextový odkaz	-
ImageView	Viz kapitola 5.1.2	-
Label	Text pouze pro čtení	Jlabel
ListView	Víceřádkový seznam	Jlist
MediaView	Viz kapitola 5.2.3	-
MenuBar	Lišta menu, do které je možné vložit rozbalovací menu (třída Menu), které obsahují jednotlivé položky (tř. MenuItem)	JMenuBar, (JMenu, JMenuItem)
Pagination	Ovladač, jehož obsah je rozdělen do stránek, mezi kterými je možné listovat.	-
PasswordField	Pole pro textový vstup, zadaný text je z bezpečnostních důvodů zobrazen pomocí zástupného symbolu, pole lze bezpečně použít k zadání hesla.	JPasswordField
ProgressBar	Znázornění průběhu operace pomocí lišty	JProgressBar
ProgressIndicator	Znázornění průběhu operace pomocí koláčového grafu	-
RadioButton	Výběrové tlačítko	JRadioButton
Slider	Posuvník (horizontální či vertikální)	JSlider
Spinner*	Výběr ze seřazené posloupnosti, umožňuje inkrementaci a dekrementaci	JSpinner
TableView	Tabulka, jejíž sloupce jsou reprezentovány třídou TableColumn	JTable
TextArea	Oblast pro vstup a výstup textu	JTextArea
TextField	Textové pole zejména pro vstup	JTextField
Tooltip	Viz kapitola 2.2.1	JToolTip
ToggleButton	Přepínací tlačítko	JToggleButton
TreeView	Hierarchická struktura	JTree
TreeTableView	Tabulka s hierarchicky organizovanými řádky	-
WebView	Spolu s třídou WebEngine umožňuje v aplikaci zobrazit webové stránky	-

*od verze JDK 8u40

Příloha C *Výběr identifikátorů CSS vlastností*

Vlastnosti třídy Node	
Identifikátor	**Poznámka**
-fx-effect	Viz kapitola 2.5.7
-fx-opacity	Průhlednost uzlu
-fx-scale-[x | y | z]	Změna měřítka podle osy
-fx-translate-[x | y | z]	Posunutí podle osy
visibility	Viditelnost Uzlu

Vlastnosti třídy Region	
Identifikátor	**Poznámka**
-fx-background-color	Barva pozadí
-fx-background-image	Obrázek na pozadí
-fx-border-color	Barva ohraničení
-fx-border-width	Tloušťka ohraničení
-fx-padding	Vnitřní okraj

Vlastnosti třídy Shape	
Identifikátor	**Poznámka**
-fx-fill	Barva výplně
-fx-stroke	Barva ohraničení
-fx-stroke-width	Tloušťka ohraničení

Vlastnosti třídy Labeled	
Identifikátor	**Poznámka**
-fx-alignment	Zarovnání uzlu
-fx-text-alignment	Zarovnání textu
-fx-font	Font textu
-fx-text-fill	Barva textu

Styly textu	
Identifikátor	**Poznámka**
-fx-font-size	Velikost písma
-fx-font-style	Styl (normální, kurzíva, sklonění)
-fx-font-weight	Tloušťka písma

Kompletní výčet je k dispozici na JavaFX CSS Reference Guide.

Příloha D *Defaultní CSS třídy*

Uzel	**Třída**	**Uzel**	**Třída**
Accordion	accordion	RadioButton	radio-button
Button	button	ScrollBar	scroll-bar
Cell	cell	ScrollPane	scroll-pane
CheckBox	check-box	Separator	separator
ChoiceBox	choice-box	Slider	slider
ColorPicker	color-picker	SplitMenuButton	split-menu-button
ComboBox	combo-box	SplitPane	split-pane
ComboBoxBase	combo-box-base	TabPane	tab-pane
Hyperlink	hyperlink	TableView	table-view
IndexedCell	indexed-cell	TextArea	text-area
Label	label	TextField	text-field
ListCell	list-cell	ToggleButton	toggle-button
ListView	list-view	ToolBar	tool-bar
Menu	menu	Tooltip	tooltip
MenuBar	menu-bar	TreeCell	tree-cell
MenuButton	menu-button	TreeView	tree-view
MenuItem	menu-item	WebView	web-view
Pagination	pagination	Chart	chart
PasswordField	password-field	Axis, NumberAxis, CategoryAxis	axis
ProgressBar	progress-bar		
ProgressIndicator	progress-indicator	Legend	chart-legend