UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Vytvoření enginu pro interaktivní procvičování matematiky

Jiří Hermann

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří Hermann**
Osobní číslo:	**I10052**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Vytvoření enginu pro interaktivní procvičování matematiky**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

V teoretické části student zdůrazní význam a potenciál využití počítačů při procvičování početních úloh a provede srovnání existujících programů (zejména s důrazem na srovnání komerčních produktů a webových stránek, které nabízí tyto programy zdarma).
Primárním cílem bakalářské práce bude vytvoření enginu pro program, který umožní procvičování intuitivních početních příkladů (násobilka, základní počty).
Engine bude navržen tak, aby splňoval princip Learn & Play.
Funkčnost enginu student doloží jeho spuštění s daty vhodnými pro procvičování matematiky pro žáky první až čtvrté třídy základní školy.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. SIERRA, Kathy, BATES, Bert. Head First Java. Vydání druhé. O?Reilly Media, 2005. ISBN 978-0596009205.
2. HEROUT, Pavel: Učebnice jazyka JAVA. České Budějovice : KOPP, 2010. ISBN 978-80-7232-398-2.
3. FLANAGAN, David. Programování v jazyce Java. Praha : Computer Press, 1997.
ISBN: 80-85896-78-8.

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra softwarových technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora

Prohlašuji, že jsem tuto práci 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 7. 5. 2014

Jiří Hermann

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Josefu Brožkovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Anotace

V teoretické části student zdůrazní význam a potenciál využití počítačů při procvičování početních úloh a provede srovnání existujících programů (zejména s důrazem na srovnání komerčních produktů a webových stránek, které nabízí tyto programy zdarma). Primárním cílem bakalářské práce bude vytvoření enginu pro program, který umožní procvičování intuitivních početních příkladů (násobilka, základní počty). Engine bude navržen tak, aby splňoval princip Learn & Play.Funkčnost enginu student doloží jeho spuštění s daty vhodnými pro procvičování matematiky pro žáky první až čtvrté třídy základní školy.

Klíčová slova

Java, Konkurence, výukový program, JavaFX, engine, animace, základní metody

Title

Creating engine for interactive math practice

Annotation

In the theoretical part of the student emphasize the importance and potential use of computers for practicing arithmetic task and make comparison with existing programs (with particular emphasis on the comparison of commercial products and websites that offer these programs free). The primary aim of this thesis is to create the engine for a program that will allow practicing intuitive numerical examples (multiplication tables, basic numbers). Engine will be designed to meet the principle of Learn & Play. Functionality engine student demonstrates his start with data suitable for practicing mathematics for pupils first to fourth grade.

Keywords

Engine, JavaFX, Java

Obsah

Seznam zkratek

Win7 - Windows 7

server-bassed middleware - název aplikace

JavaFX Scene Builder - nástroj pro návrh uživatelského rozhraní

JavaFX – softwarová platforma založená na bázi platformy Java

FXML – jazyk založený na XML

Java API – dokumentace programovacího jazyku Java

QT4 – knihovna pro vytváření programů s grafickým uživatelským rozhraním

Seznam obrázků

Seznam tabulek

Úvod

Tato bakalářská práce má za cíl vytvořit zábavný program pro výuku matematiky zacílené na žáky 1. stupně základních škol. Také se stala motivací pro vytvoření zajímavého a interaktivního prostředí, které zaujme danou skupinu dětí a výuka matematiky se tak pro ně stane zábavou. Dané téma bakalářské práce jsem si zvolil při zjištění, že na našem trhu není mnoho dostupných zábavných výukových programů matematiky. Touto prací bych chtěl ukázat dětem, že vzdělávání může být i zábavnou formou a ne jen počítáním příkladů.

Bakalářská práce je rozdělena na část teoretickou, která analyzuje konkurenci na trhu a v druhé praktické části se práce zabývá programováním. Nejvíce známá firma výukových softwarů je firma Terasoft a.s., která mě však překvapila zjištěním, že není možné spustit jejich aplikaci na systému Win7. Z tohoto důvodu jsem nemohl porovnávat program této společnosti s další konkurenční firmou E-software, kde se mi podařilo aplikaci spustit a otestovat. Dále jsem si našel pár aplikací placených i freewarových a všechny tyto aplikace si jsou velmi podobné. Z mého pohledu u nich postrádám trochu akce, animace a obrázků a na základě tohoto zjištění, bych chtěl z těchto faktů vycházet při psaní této bakalářské práce. V části praktické jsem se zaměřil na animace, díky kterým se stane tato aplikace zajímavější a odliší se tím od ostatních momentálně dostupných aplikací. Rozhodl jsem se jí napsat v jazyce JavaFX

Práce je psaná v jazyce JavaFX. Jedná se o sadu grafických a mediálních balíčků, které umožňují vytvářet a navrhovat bohaté aplikace, které fungují na různých platformách. Můžeme zde využívat různé grafické obrazce, které máme možnost různorodě nastavovat. Proto považuji tento jazyk lepším než je jazyk Java, kde nemáme možnost vytvářet animace s objekty a je těžší s nimi různé animace provádět. Jelikož jsem chtěl ve své práci tyto animace využívat, rozvíjel jsem své schopnosti v jazyce JavaFX, který nemáme možnost na naší fakultě studovat. Od základního jazyku Java se tento jazyk významně neliší, ale nabízí více možností a mezi hlavní výhodu patří tvorba animací a práce s nimi.

Pro správné pochopení této práce jsou potřebné alespoň minimální znalosti v oblasti informačních technologií.

1 Analytický pohled

Pojednává zejména o produktech dostupných na trhu, srovnávání jak placených tak i freeware aplikací. Porovnávání funkcí a ovladatelnosti. Zpracování je ze zdroje [7.].

1.1 Konkurence

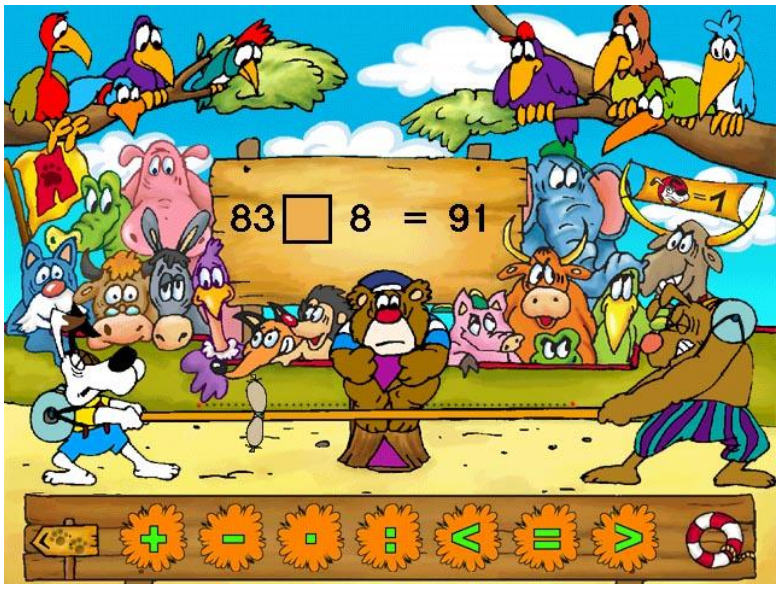
Na trhu je v současné době velká konkurence mezi dodavateli výukových programů, které je potřeba věnovat pozornost. Mezi největší konkurenci se řadí společnosti Terasoft a.s. a E-software.cz.

1.2 Terasoft a.s.

V minulosti byla společnost Terasoft a.s. známá jako dodavatel výukových programů především pro základní školy, ale po bližším prostudování společnosti jsem zjistil, že se jejich vývoj a distribuce výukových softwarů zastavil. Stáhl jsem si demoverzi jejich produktu pro výuku matematiky pro studenty 1. stupně základních škol a zjistil jsem, že tento software není kompatibilní s mým operačním systémem (Windows 7). Nemohl jsem tedy posoudit jejich grafickou stránku a strukturu programu přímo z praktického použití.

1.3 E-software

Další společností zabývající se tímto odvětvím je firma E-software, která se zabývá distribuováním výukových softwarů pro děti základních škol. E-software uveřejnilo na svých webových stránkách demoverzi matematiky pro studenty 1. stupně, kterou jsem si chtěl vyzkoušet na svém počítači. Po úspěšném rozbalení jsem nainstaloval daný program, nastala zde chyba na konci instalace, kdy se operační systém ptal, jestli je daný software řádně naistalován. Po spuštění softwaru se změnilo rozlišení mé obrazovky. Rozlišení se nepřizpůsobilo obrazovce a program nabýval dojmem, že není moc profesionální ani kompatibilní s daným operačním systémem. Při příchodu do hlavního menu jsem nevěděl jak se mám v programu pohybovat. Ani tento software ve mně nevzbudil dojem, že je vhodným nástrojem pro výuku matematiky pro děti 1. stupně.

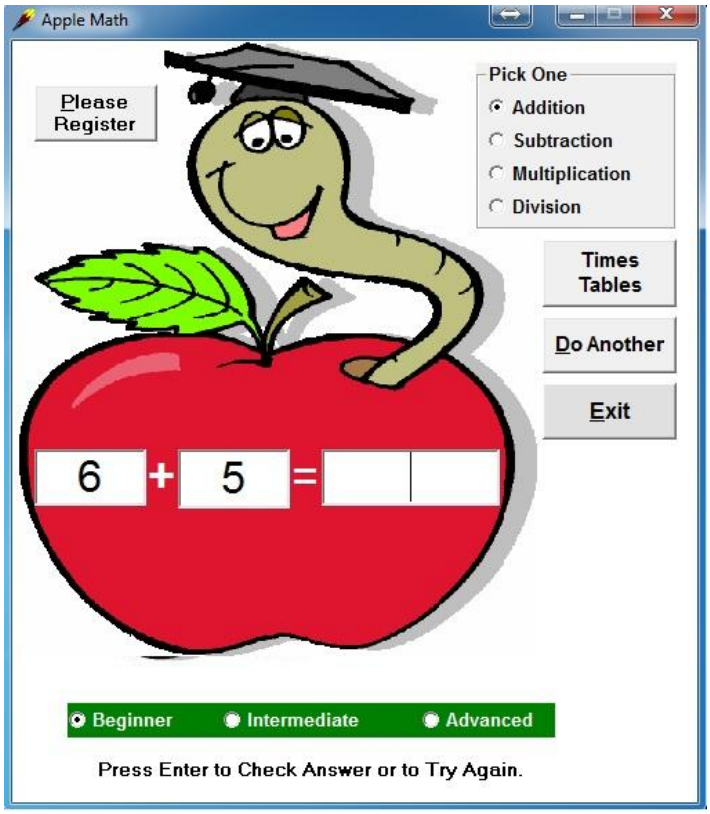


Obrázek 1: Alík veselá matematika

Obrázek 2: Alík veselá matematika

1.4 Apple Math

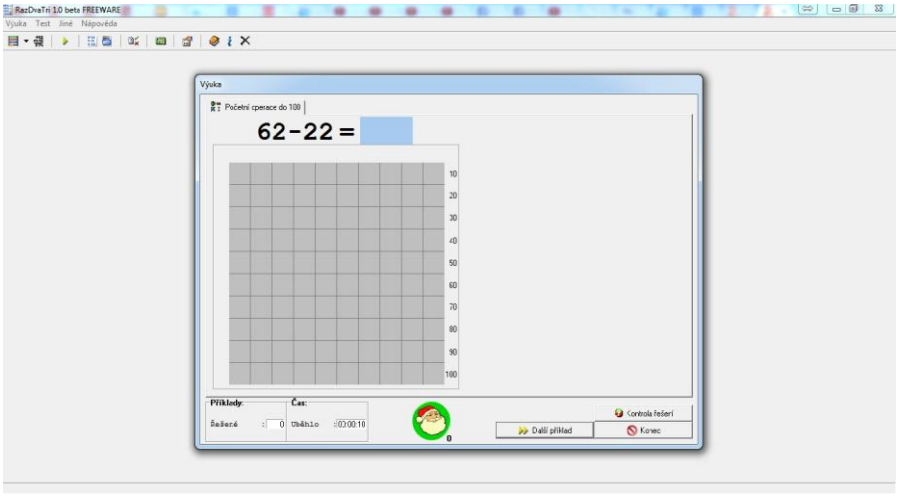
Program Apple Math je demoverzí s omezenou funkčností, pro získání plné verze je potřeba vytvořit registraci a poté se přihlásit. Hned první nevýhodou tohoto programu je jeho jazykové zpracování, které je pouze v anglickém jazyce, což může být pro porozumění problémové, neboť program je zacílen na děti 1. stupně základních škol. Program přináší početní funkce: sčítání, odčítání, násobení, dělení a také úrovně začátečník, pokročilí a profesionál. Podle těchto úrovní se generují příklady pro procvičování. Při pohledu na tento program vidíme, že se opět jedná o statický program a pro daného studenta je to opět nezáživné.



Obrázek 3: Apple Math

1.5 Program RazDvaTri

Jedná se o freeware beta program, který umožňuje procvičování matematiky pro studenty 1. stupně základních škol. Můžeme zde řešit operace sčítání, odčítání, násobení a dělení. Obsahuje vícejazyčnou podporu, máme zde také možnost automatického řešení příkladů, prohlížení výsledků testu a jejich analýzu. Program je vcelku zajímavě zpracován a je podporován i na novějších operačních systémech jako je např.: Win7, na kterém jsem jej testoval. Nicméně program mne moc nezaujal. Jedná se spíše o statický program, kde se pouze počítá. Pro studenty, respektive děti, které se pomocí tohoto softwaru budou vzdělávat, se program stane nezajímavým, a proto je brzy omrzí. Další nevýhodou je, že zde nejsou žádné obrázky či animace, které by zaujalo dětskou pozornost.



Obrázek 4: RazDvaTri

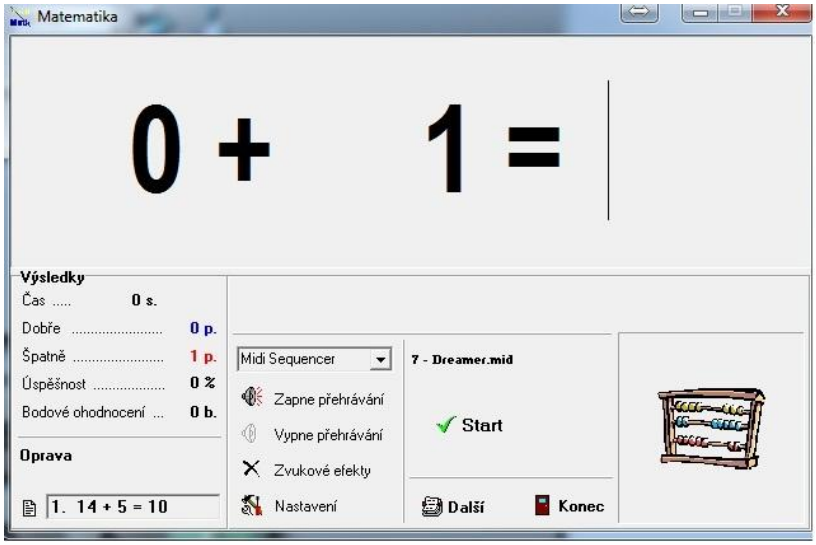
1.6 KidsMath

Volně šiřitelná (freeware) aplikace. Je zaměřena na výuku matematiky pro základní školy prvního stupně. Zabývá se matematickými operacemi sčítání, odčítání násobení a dělení. Program umožňuje 4 volby úrovně programu, jak dlouhý bude časový interval na vypracování příkladů. Můžeme si vybrat typy příkladů, které chceme generovat. Možnost volby je sčítání, odčítání, nebo všechny početní úkony dohromady. Tato aplikace je zajímavě zpracována, ale stále není dynamická. Máme zde uvedený čas, který omezuje dobu, pod kterou je potřeba odpovědět bez jakékoliv dalšího rozpracování. Aplikaci lze naistalovat a spustit na operačním systému Win7 bez problémů, problém nastává při zvětšení aplikace na celou plochu, kdy nám klávesnice s čísly ujede do strany.

Obrázek 5: KidsMath

1.7 Matematika 1.00

Matematika 1.00 je program určený pro výuku matematiky jak pro nejmenší děti, tak i pro starší žáky. Jedná se o freeware aplikaci. Opět zde máme klasické matematické operace, sčítání, odčítání, násobení a dělení. Program je doprovázen hudbou, další vlastností aplikace je klakson, zvukové oznámení při špatné, nebo správné odpovědi na příklad a také změny obrázku. Je zde možnost zobrazení chybových příkladů. Tato aplikace je opět statická, jedinou animací, která se zde nachází, je změna obrázku a zvukový klakson. Testoval jsem tento program na operačním systému Win7 bez problémů.



Obrázek 6: Matematika100

1.8 Slabiny konkurence

Slabiny konkurence jsou značně viditelné. Jedním z největších problémů je kompatibilnost s moderními operačními systémy. Softwary se špatně přizpůsobují nebo je nelze úspěšně spustit na daném operačním systému. Jejich grafické zpracování je vcelku hezké , ale ve většině případů statické. Z mého pohledu není program pro děti přehledný ani zábavný a nevzbudí v nich předpokládaný zájem pro studium matematiky

1.9 Mé požadavky

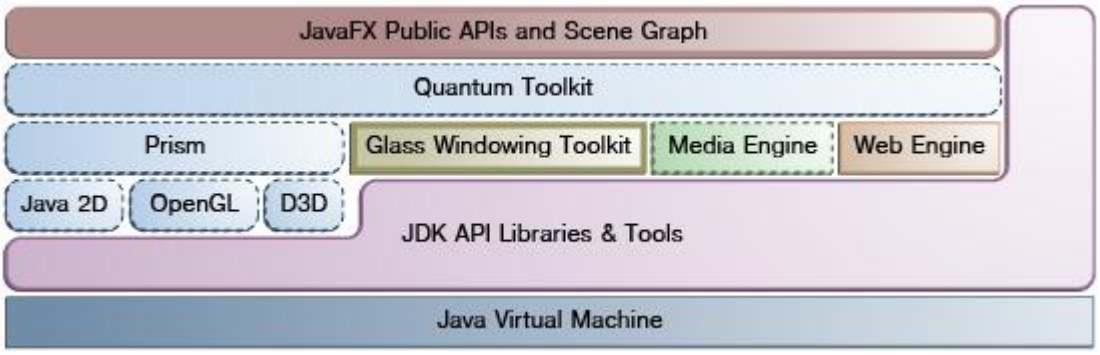
Výukový program zaměřuji na děti 1. stupně základní školy. V tomto věku děti poznávají svět kolem sebe, snaží se jej pochopit, a proto chci program těmto dětem přizpůsobit a tím je i zaujmout. Vzhledem k tomu, že předem zmíněné programy byly dosti statické, mým požadavkem je, aby byl program dynamický, aby byli vidět animace a pohyb různých objektů či věcí. Hlavním zájmem je zaujmout pozornost dětí, ukázat jim, že i matematika se lze naučit hrou a zároveň pro děti vytvořit novou zábavnou hru, která je bude bavit, a tím se mohou v matematice zlepšovat.

2 Technologický pohled

Jedná se o pohled na dostupné programovací techniky, ve kterých se dá daná aplikace naprogramovat. Zde bylo čerpáno ze zdrojů [1.] [4.] [5.].

2.1 JavaFx

JavaFx je sadou knihoven, jejichž cílem je umožnit vývojáři vytvářet bohaté klientské aplikace, které se chovají konzistentně na všech platformách. Je napsána jako Java API. JavaFX může používat knihovny jazyku Java pro přístup k nativním funkcím systému a připojení k server-bassed middleware aplikacím. Výsledný vzhled a dojem lze přizpůsobit díky možnosti použití kaskádových stylů (CSS), tím pádem se vývojář může soustředit na svůj daný kód. Vývojář může snadno přizpůsobit vzhled a styl aplikace skrze styly CSS. Pokud programujeme webové aplikace a využíváme web design, neboli chceme rozdělit uživatelské rozhraní (UI) a back-end logiky, můžeme vytvořit reprezentační aspekty uživatelského rozhraní ve FXML skriptovacím jazyce. Kód psaný v jazyce java můžeme použít pro logiku aplikace. Pokud dáváme přednost navrhování uživatelského rozhraní před psaním kódu, nabízí se nám JavaFX Scene Builder. Scene Builder poskytuje vizuální prostředí, které nám umožňuje rychle navrhnout uživatelské rozhraní pro aplikace, aniž bychom museli psát kód. FXML kód pro rozvržení je automaticky generován a Scene Builder poskytuje jednoduché, ale intuitivní rozhraní, které nám může pomoci rychle vytvořit prototyp uživatelského prostřední naší aplikace.



Obrázek 7: Architektura JavaFX

2.2 Dostupnost

Nejnovější dostupnou verzí JavaFX je verze 2.2, která je plně integrována s Java SE 7 Runtime Environment (JRE) a Java Development Kit (JDK). JDK je k dispozici pro všechny hlavní desktopové platformy jako jsou Windows, Linux a Mac OS. Aplikace psané v jazyku JavaFX jsou koncipované tak, aby byly spustitelné na všech platformách osobních počítačů. Kompatibilita umožňuje konzistentní runtime prostřední pro aplikace JavaFX jak vývojářů, tak i uživatelů.

2.3 Klíčové vlastnosti

Hlavní klíčové vlastnosti se vztahují k verzi 2.2 a novějším verzím.

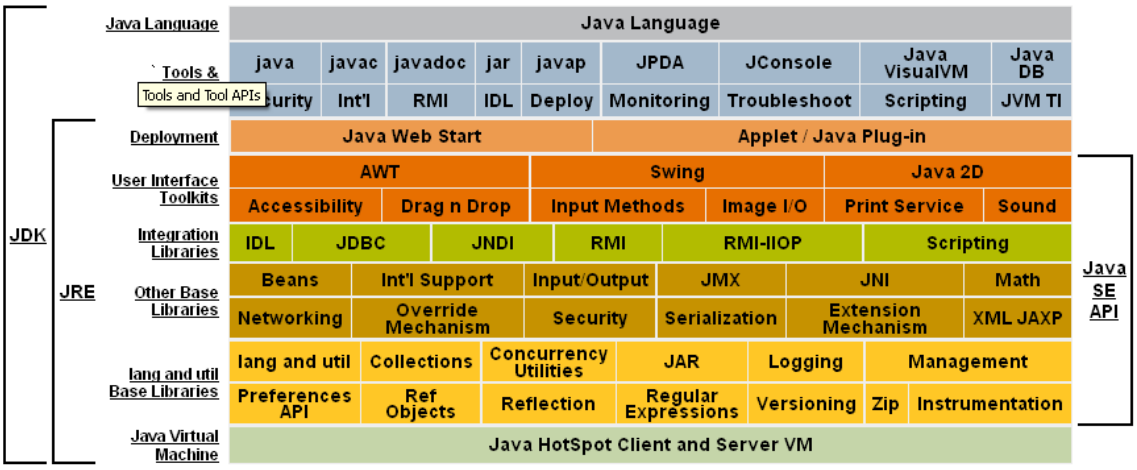
- JavaFX je knihovna, která se skládá ze tříd a rozhraní, která jsou napsaná v nativním jazyce kódu Java. API je navržena tak, aby byla přívětivou alternativou pro Java Virtual Machine (JVM) jazyků, jako je JRuby a Scala.
- FXML a Scene Builder. FXML je založen na XML deklarativním značkovacím jazyce pro vytváření uživatelského rozhraní aplikace JavaFX. Programátor může navrhnout uživatelské rozhraní skrze FXML kódu nebo může využít JavaFX Scene Builder.
- WebView – webová komponenta, která využívá technologii WebKitHTML, aby bylo možné vložit webové stránky v rámci aplikace JavaFX.
- Swing – Existující Swingové aplikace mohou být aktualizovány novými funkcemi JavaFX, bohaté na grafické přehrávání médií a integrovaného webového obsahu.
- Vestavěné ovládací prvky a uživatelské rozhraní CSS. JavaFX poskytuje všechny hlavní ovládací prvky UI potřebné pro vytvoření full-featured aplikace.
- Canvas API – umožňuje kreslení přímo v oblasti JavaFX scény, které se skládá z jednoho grafického prvku (uzlu).
- Multitouch podpora – JavaFX poskytuje podporu pro Multitouch operace, na základně schopnosti základní platform.

2.4 Historie

- Rok 2008 – Sun uvolnil první verzi platformy JavaFX 1.0
- Rok 2009 – Únor, JavaFX dostupná také pro mobilní zařízení ve verzi JavaFX 1.1
- Rok 2009 – Červen, oznámena nová verze JavaFX 1.2 která
    - Zvyšovala rychlost
    - Obsahovala vestavěné grafy
    - Beta podpora pro Linux a Solaris
    - Vestavěné ovládací prvky a layouty
- Rok 2010 – Duben, JavaFX 1.3
- Rok 2010 – Srpen, JavaFX 1.3.1
- Rok 2011 – Říjen, JavaFX 2.0
    - Nová sada Java rozhraní pro JavaFX je možností pro všechny vývojáře, kteří mohou nyní pracovat s JavaFX, aniž by bylo nutné se učit nový jazyk
    - Nový jazyk pro tvorbu uživatelského rozhraní – FXML
    - Zrušení podpory pro JavaFX Mobile a další
- Rok 2012 – Duben, JavaFX 2.1
    - První oficiální verze pro Mac OS

2.5 Porovnání s Java

JavaFX slouží pro vývoj Rich aplikací,(Rich Internet applications). RIA aplikace se snaží být co nejvíce uživatelsky přívětivé, používají zejména animace, efekty, zvuky, videa a nepoužívají standardní ovládací prvky. Pod Rich aplikací si můžeme představit hru, kde můžeme řešit její grafiku nastavování různých obtížností, ukládání pozic, vytváření uživatelského rozhraní apod. Kdež to u klasické Javy tyto možnosti nemáme, jelikož Java API nemá tyto knihovny, které obsahuje JavaFX a pokud potřebujeme vytvořit animaci, musíme si naprogramovat potřebné třídy.



Obrázek 8: Architektura Java SE

2.6 Porovnání s více technologiemi

Porovnání s více technologiemi, ve kterých můžeme danou hru napsat.

2.6.1 Jazyk C++

Jazyk C++ vznikl jako nadstavba jazyka C. Jedná se o objektově orientovaný programovací jazyk, který využívá konvence a zásady OOP a je multiplatformním jazykem. Má mnohem větší standardní knihovnu než jazyk C. Proto je i lepším programovacím jazykem pro porovnání této práce. Jazyk C++ sám o sobě nemá grafické uživatelské rozhraní, ale můžeme jej vytvořit pomocí knihovny QT4. Je to knihovna, ve které můžeme vytvořit plně funkční aplikaci. Zde je vidět že jazyk C++ nám bude sloužit pro logiku a knihovna QT4 pro grafiku. Ovšem jazyk C++ je náročný programovací jazyk, ne pro svou syntaxi, ale pro dodržování určitých zásad programování a vyhýbání se chyb. Na rozdíl od jazyku Java C++ podporuje mnohonásobnou dědičnost, která nám umožnuje dědit metody z jiných tříd tak, že jedna třída může zdědit třídy z několika jiných tříd tzv. od jejich rodičů. Jedná se o jednu z výhod tohoto jazyka, ale také je obrovskou nevýhodou, jak v programu napsat mnoho chyb, které se později obtížněji dohledávají. Dle mého názoru není tento programovací jazyk vhodný pro začátečníky, kteří mají zájem naučit se programovat. Dokonce implementace některých knihoven, například knihovna QT4, není zcela jednoduchá. Zatímco jazyk Java je více user-friendly a pro začátečníky jednodušší na pochopení.

2.6.2 Jazyk C#

Jazyk C# byl přizpůsoben pro platformu .NET Framework. Toto rozhraní podporuje několik programovacích jazyků (C#,VB.NET, J# atd.) a obsahuje také knihovny, které jsou ke všem programovacím jazykům společné. V tomto rozhraní můžeme vytvářet nejen aplikace pro Windows, ale také webové aplikace a služby pro mobilní zařízení. C# se syntaxí velmi podobá programovacímu jazyku C/C++ a Java. Pro programátory, kteří se zabývají jedním z těchto programovacích jazyků, není těžké přejít na tento jazyk. Jednou z nevýhod tohoto jazyku je, že není multiplatformní a aplikace se vytváří na operační systém Windows, proto jsem tuto variantu zavrhl.

3 Techniky JavaFX

V této kapitole jsou uvedeny nejčastější techniky a metody, které můžeme v této práci naleznout. Čerpání informací bylo ze zdrojů [4.] [5.].

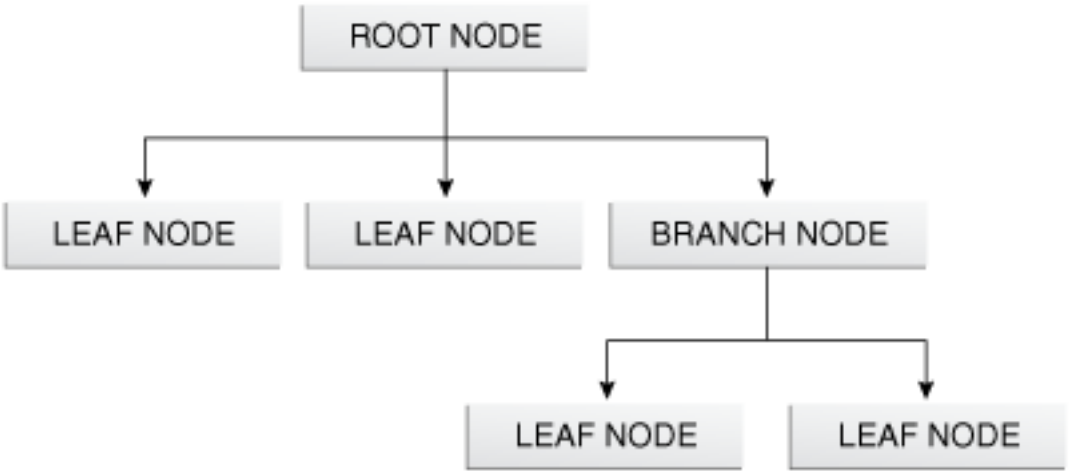
3.1 Organizace – Scene Graph

Scene Graph, jak vidíme na obr 3., je součástí horní vrstvy a výchozím bodem pro návrh a tvorbu aplikací. Jedná se o hierarchii stromu uzlů, které reprezentují dané části aplikačního uživatelského rozhraní. Jednotlivé části Scene Graph se nazývají uzle. Každý uzel obsahuje své ID, třídu stylů a ohraničenou velikost. S výjimkou kořenového uzlu je každý uzel z Scene Grahp rodičem, který má své potomky Nemusí mít žádného potomka, ale naopak jich může mít i více. Skrze Scene Graph můžeme také:

- Přidávat a odebírat efekty jako je například rozostření, stíny, neprůhlednost.
- Dále je tu transformace, kdy můžeme daný uzel otáčet, posouvat nebo zmenšovat.
- Event Handlers tzv. obslužné rutiny událostí např.: reakce klinutí na myš, klinutí na tlačítko na klávesnici.
- JavaFX Scene graph na rozdíl od Java Swing obsahuje grafické primitivy, jako jsou například kruhy, obdélníky, text kterým můžeme nastavovat kde se budou nacházet jakou budu mít barvu velikost. Také umožnuje práci s obrázky a médii.

Scene Graph nám obecně zjednodušuje práci s vytvářením animací. Rychlou animaci vytvoříme pomocí javafx.animation API, javafx.scene API umožňuje vytváření a specifikaci několika typů obsahu jako jsou:

- Uzly – které nám nabízí vytvoření různých tvarů 2D nebo 3D grafiky, práce s obrázky, médii, obsahuje také vestavěný prohlížeč, práce s textem, práce se skupinami a kontejnery.
- Efekty – jednoduché objekty, u kterých můžeme nastavovat různé efekty např.: u textu, kde můžeme nechat text blikat, zkosit nebo nastavit stín.
- Transformace – poloha a orientace daného uzlu, vizuální efekty.

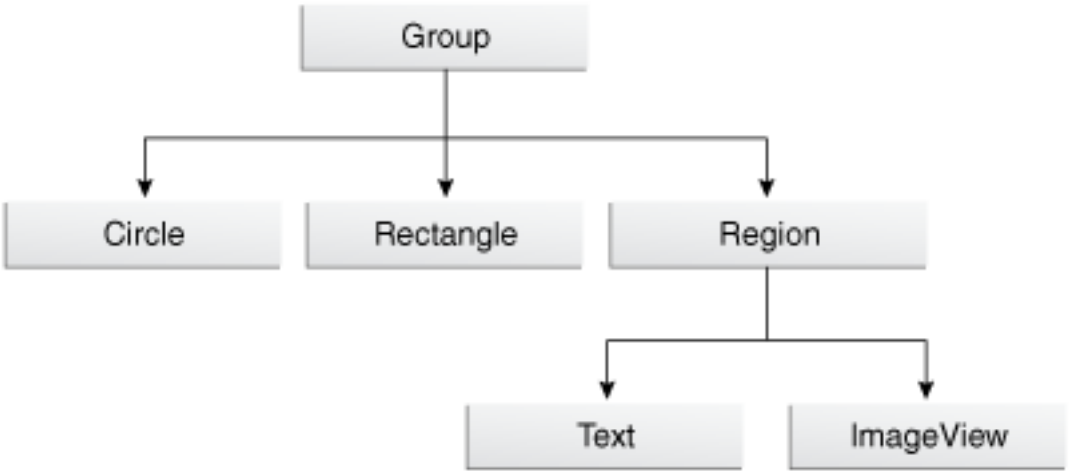


Obrázek 9: architektura uzlů

Na tomto obrázku vidíme hierarchii uzlů. První uzel v daném stromu je kořenový uzel, který nikdy nemá rodiče, ale má své potomky. JavaFX definuje několik tříd, které můžeme využít jako kořenové uzly (root node). List uzlu (leaf node) je to koncový uzel, který již nemusí mít žádného potomka, zato větev uzlu (brach node) může mít potomky. Každý uzel může mít své specifické vlastnosti, které můžeme nastavit. Obrázek je použit ze zdroje [5.].

3.2 Group

Jeden z nejzákladnějších uzlů je uzel Group. Tento uzel obsahuje ObservableList list dětí, nebo-li potomků. Tento list je poskytován především při vykreslování uzlu Group. Zajišťuje vykreslení všech jeho potomků, které obsahuje. Pokud vytvoříme transformaci nebo efekt na uzel Group, uplatní se tyto změny na všechny potomky (uzly), které se v této skupině nacházejí. V případě, kdy vytvoříme transformaci či efekt na daném potomkovi, nebude mít daná změna dopad na další potomky (uzly), které se v této skupině nacházejí. Tato skupina Group má defaultně nastavenou auto-velikost, která se dle potřeby přizpůsobuje daným potomkům.



Obrázek 10: organizace uzlů

Na tomto obrázku vidíme hierarchii objektů. Hlavní (kořenový) uzel nám představuje třída Group, do které přidáváme objekty Circle, Rectangle a Region. Zde můžeme vidět koncové (listy) uzlu, které již nemají potomky. Dále vidíme větev uzlu (Region), který dále obsahuje objekty typu Text, ImageView. Z tohoto obrázku můžeme vidět, že máme skupinu objektů, které zobrazujeme na Scene graphu. Z obrázku lze vyčíst, že na scéně (Scene Graph) budeme moci vykreslit kruh, čtverec a dále text s obrázkem. Všechny tyto objekty můžeme nastavit a specifikovat jejich vlastnosti dle naší potřeby. Obrázek je použit ze zdroje [5.].

4 Základní metody

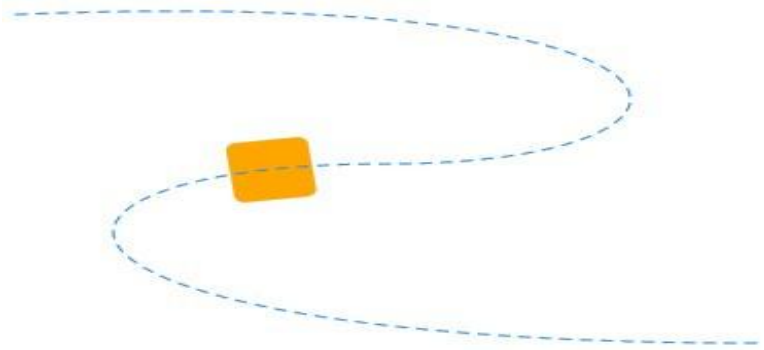
Budou zde zmíněny základní metody, které jsou nejčastěji používány v této práci. Je zde popsán jak popis jednotlivých metod, tak také ukázka kódů pro představu. Pro popis těchto metod bylo čerpáno ze zdrojů [1.] [4.] [5.].

4.1 Transition

Transition neboli přechody poskytují prostředky pro vytvoření animace. Jedná se o abstraktní třídu, která obsahuje základní přechody založené na animaci. Díky této třídě můžeme vytvářet více animací, které můžeme spouštět postupně nebo paralelně.

4.2 Path Transition

Díky této třídě můžeme vytvořit cestu, po které se bude daný objekt pohybovat. Můžeme nastavit, jak dlouho bude tato cesta trvat, kolikrát se má opakovat nebo jestli se má daný objekt vracet zpět po stejné cestě. Pro ilustraci je zde obrázek.



Obrázek 11: Path Transition

Na tomto obrázku můžeme vidět ukázku Path Transtion, kde jsme si navolili, jak daná cesta bude vypadat a také jak dlouho bude trvat. Zvolili jsme tuto cestu pro kruh, který nastavíme jako uzel dané Path Transition animace. Obrázek je použit ze zdroje [5.].

```
Circle kruh = new Circle();
Circle.setCenterX(100);
Circle.setCenterY(100);
Circle.setRadius(35);

Path path = new Path();
Path path = new Path();
path.getElements().add(new MoveTo(100,100));
path.getElements().add(new LineTo(200,200));

PathTransition pathTransition = new PathTransition();
```

```
pathTransition.setDuration(Duration.millis(5000));
pathTransition.setPath(path);
pathTransition.setNode(circle);
pathTransition.setCycleCount(1);
pathTransition.play();
```

Zde je ukázkový kód třídy Path Transition, kde nejdříve vytvoříme objekt kruh (Circle) nastavíme, aby vytvořil na pozici x = 100 a y = 100. Dále si vytvoříme cestu (Path), kde nastavíme odkud se má daný objekt pohybovat (MoveTo(100,100)) a kam se má dostat (LineTo(200,200)). Poté již vytvoříme zmiňovanou třídu PathTransition jako nový objekt. Nastavíme, jak dlouho se má pohybovat (setDuration(Duration.millis(5000)), přidáme cestu, po které se bude pohybovat (setPath(path)), s jakým objektem má pracovat (setNode(circle)) a nakonec kolikrát má animace proběhnout (setCycleCount(1)) a spustíme. Výsledná animace bude vypadat takto: kruh se nám bude pohybovat ze souřadnice x = 100, y =100 po plátně do souřadnice x=200,y=200.

4.3 Rotate transition

Tato třída nám umožňuje vytvořit rotující animaci objektu. Rotující animace nám rotuje podle úhlu, který si nastavíme. Úhel můžeme nastavit od 0° po 360°. Další možnosti nastavení animace je metoda počtu opakování otočení a metoda auto-reverse, která se vyznačuje zpětným otočením objektu.

```
Rectangle rect = new Rectangle ();
rect.setX(50);
rect.setY(50);
rect.setWidth(200);
rect.setHeight(100);
rect.setFill(Color.Black);

RotateTransition rt = new RotateTransition(Duration.millis(3000), rect);
rt.setByAngle(180);
rt.setCycleCount(4);
rt.setAutoReverse(true);
rt.play();
```

První část tohoto kódu je vytvoření obdélníku. Obdélník je vytvořen na pozici x=50,y=50 s šířkou (setWidth(200)), výškou (setHeight(100)) a vyplněn černou barvou (setFill(Color.Black)). Dále si vytvoříme objekt rotace, podle které bude obdélník rotovat. Nastavíme, pod jakým úhlem se bude rotace provádět (setByAngle(180)). V našem případě pod úhlem 180°, počet opakování animace (setCycleCount(4)), animace se bude provádět 4x a poslední je nastavení obrácené rotace (setAutoReverse(True)), kde bude objekt pořád rotovat, aniž by se zastavil.

4.4 Path

Třída Path, představuje jednoduchou cestu, po které se náš objekt bude přesouvat. Abychom daný objekt mohli přesouvat po cestě, musíme vytvořit počáteční a koncový bod odkud a kam se bude náš objekt přesouvat a tyto body vytvoříme pomocí třídy MoveTo a LineTo. Třída MoveTo vytvoří kam se má objekt přesunout a třída LineTo odkud se má daný objekt přesunout. Tuto cestu nám zahrnuje třída PathElement, jejímž abstraktním prvkem je prvek Path, který může představovat geometrické útvary, přímky, kvadratické křivky apod.

```
Path path = new Path();
path.getElements().add(new MoveTo(100,100));
path.getElements().add(new LineTo(200,200));
```

Nastavení cesty po, které se bude daný objekt pohybovat.

4.5 Image

Další třídu, se kterou se můžeme v této práci setkat, je třída Image. Třída Image nám slouží k načtení obrázku a k jeho uchování. Obrázek načítáme podle URL adresy, kterou zadáváme do konstruktoru. Abychom mohli tento obrázek přenést na plátno a zobrazit, musíme ho zavést do třídy ImageView.

```
Image image = new Image("obrazky/auto.png", 100, 150, false, false);
```

Vytvoříme si objekt obrázek s názvem image. Do konstruktoru jako první zadáváme cestu URL, kde se daný obrázek nachází ("obrazky/auto.png"), dále zadáváme jeho šířku (100), výšku (150), zachování poměru stran, aby se nám obrázek vešel celý do námi definovaného rámečku a poslední možnost jestli chceme využít kvalitnější filtrační algoritmus.

4.1 ImageView

Třída ImageView nám pomáhá zobrazit obrázek, který jsme si načetli. Kromě zobrazení obrázku na plátně nám tato třída umožňuje změnu velikosti obrázku a určení, která část obrázku se má zobrazit.

```
ImageView iv = new ImageView();
iv.setImage(image);
```

Nyní si vytvoříme objekt ImageView. Tomuto objektu nastavíme obrázek, který se vykreslí na plátno.

4.2 Pane

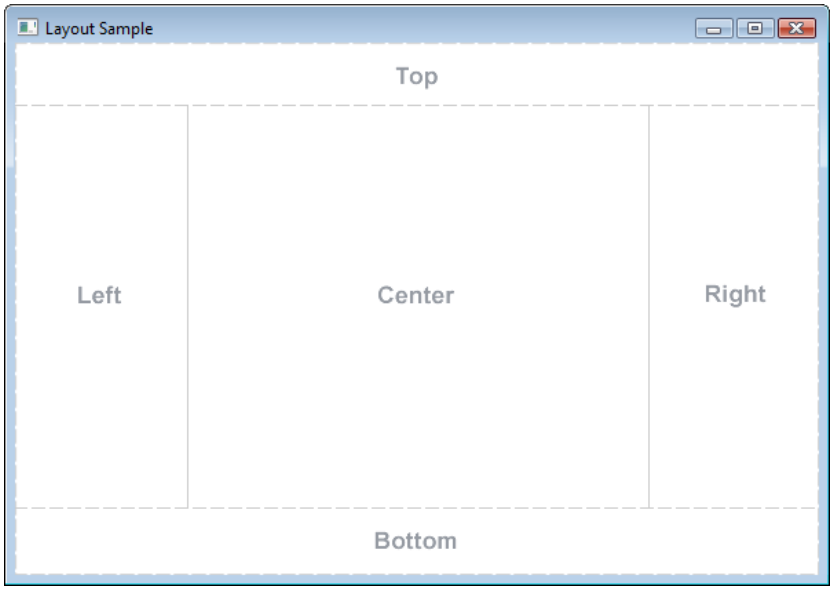
Pane je v JavaFx základní uspořádání pro dispozice, obdoba Panelu v jazyce Java. Tuto třídu můžeme využít v případech, kdy potřebujeme zveřejnit objekty tzv. děti(children), které umístíme na Pane.

```
Pane panel = new Pane();
panel.setStyle("-fx-background-color: white;");
panel.setPrefSize(100,100);
Circle kruh = new Circle(50,Color.RED);
kruh.relocate(20, 20);
Rectangle obdelnik = new Rectangle(100,100,Color.RED);
obdelnik.relocate(70,70);
panel.getChildren().addAll(kruh,obdelnik);
```

Zde vidíme příklad kódu na vytvoření objektu Pane. Vytvoření objektu Pane následné nastavení stylu, kdy pozadí bude nastaveno na bílou barvu setStyle("-fx-background-color: white;"), nastavení předdefinované velikosti setPrefSize(100,100) následné vytvoření kruhu, obdélníku a nastavení jejich velikosti a barvy. Po té změna umístění na panelu obdélníku a kruhu obdelnik.relocate(70,70), kruh.relocate(20, 20). Nakonec přidání do panelu a tím i jejich vykreslení panel.getChildren().addAll(kruh,obdelnik).

4.3 BorderPane

Obdobně jako u Pane se zde jedná o uspořádání dispozic, na rozdíl od objektu Pane, BorderPane je rozdělen na 5 pozic a to horní, dolní, pravou, levou a střední pozici, do všech těchto pozic můžeme vkládat různé objekty, které chce zobrazit. Jsou zde i nastavení u horní a dolní pozice ty objekty (děti), které budou přidány na tyto pozice a budou se měnit podle své předdefinované výšky a šířka se bude nastavovat dle BorderPane. Levá a pravá pozice, objekty, které se zde budou nacházet, šířka přizpůsobena jejich preferované šířce a výška bude prodloužena pro horní a dolní pozici. Středová pozice je k vyplnění volného místa, objekty které se zde budou nacházet, bude se jim přizpůsobovat jak výška, tak i šířka.



Obrázek 12: BorderPane

Zde na tomto obrázku můžeme vidět rozložení pozic BorderPane. Obrázek je použit ze zdroje [5.].

```
BorderPane borderpane = new BorderPane();
MenuBar menuBar = new MenuBar();
HBox hBox = new HBox();
borderPane.setTop(menuBar);
borderPane.setCenter(hBox);
```

Na tomto kódu můžeme vidět umístění objektů, které přidáváme do BorderPane. V prostředku máme umístění hBox, který nám znázorňuje horizontální box, kam se nechávají umísťovat například tlačítka (Button) a nahoře máme umístěn MenuBar, ten znázorňuje klasickou lištu s funkcemi jako například ukončit program.

4.4 Button

Button je v JavaFX jednoduché ovládací tlačítko, které může obsahovat text nebo grafický obrázek. Tlačítko má tři různé režimy ovládání.

Normal: tlačítko normálně zmáčkneme, většinou myší

Default: Defaultní tlačítko je tlačítko, které obdrží tlačítko klávesnice VK_ENTER a stiskne se, pokud tuto klávesnici žádný jiný uzel nepotřebuje.

Cancel: Tlačítko zrušit je tlačítko, které reaguje na zmáčknutí klávesnice VK_ESC, pokud žádný jiný uzel tuto klávesnici na scéně nepotřebuje.

```
Button tlacitko = new Button("Enter");
```

Vytvoření tlačítka s názvem „Enter“.

4.5 HBox

HBox má všechny své objekty nashromážděné ve vodorovné řadě. HBox má vlastnost měnit svou velikost k svým objektům, které vlastní a přizpůsobovat se jim. Můžeme zde zarovnávat objekty, které jsou standardně nastaveny na pozici Pos.TOP_LEFT.

```
Hbox hbox = new hbox(8) ;
hbox.getChildren( )add(new Label("Nazev"));
```

Vytvoření HBox s spacing(8), přidání nového Label s názvem „Název“

4.6 MenuBar

Klasické menu, které zajisté znáte z mnoha aplikací. Většinou se u aplikací umisťuje nahoře. Pro každou nabídku v menu, kterou chceme přidat, se tak přidá do menu ObservableList. Ve výchozím nastavení jsou tyto nabídky reprezentována tlačítky s názvem nabídky, kterou si definujeme.

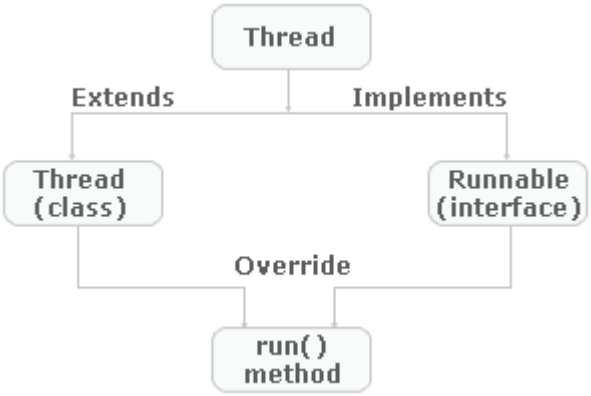
```
Menu menuFile = new Menu("File");
```

```
Menu menuOptions = new Menu("Options");
Menu menuHelp = new Menu("Help");
MenuBar menuBar = new MenuBar();
menuBar.getMenus().addAll(menuFile, menuOptions, menuHelp);
```

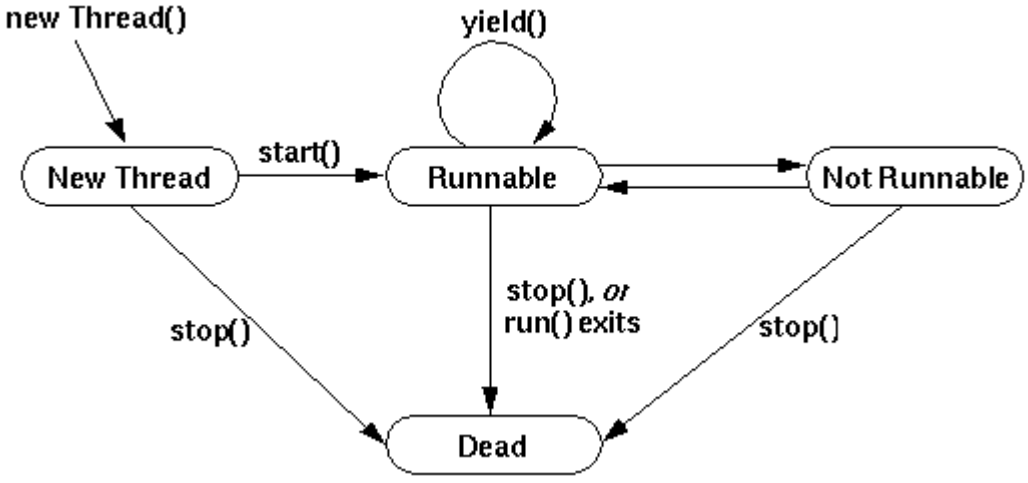
Zde je znázorněn zdrojový kód. Vytvoření objektu MenuBar s položkami File, Options a Help.

4.7 Vlákna – Threads

Java umožňuje multithreading tzv. multiprogramování běh více částí kódu na jednou. Každá tato část běží paralelně a Javě se nazývá vlákno (thread). Podpora vláken v Javě je velmi obsáhlá. Vlákna jsou na sobě nezávislá, každé vlákno vykonává svoji část programu. Vlákna spolu sdílí data procesu. V Javě máme dvě možnosti, jak můžeme postupovat, buď to zvolíme cestu rozšíření třídy Thread, nebo implementujeme rozhraní Runnable. V této prací jsem postupoval implementací rozhraní Runnable a rozšíření metody run. Obrázky jsou čerpány ze zdrojů [8.] [9.].



Obrázek 13: Diagram třídy vlákna



Obrázek 14: Vytvoření nového vlákna

5 Praktická část

Praktická část se zabývá částí programovou, jsou zde zmíněny základní problémy, které v té to aplikaci nastali, jejich postupné řešení.

5.1 Popis programu

V této kapitole vám přiblížím nejzajímavější metody, na které jsem při programování této aplikace narazil.

5.1.1 Třída Wait

public class Wait implements Runnable {

```
    private AsteroidScene asteroidScene;
    private Thread thread;
    private int pom;
    Random rn = new Random();
    public Wait(AsteroidScene as) {
        this.thread = new Thread(this);
        this.asteroidScene = as;
        this.pom = asteroidScene.getSizeOfAsteroidList();
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < pom; i++) {
                Platform.runLater(new Runnable() {
                    @Override
                    public void run() {
                        asteroidScene.putAsteroidOnScreen();
                    }
                });
                Thread.sleep((rn.nextInt(6000) + 3000));
            }
        } catch (InterruptedException ex) {
            Logger.getLogger(Wait.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
    public void statrVlakno() {
        thread.start();
```

```
    }
}
```

V této třídě bych chtěl zmínit zejména metodu *run (), kde* jsem se setkal s jistým úskalím při přidávání objektů na scénu. Bylo potřeba zabezpečit, aby objekt se na scéně vykreslil jen jednou, a také aby se spustila animace a poté se přidal další objekt. V programu je vždy vygenerovaný určitý počet objektů, které se uchovávají v ArrayListu. Při vypsání na scénu se vypsaly všechny objekty a spustila se i jejich animace. Pro zabránění vypisování jsem zvolil použití vláken, abych zaručil, že se dané objekty budou přidávat na scénu postupně a scéna se zatím vykreslí. Při vytvoření vlákna a postupného přidávání jsem narazil na chybu, které neumožňovala přidávání dalších objektů. Metoda *Platform.runLater(Runnable)*, je to metoda, která může být volána z jakéhokoliv vlákna, zajišťuje spuštění aplikace v brzké době, v případě tohoto kódu zaručuje spuštění vlákna a tím pádem spuštění metody *AsteroidScene.putAsteroidOnScreen()*, která vláká objekty postupně na scénu. Metoda Thread.Sleep zaručuje, na jak dlouhou dobu se vlákno uspí a poté vloží další objekt na scénu.

5.1.2 Metoda generateTrasitionAsteroid

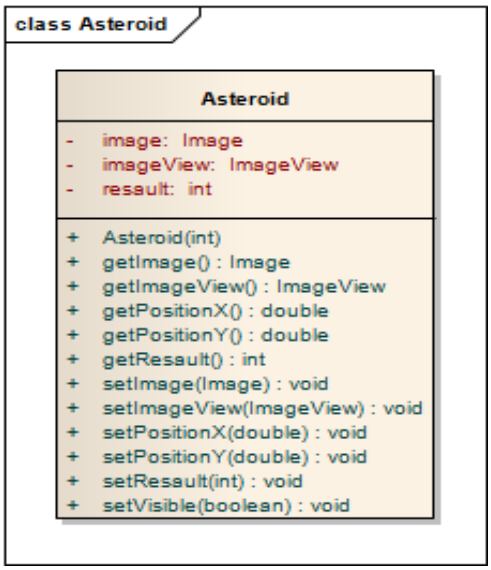
```
private void generateTransitionAsteroid() {
        PathTransition pathTrAsteroid;
        RotateTransition rt;
        if (listOfAsteroids.size() == 0) {
            System.err.println("Prazdno");
        } else {
            for (int i = 0; i < listOfAsteroids.size(); i++) {
                pathTrAsteroid                                          =
generateAsteroidTransition(listOfAsteroids.get(i).getImageView(),
Duration.seconds(3 + i), Duration.seconds(1), listOfPath.get(i));
                rt    =    new    RotateTransition(Duration.millis(5000),
listOfAsteroids.get(i).getImageView());
                rt.setFromAngle(10);
                rt.setToAngle(190);
                rt.setCycleCount(6);
                listOfTravelingAst.add(pathTrAsteroid);
                listOfRotate.add(rt);
                pathTrAsteroid.setOnFinished(new
EventHandler<ActionEvent>() {
                    @Override
                    public void handle(ActionEvent t) {
                        PathTransitio pt = (PathTransition)t.getSource();
                        Node nd = pt.getNode();
                        mainGroup.getChildren().removeAll(nd);
                        final ImageView imageView = new ImageView(IMAGE);
```

```
                        imageView.setViewport(new   Rectangle2D(OFFSET_X,
OFFSET_Y, WIDTH, HEIGHT));
                        final Animation animation = new SpriteAnimation(
                                imageView,
                                Duration.millis(1000),
                                COUNT, COLUMNS,
                                OFFSET_X, OFFSET_Y,
                                WIDTH, HEIGHT
                        );
                        animation.setCycleCount(1);
                        animation.play();
                        mainGroup.getChildren().add(imageView);
                    }
                });
            }
        }
    }
```

Zde máme metodu na generování PathTransition, co je to PathTransition je popsáno v kapitole 4.2. Nejdůležitější částí v tomto kódu je metoda *pathTrAsteroid.setOnFinished(new EventHandler<ActionEvent>()*, která reaguje na dokončení cesty objektu. Při dokončení objektu kdy dorazí na konec své cesty, se objekt se odstraní ze scény příkazem *mainGroup.getChildren().removeAll(nd);* a dále se provede animace výbuchu.

5.2 Tabulky (objektů)

5.2.1 Třída Asteroid



Tabulka 1: Třída Asteroid

V třídě asteroid se nachází tyto metody:

- *Asteroid (int)* – Konstruktor třídy Asteroid, kde se vkládá celočíselná proměnná Integer, kde se uchovává výsledek
- *getImage()* – získání obrázku
- *getImageView()* – získání objektu ImageView
- *getPositionX()* – získání pozice x objektu ImageView
- *getPositionY()* – získání pozice y objektu ImageView
- *getResault()* – získání výsledku
- *setImage(Image)* – nastavení obrázku
- *setImageView(ImageView)* – nastavení objektu ImageView
- *setPositionX(double)* – nastavení pozice x objektu ImageView
- *setPositionY(double)* – nastavení pozice y objektu ImageView
- *setResault(int)* – nastavení výsledku
- *setVisible(boolean)* – nastavení viditelnosti objektu ImageView

5.2.2 Třída AsteroidScene

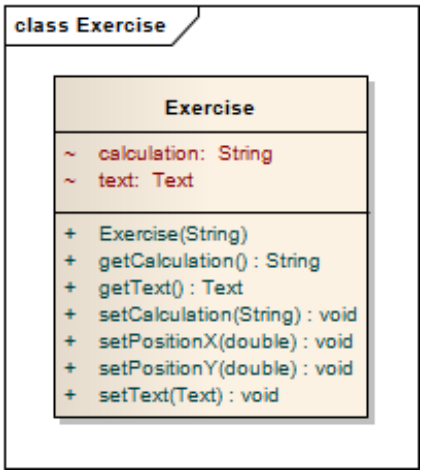
Tabulka 2: Třída AsteroidScene

V třídě AsteroidScene se nachází tyto metody:

- *addHbox()* – přidání ovládacího panelu
- *addMenu()* – přidání menu
- *AsteroidScene()* – konstruktor třídy AsteroidScene

- *clear()* – metoda, která vyprazdňuje všechny pole, ArrayListy a proměnné
- *generateAsteroids(int)* – generování asteroidu, dle daného počtu
- *generateAsteroidTransition(Node,Duration,Duration,Path)* – generování objektu PathTransition, kde do konstruktoru zadáváme uzel, dobu čekání, dobu prodlevy a cestu
- *generateExercise()* – generování příkladu
- *generateExercises(int)* – generování příkladů, dle zadaného počtu
- *generateJourney(int)* – generování cesty
- *generateObjects(int,int,int)*- generování objektů, do konstruktoru se zadávají časové prodlevy
- *generateRotateTransition(int)* – generování rotací
- *generateTransitionAsteroid(int[])* – generování animace pro asteroid, do konstruktoru zadáváme pole intů
- *generateTransitionExcercise(int[])* – generování animace pro exercise, do konstruktoru zadáváme pole intů
- *getSizeOfAsteroidList()* – vracení velikosti pole Asteroidů
- *init(Stage)* – inicializační metoda, kde předáváme skrze konstruktor objekt Stage
- *levels()* – úrovně aplikace
- *mouseEventClicked ()* – nastavení zpracování požadavků při stisku myši u jednotlivých objektů typu Exercises
- *putAsteroidOnScreen()* – vkládání asteroidů na scénu
- *setButtonExercises()* – nastavení herních tlačítek

5.2.3 Třída Exercise

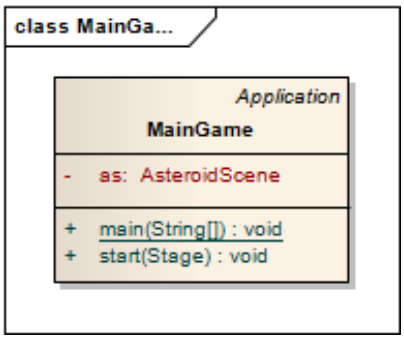


Tabulka 3: Třída Exercise

V třídě Exercise se nachází tyto metody:

- *Exercise(String)* – konstruktor třídy Exercise, předáváme datový typ String
- *getCalculation()* – získáváme výpočet příkladu
- *getText()* – získáváme podobu příkladu v objektu text
- *setCalculation()* – nastavujeme výpočet příkladu
- *setText()* – nastavujeme text
- *setPositionX()* – nastavujeme pozici x objektu Text
- *setPositionY()* – nastavujeme pozici y objektu Text

5.2.4 Třída Main

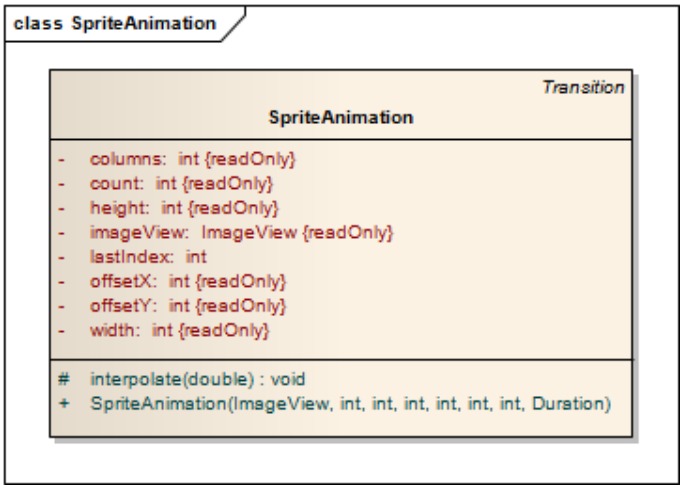


Tabulka 4: Třída MainGame

V této třídě se nachází metody:

- *main(String[])* – klasická spouštěcí metoda pro Java aplikace
- *start(Stage)* – spouštěcí metoda pro JavaFX aplikace

5.2.5 Třída SpriteAnimation

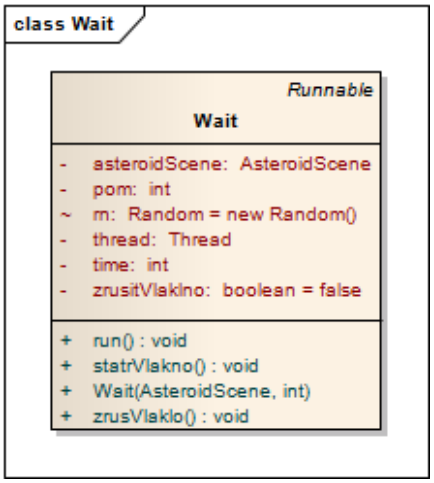


Tabulka 5: Třída SpriteAnimation

Tato metoda je převzatá z návodu, který jsem čerpal ze zdroje [6.]. V třídě SpriteAnimation se nachází metody:

- *interpolate(double)* – výpočet pro animaci obrázku
- *SpriteaAnimation(ImageView,int,int,int,int,Duration)* – konstruktor třídy SpriteAnimation, kde se zadává jako první argument, objekt ImageView, následně souřadnice x, souřadnice y, šířka, výška a doba zpoždění

5.2.6 Třída Wait



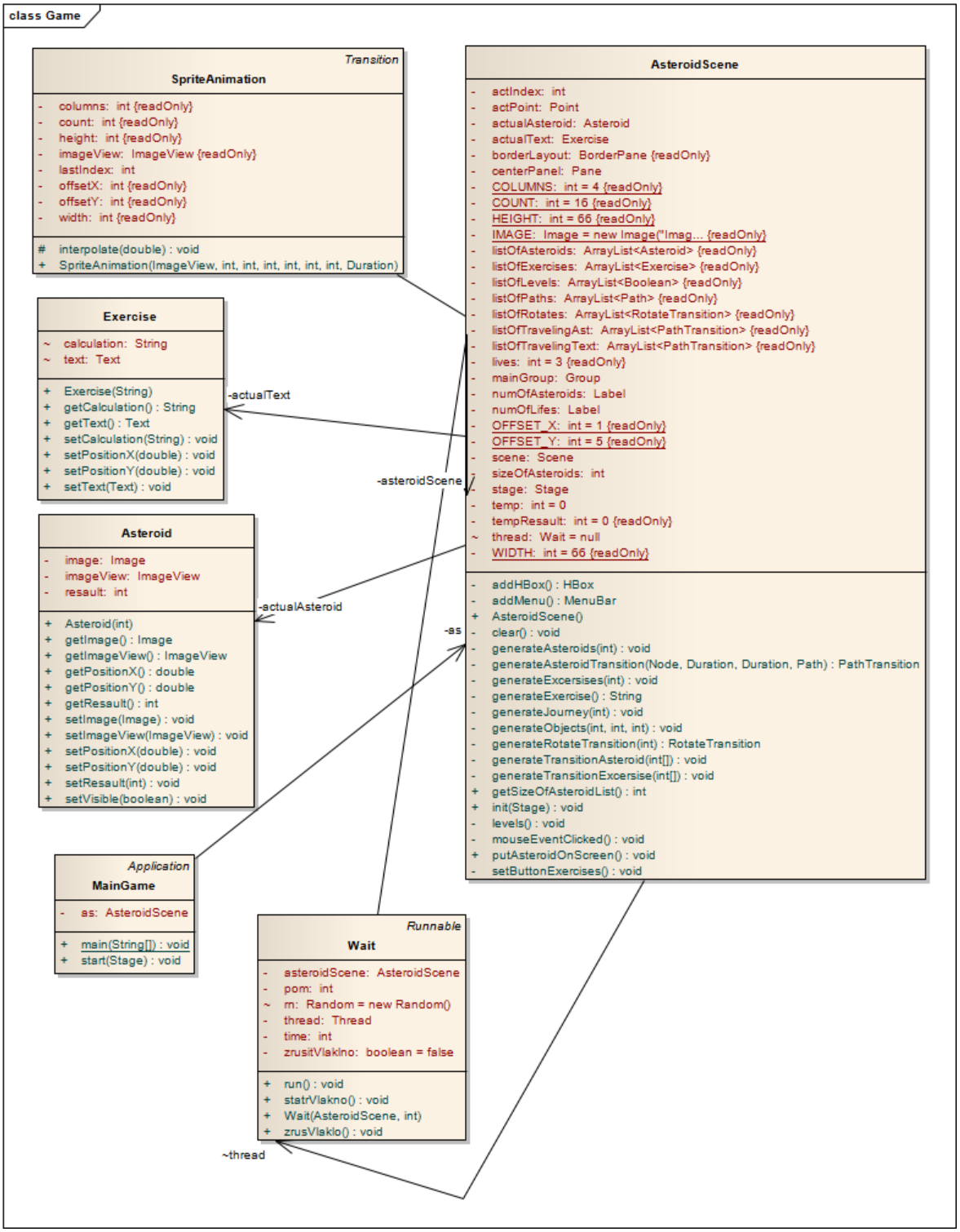
Tabulka 6: Třída Wait

Třída Wait obsahuje metody :

- *run()* – kód vlákna a přidané metody, kterou vlákno ovládá
- *startVlakno()* – spouštění vlákna

- *Wait(AsteroidScene,int)* – konstruktor třídy Wait, skrze parametr přidáváme třídu AsteroidScene, kde využíváme její metody *putAsteroidOnScreen()* a dále časovou konstantu v datové proměnné int

5.3 Class Diagram



Obrázek 15: Class diagram

Závěr

Jak jsem se již zmiňoval v úvodu, tato práce vznikla myšlenkou přiblížit dětem výuku matematiky zábavnou formou. Nejprve jsem potřeboval zanalyzovat situaci na našem trhu, jaké programy nabízí konkurence a co všechno dnešní aplikace umožňují. Analýzou a testováním jsem zjistil, že všechny tyto aplikace jsou zastaralé s porovnáním jiných počítačových her dnešní doby, které se již téměř přibližují reálnému světu, jak po stránce grafické, tak po stránce ovladatelnosti. Proto jsem chtěl vytvořit aplikaci, která by nebyla statická, ale naopak dynamická, aby uživatel, respektive žák, byl nucen reagovat a sledovat situaci na obrazovce.

Podařila se mi naprogramovat engine, kde je uživatel nucen reagovat na létající objekty připomínající asteroidy a každý tento asteroid si nese textový obraz, který znázorňuje příklad. Uživatel musí myší označit daný objekt a v dolní liště se mu objeví tlačítka s výsledky. Při zmáčknutí správného tlačítka s výsledkem se objekt zničí, při špatném zmáčknutí tlačítka se asteroid také zničí, ale uživatel přijde o jeden ze tří životů, které má hráč od začátku hry. Pokud tyto životy vyprchají, hra skončila.

V diplomové práci bych rád tento engine vylepšil s větším množstvím matematických příkladů, dále bych postupně přidal těžší příklady zacílené také na žáky druhého stupně základních škol, kde budou mít studenti možnost řešit náročnější příklady. Cílem je proniknout ne jen do klasické matematiky a početních úloh, ale také do geometrie, konstrukční úlohy, statistiky, objemu a povrchu tělesa. Dalším směrem, kterým bych chtěl tento engine směřovat, je fyzika, jelikož JavaFX podporuje i 3D animace, které jsou velmi dobrým nástrojem pro vytvoření enginu na výuky fyziky, které můžeme využít pro vytvoření příkladů a přiblížit tak i tuto vědu studentům zábavnou a poučnou formou.

Literatura

1. SIERRA, Kathy, BATES, Bert. Head First Java. Vydání druhé. O?Reilly Media, 2005. ISBN 978-0596009205.

2. HEROUT, Pavel: Učebnice jazyka JAVA. České Budějovice : KOPP, 2010.

   ISBN 978-80-7232-398-2.

3. FLANAGAN, David. Programování v jazyce Java. Praha : Computer Press, 1997.

   ISBN: 80-85896-78-8.

4. Java Platform, Standard Edition (Java SE) 8. *Java Platform, Standard Edition (Java SE) 8* [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

5. JavaFX 2 Documentation. *JavaFX 2 Documentation* [online]. 2011, 2014 [cit. 2014-05-04]. Dostupné z:http://docs.oracle.com/javafx/2/

6. HEINRICHS, Michael. Java Code Geeks: JavaFX: Creating a Sprite Animation. *Java Code Geeks: JavaFX: Creating a Sprite Animation* [online]. 2012 [cit. 2014-05-05]. Dostupné z: http://www.javacodegeeks.com/2012/03/javafx-creating-sprite-animation.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+JavaCodeGeeks+%28Java+Code+Geeks%29

7. Stahuj.cz: Výukové hry. *Stahuj.cz: Výukové hry* [online]. 1999, 2014 [cit. 2014-05-05]. Dostupné z:http://www.stahuj.centrum.cz/hry_a_zabava/vyukove/

8. Rose India: Thread Creation. *Rose India: Thread Creation* [online]. 2007 [cit. 2014-05-05]. Dostupné z:http://www.roseindia.net/java/thread/thread-creation.shtml

9. CAMPIONE, Mary a Kathy WALRATH. Thread State. *Thread State* [online]. 1997 [cit. 2014-05-05]. Dostupné z:http://www.science.uva.nl/ict/ossdocs/java/tutorial/java/threads/states.html