UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

HTML5 canvas a SVG jako moderní technologie pro tvorbu animací do webových stránek

Lukáš Janda

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš Janda**
Osobní číslo:	**I11382**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**HTML5 canvas a SVG jako moderní technologie pro tvorbu animací do webových stránek**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části práce je nutné provést přehled problematiky implementace animací do webových stránek obecně.
Primárním cílem diplomové práce je návrh a implementace aplikace umožňující pohodlné sestavení jednoduchých animací využívajících HTML5 canvas a SVG. Takto sestavené animace bude umět aplikace, volitelně také včetně základního ovládacího panelu (začátek, konec, změna rychlosti animace, vyexportovat v podobě HTML5 kódu, uložit a následně zpřístupnit pod unikátní URL. Návrhu a implementaci aplikace by měla předcházet rešerše dostupných technologií, na jejímž základě si student vybere optimální platformu pro vývoj.
V závěrečné části práce bude provedena analýza zatížení hardware, zejména pak CPU a GPU, uživatele při přehrávání animace v různých dominantních webových prohlížečích (MS Internet Explorer, Mozilla FireFox, Google Chrome). Součástí analýzy bude též porovnání se zatížením hardware při použití jiných technologií umožňujících přehrávání podobných animací (např.: Adobe Flash, Microsoft Silverlight, Java Plug-in)

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

CAGLE, Kurt. HTML5 Graphics: with SVG & CSS3. O'Reilly Media, 2011. 120 s. ISBN 978-1-4493-0447-8.
W3C pracovní specifikace HTML5: [online]. 2011 [cit. 2011-10-11]. Dostupné z WWW: [http://dev.w3.org/html5/spec/Overview.html]
W3C specifikace: SCALABLE VECTOR GRAPHICS (SVG) [online]. 2011 [cit. 2011-10-11]. Dostupné z WWW: [http://www.w3.org/Graphics/SVG/
WAGSTAFF, Sean. Animation on the Web. 1st edition. Peachpit Press, 2000. 512 s. ISBN 978-0201696875.

Vedoucí diplomové práce: **Ing. Jan Hřídel**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

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 28. 4. 2013 Lukáš Janda

Poděkování

Na tomto místě chci poděkovat hlavně mé rodině za podporu při studiu i vytváření této diplomové práce a také vedoucímu mé práce panu Ing. Janu Hřídelovi za trpělivost, cenné rady a vedení mé diplomové práce.

Anotace

Tato diplomová práce obsahuje základní informace o problematice webových animací. Také popisuje implementaci praktické části této práce, tedy aplikace, která generuje animace ve formátu HTML5 a SVG. Dále jsou vygenerované animace podrobeny testům, které zkoumají náročnost na HW v jednotlivých prohlížečích.

Klíčová slova

HTML5, canvas, SVG, animace, JavaScript, XML

Title

HTML5 canvas and SVG as modern technology to create animations for web pages.

Annotation

This thesis contains basic information about web animations. It also describes the practical part of this thesis thus implementation of an application that generates animations in HTML5 and SVG. Then the generated animations are tested on the performance of the hardware in different browsers.

Keywords

HTML5, canvas, SVG, animation, JavaScript, XML

Obsah

Seznam zkratek **8**

Seznam obrázků **9**

Seznam tabulek **9**

Úvod **10**

1 Využití animací na webu **11**

1.1 Co je to animace 11

1.2 Proč využívat animace 11

1.3 Jak využívat animace na webu 12

1.4 Technologie tvorby animací 14

1.4.1 SVG 15

1.4.2 Flash 17

1.4.3 Animované obrázky GIF 19

1.4.4 JavaScript 19

1.5 Srovnání jednotlivých metod 20

2 Platformy pro vývoj aplikace **20**

2.1 Výsledné animace ve formátu HTML5 a SVG 21

2.2 Uživatelská část pomocí C# 21

2.3 Uživatelská část pomocí php 22

2.4 Další alternativy 23

2.5 Srovnání a výběr architektury 23

3 Vývoj aplikace **24**

3.1 Základní popis 24

3.2 Architektura 25

3.2.1 Uživatelská část 26

3.2.2 Veřejná část 29

3.2.3 Databáze 29

3.3 Implementace 30

3.3.1 Uživatelská část 32

3.3.2 Veřejná část 35

3.4 Popis funkcionality 36

3.4.1 Hlavní menu 36

3.4.2 Vkládání prvků 38
3.4.3 Vlož obdélník 39
3.4.4 Vlož kruh 39
3.4.5 Vlož Bézierovu křivku 40
3.4.6 Vlož kvadratickou křivku 41
3.4.7 Vlož text 41
3.4.8 Vlož Obrázek 42
3.4.9 Vlož koťátko 42
3.4.10 Log a základní operace 43
3.4.11 Vlastnosti kreslícího plátna 44
3.4.12 Vlastnosti animace 44
4 Uživatelské testování aplikace 45
5 Testování výkonu 46
5.1 Naměřené hodnoty 48
5.2 Porovnání zátěže 49
Závěr 51
Literatura 52
Příloha A – Animace ve formátu XML 54
Příloha B – Animace ve formátu SVG 56
Příloha C – Výsledky uživatelského testování 57

Seznam zkratek

HTML5	HyperText Markup Language verze 5
SVG	Scalable Vector Graphics
GIF	Graphics Interchange Format
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Seznam obrázků

Obrázek 1 - Průběh animace [1] 11
Obrázek 2 - Animovaný graf [17] 12
Obrázek 3 - Animovaný layout [18] 13
Obrázek 4 - Animovaná reklama [19] 14
Obrázek 5 - Rozdíl mezi vektorovou a bitmapovou grafikou [4] 15
Obrázek 6 - Kreslení pomocí SVG [9] 17
Obrázek 7 - Vývojové prostředí Adobe Flash Catalyst [10] 18
Obrázek 8 - Animovaný obrázek GIF [11] 19
Obrázek 9 - Výsledná animace. Zdroj: autor 21
Obrázek 10 - Implementace v C# a php. Zdroj: autor 22
Obrázek 11 - Implementace v php. Zdroj: autor 23
Obrázek 12 - Architektura aplikace. Zdroj: autor 26
Obrázek 13 - Návrh databáze. Zdroj: autor 30
Obrázek 14 - Časová osa. Zdroj: autor 34
Obrázek 15 - Editor animací. Zdroj: autor 36
Obrázek 16 - Hlavní menu. Zdroj: autor 37
Obrázek 17 - Vkládání tvaru. Zdroj: autor 38
Obrázek 18 - Vlastnosti nového tvaru. Zdroj: autor 38
Obrázek 19 - Změna vlastností tvaru. Zdroj: autor 39
Obrázek 20 - Obdélník. Zdroj: autor 39
Obrázek 21 - Kruh. Zdroj: autor 39
Obrázek 22 - Definice Bézierovy křivky[22] 40
Obrázek 23 - Bézierova křivka. Zdroj: autor 40
Obrázek 24 - Kvadratická křivka. Zdroj: autor 41
Obrázek 25 - Text. Zdroj: autor 42
Obrázek 26 - Obrázek. Zdroj: autor 42
Obrázek 27 - Služba placekitten.com [20] 43
Obrázek 28 - Služba placehold.it [21] 43
Obrázek 29 - Log a základní operace. Zdroj: autor 44
Obrázek 30 - Vlastnosti kreslícího plátna. Zdroj: autor 44
Obrázek 31 - vlastnosti animace. Zdroj: autor 45
Obrázek 32 - Složitější animace. Zdroj: autor 48
Obrázek 33 - Jednoduchá animace. Zdroj: autor 48

Seznam tabulek

Tabulka 1 - Testovací sestava 1 46
Tabulka 2 - Testovací sestava 2 46
Tabulka 3 - Naměřené hodnoty zatížení počítače na testovací sestavě 1 49
Tabulka 4 - Naměřené hodnoty zatížení počítače na testovací sestavě 2 49

Úvod

Úvodní část bude obsahovat rešerši o technologiích pro tvorbu animací. Tyto technologie se v poslední době velmi rychle vyvíjí a vznikají díky tomu nové funkce a nové metody, jak animace vytvářet. Proto ze začátku rozeberu animace jako takové, poté využití animací na webu. Detailně se zaměřím na technologie tvorby animací do webových stránek, zpočátku s technologií SVG, která bude poté implementována v praktické části. Pak budu pokračovat s flashovými animacemi, které jsou v současné době pro SVG velkým konkurentem.

V další části bude navržena a implementována aplikace, která bude využívat HTML5 a SVG pro tvorbu animací. V tomto návrhu aplikace bude provedena analýza možných platforem pro implementaci a následně vybrána ta nejvhodnější. Pro komfortní zobrazení kreslených tvarů a pro vytvoření uživatelského prostředí pro tvorbu animací bude použito již zmíněné HTML5. Výsledná aplikace bude umožňovat uživateli vytváření vlastních jednoduchých animací, které pak bude možné exportovat ve formátu SVG a zpřístupnit je na internetu pod unikátní URL adresou. Mimo to bude uživateli umožněno ukládání i načítání rozpracovaných animací. Tyto animace se budou uchovávat v databázi MySQL. Z důvodu bezpečnosti bude vytvořeno jednoduché rozhraní pro registraci a přihlášení uživatele.

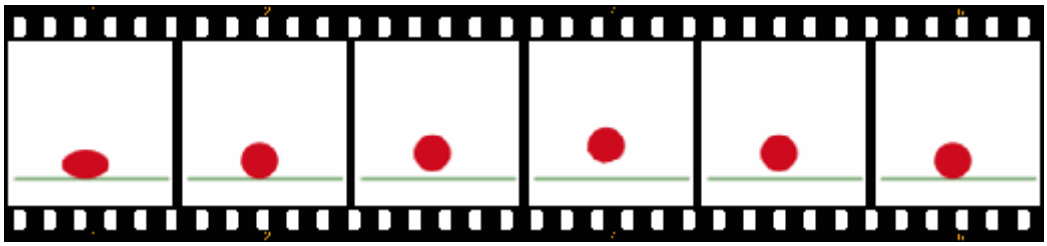
V závěrečné části této práce pak bude provedeno srovnání flashových animací a animací ve formátu SVG, vytvořených vyvinutou aplikací. Na internetu bude vybráno několik zajímavých flashových animací, pro které bude možné vytvořit ve vyvinuté aplikaci ekvivalentní animaci ve formátu SVG. Poté se bude zkoumat zatížení počítače v různých prostředích při přehrávání těchto dvou typů animací a následně bude provedeno srovnání naměřených hodnot a vyhodnocení obou technologií.

1 Využití animací na webu

Jak již bylo řečeno v úvodu, tato diplomová práce se bude týkat animací, které jsou primárně určené k použití na webových stránkách. V následujících kapitolách popíšu, jakým způsobem efektivně a moderně vytvářet animace pro webové stránky. Animace je totiž potřeba tvořit nejen přehledně, a aby hezky vypadaly, ale také optimalizovat jejich velikost i chod. Nepřehledné animace totiž návštěvníka stránek nezaujmou a velké a náročné animace se zase budou načítat pomalu a budou vyžadovat větší výkon počítače. V současné době se pro tvorbu animací používá velké množství technologií, některé v této kapitole popíši a uvedu jejich klady i zápory.

1.1 Co je to animace

Animace je technika, jak uvést statické objekty do pohybu. Skládá se z jednotlivých snímků, které když se promítají dostatečně rychle za sebou, tak lidské oko bere jejich změnu jako plynulou. Díky tomu objekty vypadají, že se pohybují tak, jak to můžeme vidět na Obrázku 1. [1]



Obrázek 1 - Průběh animace [1]

Abychom docílili plynulého pohybu, musíme pustit animaci rychlostí 24 snímků za vteřinu. Při této rychlosti bude brát naše oko obraz jako spojitý a nám se zobrazí poskakující míček. [1]

1.2 Proč využívat animace

Důvodů, proč využívat na internetových stránkách animace, je hned několik. Například když chceme:

- zaujmout návštěvníky,
- umístit na web reklamu,
- dynamicky prezentovat důležité informace,
- znázornit průběh nějaké funkce či graf,
- vytvořit webovou hru nebo jakoukoliv jinou interaktivní součást webu.

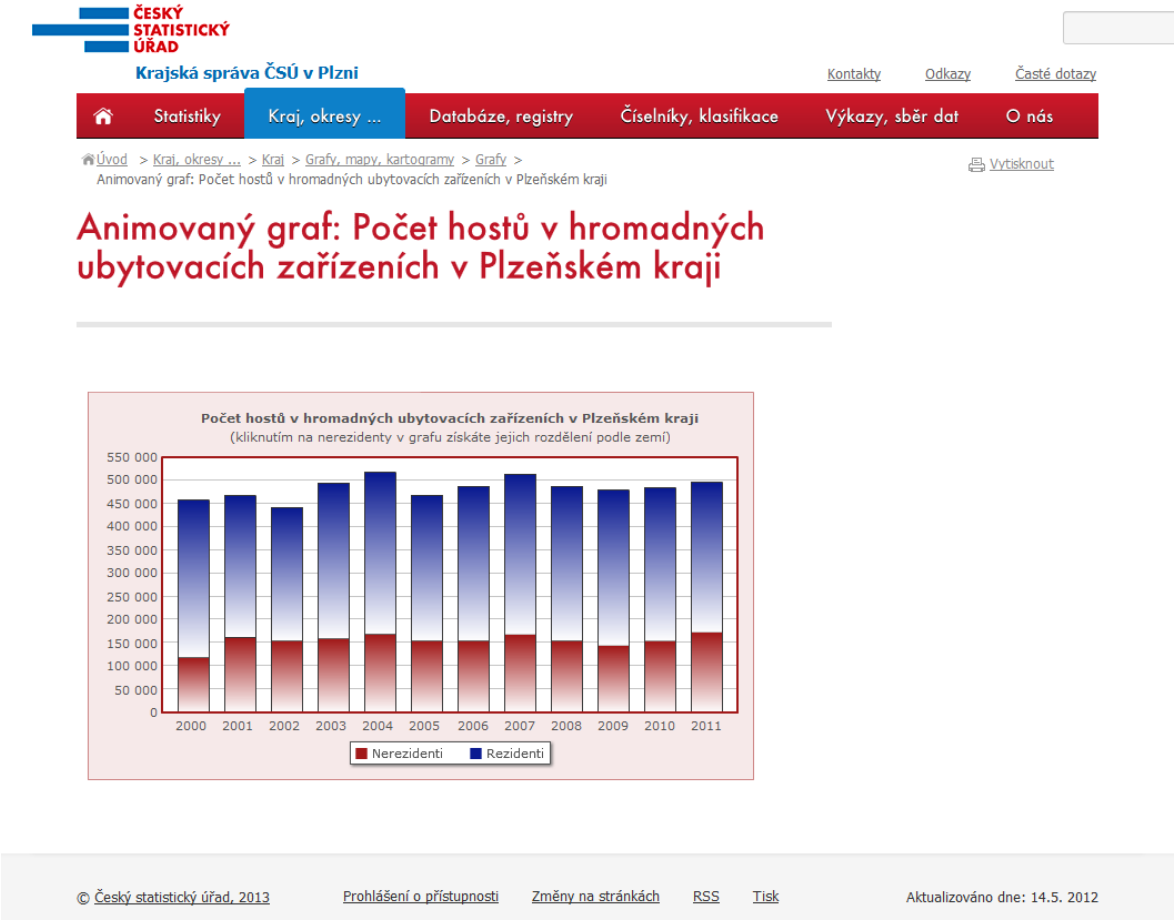
Díky animacím jsou totiž naše stránky živější a více připomínají desktopové aplikace. Tato metoda nám také přináší velké množství rozmanitých způsobů, jak prezentovat informace nebo vytvořit zajímavější layout[1] stránek. V současné době se animace stávají základním

[1] Layout - šablona webové stránky

stavebním prvkem při tvorbě internetových reklam. Efektivně totiž dokážou prezentovat větší množství informací i na menších plochách, neboli bannerech[2].

1.3 Jak využívat animace na webu

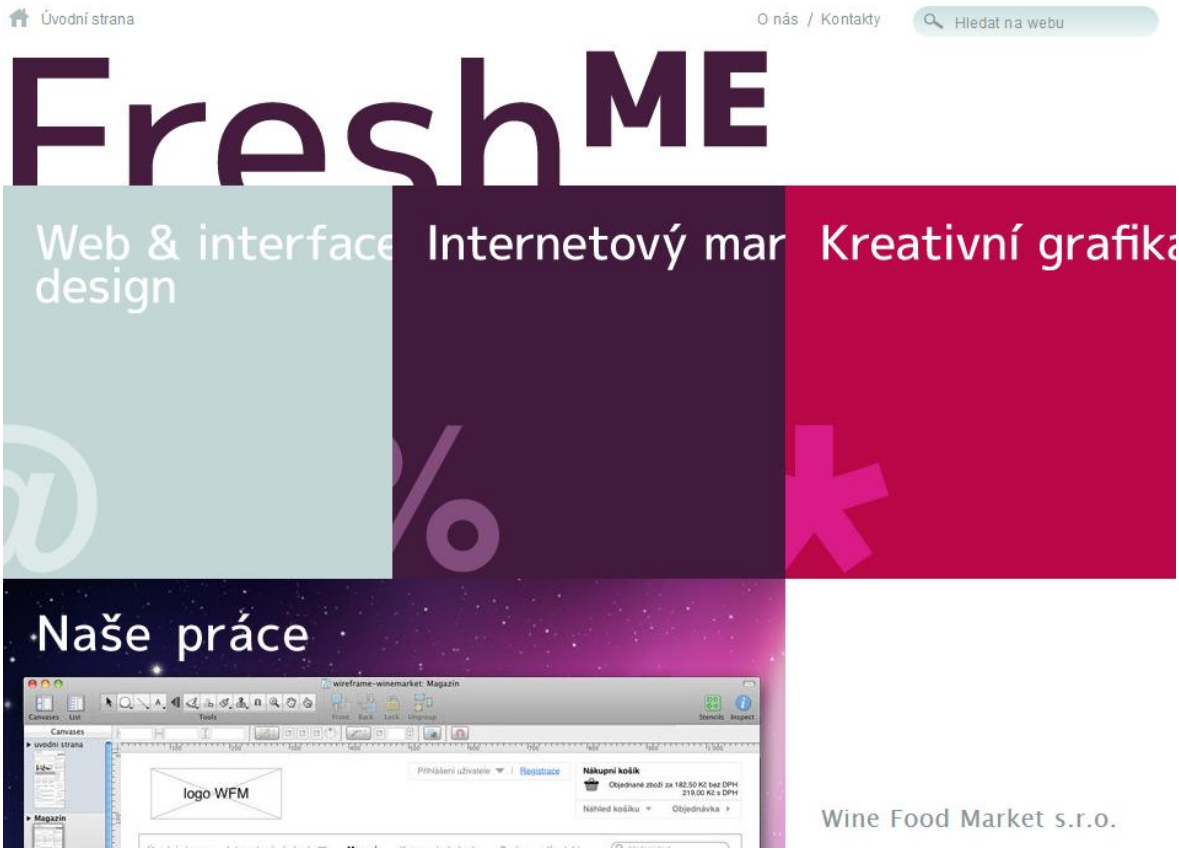
Jednou z možností, jak můžeme animace využívat, je vytvořit je jako prvky, které vložíme do našich webových stránek. Tím se stanou součástí obsahu a běžně zobrazují například průběh matematické funkce, hodnoty v grafu nebo další informace. Použití animovaného grafu můžeme vidět na Obrázku 2. Hlavně na tuto metodu se bude zaměřovat aplikace, kterou budu vyvíjet v rámci praktické části této diplomové práce.



Obrázek 2 - Animovaný graf [17]

Další možností může být začlenění animací do šablony stránek, čímž uděláme vzhled stránek zajímavějším. Můžeme přidat různé měnící se efekty, ať už do loga, nebo třeba vytvořit efektní měnící se menu nebo různá textová pole. Kompletně animovaný layout můžeme vidět například na Obrázku 3. Technologie použití animovaných částí webu zde přináší hned několik výhod. Vytváří velmi zajímavý design stránek, ale také umožňuje na menší ploše zobrazovat větší množství informací, tím, že postupně zviditelňuje a skrývá jednotlivé texty.

[2] Banner - typicky obdélníkový prostor pro zobrazení reklamy



Obrázek 3 - Animovaný layout [18]

Jiným odvětvím, ve kterém se animace velmi často využívají, je internetová reklama. V současné době tvoří velkou část obsahu webových stránek a vzhledem k množství reklamy na internetu se její tvůrci snaží návštěvníky na sebe upoutat a získat si jejich pozornost. Proto přichází s různými dynamickými efekty a způsoby, jak reklamní plochy udělat výraznými a zajímavými. Použití v praxi můžeme vidět na Obrázku 4. Výraznou animovanou reklamu od společnosti McDonald's můžeme vidět v horní a pravé části webu. Zde se snaží získat si pozornost návštěvníka možností přetáhnout jednotlivé produkty z horní části do čtverce vpravo a získat tak informace o daných produktech. Následně pak dojde k přesměrování na stránky inzerenta. Podobnou koncepci využívá velké množství inzerentů, kteří nejdříve návštěvníka nalákají na jednoduchou chytlavou hru a poté ho přesměrují na vlastní stránku.

Obrázek 4 - Animovaná reklama [19]

1.4 Technologie tvorby animací

Pro tvorbu animací na webové stránky můžeme využít velké množství technologií. Každá z nich se hodí na něco jiného a každá má svoje klady i zápory. Při rozhodování, kterou použijeme, bychom ze začátku určitě měli zhodnotit jednak naše znalosti a také jakou animaci budeme tvořit. Pokud například máme zkušenosti s tvorbou animací pomocí SVG[3] a animaci, kterou chceme vytvořit, je možné pomocí této technologie bez problémů udělat, je zbytečné využít například Flash, který je komplikovanější a vyžaduje přehrávač. Této skutečnosti se v současné době snaží využít tvůrci programovacího jazyka HTML a ve verzi 5 proto přinášejí prvky, které umožňují podobné funkce jako Flash. Hlavní výhoda programovacího jazyka HTML je nativní podpora novějších prohlížečů, takže není potřeba přehrávač. Detailněji tyto technologie popíšu v následujících kapitolách.[5]

Rozdíl je také ve vektorové a bitmapové grafice. Vektorová bere obrazovou scénu tak, že ji rozloží na jednotlivé prvky a shromažďuje informace o jejich tvaru a transformacích. Z těchto informací se pak vypočítá výsledná scéna, tedy to, co vidí uživatel. Výhodou této technologie je menší velikost animací, tudíž menší datový přenos pro uživatele. S tím však

[3] popisovací jazyk, který popisuje 2D vektorovou grafiku pomocí XML [2]

přichází také hlavní nevýhoda a tou je nutnost vypočítat výslednou scénu a tedy větší zátěž na hardware počítače. Tuto metodu využívají například technologie SVG a Flash, které popíšu v následujících kapitolách. Oproti vektorové grafice bitmapová rozdělí animační scénu na jednotlivé snímky, které jdou za sebou (viz. Obrázek 1) a tyto snímky na jednotlivé pixely[4], u nichž určí barvu. Dalo by se říct, že výhody a nevýhody jsou oproti vektorové grafice přesně opačné. Výhoda tedy je, že je tato metoda méně náročná na hardware počítače, protože pouze zobrazí obrázky tak, jak jdou za sebou a nemusí je vypočítávat. Nevýhoda je ovšem větší datová náročnost, protože se přenáší k uživateli celé obrázky. Tyto obrázky jsou téměř vždy komprimovány, takže velikost výsledné animace není až tak velká. Oproti vektorové grafice a jejím animacím tu však rozdíl je a celkem znatelný. Další rozdíl mezi těmito metodami je i velmi dobře viditelný například na Obrázku 5. Při zvětšení vektorového tvaru, tedy kruhu, zůstane jeho kvalita stejná, protože se pouze daný tvar vykreslí jinak. Stejná zůstane i velikost daného tvaru, na rozdíl od bitmapové grafiky, kde se velikost zvětší, protože zvětšený tvar bude reprezentovat větší množství pixelů. Hlavně ale dojde ke zhoršení kvality, protože dojde k dopočtení barvev jednotlivých pixelů, které se počítají většinou z krajních bodů a jak vidíme, není ani zdaleka tak kvalitní, jako u vektorové grafiky. [4][5]

Obrázek 5 - Rozdíl mezi vektorovou a bitmapovou grafikou [4]

Obecně se tedy musíme při tvorbě animací rozhodnout, jakou grafiku použijeme, zda vektorovou, nebo bitmapovou a podle toho pak některou z dostupných technologií. I když to vypadá, že jsem v předchozím odstavci bitmapovou grafiku více kritizoval, než chválil, i ona má své výhody. U animace, která se nebude skládat z lehce popsatelných prvků, tedy písem, geometrických tvarů a podobně, ale naopak se bude skládat například z fotek a složitějších efektů, představuje bitmapová grafika zajímavou alternativu.

1.4.1 SVG

Tento standard, který popisuje prvky vektorového grafického formátu, byl vyvinut pod záštitou W3C[5]. Využívá se ke značkování dvojrozměrné grafiky pomocí XML. Výsledný datový soubor tedy obsahuje pouze popsané jednotlivé prvky, které pak zpracuje prohlížeč a vykreslí grafickou scénu. Stejně jako u Flashe se využívá vektorová reprezentace a proto je ideální pro tvorbu webové grafiky. Velikost takto vytvořených obrázků, či animací, je

[4] Pixel - jeden bod obrazu, nejmenší částice
[5] Mezinárodní organizace, která vyvíjí webové standardy

pak rovněž mnohonásobně menší, než při použití bitmapové grafiky a animovaných obrázků. [7]

Výhody:

- malá velikost - ukládají se pouze zdrojová data, nikoliv grafické prvky,
- velké možnosti, díky otevřenému formátu,
- možnost použití různých stylů - např. různé fonty pro texty, barvy pro tvary, atd.,
- postupné načítání stránky - podobně, jako to dělají prohlížeče při zpracování HTML, při načítání z pomalé sítě, se načtené části ihned můžou zobrazovat a není nutné čekat na načtení celé stránky, což může být zdlouhavé a nepříjemné, jako je tomu například u Flashe, [7][8]
- zapisuje se pomocí XML.

Nevýhody:

- prozatím tento standard není tak rozšířený,
- prozatím nenabízí nějaké vyspělejší vývojové prostředí, jako například Flash. Tohle je, stejně jako předchozí bod, ale pouze otázka času.[8]

Hezky si tuto technologii můžeme předvést na jednoduchém příkladu. Pomocí SVG si vytvoříme kruh, u kterého definujeme jeho styl, tedy výplň a barvu a šířku okraje, dále jeho vlastnosti, konkrétně průměr, pozici na ose X a Y. K tomu nám stačí následující kód.

```
<circle style="fill: none; stroke: black; stroke-width: 3px" cx="100px"
cy="100px" r="60px"/>
```

Podobně budeme postupovat při tvorbě obdélníku, pouze místo průměru, který jsme definovali u kruhu, určíme jeho výšku a šířku.

```
<rect style="fill: red; stroke: red" x="40px" y="20px" width="70px" he-
ight="40px"/>
```

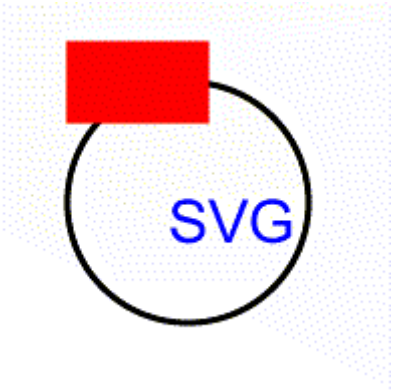
Rovněž můžeme do výsledného obrázku vkládat text. To provedeme pomocí následujícího kódu, ve kterém nejprve definujeme souřadnice na osách X a Y a poté jeho styl, tedy velikost písma a barvu. Zde vidíme, že máme párový tag text, do kterého se pak píše výsledný řetězec, který se nakonec zobrazí.

```
<text x="90px" y="120px" style="font-size: 30px; color: blue">
      SVG
</text>
```

Výsledný dokument, který nám všechny 3 prvky vykreslí do obrázku o velikosti 200 na 200 pixelů, pak vypadá následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="200px" height="200px">
      <desc>Image</desc>
      <g>
            <circle style="fill: none; stroke: black; stroke-width: 3px"
            cx="100px" cy="100px" r="60px" />
            <rect style="fill: red; stroke: red" x="40px" y="20px"
            width="70px" height="40px" />
            <text x="90px" y="120px" style="font-size: 30px; color:
            blue">
                  SVG
            </text>
      </g>
</svg>
```

Jak bude vypadat výsledná kompozice, nám ukazuje Obrázek 7. [9]



Obrázek 6 - Kreslení pomocí SVG [9]

1.4.2 Flash

Flash je technologie, která se zabývá tvorbou vektorové animované grafiky. Její největší výhodou a zároveň vlastností, díky které se Flash stal tak oblíbený, je malá velikost výsledných animací. To je možné díky uchovávání informací o animaci a grafických prvcích pomocí vektorové grafiky. Narozdíl od animovaných obrázků GIF, které byly v minulosti velkým konkurentem, Flash neuchovává samotné pixely, ale grafická scéna se rozdělí na jednotlivé tvary a uchovávají se informace o nich a jejich transformacích, jak již bylo popsáno v předchozí kapitole. [3]

Flash využívá vlastní programovací jazyk, zvaný ActionScript. Ten vychází z Java Scriptu a umožňuje vytvářet různé interaktivní prvky, např. události po kliknutí myší, nebo různé akce. V současných verzích už se jedná o poměrně vyspělý, objektově orientovaný programovací jazyk. [3][6]

Výstupní animace můžeme realizovat pomocí dvojice formátů:

- svf - tento formát se vyznačuje malou velikostí, obsahuje pouze ActionScripty, ale k jeho spuštění je potřeba Flash Player,

- exe - formát, který je určen pro spouštění přímo v operačním systému Windows, mimo webový prohlížeč. Má však mnohonásobně větší velikost, protože má Flash Player implementovaný v sobě. [3]

Tato technologie má nesporné výhody:

- platformní nezávislost - nezáleží v jakém operačním systému nebo webovém prohlížečí animaci spouštíme,
- stejný vzhled na všech platformách - díky tomu, že animaci nevykresluje prohlížeč, ale flashový přehrávač, je výsledek vždy stejný,
- jednoduchý návrh designu, např. oproti HTML se nemusí složitě programovat, ale jednoduše se nakreslí,
- menší datové přenosy - nepřenáší se celé animace, pouze ActionScripty, které pak zpracovává prohlížeč,
- přehrávání i streamování videí i zvuků. [3][5]

S výhodami však přichází také celkem zásadní nevýhody:

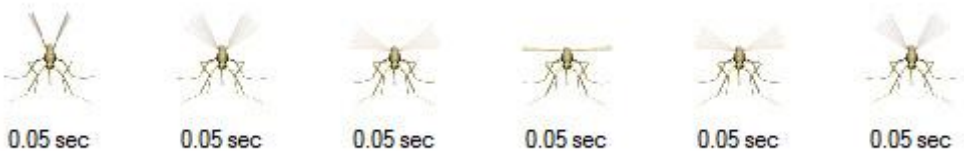
- potřebujeme Flash Player,
- nemožnost použití funkcí prohlížeče, jako např. tlačítka "Zpět", vyhledávání "Ctrl + F" a dalších
- vyšší hardwarové nároky,
- obsah je prakticky neviditelný pro internetové vyhledávače. [3]

Technologií Flash se v této práci budu zabývat pouze teoreticky, ale animace ve Flashi později budou sloužit ke srovnání náročnosti s animacemi v HTML5 a SVG. Pro vytvoření představy, jak vypadá vývojové prostředí Adobe Flash Catalyst, se můžete podívat na Obrázek 7.

Obrázek 7 - Vývojové prostředí Adobe Flash Catalyst [10]

1.4.3 Animované obrázky GIF

Tato technologie, která už se příliš často nepoužívá, vytváří animace pomocí po sobě jdoucích obrázků ve formátu GIF. V praxi to funguje tak, že si animátor vytvoří obrázky v nějakém grafickém editoru a tyto obrázky pak v animačním programu spojí, nastaví časování a uloží je jako GIF. Když si výslednou animaci rozložíme na jednotlivé obrázky, bude to vypadat jako na Obrázku 8. Přehledně tu vidíme změny mezi jednotlivými obrázky a také časování. Ve výsledku se tyto obrázky budou měnit po pěti setinách sekund a to komára přivede do pohybu. Tyto obrázky jsou však limitované samotným formátem GIF, který zvládá pouze 256 barev. To přináší drobnou výhodu ve formě menší velikosti výsledného souboru, ovšem výrazně na úkor kvality. Animované gify pak umožňují mít paletu barev pro každý snímek různou, což umožňuje alespoň trochu kvalitu obrázků zvýšit, protože je nutné uložit menší počet barev a tím pádem se může zaměřit pouze na určité odstíny. Když například budeme animovat rotující zeměkouli, obrázky, které budou znázorňovat oceány, budou mít dominantní barvu modrou a naopak obrázky, které budou znázorňovat pevninu, budou mít hlavní barvy v okolí zelené a žluté. [11][12]



Obrázek 8 - Animovaný obrázek GIF [11]

I přesto, že už se tento formát animací moc nepoužívá, má stále své výhody:

- animace může mít průhledné části,
- použití stejné, jako u klasických obrázků,
- platformní nezávislost - nezáleží na operačním systému nebo webovém prohlížeči, všude se zobrazí stejně,
- prohlížeč nepotřebuje žádný doinstalovaný plug-in ani přehrávač. [11]

Vzhledem k tomu, že se ale jedná o poměrně zastaralou technologii, má i své nevýhody:

- animace mají pouze 256 barev,
- u složitějších animací výrazně větší datový přenos,
- složitější tvorba animací a efektů. [11]

1.4.4 JavaScript

JavaScript jako takový neslouží přímo pro tvorbu animací, spíše se zaměřuje na okamžitou změnu a validaci obsahu webových stránek. Nicméně je možné ho použít i pro kreslení různých tvarů a vkládání obrázků na element Canvas jazyka HTML5. Při větším úsilí samozřejmě bude možné animace pomocí tohoto programovacího jazyka vytvořit, ale řešení rozhodně nebude úplně optimální. Hlavní důvod je absence pokročilejších funkcí určených

ke tvorbě složitějších animací. A proto pokud budeme chtít nějakou složitější animaci vytvořit, bude potřeba například dopočítávat průběhy různých funkcí a ručně překreslovat animované tvary na vypočítané souřadnice. Proto je výhodnější tuto platformu použít pouze pro okamžitou změnu obsahu a pro animování použít technologii k tomu přímo určenou. I přesto má tvorba animací pomocí JavaScriptu své výhody:

- tvorba jednoduchých animací, například změna velikosti tvaru, jeho natočení, výměnu obsahu a podobně, bude velmi jednoduchá,
- velikost takové animace bude rovněž velmi malá, protože se bude jednat pouze o zdrojový kód a případné obrázky, nebo obsah, který se bude měnit.

Důležitější ovšem budou nevýhody:

- velmi obtížná realizace složitějších transformací,
- v případě, že má uživatel ve svém prohlížeči vypnutý JavaScript, žádná animace se mu nezobrazí.

1.5 Srovnání jednotlivých metod

Obecně je srovnání jednotlivých metod velmi obtížné. Každá je totiž specializovaná na jiný typ animací, byla vytvořena v jiné době a každá má své výhody i nevýhody. Zatímco například při použití SVG, budou nároky hlavně na aktuální internetový prohlížeč, aby podporoval tuto technologii. U Flash animací bude zase potřeba Flash Player. JavaScriptové animace sice bude podporovat drtivá většina internetových prohlížečů, ale uživatel může mít vypnutý JavaScript a efekty nebudou takové, jako u předchozích dvou metod. Jediný typ animací, který se dá prohlížet snad ve všech prohlížečích a na všech zařízeních jsou animace ve formě animovaných GIFů. Ty jsou ovšem zase výrazně omezeny nízkou kvalitou. Obecně tedy nelze říci, který typ animace je nejlepší a který nejhorší. Každý z nich ovšem své uplatnění ještě najde. Záleží tedy na konkrétním využití.

2 Platformy pro vývoj aplikace

Jak již bylo popsáno v úvodu, aplikace bude mít za úkol vytvářet animace a ty pak exportovat a zpřístupnit na internetu pod jedinečnou url adresou. Aplikaci proto bude z hlediska implementace nejlepší rozdělit na dvě části. Jednu, která se bude starat výhradně o tvorbu animace a druhou, která bude zpřístupňovat a spravovat vytvořené animace. Vzhledem k tomu, že současné technologie umožňují použít komunikaci i mezi aplikacemi na různých platformách, například využitím webových služeb, není tedy nutné, aby obě části aplikace byly implementovány na stejné platformě.

Jelikož ne všechny platformy pro vývoj aplikace jsou pevně dané v zadání, bylo potřeba se rozhodnout, jaké technologie využít při implementaci. Ohledně formátu animací a zobrazování průběžných stavů animace to bylo jednoduché. Tahle část v zadání specifikována byla a jedná se konkrétně o kombinaci HTML5 a SVG. HTML5 bude použito pro průběž-

né vykreslování vytvořených objektů a SVG pro popsání vytvořených animací. Specifikována ale nebyla platforma pro vývoj samotné aplikace, která animace bude vytvářet. Jako nejlepší možné volby se nabízejí kombinace php a C# nebo kompletní implementace pomocí php.

2.1 Výsledné animace ve formátu HTML5 a SVG

Jak již bylo uvedeno v úvodu této kapitoly, pro zobrazování rozpracovaných animací bylo určeno použití HTML5 a pro výsledné animace popisovací jazyk SVG. V praxi to znamená, že při tvorbě animací, se jednotlivé vytvořené prvky uchovávají v aplikaci, jako instance nějakého definovaného objektu a tyto objekty se pak kreslí pomocí JavaScriptu na element jazyka HTML5, zvaný Canvas. Výhoda této technologie je v možnosti realizace drag'n'drop manipulací s vytvořenými objekty. Například při posunování objektů se jednoduše pouze mění souřadnice daného prvku a vše se vykresluje na Canvas. Toto kreslení probíhá s velmi malou periodou, aby lidské oko zaznamenávalo vykreslování jako plynulé a při každém kreslení se nejprve kreslící plátno smaže a poté se na něj vykreslí všechny objekty. Tento postup samozřejmě není úplně optimální, ale vzhledem k faktu, že je tento postup prováděn pouze při tvorbě animace, to není až tak zásadní.

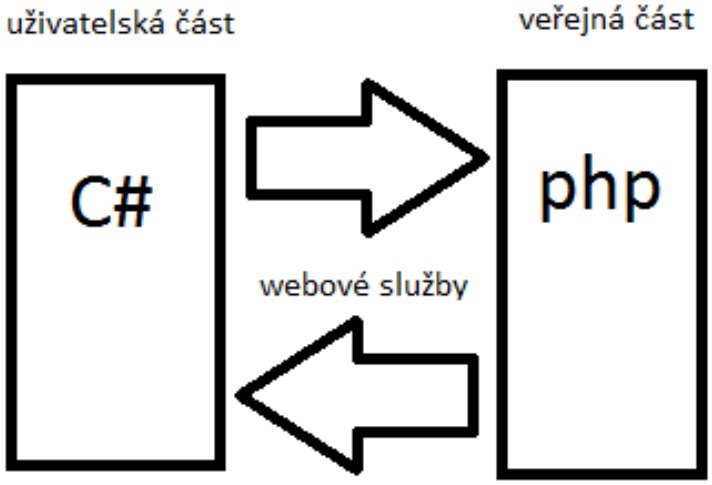
Obrázek 9 - Výsledná animace. Zdroj: autor

Výsledná animace je pak popsána popisovacím jazykem SVG, který vychází z jazyka XML a který zpracovává internetový prohlížeč. Výstupem je tedy SVG dokument, který kompletně popisuje vytvořenou animaci. Bude proto potřeba implementovat generátor, který z vytvořených instancí jednotlivých prvků bude generovat kód podle standardů SVG. Tento kód se pak zobrazí jako animace, aby měl uživatel možnost si výsledek své práce prohlédnout, ale také v textové podobě, aby si mohl tento kód zkopírovat a vložit do svých internetových stránek.

2.2 Uživatelská část pomocí C#

Při použití tohoto návrhu by uživatelská část určená k tvorbě animací byla implementována v jazyce C#. Konkrétně by se jednalo o desktopovou aplikaci, která by po vytvoření animace vygenerovala její popis v jazyce SVG a ten pomocí webové služby předala veřejné části. Tato veřejná část by byla implementována v jazyce php a to hned z několika důvodů. Vytvořené animace by bylo potřeba zpřístupňovat online pod jedinečnou url adresou, takže by bylo potřeba vytvořit webovou stránku, která tohle bude zabezpečovat. K tomu je php přímo určeno a jelikož by se nejednalo o velmi složité řešení, jeho potenciál by byl naprosto dostačující. Další výhoda je umístění na internet, neboli využití webhostingu, kterých je

na internetu velké množství a není problém najít použitelný hosting zdarma. Výhoda tohoto návrhu by spočívala v použití C# jako kreslící aplikace, protože je zde velká podpora pro kreslení tvarů. Drobná komplikace by bylo pouze vyřešení komunikace obou částí aplikace pomocí webových služeb. Ty by nebyly nijak složité, protože by šlo v podstatě pouze o předání vygenerovaného kódu a zaslání odpovědi o výsledku této akce. Celkové řešení, oproti kompletní implementaci v php, by bylo o něco složitější, protože řešení této komunikace v php odpadá. Výhodou tohoto návrhu by byl způsob využívání uživatelské části. Uživatel by nemusel pro práci s ní být připojen k internetu, protože by se jednalo o desktopovou aplikaci, kterou by měl nainstalovanou na svém počítači. Pouze pro zveřejnění výsledných animací by se musel připojit na internet a použitím webových služeb animace zveřejnit. S tím spojenou nevýhodou by byla ale nutnost aplikaci instalovat na každé zařízení, kde by chtěl uživatel tuto aplikaci používat.

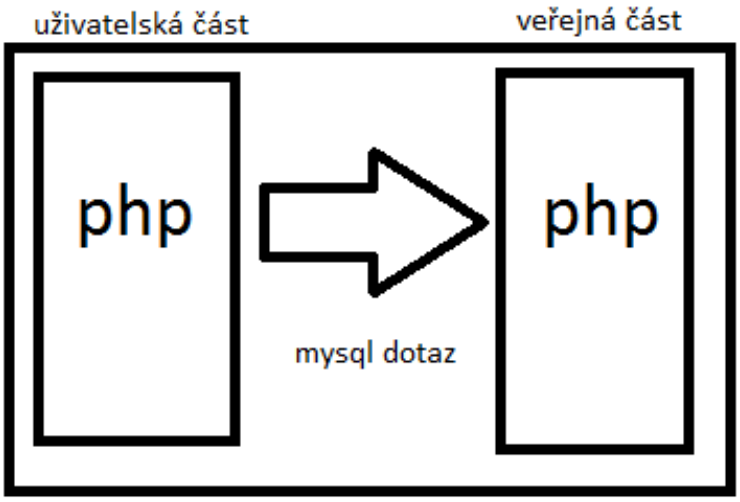


Obrázek 10 - Implementace v C# a php. Zdroj: autor

2.3 Uživatelská část pomocí php

U tohoto návrhu by byly obě části implementovány v php. Způsob implementace veřejné části byl už popsán u předchozího návrhu, a jelikož se to v rámci možností jeví jako nejlepší možné řešení, počítalo by se s implementací veřejné části v php i u tohoto návrhu. Komunikace mezi jednotlivými částmi aplikace by tedy probíhaly například SQL dotazem. To by výrazně ulehčilo propojení obou částí aplikace. Jednalo by se tedy kompletně o webové řešení. Uživatelská část by měla za úkol vytvářet animace pomocí HTML5 a JavaScriptu. V této části by se po dokončení animace vygeneroval její kód, který by se uložil do databáze zavoláním příslušného dotazu. Veřejná část by pak měla za úkol načítat animace z databáze, respektive jejich kód v nich uložený, a zobrazovat je. Drobné komplikace by u tohoto návrhu mohly nastat v oblasti kreslení jednotlivých tvarů a manipulací s nimi v uživatelské části. Jelikož je HTML5 poměrně mladé, mohl by toto být kámen úrazu. Další výhodou tohoto návrhu pak je distribuce aplikace. Tím, že by se kompletně umís-

tila na internet jako webová stránka, by se uživatel vyhnul instalaci programu a mohl by ji využívat online a v podstatě také na jakémkoliv zařízení. Tato aplikace by také byla platformně nezávislá, protože by pro její použití byl potřeba pouze webový prohlížeč, které už dnes najdeme například i v chytrých televizích, takže by zde byla daleko větší škála zařízení, na kterých by šlo aplikaci používat. Nevýhodou tohoto řešení by ovšem byla nutnost internetového připojení, i když to už je v dnešní době bráno jako samozřejmost, a proto tato skutečnost není až tak zásadní komplikací.



Obrázek 11 - Implementace v php. Zdroj: autor

2.4 Další alternativy

Dalších alternativ, jakou koncepci použít, je samozřejmě několik. Aplikaci by bylo možné implementovat také pomocí Javy a jejích webových technologií. Ovšem vzhledem k velmi malým znalostem této technologie, téměř nulovým zkušenostem s ní a nenalezení žádných významných výhod této platformy, které by učinily použití výhodnější, byla samotná analýza této technologie zavržena. Výhodnější by tato technologie samozřejmě byla v případě, že by se jednalo o komerční projekt, kde Java je dostupná zdarma a s ní i velké množství vývojových prostředí. A to hlavně ve srovnání s technologií C#, kde cena Visual Studia, které je výborným nástrojem pro vývoj na této platformě, už není zanedbatelná. V případě využití studentských licencí, které by na tvorbu této práce šly použít, toto ale nebylo relevantní. Proto bylo rozhodnuto držet se předchozích dvou architektur, tedy php a C#.

2.5 Srovnání a výběr architektury

Při výběru architektury bylo potřeba shrnout si zjištěné poznatky a rozhodnout se, jaká platforma bude použita. Po vyloučení Javy z výběru potencionálních platforem, zde zbyly php a C#. Obě tyto platformy se jevily jako zajímavé možnosti a obě měly své klady i zápory. Důvody pro implementaci kompletně pomocí php tedy byly:

- jednodušší komunikace obou částí aplikace,
- snadnější distribuce,
- větší znalosti v této oblasti,
- platformní nezávislost.

Na druhou stranu přesvědčivé argumenty měla i architektura založená na implementaci uživatelské části pomocí jazyka C# a to konkrétně:

- pro používání aplikace není vyžadováno internetové připojení,
- podpora kreslení v C#.

Pro oba návrhy zde záměrně nejsou uvedeny nevýhody, protože v podstatě výhoda jednoho návrhu je nevýhoda toho druhého. Z tohoto důvodu by se jednotlivé vlastnosti pouze opakovaly. Po shrnutí a důkladném promyšlení všech aspektů této analýzy se jako lepší volba zdála kompletní implementace v php. Hlavním důvodem tohoto rozhodnutí byla jednodušší komunikace mezi jednotlivými částmi aplikace, tedy fakt, že nebude nutné používat webové služby. Výrazně se také ulehčí použití aplikace, protože odpadne i nutnost aplikaci instalovat, jelikož poběží na internetu jako webová stránka. Ani nevýhody návrhu založeném na kompletní implementaci v php se nakonec neprokázaly jako zásadní a i toto nakonec výrazně ulehčilo rozhodování. Nevýhodu v podobě kreslení tento návrh vynahrazuje velkým množství návodů a precizně zpracovanou internetovou dokumentací pro php i JavaScript a samotnou nutnost internetového připojení už dnes nepovažuji jako zásadní problém.

3 Vývoj aplikace

3.1 Základní popis

Podle zadání bylo potřeba vyvinout aplikaci, ve které je možné si vytvořit animaci bez znalostí jazyka HTML nebo SVG. Tuto aplikaci bylo potřeba udělat uživatelsky co nepřívětivější, ovšem místy bylo nutné od tohoto záměru mírně ustoupit. Jednalo se většinou o nedostatečnou funkčnost JavaScriptu, nebo velmi komplikovanou realizaci některých funkcí a vylepšení. Asi jako největší problém se ukázalo získávání uživatelských podmětů, tedy ovládání aplikace. Komfortní a v dnešní době už v podstatě běžné, se považuje ovládání aplikace pomocí kláves. Například mazat označené prvky klávesou Delete, zrušit označení klávesou Esc, nebo potvrzení různých nastavení Enterem. Zde se ovšem projevily limity JavaScriptu, protože nebylo možné stisk těchto kláves nějakým způsobem zachytávat, a proto bylo jediné řešení od této funkcionality upustit. To samozřejmě neznamená, že není možné tuto funkcionalitu JavaScriptem realizovat. V tomto případě to ovšem bylo velmi komplikované, a proto bylo praktičtější řešit ovládání pomocí myši. Prvky se vkládají na plátno na místo, kam uživatel dvakrát klikne levým tlačítkem myši, a vlastnosti prvků se definují pomocí HTML formulářů. Tyto formuláře jsou ovšem využívány pouze jako

vstupy, ze kterých se pak JavaScriptem čtou hodnoty. Pouze zřídkakdy dojde ke klasickému odeslání formuláře. To se děje pouze při manipulaci s již vygenerovaným SVG kódem výsledné animace, například v náhledu pro zobrazení,nebo pro ukládání do databáze. K odesílání formulářů nedochází hlavně proto, že při odeslání formuláře se odešlou data na server, ten je zpracuje a zobrazí výsledek. Vzhledem k tomu, že celá aplikace běží u uživatele, jelikož JavaScript se nevykonává na straně serveru, nýbrž u klienta, byla by veškerá práce ztracena. Proto jsou veškeré operace vykonávány u klienta a v reálném čase. Když se tedy shrnou vlastnosti aplikace jako takové, umožňuje uživateli jednoduše si vytvořit animaci, uložit ji do databáze, případně si načíst již uloženou animaci a generovat ze svých animací SVG kód, který pak může vložit do svých webových stránek. K tomu, aby uživatel mohl aplikaci používat, se musí přihlásit. Proto je zde registrace uživatelů, ovšem je možné využít i univerzální účet host, který je veřejný. Animaci si může rovněž exportovat ve formátu XML, ze kterého jde poté zpětně animace načíst. Tuto alternativu je možné využít v případě, že uživatel využívá veřejný účet a nechce ukládat animace na server. Důkladněji je veškerá funkcionalita popsána v kapitole 3.4 Popis funkcionality.

3.2 Architektura

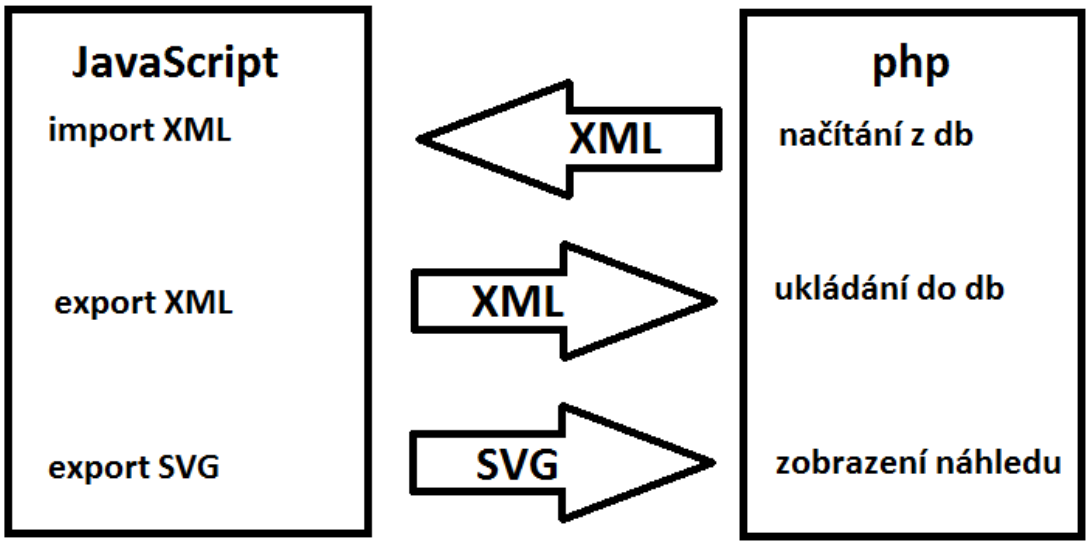
Jak již bylo zmíněno výše, zvítězila nakonec realizace aplikace pomocí jazyka php a postavení celé architektury na něm. Od původního konceptu rozdělení funkcionality na uživatelskou a veřejnou část, se implementace a plánování držely tohoto konceptu. Jediná funkce, která nakonec nebyla zahrnuta do plánování, ani implementace, bylo veřejné zobrazení všech vytvořených animací. Konstrukčně by to nebylo až natolik složité, nicméně byla tato funkcionalita značně komplikovaná a velmi obsáhlá, a proto bylo v rámci první fáze vývoje výhodnější tuto funkčnost prozatím neimplementovat.

Aplikace se proto dělí v podstatě na dvě části. Jedna se stará kompletně o vytvoření a sestavení animací. To probíhá kreslením na HTML5 Canvas pomocí JavaScriptu. Tato část pak rovněž generuje XML a SVG kódy, ve kterých jsou uložené veškeré informace o animaci.

Druhá část je pak postavená na php a slouží k ukládání a načítání animací z a do databáze. Také je zde jednoduché uživatelské rozhraní, které umožňuje uživateli se přihlásit, měnit své údaje a odhlásit se.

Propojení si v praxi můžeme představit tak, že máme první část, která představuje spodní vrstvu aplikace, která zpracovává uživatelovy akce, a druhou část, která představuje horní vrstvu aplikace, která pak zobrazuje výsledky. Jednotlivé vrstvy budou podrobněji představeny v následujících podkapitolách. To, jak jsou propojeny, je možné vidět na následujícím obrázku. Vrstvy mezi sebou komunikují tak, že si předávají informace ve formátu XML nebo SVG. V případě, že chceme animaci uložit, předá spodní vrstva XML kód s informacemi o animaci horní vrstvě. Ta pak tento kód uloží do databáze. V případě, že chceme naopak animaci načíst, je postup přesně opačný. Horní vrstva načte XML kód popisující animaci z databáze a předá ho spodní vrstvě, která z něj importuje animaci. V případě, že chceme animaci zobrazit, je postup úplně stejný jako při ukládání, pouze se pře-

dává SVG kód a ten se pak zobrazí uživateli. Buď jako čistý kód, který si pak uživatel může zkopírovat a vložit do svých webových stránek, nebo jako náhled, kde uživatel vidí konkrétní animaci.



Obrázek 12 - Architektura aplikace. Zdroj: autor

3.2.1 Uživatelská část

Uživatelskou částí je myšlena ta část aplikace, která slouží k vytvoření samotné aplikace. Je kompletně postavena na JavaScriptu a provádí veškeré drag'n'drop operace, výpočty, generování kódů, export i import. Je řešena objektově a obsahuje dvě hlavní objektové třídy.

První z nich je třída Tvar, která reprezentuje jednotlivé objekty, které se budou vykreslovat. Tyto tvary v sobě nesou informace o typu, velikosti, barvách, ohraničení a souřadnicích v rámci jednotlivých časových stavů. Vzhledem k tomu, že se objekty během animování můžou pohybovat, je potřeba zaznamenat pro každý časový stav polohu objektu. Pro většinu prvků stačí zaznamenat pouze jeden bod, tedy jeho souřadnice X a Y. Například Bézierova křivka ale potřebuje body 4. Počáteční, koncový a dva řídící. Tyto body se ukládají v polích, kde index jednotlivých záznamů vyjadřuje údaj o časovém stavu.

Druhá třída se jmenuje CanvasState a vyjadřuje v podstatě animaci jako takovou. Obsahuje pole prvků, které představují jednotlivé tvary, informace o plátně, tedy jeho rozměry, ale také informace o animaci samotné, jako je například její délka, nebo aktuální čas v rámci animace. Tato třída se stará o vykreslování aktuálních časových stavů animace. Vykresluje veškeré prvky, které jsou v ní uložené do elementu s názvem Canvas, jenž je novinkou HTML5. Tento prvek si můžeme představit jako kreslící plátno, na které pomocí JavaScriptu kreslíme. Samotné kreslení probíhá tak, že se postupně vykreslují všechny prvky v této třídě uložené a to na těch místech, které jsou definované pro aktuální časový stav. Tato třída rovněž ve velmi krátkém intervalu ověřuje, zda je plátno aktuální. V případě, že

není, vykreslí všechny prvky na aktuálních pozicích. Díky tomu i drag'n'drop přesouvání prvků nebo kreslení křivek vypadá plynule.

Dále má tato část za úkol přiřazovat vlastnosti novým tvarům a upravovat existující. Je možné měnit rozměry, ohraničení, barvy, počet opakování v rámci animace a další vlastnosti. Rovněž je možné určit typ animování, zda chceme daný tvar animovat přímo průchodem jednotlivých bodů, nebo po křivce procházející těmito body.

Další funkcí, která je při tvorbě animace velmi praktická, je log. Do tohoto logu se vypisuje záznam při každé provedené akci, například po vložení nového tvaru, přesunutí ho na jiné místo, ale také import a export celé animace.

Postupně se dostáváme k velmi důležité funkcionalitě a tou je import a export. Tyto operace je možné provádět dvěma způsoby a to podle toho, jaký výstup nebo vstup očekáváme. Pokud chceme uložit informace o animaci pro pozdější znovu načtení, použijeme formát XML. Tento formát se využívá také k ukládání a načítání animace z databáze. Formát animací je v XML hezky čitelný. Znázornit to můžeme na jednoduché animaci s modrým kolem, které se ve 3 časových stavech pohybuje a výsledná animace se sedmkrát zopakuje.

```
<?xml version="1.0" encoding="UTF-8"?>
<tvary>
      <tvar typ="kruh" opakovani="7" fill="blue" hodnota="greenx2" anima-
      ce="Přímá" nastaveni="" nastaveni2="" w="50" h="50" id="9401" >
            <stavBod1>
                  <bod1 id="9401" cas="0" x="158" y="119"></bod1>
                  <bod1 id="9401" cas="4" x="218" y="175"></bod1>
                  <bod1 id="9401" cas="8" x="323" y="219"></bod1>
            </stavBod1>
      </tvar>
</tvary>
```

Dokument začíná hlavičkou, která ho definuje jako XML. Potom následuje takzvaný root element, v tomto případě <tvary>, který znázorňuje kořenový element a pro každý XML dokument je povinný. Definuje totiž strukturu, ve které se uchovávají informace, v tomto případě tvary, čímž pokračuje i tato ukázka. Tento element definuje konkrétní tvar a jeho vlastnosti. Vidíme, že se jedná o kruh, jehož animace se bude sedmkrát opakovat, má modrou výplň, 2 pixely široký zelený okraj. Animovat se bude přímo průchodem časových bodů, nikoli podle křivky, jeho průměr je 50 pixelů a id je 9401. To se používá pouze k identifikaci. Dále se dostáváme k další větvi hodnot a tou jsou hodnoty časových bodů, tedy informace o pozicích kruhu v čase. I tyto hodnoty jsou velmi dobře čitelné a vidíme zde, že v čase 0 na začátku animace se kruh nachází na souřadnicích 158 x 119. Následuje přesun, který bude trvat 4 vteřiny a to na souřadnice 218 x 175. Stejně bychom si pak mohli odvodit 3. bod. Nyní už následují pouze ukončovací tagy jednotlivých elementů. Tenhle případ vypadá jednoduše, ale kompletní řešení jednotlivých animací bývá samozřejmě rozsáhlejší. Výslednou animaci ve formátu XML, která zobrazuje vlnobití a obsahuje 3 časové stavy, můžete vidět v Příloze A.

Popisování animací ve formátu XML pro import a export nebylo vybráno náhodou. Jedná se, v současné době, asi o nejpoužívanější formát, jež předává informace napříč aplikacemi s různými platformami. Je snadno čitelný a ve většině programovacích jazyků je již vytvořena podpora pro tento jazyk a je velmi jednoduché s ním pracovat. Rovněž je chápán jako jakýsi standard v oblasti přenosu dat, na jeho principu pracují webové služby a vychází z něho i samotné SVG.

Samotný import a export pak fungují celkem jednoduše. V případě exportu, čili generování XML kódu, se postupně projdou všechny prvky a pro každý zvlášť se vygeneruje sada elementů, jež je popisuje. Výsledek je pak kompletní XML soubor, popisující celou animaci. Tento kód pak můžeme uložit buď do databáze, kde k němu budeme mít přístup a budeme moci animaci kdykoliv otevřít, nebo na disk počítače do XML souboru. Ten pak můžeme použít i v přímém XML importu. Jak již bylo zmíněno, tato možnost je zde pro uživatele, kteří nechtějí ukládat animace do databáze, ale chtějí je mít ve svých dokumentech. Hlavně však pro ty, kteří se rozhodnou neregistrovat a budou používat univerzální veřejný účet. Zde by totiž k uloženým animacím měl přístup kdokoliv, kdo by tento účet používal. Na podobném principu jako export funguje i import. Pouze je proces opačný. Vstupem je XML dokument, kde je popsaná celá animace. Samotný import provádí jednoduchý kompilátor, který postupně přečte informace o animaci a nastaví její základní vlastnosti, jako jsou rozměry plátna a doba animace. Poté postupně prochází jednotlivé tvary a vytváří jejich instance. Zdrojový XML dokument se může číst buď z databáze, při otevírání uložené animace, nebo ze vstupu pro XML import, kam se jednoduše vloží XML kód.

Druhý způsob, jak řešit export, je generovat kód SVG, který se pak používá pro samotné zobrazení animace. Jelikož se pro import kódu používá jazyk XML, nebylo nutné implementovat tuto funkcionalitu i pro SVG, a proto je pro tento typ výstupu použitelný pouze export. Při návrhu aplikace bylo důležité se rozhodnout, zda by nebylo efektivnější výslednou aplikaci ukládat rovnou v jazyce SVG a ve stejném formátu ji i načítat. Vyhnout se kompletně použití jazyka XML. Po krátké analýze této problematiky se ovšem lepší volbou jevilo využití XML a to hned z několika důvodů:

- SVG sice vychází z XML, ale jeho syntaxe je pevně daná, není určená k uchovávání informací, ale popisuje konkrétní animační kroky. Proto by bylo velmi komplikované implementovat kompilátor, který by tento kód rozebral a vytvořil z něj instance jednotlivých objektů,
- u jazyka XML je možné definovat si vlastní strukturu, tedy takovou, aby bylo možné pohodlné získání informace o jednotlivých tvarech a nebylo nutné pracně zkoumat například rozsah animace a časové stavy, jak by tomu bylo u SVG,
- kompatibilita a univerzálnost do budoucna. V případě, že by někdy bylo potřeba propojit vyvíjenou aplikaci s nějakou jinou, bude velmi jednoduché jednu, či druhou aplikaci nastavit tak, aby přijímala výsledný XML dokument, jelikož se jedná o univerzální metodu a je možné libovolně definovat výsledný soubor.

Když se nyní podíváme na výsledný SVG dokument, je opět relativně dobře čitelný a snadno pochopitelný. Nejprve definujeme typ souboru, v tomto případě SVG dokument. Ve stejném elementu jsou ale také informace o rozměrech výsledné animace a okrajích výsledného obrázku, které jsou zde nastavené na tloušťku 2 pixely a vykreslí se černou barvou. Následuje element <g>, který definuje skupinu. V této skupině se pak nachází element kruh, který znázorňuje modrý kruh, jehož průměr je 50 pixelů, výplň modrá a jeho ohraničení bude tlusté 2 pixely a vykresleno zelenou barvou. Jedná se tedy o stejnou animaci, kterou jsem výše popisoval pomoci XML. Součástí této skupiny je ještě samotný element, který tuto skupinu animuje, v tomto případě modrý kruh. Skupiny se používají proto, aby bylo možné použít více transformací a definovat je pouze pro konkrétní elementy. Bylo by zde ještě možné přidat další transformace, jako například otočení, změnu rozměru, zkosení a další. Zde zase vidíme, že se kruh bude pohybovat ze souřadnic 158 x 119, na 218 x 175 a nakonec dorazí na souřadnice 323 x 219. Animace se provede 7 krát a jedno opakování bude trvat 4 vteřiny. Nakonec je opět nutné jednotlivé elementy uzavřít, jak konkrétní skupinu, tak i samotný svg element.

```
<svg xmlns="http://www.w3.org/2000/svg" height="300px" width="400px" sty-
le="border:2px solid black;" version="1.1">
      <g>
            <circle id="9401" cx="0" cy="0" r="50" stroke="green" stroke-
            width="2" fill="blue"/>
            <animateMotion fill="freeze" path="M 158, 119 T 218, 175
            T 323, 219 Z" dur="4s" repeatCount="7"/>
      </g>
</svg>
```

V předchozím odstavci byl popsán jednoduchý příklad výsledné animace v SVG. Reálnou animaci, která zobrazuje vlnobití a je popsaná jazykem SVG je možné vidět v Příloze B a hned na první pohled je patrné, že je složitější a obsáhlejší.

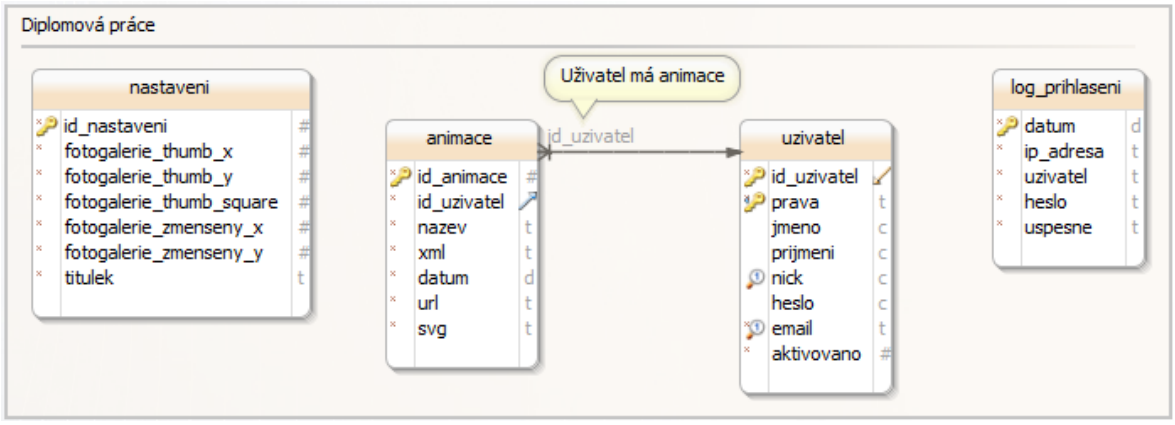
3.2.2 Veřejná část

Veřejnou část si můžeme představit jako jakousi nástavbu uživatelské části. Obaluje základní funkcionalitu a přidává uživatelské rozhraní. Hlavním účelem této části je spolupráce s databází, ukládání a načítání animací, registrace, přihlašování a úprava údajů uživatelů. Jelikož JavaScript nemá přímou podporu pro práci s databází, rozhodl jsem se tuto část implementovat pomocí php. V praxi propojení s uživatelskou vrstvou funguje tak, že od ní například získá XML kód animace a uloží ho do databáze. Při načítání animace si zase načte XML kód z databáze a předá ho uživatelské části pro editaci. Nad rámec spolupráce s touto vrstvou, provádí ještě další úlohy. Jednou z nich je správa uživatelů, což znamená jejich registraci, přihlašování a editaci jejich údajů. Další je pak zobrazení výsledné animace dostupné přes konkrétní URL adresu a asi poslední významnou úlohou je zobrazení Helpu, což je klasický návod použití, který najdeme v drtivě většině aplikací.

3.2.3 Databáze

Databáze zde plní pouze základní úlohu. Konkrétně má shromažďovat informace o uživatelích, zejména jejich přihlašovací údaje, a ukládat animace. Její podoba je velmi jednoduchá. Skládá se v podstatě pouze ze 4 tabulek, z nichž jedna nakonec nebyla využita, ale

bude sloužit pro případné vylepšení do budoucna. Začneme do nejjednodušší tabulky, kterou je log_přihlášení. Tato tabulka v sobě shromažďuje informace o pokusech o přihlášení do aplikace. A to jak úspěšné, tak neúspěšné. Ukládá se zde ip adresa počítače, ze kterého byl pokus o přihlášení uveden, výsledek, zda bylo přihlášení úspěšné a pak zadané hodnoty, jako například jméno a heslo. Ty se uchovávají pro případ, že se někdo bude pokoušet například prolomit zabezpečení zkoušením kombinace uživatelského jména a generováním hesel. To půjde z logu vyčíst a bude možné podniknout proti takovýmto praktikám kroky, které je znemožní, nebo alespoň znepříjemní. Další tabulka v sobě uchovává informace o samotných uživatelích a nese příznačný název uživatel. Opět jsou zde některé položky uvedené, i když pro ně zatím není funkcionalita, jako například práva, aktivováno a email. Je ale velmi pravděpodobné, že v případě dalšího vývoje aplikace by byly využity. Ať už k aktivaci účtu přes mail, jako dokončení registrace, nebo odlišení různých druhů uživatelských účtů. Tato funkcionalita ovšem v základní verzi této aplikace zahrnuta není. Na tuto tabulku navazuje tabulka animace, která zase, jak už název napovídá, uchovává jednotlivé uložené animace. V ní najdeme mimo atributy, uchovávající konkrétní informace o animaci, také hodnotu cizího klíče id_uzivatel, určující konkrétního uživatele, který tuto animaci vytvořil. Poslední z tabulek je tabulka nastavení, která zatím není využita, ale opět je zde potenciální a velká šance uplatnit se v budoucnu. Nese informace o různém nastavení. V nynější podobě tedy hlavně informace o vytvářených obrázcích. Ty by mohly být využity při implementaci funkcionality, která by umožňovala uživatelům nahrávat na server obrázky a ty pak vkládat do animací. Plán budoucí implementace této funkcionality vznikl na základě poznatků z uživatelského testování, které bude detailně popsané v kapitole 4 Uživatelské testování aplikace.



Obrázek 13 - Návrh databáze. Zdroj: autor

3.3 Implementace

Samotná implementace probíhala celkem hladce. Díky výbornému helpu a zkušenostem s programováním veřejné části v php nenastaly žádné komplikace. U JavaScriptu to bylo ovšem lehce rozdílné. Pouze základní zkušenosti zpočátku vyvolávaly lehkou skepsi, ale po pochopení základní syntaxe a zjištění, že se jedná o objektový programovací jazyk, se najednou objevil jeho obrovský potenciál a možnosti. K ovládnutí základních metod tohoto

programovacího jazyka dopomohl i fakt, že na internetu se nachází velké množství různých tutoriálů, návodů a stránek, které specifikují vlastnosti tohoto jazyka. Jedny z nejlepších a nejpřehlednějších jsou bezpochyby w3schools.com[14], kde je popsaná drtivá většina všech funkcí JavaScriptu a ke všem jsou velmi dobře řešené tutoriály a příklady. Stejně tak tento web poskytuje i definici popisovacího jazyka SVG. Celkově by se tyto stránky daly přirovnat k dokumentaci jazyka php[13], která je rovněž velmi dobře řešená a většina php programátorů zde vyhledává podrobnější definice méně často používaných funkcí..

Implementace byla prováděna podle zásad agilního programování, tedy vyvíjení aplikaci pomocí iterativních stavů. Práce byla vždy rozdělena do malých bloků. Začalo se s uživatelskou částí, kde se nejprve řešilo samotné kreslení na Canvas a drag'n'drop editaci tvarů. V úplně prvním bloku bylo cílem vytvořit aplikaci, která bude umět vykreslit obdélník a bude možné s ním pohybovat. Po dokončení a odladění tohoto bloku, se plánování zaměřilo na přidávání další doplňkové funkcionality, což je editace vlastností tohoto tvaru. Až poté, co byla kompletně dokončena editace obdélníka, byl definován další bod a to konkrétně návrh designu samotného animačního programu. V této době už bylo připravené uživatelské prostředí, které se vzhledem podobalo výsledné aplikaci, pouze chyběla funkcionalita. Další krok byla postupná implementace dalších tvarů. Každý tvar byl definován jako jeden blok a až po kompletním odladění, se realizoval další. Po dokončení všech 7 typů tvarů byl navržen další krok a to export a import animace. Nejdříve přišel na řadu export, díky kterému pak bylo k dispozici velké množství pokusných vzorových XML dokumentů pro testování a ladění importu. Po dokončení exportu a importu pomocí XML dokumentů bylo nutné realizovat export výsledné animace ve formátu SVG a při návrhu tohoto kroku došlo k v podstatě jediné chybě, která během tohoto plánování nastala. V prvotním plánování nebyly více zohledněny časové stavy, takže nebylo možné jinak zaznamenávat pozice objektů pro tyto jednotlivé časové stavy. Proto bylo potřeba vrátit zpět a editovat všechny tvary. Tato drobná chyba v plánování, která na druhou stranu nebyla snadno předvídatelná, zdržela vývoj o několik hodin, které byly ztraceny při jejím napravování. Díky definování malých bloků v průběhu implementace a jejich postupnému ladění a testování však byl tento nedostatek objeven velmi brzo a jeho napravení nebylo tak komplikované. V tomto kroku byla také navržena konečná podoba časové osy. Po jejím dokončení už chybělo v podstatě jen vytvořit generátor SVG kódu a po jeho dokončení, už byla v podstatě funkční aplikace, která již byla schopná vytvářet animace ve formátu SVG. Nastal tak čas navrhnout veřejnou část, která bude mít za úkol správu animací a uživatelů. Jelikož funkcionalita nebyla velmi náročná a s plánováním aplikace v jazyce php pomohla několikaletá praxe, celý návrh této části byl proveden v podobě jednoho bloku. Po jeho dokončení už pak následovalo pouze finální testování, sepsání uživatelské dokumentace a odladění odhalených chyb. Dalo by se tudíž říct, že tato metoda programování velmi ulehčila realizaci celé aplikace. A to hlavně z toho důvodu, že bylo možné plánovat menší kousky aplikace a z poznatků získaných při vývoji předchozích kroků, odvozovat důsledky a zohledňovat je při dalším plánování. Až na problém s časovými stavy, který by bylo velmi těžké předvídat, by se klidně dalo prohlásit, že metoda fungovala bezchybně. Opět bylo zajímavé analyzovat a zpětně porovnat, zda by nebylo lepší si před začátkem imple-

mentace aplikace všechno důkladně promyslet a navrhnout celou aplikaci kompletně do detailů. Po zhodnocení dojmů a výhod této metody vyplynulo, že by to však pravděpodobně lepší nebylo. Postup vývoje aplikace byl samozřejmě promýšlen po celou dobu implementace, ale ne do úplně podrobných detailů, které byly řešeny pouze v rámci jednotlivých kroků. To se ukázalo jako obrovská výhoda, protože kvůli malým znalostem JavaScriptu na počátku by větší plánování bylo velmi obtížné. Velké množství problémů bylo potřeba řešit až při jejich realizaci a při té příležitosti se mi naskytla možnost využít jejich vyřešení k dalšímu plánování, což by v případě kompletního plánování do detailů už na začátku nebylo možné, nebo by bylo potřeba naplánovaný návrh měnit. Způsoby implementace jednotlivých částí budou představeny detailněji v následujících podkapitolách.

3.3.1 Uživatelská část

Jak již bylo naznačeno v architektuře uživatelské části, v podstatě celá tahle vrstva aplikace je programovaná v JavaScriptu. Při prvotním plánování, kdy se rozhodovalo mezi php a C#, tedy jestli se bude vyvíjet webová, nebo desktopová aplikace, bylo nakonec rozhodnuto pro php. Na začátku implementace a při plánování prvního bloku, se ovšem objevila skutečnost, že implementace této vrstvy by byla takřka nereálná, nebo minimálně velmi komplikovaná. Celá vrstva totiž pracuje tak, že snímá uživatelovy podněty a na základě nich vytváří, nebo upravuje jednotlivé tvary, které se pak animují. A tady se projevuje obrovská výhoda JavaScriptu, tedy ta, že se provádí přímo u klienta a není potřeba posílat požadavky na server a přijímat jeho dopovědi. V praxi to vlastně funguje tak, že veškeré kreslení a tvorba animace se děje u uživatele. Je zpracováváno již zmíněným JavaScriptem a to i včetně generováním výstupu. Ten je buď ve formátu XML nebo SVG, podle způsobu pozdějšího použití. Veškeré operace tak zpracovává internetový prohlížeč uživatele a až po dokončení animace se pošle serveru hotový kód, který se buď v případě XML uloží do databáze, nebo se v případě SVG zobrazí jako náhled výsledné animace.

Tato část je v podstatě implementována pomocí dvou tříd.

První je globální (CanvasState), a obsahuje v sobě velké množství atributů a vlastností, kompletně popisující celou animaci i kreslící plátno. Hlavně ale také snímá uživatelské podněty a to konkrétně pomocí událostí. Událost si můžeme představit jako jakousi funkci, která se provede vždy, když se splní některá definovaná podmínka. Například událost mousedown snímá stisk levého tlačítka myši a ve funkci k této události se zkoumá, jestli bylo kliknuto na již vytvořený prvek, který se tímto označí, nebo na prázdné místo kreslícího plátna, což by znamenalo, že se nebude nic dít. V případě, že narazíme na nějaký existující prvek, označí se a zobrazí se panel na jeho úpravu, tedy k definici jeho vlastností. Vraťmě se teď ale ke kontrole kliknutí na prvek. Tato událost provede funkci, ve které máme dostupné souřadnice místa na kreslícím plátně, na které uživatel klikl. Tím testujeme, jestli tento bod přísluší některému prvku. Pro většinu tvarů se tato metoda liší. Například pro obdélník, obrázek, koťátko a text se jednoduše testuje, zda je bod uvnitř daného tvaru, to znamená mezi horním levým a pravým dolním rohem. Tyto rohy se vypočítají pomocí souřadnice, kam se tvar vykresluje, jeho šířky a délky. Komplikovanější je to například u kruhu, kde máme pouze souřadnice středu a jeho poloměr. Zde se potom počítá

vzdálenost bodu, na který uživatel klikl, a středu kruhu a porovná s jeho poloměrem. V případě, že je vzdálenost menší, vyhodnotí se funkce tak, že bylo kliknuto uvnitř kruhu a daný tvar se označí. V opačném případě se prohledává dál. Podobně je řešené i rozpoznávání křivek. Jelikož se jejich průběh počítá i vykresluje automaticky, bylo by značně komplikované testovat, zda bylo kliknuto na povrch křivky. Proto se jako lepší možnost jevilo tyto tvary, tedy jak kvadratické, tak Bézierovy křivky, vybírat podle řídících bodů, jimiž jsou křivky definovány a které vlastně určují jejich tvar. Tyto body jsou tedy reprezentovány kruhy s poloměrem 5 pixelů, což je velikost, která byla zvolena s ohledem na to, aby tyto body byly dobře viditelné, ale také aby při vykreslování na kreslící plátno nepřekážely. Při testování křivky, zdali má být vybrána, se proto testuje vzdálenost od uživatelského kliknutí k pozici řídících bodů křivky. V případě, že je vzdálenost některého bodu menší než 5, bod se označí. Je ho tedy možné přenést na jiné místo a také editovat vlastnosti křivky. V případě, že se narazí na odpovídající tvar, označí se jako aktivní prvek, který je potom možné přemístit na jiné místo, nebo změnit jeho vlastnosti.

Další poměrně důležitá událost, která je obsažena v této třídě, se nazývá mousemove. Jak již napovídá anglický význam tohoto slova, spouští se tato funkce při pohybu myši. Pokud mám vybraný prvek a je aktivní, znamená to, že se ho uživatel snaží pomocí metody drag'n'drop přemístit na jiné místo. Aktualizuje se tím pozice vybraného tvaru a plátno se neustále překresluje. Díky tomu je možné dosáhnout cíleného efektu, kdy je daný prvek přenášen. Samozřejmě se ale nejedná pouze o přenesení celého prvku. U křivek se mění souřadnice pouze jednoho vybraného bodu, takže křivka se postupně překresluje tak, jak se mění její definice, s tím, že ostatní body zůstávají na stejných souřadnicích. Mimo testování vybraného bodu, se tato funkce ještě používá k dočasnému vykreslení nové křivky. Ta se totiž definuje výběrem počátečního a koncového bodu. S tím, že po té, co uživatel vybere počáteční bod, začne se vykreslovat křivka z tohoto bodu po bod, na kterém se nachází kurzor myši na kreslícím plátně. Díky tomu uživatel vidí, jak bude výsledná křivka vypadat. Po vybrání koncového bodu se pak křivka dokončí a vytvoří se její instance v této základní třídě.

Jako aktivátor pro tvorbu nových prvků a řídících bodů křivek, byl použit dvojitý klik levého tlačítka myši. Ten obsluhuje další událost, která je se nazývá dblclick. Ta po zaregistrování dvojitého kliknutí na levé tlačítko myši vytvoří aktuálně vybraný tvar. V případě, že se jedná o křivku, tak vytvoří pouze počáteční bod a spustí průběžné vykreslování dané křivky až po vytvoření koncového bodu. Veškeré body i tvary, které se touto funkcí vytvoří, se vkládají na aktuální souřadnice kurzoru myši, nad kterými byl dvojklik proveden.

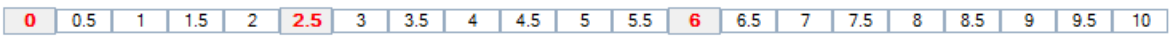
Tato třída ještě obsahuje poměrně důležitou funkci a to konkrétně funkci draw. Ta se stará o vykreslování všech vytvořených tvarů. Je definován interval s periodou 30 ms, který funkci neustále volá a testuje, zda je potřeba překreslit kreslící plátno. V případě že ano, tvary se překreslí.

Druhou třídou, která je v této části použita, je třída Tvar, která reprezentuje jednotlivé tvary a obsahuje informace o jeho pozici na kreslícím plátně v rámci časových stavů, jeho typ,

rozměry, barvy, ohraničení, vlastnosti animování a případnou hodnotu, například u textu. Ke každému typu tvaru jsou navíc implementovány funkce, které umožňují editaci jeho vlastností, protože struktura vlastností pro každý tvar se liší. Pro tento účel jsou určeny vždy dvě funkce. Jedna zadané nové údaje upraví přímo v instanci tvaru a druhá pak tyto změněné údaje zobrazí v editačních polích.

Pro tuto třídu jsou vytvořeny ještě další poměrně důležité funkce. Například funkce vytáhnout, která se stará o zvýraznění tvaru, po tom co ho uživatel vybere pro editaci. U většiny tvarů se zobrazí jejich obrys, u křivek se zobrazí konkrétní vybraný řídící bod. Dále jsou zde pak definované getry a setry pro získání a nastavení aktuálních souřadnic, na kterých se tvar nachází. Tato hodnota je závislá na aktuálním časovém razítku. Velmi důležitou funkcí, o které jsem se již zmínil u třídy CanvasState, která ji využívá, je metoda obsahujeBod, která zkoumá, zda se uživatelovy podněty týkají konkrétního tvaru. A tímto se dostáváme k poslední a nejdůležitější funkci, kterou je metoda draw. Ta obsahuje definici všech tvarů a pro každý zvlášť je definováno jeho vykreslení.

V této části aplikace se také nachází časová osa, která je velmi důležitým prvkem pro každou animaci. Ovlivňuje jednotlivé kroky animace, tedy jak jejich polohy, tak jejich délku. Při plánování jsem si celou animaci rozdělil na jednotlivé časové stavy, které reprezentují polohy jednotlivých vytvořených tvarů v závislosti na čase. Při plánování implementace časové osy, bylo důležité promyslet, jakým způsobem tento ovládací prvek vytvořit. Jedna možnost realizace byla pomocí dynamického generování HTML prvků div, které by měly staticky definovanou funkci v JavaScriptu, která by se starala o aktivaci daného časového stavu nebo jeho vytvoření, či zrušení. Tato koncepce měla ale jeden zásadní problém. Tím bylo vytváření nových časových stavů. Aby bylo možné stávající nápad realizovat, bylo by nutné definovat určitou délku této časové osy a pak snímat, kam uživatel klikl, když vytvářel nový časový stav. Potom by bylo potřeba přepočítat vzdálenost od počátku osy a získat poměr ke konci osy. Pomocí tohoto poměru a celkové délky animace by pak bylo možné získat přesnou časovou hodnotu nového časového stavu a vykreslit ho na časovou osu. Nicméně tohle řešení bylo velmi komplikované a také složité na realizaci. Proto bylo nutné se zamyslet nad jinými metodami a nakonec se podařilo nalézt výrazně přívětivější metodu. Časová osa je složená z dvaceti tlačítek, které reprezentují jednotlivé časové stavy a k ním definované funkce, které je obsluhují. Celá animace se rozdělí na jednotlivé kroky a hodnoty těchto kroků se počítají podle délky animace dělené počtem kroků. Tímto se získá rovnoměrně rozdělená časová osa o 20 možných animačních krocích. Což je pro tvorbu jednoduchých animací dostatečný počet a uživatele nijak výrazně při tvorbě animace limitovat. Konkrétní podobu časové osy je možné vidět na následujícím obrázku. Jsou zde viditelné definované časové stavy. Kromě počátku také v čase 2,5 vteřiny a v šesti vteřinách.



Obrázek 14 - Časová osa. Zdroj: autor

Po rozkliknutí konkrétního stavu, se překreslí plátno a všechny objekty se vykreslí na souřadnicích, které jsou pro aktuální časový stav definované. Daný časový stav je pak samo-

zřejmě možné kdykoliv zrušit. Vytváření časových stavů opět funguje celkem jednoduše. Při vytvoření nového stavu v konkrétním čase se zkopírují hodnoty souřadnic jednotlivých objektů od nejbližšího předchozího stavu. To se děje proto, aby když budeme chtít přidat do již rozpracované animace další časový stav, například pro zjemnění průběhů jednotlivých tvarů. Nebudeme muset jednotlivé objekty pozicovat z výchozího nulového stavu, ale budeme mít hodnoty přesnější, které již navazují na průběžné souřadnice.

3.3.2 Veřejná část

Tvorba veřejné části byla celkem jednoduchá a příjemná záležitost. Jelikož s programováním v jazyce php mám již poměrně hodně zkušeností, jediným teoretickým problémem pro mě bylo předávání hodnot z JavaScriptové uživatelské části, do této veřejné. Nakonec jsem se rozhodl pro předávání hodnot přes formuláře. JavaScript umožňuje za chodu měnit hodnotu jednotlivých formulářů, ale i tuto hodnotu číst. Takže když jsem potřeboval například XML kód animace předat veřejné části pro uložení, po jeho vygenerování jsem hodnotu uložil do formuláře typu hidden, takže se nikde v šabloně editoru nezobrazuje. Tento formulář jsem poté odeslal na php soubor, který tento kód přijal a uložil do databáze. Úplně stejně to fungovalo i s exportem animací ve formátu SVG. Pouze se vygeneroval dokument podle zásad SVG popisování grafiky a animací a výsledný kód zobrazil. Podobný byl i postup v případě načítání hotových animací k úpravě či exportu. Z databáze jsem získal XML dokument popisující animaci a tento dokument uložil do stejného formuláře, jako při ukládání. Z něho jsem pak v JavaScriptu načetl hodnotu a pomocí jednoduchého kompilátoru vygeneroval objekty jednotlivých tvarů.

Pokud se tedy zaměřím přímo na implementaci veřejné části, její struktura je velice jednoduchá a k tomu mi výrazně dopomohlo objektové programování v php. Tato část se skládá z několika objektů, kde základní objekt je třída main. V této třídě jsou definované základní a univerzální metody, jako například přihlášení, odhlášení, výpis nastavení, informací o uživateli a podobně. Tato třída je předkem třídy animace, která zabezpečuje veškerou funkcionalitu, tedy načítání, mazání a ukládání animací, změnu uživatelských údajů nebo registraci. V případě složitější animace by se určitě slušelo tuto třídu rozdělit na třídu, která by se starala o animace a na třídu, která by zabezpečovala registraci a změnu údajů uživatelů. Vzhledem k tomu, že tady nároky na implementaci nebyly tak velké a soubory jsou i tak velmi malé, rozhodl jsem se tuto část příliš nekomplikovat a použít pouze jednu třídu, která bude mít na starost obě tyto činnosti.

Veškeré operace s databází jsou realizovány přes třídu db, kterou jsem použil už ve své bakalářské práci a v podstatě beze změn ji používám již několik let. Využívá databázového frameworku Dibi[6], který je programovaný v php a výrazně ulehčuje práci s databázemi. Tato šikovná utilita umožňuje automatické nastavení připojení podle typu serveru. V případě, že se dotazy a data předávají k provedení ve správném tvaru, nezáleží pak, jestli se používá databázový server MySQL, Oracle, MsSQL nebo jiný. Po předání dotazu Dibi se tento doraz rozparsuje na jednotlivé části a poté spojí v závislosti na použitém typu serve-

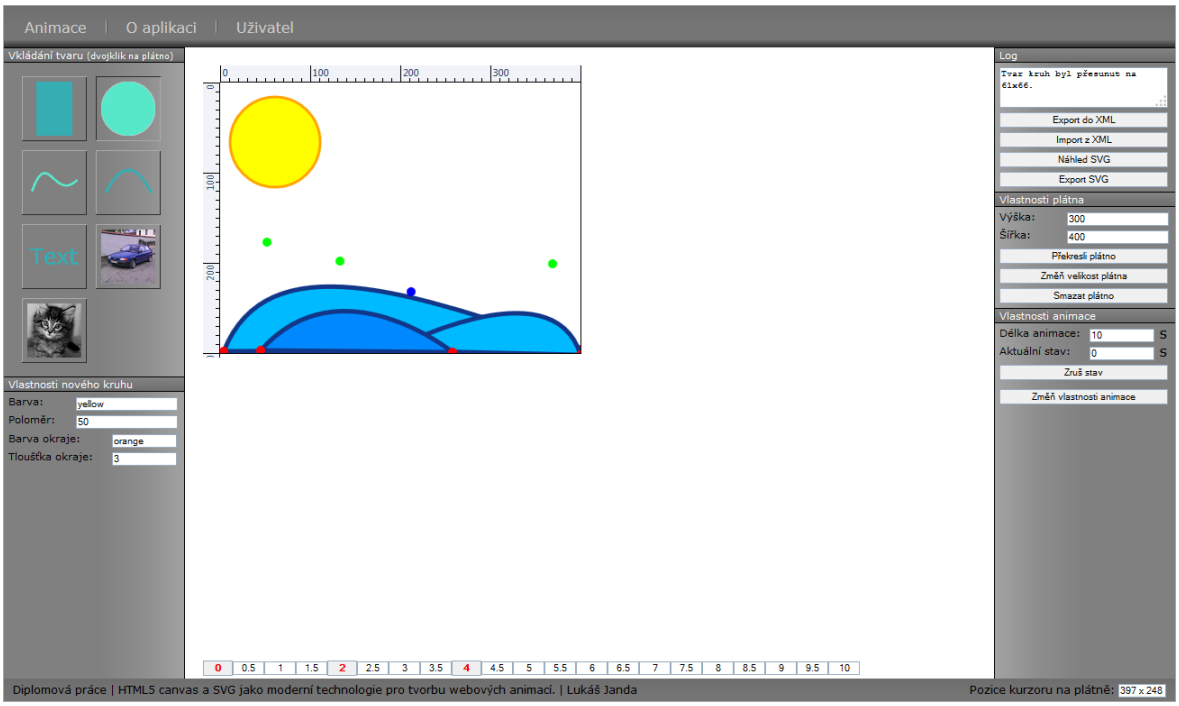
[6] domovská stránka tohoto projektu je dostupná pod touto url adresou http://dibiphp.com/

ru. Třída db má definované různé metody pro práci s databází, jako je například vkládání, editace, mazání, ale i čtení informací z databáze. Tuto třídu spolu s dibi stačí pouze připojit k jakékoliv aplikaci v php a není potřeba jinak řešit spolupráci s databází.

Tato část aplikace zajišťuje i jakousi jednoduchou administraci uživatelů. Můžou se skrze tuto část registrovat, přihlašovat, odhlašovat a měnit své údaje. Nakonec jsem se rozhodl pro povinné přihlašování uživatelů zejména ze dvou důvodů. Jedním z nich bylo ukládání animací do databáze. Pokud by bylo možné ukládat animace do databáze bez přihlášení, obával jsem se možných spamových nebo jiných škodlivých útoků. Registrace účtu není nijak složitá a spamoví roboti by mohli zneužít i ji. I tak se mi ale jeví jako alespoň nějaká základní ochrana. A druhý důvod byl hlavně komfort pro uživatele, protože po přihlášení si můžou načíst kteroukoliv již uloženou animaci, což by bez přihlášení také nebylo možné.

3.4 Popis funkcionality

V této kapitole bych se rád zaměřil na funkcionalitu, kterou se mi podařilo implementovat. V podstatě jsem se po celou dobu držel původního zadání a kromě funkčnosti, jsem se také snažil výsledný editor navrhnout tak, aby byl přehledný. Celý editor animací si můžeme prohlédnout na následujícím obrázku.



Obrázek 15 - Editor animací. Zdroj: autor

3.4.1 Hlavní menu

V horní části se rozprostírá hlavní menu aplikace, které má 3 základní skupiny tlačítek. Po najetí myší na tyto hlavní tlačítka se zobrazí další tlačítka, která se vždy týkají této hlavní kategorie. První se týká samotné animace a najdeme zde následující tlačítka:

- Vytvořit - vytvoří novou animaci. Při této akci se uživateli zobrazí jednoduché dialogové okno, kde vyplní rozměry a délku nové animace,
- Otevřít - zde si může načíst již uloženou animaci. Zde se zobrazí okno, ve kterém je výpis veškerých uživatelových animací, spolu se základním popisem, jako je například název nebo datum uložení,
- Uložit - uloží rozpracovanou animaci. V dialogovém okně je uživatel dotázán na název nové animace a poté je animace uložena do databáze pro pozdější znovunačtení,
- Import XML - v případě, že si uživatel vytvořil animaci a má její popis pomocí XML vygenerovaný v tomto programu, může pomocí tohoto kódu animaci importovat,
- Export XML - pokud si uživatel nepřeje uložit animaci do databáze, může si nechat vygenerovat animaci ve formátu XML. Tento kód pak může kdykoliv použít pro opětovné načtení animace pomocí Importu XML,
- Náhled SVG - vygeneruje se SVG kód animace a zobrazí náhled tak, jak se pak výsledná animace bude zobrazovat na webových stránkách,
- Export SVG - toto tlačítko slouží pro export do SVG. Po jeho stisknutí se vygeneruje popis animace v jazyce SVG a tento kód zobrazí uživateli. Ten si ho pak může zkopírovat a vložit na své internetové stránky.

Další kategorií v menu, je tlačítko O aplikaci, které má v podstatě pouze informativní charakter. Nacházejí se pod ním dvě tlačítka, které zobrazí buď informace o aplikaci, nebo návod na použití. V tomto návodu je popsáno rozhraní pro použití tohoto programu a stejně tak i vkládání prvků. V případě, že si uživatel nebude vědět s ovládáním tohoto programu rady, tohle je to pravé místo, kde by měl hledat informace a rady.

V poslední části, která nese název Uživatel, se nachází 3 tlačítka:

- Moje animace - zde si uživatel může zobrazit všechny animace, které si sem uložil. Zobrazí se mu přehledný výpis, kde jsou animace vypsané. Tento výpis slouží také k administraci, takže je možné animace otevírat i mazat,

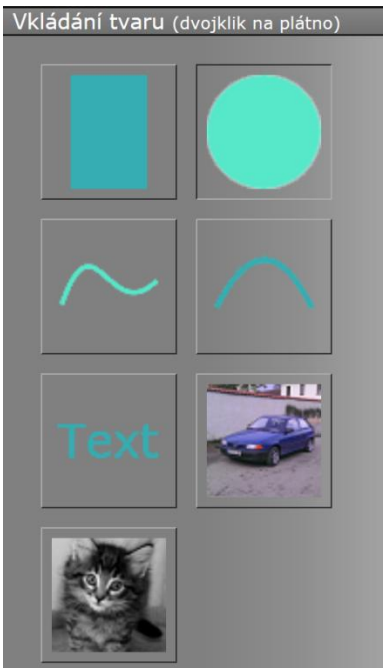
- Změna údajů - po stisknutí tohoto tlačítka se zobrazí dialogové okno, kde si uživatel může změnit základní údaje jako svoje jméno, příjmení, přihlašovací jméno, heslo a email,

- Odhlásit - tohle tlačítko slouží k odhlášení uživatele.

Obrázek 16 - Hlavní menu. Zdroj: autor

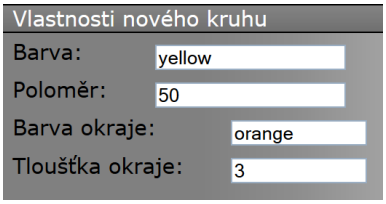
3.4.2 Vkládání prvků

Pro vložení tvaru si stačí vybrat z nabídky v levé části editoru a poté dvojklikem na kreslícím plátně umístit prvek tak, kam chceme. Na výběr zde máme celkem sedm tvarů, které si podrobně představíme později.



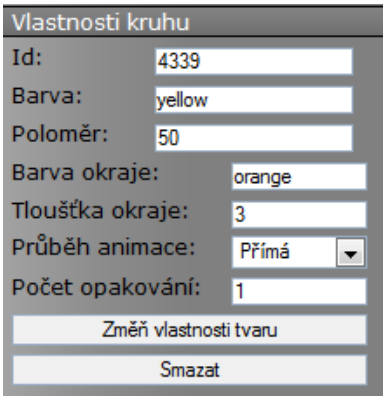
Obrázek 17 - Vkládání tvaru. Zdroj: autor

Při vytváření každého tvaru, si můžeme definovat jeho základní vlastnosti. Tyto vlastnosti se pro každý prvek liší. Například pro kruh si můžeme definovat jeho výplň, která ale může být transparentní, jeho poloměr, tloušťku a barvu okraje, který ale taky nemusíme definovat. Pro ostatní tvary se toto nastavení samozřejmě mírně liší, představím ho tedy podrobněji spolu s vlastnostmi každého tvaru zvlášť.



Obrázek 18 - Vlastnosti nového tvaru. Zdroj: autor

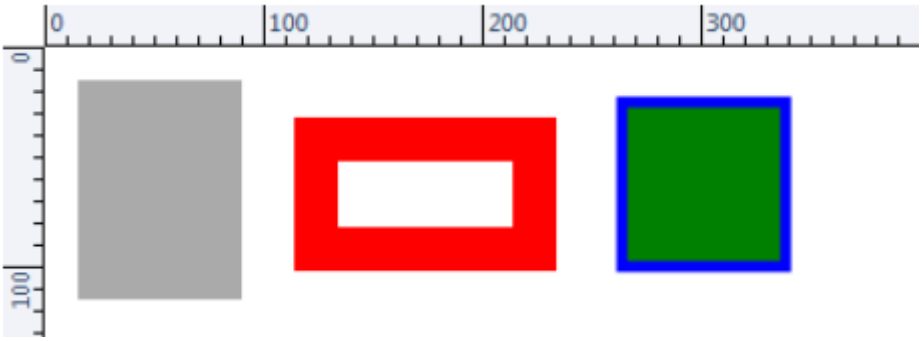
U každého vloženého tvaru je možné po jeho vybrání editovat jeho vlastnosti. Mírně se ovšem liší od těch, které se přiřazují novému tvaru. Jelikož ne všechny prvky se budou animovat, rozhodl jsem se vytváření prvku co nejvíce zjednodušit a kompletní nastavení zpřístupnit až při jeho editaci.



Obrázek 19 - Změna vlastností tvaru. Zdroj: autor

3.4.3 Vlož obdélník

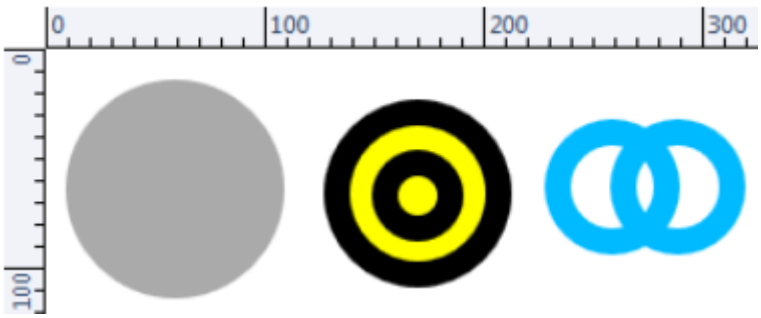
Tato funkce vloží na plátno obdélník. U tohoto tvaru je možné nastavit jeho výšku, šířku, barvu výplně, barvu a tloušťku okraje, počet opakování, průběh animace. Barva výplně i okraje může být transparentní, a proto je možné vytvářet poměrně zajímavé tvary a skládat z nich složitější obrazce.



Obrázek 20 - Obdélník. Zdroj: autor

3.4.4 Vlož kruh

Vkládání kruhu je podobné jako vkládání obdélníku. Pouze se zde nenastavuje šířka a výška tvaru, ale jeho poloměr. Jinak lze i u tohoto tvaru nastavovat barvu výplně, barvu a tloušťku okraje, již zmíněný poloměr, počet opakování animace a průběh animace.



Obrázek 21 - Kruh. Zdroj: autor

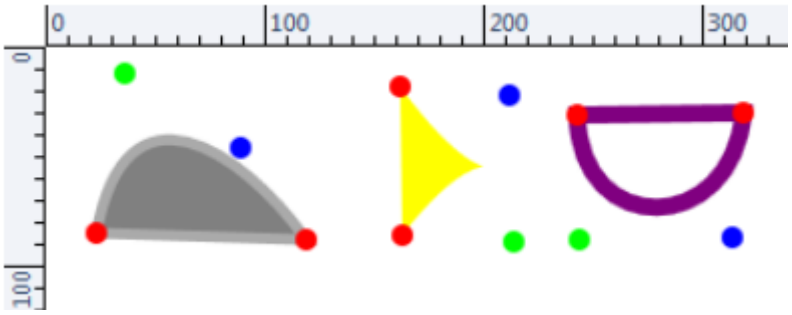
3.4.5 Vlož Bézierovu křivku

Nyní už se dostáváme k zajímavější funkci. Její realizace byla spolu s kvadratickou křivkou nejobtížnější. A to jak vkládání, tak editace. Ostatní tvary se totiž vkládají pouze s jedním řídícím bodem, ovšem u křivky jsou potřeba 2, tedy počáteční a koncový. Aby se vkládání zjednodušilo, zbývající dva řídící body se při vkládání křivky dopočítávají automaticky a je možné je po vytvoření křivky editovat. Křivka se tedy vkládá tak, že si uživatel vybere dvojklikem na plátně první bod. Při hledání druhého bodu se vykresluje křivka tak, jak by vypadala, kdyby její vytvoření potvrdil druhým dvojklikem. Ve chvíli kdy to udělá, se vytvoří instance křivky a bude se stabilně vykreslovat s ostatními tvary. Implementace byla poměrně náročná, protože jsem musel vytvořit další atribut v základní třídě CanvasState, který představuje ukazatel, zda se zrovna vytváří křivka. V případě že ano, vykresluje se podle jejího typu provizorní křivka z počátečního bodu do bodu, kde se zrovna nachází kurzor. Po dokončení křivky se změní hodnota tohoto ukazatele a křivka se přestane vykreslovat. Samotná Bézierova křivka je definována rovnicí, kterou je možné vidět na následujícím obrázku.

$$C(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} P_i$$

Obrázek 22 - Definice Bézierovy křivky[22]

U tohoto tvaru je také možné definovat výrazně více parametrů. Zpočátku se při vytváření jedná o klasickou barvu výplně, tloušťku a barvu ohraničení. Přibyla zde ale možnost spojit počáteční a koncový bod. Touto volbou se nám naskýtají další možnosti, co se týká vytváření různých tvarů. Opět i výplň i ohraničení může být transparentní, což ještě více rozšiřuje tyto možnosti. Původně jsem počítal s tímto tvarem pouze jako s jakýmsi vylepšením na zobrazování průběhu funkcí, ale poté, co jsem si uvědomil, že například přidáním výplně, nebo již zmíněného spojení počátečního a koncového bodu, můžou vznikat opravdu zajímavé tvary. Při editaci křivky opět přibudou možnosti nastavit počet opakování a průběh animace. A k tomu také nastavení souřadnic řídících bodů. Může se totiž stát, že některý řídící bod se bude nacházet mimo kreslící plátno a proto nebude vidět. V tom případě ho půjde jednoduše nastavit pomocí editačních polí, kde se vepíšou souřadnice.

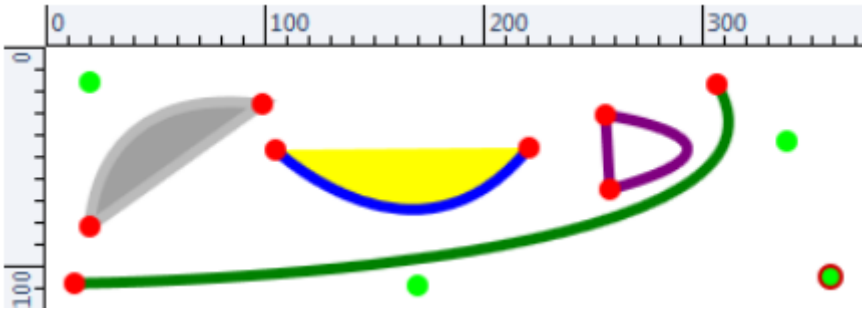


Obrázek 23 - Bézierova křivka. Zdroj: autor

Na předchozím obrázku můžeme vidět tři tvary, vzniklé modifikací Bézierovy křivky. Poblíž těchto tvarů se vždy nachází 4 body. Konkrétně dva červené, jeden zelený a jeden modrý. Červené body reprezentují počáteční a koncový stav. Zelený bod pak znázorňuje první řídící bod křivky a modrý zase ten druhý. Tyto body jsem přidal na kreslící plátno proto, aby bylo možné s křivkou manipulovat. Jednoduchým kliknutím na konkrétní bod, který chceme přetáhnout na jiné místo, se nám označí a můžeme ho pomocí drag'n'drop přesunout na jiné místo. Během manipulace s kterýmkoliv bodem dané křivky, se křivka neustále překresluje, takže máme pořád přehled o její aktuální podobě.

3.4.6 Vlož kvadratickou křivku

Vkládání kvadratické křivky je velmi podobné předchozí funkci. Dalo by se říct, že je skoro stejné a jeho editace je rovněž. Pouze s výjimkou editace druhého řídícího bodu, který zde není potřeba. Na následujícím obrázku si můžeme prohlédnout příklady vytvořených tvarů pomocí kvadratické křivky. Jak již bylo zmíněno, je zde pouze jeden řídící bod a ten je znázorněn zeleným kruhem. Zbývající dva červené i zde reprezentují počáteční a koncový bod. V pravém dolním rohu si můžeme všimnout označeného bodu, který je vybraný k editaci a zvýrazněn červeným okrajem.



Obrázek 24 - Kvadratická křivka. Zdroj: autor

3.4.7 Vlož text

Další poměrně zajímavou a užitečnou funkcí je vkládání textu. Na výběr zde máme čtyři předdefinované fonty, které můžeme vidět na následujícím obrázku. Můžeme tedy použít font Georgia, Tahoma, Times New Roman a Verdana. Při vkládání nového textu si tedy vybereme jedem z těchto fontů, zvolíme velikost písma v pixelech a vybereme barvu textu. Poslední hodnota, kterou bychom měli nastavit je samotný text, který se má na plátno vykreslit. V případě, že tak neučiníme, zobrazí se text „Chyba! Nebyl zadán text.", jak můžeme vidět v levé horní části obrázku, kde se nachází šedý text s touto zprávou. Pokud hodnotu textu zapomeneme uvést, stačí pak kliknutím daný text označit a jeho hodnotu změnit. V editaci rovněž potom přibudou editační políčka pro nastavení animace, tedy její průběh a počet opakování. Na následujícím obrázku můžeme také vidět různé kombinace fontů, barev a velikostí písem. Jsou zde kombinace textů a jiných tvarů. Například žlutého textu s černým pozadím, nebo černý text s červeným kulatým okrajem, připomínající dopravní značku.

Obrázek 25 - Text. Zdroj: autor

3.4.8 Vlož Obrázek

Jelikož by tvorba samotných obrázků pouze z jednoduchých tvarů něco postrádala, bylo nutné implementovat i funkci na vkládání obrázků. Ta je sama o sobě velmi jednoduchá. Při vkládání obrázku se zadává pouze adresa obrázku dostupného na internetu. Obrázek se klasicky vykreslí na místo, kde uživatel dvojklikem určí jeho novou pozici. Při editaci je možné cestu k obrázku změnit, což znamená, že můžeme kdykoliv měnit animovaný obrázek. Při editaci je také možné natavení počtu opakování a průběh animace.

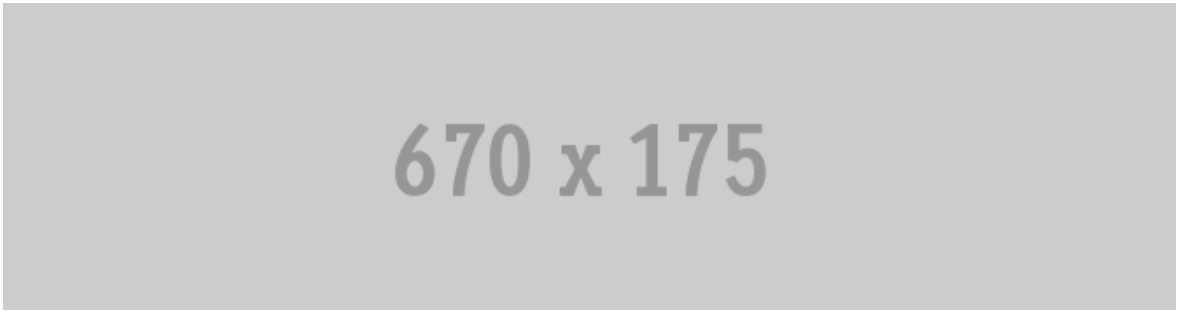
Obrázek 26 - Obrázek. Zdroj: autor

3.4.9 Vlož koťátko

Tato funkce vloží na kreslící plátno obrázek koťátka o zadaném rozměru. Při vkládání se vždy zadává pouze výška a šířka požadovaného obrázku. Při editaci je pak možné nastavení animace, jako u všech předchozích tvarů. Funkce využívá webových služeb http://placekitten.com/, které poskytují obrázky koťátka v rozměru, který si uživatel definuje. Využití této služby je jednoduché, pokud chcete někam umístit provizorní obrázek o určitých rozměrech, stačí se odkázat na stránku http://placekitten.com/šířka/výška. Kde šířka znamená šířku výsledného obrázku v pixelech a výška zase výslednou výšku. Takže například na adrese http://placekitten.com/670/175 se skrývá následující obrázek.

Obrázek 27 - Služba placekitten.com [20]

Tato služba vychází ze služby placehold.it, která ovšem generuje pouze klasický obrázek s rozměrem, který potřebujeme, jak můžeme vidět na následujícím obrázku. Výhodou je, že můžeme nastavovat různé texty místo pouhého rozměru obrázku, barvu pozadí a jiné věci. Na druhou stranu předchozí služba a jí vygenerované obrázky jsou pořád trošku veselejší. Alternativ k těmto službám je několik. Při hledání na internetu jsem narazil na různé variace této služby například služba poskytující obrázky ze serveru Flickr[7]. Když jsem tuto službu ale testoval, obrázky se načítaly velmi pomalu, a proto její použití bylo velmi komplikované a ne zrovna pohodlné. Osobně by se mi líbila podobná služba, poskytující místo zvířat fotky amerických automobilových veteránů, ale takovou jsem nikde nenašel. Proto jsem se v rámci testování rozhodl využít službu placekitten.com.



Obrázek 28 - Služba placehold.it [21]

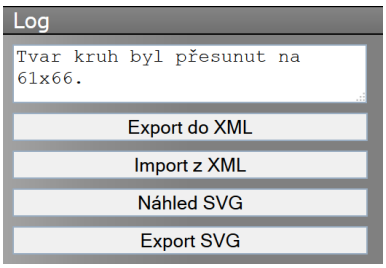
Funkce vlož koťátko tedy v této aplikaci znázorňuje přehlednou možnost, jak otestovat veškerou funkcionalitu. Můžete vložit provizorní obrázek o určitých rozměrech, provádět s ním různé transformace a poté ho nahradit skutečným, který chcete animovat. Podobně se tato služba využívá i při návrhu internetových stránek, kde většinou ještě nemám konkrétní obsahové obrázky nebo například reklamy. Proto do šablony stránek vložíme podobné vygenerované obrázky a testujeme šablonu, jak bude vypadat a chovat se v různých situacích.

3.4.10 Log a základní operace

Pro lepší přehled o prováděných operacích byl v této aplikaci implementován i jednoduchý log, kde se vypisují zprávy o všech uživatelových akcích. V tomto ovládacím panelu se rovněž nachází tlačítko pro import a export animace. Provádí v podstatě stejné akce, jako tlačítka v hlavním menu v položce „Animace". Umožňují tedy animaci exportovat ve for-

[7] Internetové stránky určené pro sdílení fotografií, dostupné na http://www.flickr.com/

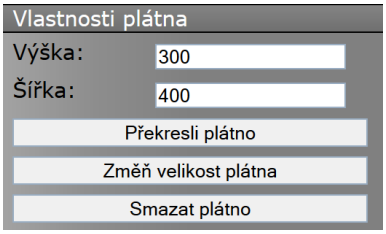
mátu XML, tento vygenerovaný kód je potom možné zpátky importovat a vytvořit původní animaci. Jsou zde také dvě možnosti exportu animace ve formátu SVG. Jedna možnost zobrazí náhled výsledné animace a druhá vygenerovaný kód pro vložení do webových stránek.



Obrázek 29 - Log a základní operace. Zdroj: autor

3.4.11 Vlastnosti kreslícího plátna

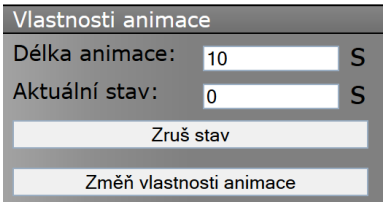
I v průběhu tvorby animace je možné měnit její velikost. Pouze je potřeba si dát pozor na změnu rozměrů, konkrétně pokud budeme rozměr animace zmenšovat, může se stát, že některé tvary se budou nacházet mimo kreslící plátno a v tomto stavu nebude možné je nijak editovat. Změnu rozměrů plátna je nutné potvrdit stiskem tlačítka „Změň velikost plátna". V této části editačního panelu se rovněž nachází tlačítko „Překreslit plátno", které slouží k překreslení kreslícího plátna. To se hodí zejména v případě, kdy se nějaká akce neprovede správně. Respektive se vypíše do logu, že byla provedena, ale nevykreslí na kreslícím plátně. Děje se to v některých případech, kdy se nestihne po vložení obrázek vykreslit, ale plátno už je označeno jako aktuální. Poslední tlačítko, které se nachází v tomto bloku, slouží ke smazání celé animace. Po jeho stisknutí dojde k vymazání celého kreslícího plátna, všech instancí vytvořených tvarů i všech nastavení, týkajících se animace.



Obrázek 30 - Vlastnosti kreslícího plátna. Zdroj: autor

3.4.12 Vlastnosti animace

K nastavení délky animace slouží tato část ovládacího panelu. Můžeme zde definovat délku animace, což bude mít vliv na délku jednotlivých kroků, které budeme později animovat. Změnu délky animace je potřeba potvrdit stiskem tlačítka „Změň vlastnosti animace". Dále se zde nachází informativní pole, které nám ukazuje čas aktuálního časového stavu. Na následujícím obrázku je vidět, že se nacházíme na začátku animace, tedy v čase 0. Po výběru jednotlivých časových stavů máme rovněž možnost je mazat. K tomu slouží tlačítko „Zruš stav", které zruší vybraný stav a jako aktuální nastaví výchozí stav v čase 0, tedy začátek animace.



Obrázek 31 - vlastnosti animace. Zdroj: autor

4 Uživatelské testování aplikace

V rámci tohoto testování byl vytvořen testovací scénář, který pak byl předložen uživateli, který tuto aplikaci nikdy předtím neviděl, a byly zaznamenávány jeho reakce a postřehy. Tento scénář zahrnoval posloupnost akcí potřebných k vytvoření a exportu animace. Konkrétně tyto kroky:

1. Přihlásit se do aplikace.
2. Vytvořit novou animaci.
3. Vytvořit průhledný čtverec kdekoliv na kreslícím plátně, který bude mít modrý okraj.
4. Uložit animaci.
5. Načíst animaci.
6. Definovat na časové ose nový bod.
7. Upravit animaci pro tento časový stav.
8. Přidat libovolnou křivku, která bude mít zelenou barvu.
9. Zobrazení náhledu animace
10. Smazání čtverce.

Po celou dobu byly zaznamenávány reakce uživatelů pro každý definovaný úkon, například kde hledal danou funkcionalitu, zda ji našel, nebo nenašel a jak mu to přibližně dlouho trvalo. Je totiž velmi složité po několikaměsíčním vývoji odhadnout míru uživatelské přívětivosti vyvíjené aplikace, protože vývojář přesně ví, kde jakou funkcionalitu má hledat. Tato metoda je sama o sobě velmi neurčitá a nelze z ní zcela vycházet, máme zde ale zpětnou vazbu od uživatelů a podněty na vylepšení aplikace.

Organizovaného uživatelského testování se účastnily celkem 2 osoby. V případě, že si nebyli jistí, jak daný krok provést, měli k dispozici nápovědu v aplikaci. Pokud ani tam odpověď nenašli, přišly na řadu dotazy. Detailní postřehy z jejich testování je možné najít v příloze C, kde jsou ke každému bodu uvedeny podrobnosti ohledně dokončení úkonu a případný komentář. Toto uživatelské testování bylo provedeno po dokončení vývoje apli-

kace, aby bylo možné zhodnotit, zda je program dostatečně intuitivní i pro někoho, kdo ho nikdy před tím neviděl. Tato zpětná vazba bude sloužit také jako zdroj námětů na budoucí vylepšení aplikace a také jako inspirace pro plánování implementace další funkcionality.

Samotné testování dopadlo nakonec velmi dobře. Oba uživatelé splnili bez větších problémů úkoly definované v testovacím scénáři. Z tohoto testování nakonec vyplynulo, že aplikace je relativně přehledná a intuitivní. Do budoucna by bylo dobré implementovat nástroj pro výběr barev, který by sloužil pro definici barevného odstínu pro výplně nebo okraje prvků. Rovněž by bylo dobré přepracovat časovou osu, která o ovšem sama o sobě velmi komplikovaný prvek. Dále by bylo vhodné drobně poupravit a reorganizovat některé ovládací prvky.

5 Testování výkonu

Velmi zásadní část této práce je i testování výsledných animací. To se bude zakládat zejména na analýze nároků na výkon počítače. Hlavní pilíře v této analýze budou procesor, grafická karta a operační paměť. U nich se bude sledovat zatížení při přehrávání animací. Jedna skupina budou flashové animace, druhá pak animace ve formátu SVG, vytvořené pomocí aplikace, která byla implementována v rámci této práce. Jako prostředí, ve kterých se budou animace testovat, byly vybrány Mozilla Firefox a Google Chrome. Výběr těchto prostředí není náhodný, ba naopak jedná se o dva nejpoužívanější prohlížeče, a proto využijeme k testování zrovna je. Jako aplikace, která bude sledovat zatížení jednotlivých komponent, byl vybrán program Process Explorer. Jedná se o poměrně jednoduchou aplikaci, která je k dispozici zdarma a přehledně zobrazuje zatížení jednotlivých částí systému. I přesto, že se jedná o relativně malou aplikaci, máme zde na výběr velké množství nastavení a možností. Pro nás tedy bude hlavní kritérium zatížení procesoru, grafické karty a operační paměti. Celé testování bude prováděno na dvou počítačích. Detaily testovacích sestav můžete vidět v následujících tabulkách.

Tabulka 1 - Testovací sestava 1

Notebook VBI Salford	
Procesor	Intel Core 2 Duo 2,53 GHz
Operační paměť	4GB DDR2
Grafická karta	nVidia GeForce 9650m GT 1GB DDR2
Operační systém	Windows 7 64bit

Tabulka 2 - Testovací sestava 2

Notebook Acer Travelmate	
Procesor	Intel Celeron 1.46 GHz
Operační paměť	1GB DDR
Grafická karta	Integrovaná
Operační systém	Windows XP

Testovací scénáře byly definovány následovně:

1. Vybereme na internetu flashovou animaci, ke které bude možné v připravené aplikaci vytvořit ekvivalentní animaci. Tím se rozumí, že bude obsahovat podobné akce a efekty.

2. Vytvoříme ekvivalentní animaci ve formátu SVG.

3. Zaznamenávat budeme hodnoty zatížení procesoru, grafické karty a využití operační paměti.

4. Nejdříve budeme testovat animace v prostředí Mozilla Firefox na první testovací stanici. Přehrajeme první animaci ve formátu SVG a zaznamenáme zatížení jednotlivých komponent počítače. Poté přehrajeme druhou SVG animaci a opět zaznamenáme naměřené hodnoty.

5. Potom na stejné sestavě a ve stejném prohlížeči přehrajeme animace ve formátu Flash a opět u nich zaznamenáme zatížení počítače.

6. Následně veškeré úkony zopakujeme při testování v prohlížeči Google Chrome.

7. Pokračovat budeme stejným testováním na druhé stanici. Přehrajeme SVG animace v prostředí Mozilla Firefox a zaznamenáme pro ně hodnoty zatížení jednotlivých sledovaných komponent.

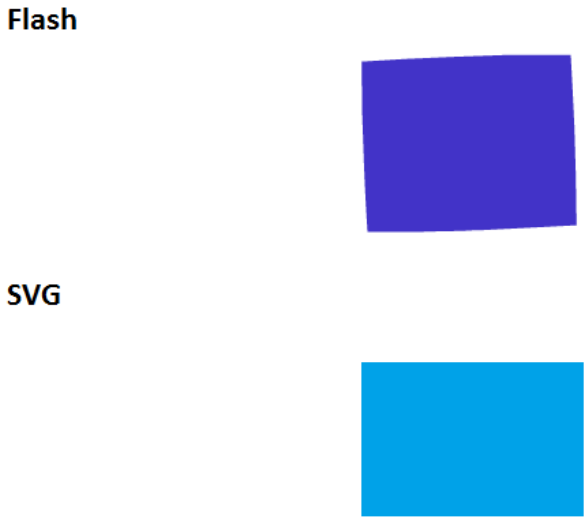
8. Poté na stejné stanici přehrajeme v Mozille Firefox flashové animace a opět zaznamenáme hodnoty.

9. Stejné testování provedeme na druhé stanici i v prostředí Google Chrome.

Dokončením tohoto testování získáme přibližné hodnoty zatížení počítače při přehrávání jednotlivých animací. Budeme zde porovnávat rozdíly v zatížení u jednoduché a složitější animace. Testované animace budou dvě. Jedna složitější, která bude v případě SVG inzerovat automobily a ve flashové podobě motocykly. Budou se při tom porovnávat bannery, které můžete vidět na následujícím obrázku. Jedná se o animaci s pohyblivými texty a obrázky. Flashová podoba nabízí mírně pokročilejší efekty, jako jsou blikající stíny písma a je dostupná na internetovém portálu Flashexplained[15]. Jinak jsou však animace relativně podobně složité. Animace ve formátu SVG je pak dostupná, po přihlášení pomocí veřejného účtu, v uložených aplikacích pod názvem Testovací animace 1.

Obrázek 32 - Složitější animace. Zdroj: autor

Druhá animace bude o poznání jednodušší. Bude se jednat pouze o jeden tvar, který bude oscilovat na vodorovné ose x. V případě SVG se bude jednat pouze o animované posouvání obdélníků, protože zde složitější transformace nebyly implementovány. Tato animace je dostupná v aplikaci po přihlášení pod veřejným účtem jako Testovací animace 2. Flashová animace zobrazí navíc změnu barvy a tvaru animovaného tělesa. I když flashová animace, pocházející z webu techbyte[16], nabízí pokročilejší efekty, z hlediska náročnosti by měly být obě animace podobně složité. Jejich srovnání lze vidět na následujícím obrázku.



Obrázek 33 - Jednoduchá animace. Zdroj: autor

5.1 Naměřené hodnoty

Přibližné naměřené hodnoty je možné vidět v následujících tabulkách. O zatížení procesoru vypovídá řádek CPU, grafické karty řádek GPU a operační pamětí řádek RAM. Hodnoty zatížení procesoru a grafické karty znázorňuje průměr hodnot při přehrávání aplikace.

U paměti RAM zase rozdíl využité paměti před a po načtení animace. U druhé testovací sestavy nakonec nebylo sledováno zatížení grafické karty, protože zde byla pouze integrovaná, a proto se zkoumalo pouze zatížení procesoru.

Tabulka 3 - Naměřené hodnoty zatížení počítače na testovací sestavě 1

SVG animace	**Testovací animace 1**		**Testovací animace 2**	
Prohlížeč	Mozilla Firefox	Google chrome	Mozilla Firefox	Google chrome
CPU	38%	9%	21%	9%
GPU	25%	30%	16%	8%
RAM	4 MB	5 MB	3 MB	2 MB
Flash animace	**Testovací animace 1**		**Testovací animace 2**	
Prohlížeč	Mozilla Firefox	Google chrome	Mozilla Firefox	Google chrome
CPU	14%	9%	7%	7%
GPU	27%	5%	2%	3%
RAM	32 MB	13 MB	12 MB	9 MB

Tabulka 4 - Naměřené hodnoty zatížení počítače na testovací sestavě 2

SVG animace	**Testovací animace 1**		**Testovací animace 2**	
Prohlížeč	Mozilla Firefox	Google chrome	Mozilla Firefox	Google chrome
CPU	65%	9%	8%	6%
RAM	6 MB	5 MB	4 MB	3 MB
Flash animace	**Testovací animace 1**		**Testovací animace 2**	
Prohlížeč	Mozilla Firefox	Google chrome	Mozilla Firefox	Google chrome
CPU	16%	8%	13%	3%
RAM	16 MB	9 MB	9 MB	4 MB

5.2 Porovnání zátěže

Předem je třeba říci, že testování výkonu v žádném případě nebylo provedeno na laboratorní úrovni. Spíše by se dalo přirovnat k běžnému uživatelskému testování a to z několika důvodů. Program, vytvořený v rámci této práce, je určen pro tvorbu jednoduchých animací pro běžné uživatele, kteří nemají potřebné znalosti, aby si animace vytvořili bez něj. Proto i testování se zaměřovalo na jednoduché porovnání zátěže hardwarových částí, které jsou pro běžného uživatele klíčové, tedy operační paměť, procesor a grafickou kartu. Účelem tohoto testování totiž nebylo naprosto přesné získání nároků na hardware počítače, ale přibližné srovnání obou porovnávaných technologií. Dalším důvodem je stav operačních systémů. Testovací sestavy jsou v provozu už dlouhou dobu a operační systémy obsahují mnoho dalších nainstalovaných programů, které mohou výsledné hodnoty ovlivňovat. Spíše než hodnoty samotné bude relevantní srovnání těchto hodnot.

Nejprve se zaměříme na srovnání jednotlivých prohlížečů a jejich využití počítače. Při pohledu na naměřené hodnoty jsou poměrně zřetelné rozdíly v zatížení počítače u obou prohlížečů. Výrazně lépe zde vychází Google Chrome, který jednak spotřebovává méně operační paměti jako taký, ale rovněž při přehrávání aplikací. Zatížení procesoru je u něj vždy menší než u Mozilly Forefox. U využití grafické karty se dá říct to samé, pouze s výjimkou

složitější animace ve formátu SVG. V tomto případě využíval Google Chrome více potenciálu grafické karty než jeho sok. Úplně stejně tomu tak bylo i s využitím operační paměti, které Google Chrome ve stejném případě spotřeboval více.

Když se podíváme na srovnání jednotlivých technologií - SVG a Flash - tady už jsou hodnoty zajímavější. Zatímco Flash má větší nároky na operační paměť, má výrazně nižší nároky na procesor a grafickou kartu. Konkrétně využití počítače bylo vždy menší, než v případě SVG animace a při zkoumání zátěže grafické karty bylo SVG pouze jednou úspornější. Ostatně když si uvědomíme formát jednotlivých grafických formátů, jsou získané výsledky i celkem očekávatelné. SVG animace jsou v podstatě XML dokument, který obsahuje informace o grafických tvarech, které se mají vykreslit. Proto zabírají menší množství paměti a více zatěžují procesor a grafickou kartu. Je zde totiž nutnost vykreslit a zobrazit tyto tvary. Naproti tomu flashové animace jsou již vygenerovaný soubor, určený pro flashový přehrávač, kde jsou již nějaká metadata. Tento formát bezpochyby také bude za dlouhá léta vývoje optimalizovaný, a proto zde vidíme menší zatížení procesoru a grafické karty, ale vyšší využití operační paměti.

Využití počítače a jeho jednotlivých testovaných komponent se ovšem nelišilo v nějaké zásadní míře. Rovněž použití na notebooku, který představoval druhou testovací stanici a byl starý více než 7 let, bylo plynulé a bezproblémové. Podle měření tedy vychází v testu zátěže lépe flashová animace. Jejich větší spotřeba operační paměti je možná dána i trendem vývoje dnešních aplikací, který se už ani zdaleka nezaměřuje na optimalizaci tak jako dříve. Také operační paměť v počítačích v této době výrazně narostla. Velikost operační paměti, která byla osazena v první testovací sestavě dnes představuje v mnohých případech snesitelné minimum. Velmi často se už u běžných počítačů setkáváme dvojnásobnou hodnotou. Zejména díky tomuto faktu se tedy flashové animace zdají v porovnání s SVG jako ty méně náročně.

Závěr

Hlavním cílem úvodní části této diplomové práce bylo teoreticky shrnout problematiku webových animací. V rámci praktické části se pak měla implementovat aplikace, která umožní uživateli vytvářet si jednoduché animace a exportovat je ve formátu SVG. V závěrečné části se pak mělo srovnat zatížení počítače při přehrávání flashových a SVG animací.

Vypracovaná diplomová práce nakonec splnila všechny požadované cíle. Byla zpracována rešerše, kde byly rozebrány technologie, které se v současné době používají pro tvorbu animací do internetových stránek. Důraz byl kladen obzvlášť na SVG animace, které měla vytvářet implementovaná aplikace a flashové animace, se kterými se pak vytvořené animace měly porovnat.

Rovněž se podařilo i úspěšně implementovat požadovanou aplikaci. Ta umožňuje uživateli vytvářet animace kombinováním sedmi typů prvků. Na časové ose si může definovat jednotlivé časové stavy a díky tomu dynamicky měnit dobu animace. Rozpracované animace je možné uložit a později zase načíst. Toto ukládání je realizováno dvěma způsoby. V původních požadavcích bylo uvedeno, aby se animace ukládaly do databáze MySQL, což bylo splněno. Nad rámec původního zadání byla ještě implementována druhá možnost, která uživateli umožňuje animace exportovat ve formátu XML. V tomto formátu pak může animaci samozřejmě i importovat. Tato metoda byla implementována zejména jako alternativa pro ukládání do databáze pro uživatele, kteří se rozhodnou využívat veřejný účet a nechtějí animace ukládat do databáze, kde by k nim měli přístup všichni uživatelé, využívající tento účet.

Aplikace je tedy plně použitelná pro vytváření jednoduchých animací. Do budoucna by se ale hodilo rozšíření možností pro jejich tvorbu. Určitě by bylo vhodné přidat další typy tvarů, ze kterých se animace skládají a rovněž implementovat i více druhů transformací. V současné podobě ale představuje aplikace kvalitní základ pro budoucí rozšíření. Tento fakt se potvrdil i v průběhu uživatelského testování, kterého se účastnili dva uživatelé. Ti měli za úkol provést jednoduché úkony v prostředí aplikace a vytvořit jednoduchou animaci. V průběhu tohoto testování byly zaznamenávány jejich reakce a připomínky. Oba nakonec tyto úkony bez větších problémů splnili a výsledky tohoto testování pouze potvrdily použitelnost aplikace. Z tohoto testování by se mohlo vycházet i při budoucím vývoji aplikace, protože oba uživatelé mimo jiné uvedli, že by bylo dobré přepracovat časovou osu a implementovat nástroj pro výběr barev.

Na závěr práce mělo být provedeno srovnání samotných animací. Konkrétně byla tato část zaměřena na srovnávání flashových a SVG animací, které byly vytvořeny s podobnou složitostí, strukturou a obsahem. Následovalo pak srovnávání nároků na počítač a využití jednotlivých jeho komponent. V tomhle srovnání poté vyšly nepatrně lépe flashové animace, které méně využívaly potenciálu procesoru a grafické karty, ovšem potřebovaly pro svůj chod více operační paměti. Rozdíly v zjištěných hodnotách ovšem byly velmi malé a i na 7 let starém počítači, na kterém byly obě technologie testovány, se animace zobrazovaly plynule. I to dokazuje kvalitu obou grafických formátů.

Literatura

[1] Animace. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2004, 10. 4. 2013 [cit. 2013-04-29]. Dostupné z: http://cs.wikipedia.org/wiki/Animace

[2] Scalable Vector Graphics. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-05-01]. Dostupné z: https://cs.wikipedia.org/wiki/Scalable_Vector_Graphics

[3] Adobe Flash. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2006, 9. 3. 2013 [cit. 2013-05-11]. Dostupné z: http://cs.wikipedia.org/wiki/Adobe_Flash

[4] Grafika.sk. *Flash - Výhody vs. nevýhody* [online]. 2006 [cit. 2013-05-11]. Dostupné z: http://grafika.sk/clanok/flash-vyhody-vs-nevyhody/

[5] Bye, bye, flash. *Chip.cz* [online]. 2011 [cit. 2013-05-12]. Dostupné z: http://earchiv.chip.cz/cs/earchiv/vydani/r-2011/chip-02-2011/bye-flash.html

[6] ActionScript. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2006, 19. 4. 2013 [cit. 2013-05-12]. Dostupné z: http://cs.wikipedia.org/wiki/ActionScript

[7] SVG - technologie 21. století pro web. *Interval.cz* [online]. 1999 [cit. 2013-05-12]. Dostupné z: http://interval.cz/clanky/svg-technologie-21-stoleti-pro-web/

[8] Průvodce SVG - SVG versus Flash. *Interval.cz* [online]. 2003 [cit. 2013-05-12]. Dostupné z: http://interval.cz/clanky/pruvodce-svg-svg-versus-flash/

[9] SVG má zelenou. *Zive.cz* [online]. 2001 [cit. 2013-05-12]. Dostupné z: http://www.zive.cz/clanky/svg-ma-zelenou/sc-3-a-102835/default.aspx

[10] Adobe Flash Catalyst: interaktivní designér Flash projektů. *Grafika.cz* [online]. 2009 [cit. 2013-05-13]. Dostupné z: http://www.grafika.cz/rubriky/software/adobe-flash-catalyst-interaktivni-designer-flash-projektu-136940cz

[11] Rozdíl mezi GIF a FLASH formátem. *Web Design* [online]. 2013 [cit. 2013-05-13]. Dostupné z: http://www.webdesign.izde.cz/clanek_gif_flash.asp

[12] Příprava obrázků. *Jak psát web* [online]. 2012 [cit. 2013-05-14]. Dostupné z: http://www.jakpsatweb.cz/obrazky-priprava.html

[13] THE PHP DOCUMENTATION GROUP. *PHP: PHP Manual* [online]. 1997, 2013-08-16 [cit. 2013-08-17]. Dostupné z: http://www.php.net/manual/en/

[14] *W3schools.com* [online]. 1999 [cit. 2013-08-17]. Dostupné z: http://www.w3schools.com/

[15] How to create ad banners in Flash CS4. In: *Flashexplained.* [online]. 2010 [cit. 2013-08-24]. Dostupné z: http://flashexplained.com/banners/how-to-create-ad-banners-in-flash-cs4/

[16] Flash Animation. In: *Techbyte* [online]. 2008 [cit. 2013-08-24]. Dostupné z: http://www.sfu.ca/~tutor/techbytes/Flash/fl1.html

[17] Český statistický úřad: Animovaný graf: Počet hostů v hromadných ubytovacích zařízeních v Plzeňském kraji. *Krajská správa ČSÚ v Plzni* [online]. 2013, 14.5. 2012 [cit. 2013-04-29]. Dostupné z: http://www.czso.cz/xp/redakce.nsf/i/animovany_graf:_pocet_hostu_v_hromadnych_ubytovacich_zarizenich_v_plzenskem_kraji

[18] Fresh ME. *Fresh ME* [online]. 2010 [cit. 2013-04-29]. Dostupné z: http://www.freshme.cz/

[19] *AutoRevue.cz – Auta, testy, novinky, fotografie* [online]. 2013 [cit. 2013-08-27]. Dostupné z: http://www.autorevue.cz/

[20] *Placekitten* [online]. [cit. 2013-08-27]. Dostupné z: http://placekitten.com/

[21] *Placehold.it* [online]. [cit. 2013-08-27]. Dostupné z: http://placehold.it/

[22] Bézierova křivka. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 2013-08-05 [cit. 2013-08-27]. Dostupné z: http://cs.wikipedia.org/wiki/B%C3%A9zierova_k%C5%99ivka

Příloha A – Animace ve formátu XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tvary>
	<tvar typ="bezier" opakovani="3" fill="#123789" hodnota="5"
	animace="Přímá" nastaveni="134x198" nastaveni2="Anox#00baff"
	w="394" h="298" id="3436" >
		<stavBod1>
			<bod1 id="3436" cas="0" x="4" y="298"></bod1>
			<bod1 id="3436" cas="4" x="4" y="298"></bod1>
			<bod1 id="3436" cas="8" x="4" y="298"></bod1>
		</stavBod1>
		<stavBod2>
			<bod2 id="3436" cas="0" x="52" y="177"></bod2>
			<bod2 id="3436" cas="4" x="115" y="212"></bod2>
			<bod2 id="3436" cas="8" x="285" y="215"></bod2>
		</stavBod2>
		<stavBod3>
			<bod3 id="3436" cas="0" x="212" y="232"></bod3>
			<bod3 id="3436" cas="4" x="370" y="232"></bod3>
			<bod3 id="3436" cas="8" x="394" y="245"></bod3>
		</stavBod3>
		<stavBod4>
			<bod4 id="3436" cas="0" x="398" y="296"></bod4>
			<bod4 id="3436" cas="4" x="398" y="296"></bod4>
			<bod4 id="3436" cas="8" x="398" y="296"></bod4>
		</stavBod4>
	</tvar>
	<tvar typ="kvadraticka" opakovani="3" fill="#123789" hodnota="5"
	animace="Přímá" nastaveni="214x199" nastaveni2="Anox#00baff"
	w="365" h="294" id="266" >
		<stavBod1>
			<bod1 id="266" cas="0" x="190" y="297"></bod1>
			<bod1 id="266" cas="4" x="190" y="297"></bod1>
			<bod1 id="266" cas="8" x="190" y="297"></bod1>
		</stavBod1>
		<stavBod2>
			<bod2 id="266" cas="0" x="369" y="201"></bod2>
			<bod2 id="266" cas="4" x="254" y="225"></bod2>
			<bod2 id="266" cas="8" x="176" y="223"></bod2>
		</stavBod2>
		<stavBod4>
			<bod4 id="266" cas="0" x="400" y="302"></bod4>
			<bod4 id="266" cas="4" x="400" y="302"></bod4>
			<bod4 id="266" cas="8" x="400" y="302"></bod4>
		</stavBod4>
	</tvar>
	<tvar typ="kvadraticka" opakovani="3" fill="#123789" hodnota="5"
	animace="Přímá" nastaveni="133x198" nastaveni2="Anox#0088ff" w="45"
	h="297" id="1573" >
		<stavBod1>
			<bod1 id="1573" cas="0" x="258" y="299"></bod1>
			<bod1 id="1573" cas="4" x="258" y="299"></bod1>
			<bod1 id="1573" cas="8" x="258" y="299"></bod1>
		</stavBod1>
		<stavBod2>
			<bod2 id="1573" cas="0" x="133" y="198"></bod2>
			<bod2 id="1573" cas="4" x="239" y="223"></bod2>
			<bod2 id="1573" cas="8" x="285" y="224"></bod2>
		</stavBod2>
```

```
            <stavBod4>
                  <bod4 id="1573" cas="0" x="45" y="297"></bod4>
                  <bod4 id="1573" cas="4" x="45" y="297"></bod4>
                  <bod4 id="1573" cas="8" x="45" y="297"></bod4>
            </stavBod4>
      </tvar>
</tvary>
```

Příloha B – Animace ve formátu SVG

```
<svg xmlns="http://www.w3.org/2000/svg" height="300px" width="400px" sty-
le="border:2px solid black;" version="1.1">
      <g>
            <path id="3436" d="M 4,298 C 52,177 212,232 398,296 Z"
                  stroke="#123789" stroke-width="5" fill="#00baff" >
                  <animate id="move" repeatCount="3" values="M 4,298
                  C 52,177 212,232 398,296 Z;M 4,298 C 115,212 370,232
                  398,296 Z;M 4,298 C 285,215 394,245 398,296 Z;M 4,298
                  C 52,177 212,232 398,296 Z;" dur="4s" attributeName="d"
                  begin="0s">
            </path>
      </g>
      <g>
            <path id="266" d="M 190 297 q 179 -96 210 5 Z"
            stroke="#123789" stroke-width="5" fill="#00baff" >
                  <animate id="move" repeatCount="3" values="M 190 297
                  q 179 -96 210 5 Z;M 190 297 q 64 -72 210 5 Z;M 190 297
                  q -14 -74 210 5 Z;M 190 297 q 179 -96 210 5 Z;"
                  dur="4s" attributeName="d" begin="0s">
            </path>
      </g>
      <g>
            <path id="1573" d="M 258 299 q -125 -101 -213 -2 Z"
            stroke="#123789" stroke-width="5" fill="#0088ff" >
                  <animate id="move" repeatCount="3" values="M 258 299
                  q -125 -101 -213 -2 Z;M 258 299 q -19 -76 -213 -2 Z;
                  M 258 299 q 27 -75 -213 -2 Z;M 258 299 q -125 -101 -213
                  -2 Z;" dur="4s" attributeName="d" begin="0s">
            </path>
      </g>
</svg>
```

Příloha C – Výsledky uživatelského testování

První uživatel, který aplikaci testoval, studuje Informatiku. Zde jsou poznatky z jeho testování:

1. Našel a provedl bez problémů.
2. Našel a provedl bez problémů.
3. Problém s průhledností, nevěděl, jak jí dosáhnout. Také dvojklik pro kreslení je nedostatečně výrazně popsán.
4. Našel a provedl bez problémů.
5. Našel a provedl bez problémů.
6. Časová osa je prý trošku nepřehledná. Doporučuje udělat jí pomocí posuvníku, jako například v programu After Effects. Jinak v pořádku.
7. Při manipulaci sem tam došlo k neúmyslnému vytvoření nového tvaru. Jinak v pořádku.
8. Našel a provedl bez problémů.
9. Našel a provedl bez problémů.
10. Našel a provedl bez problémů. Pouze je zde zbytečné dotázání, zda chce skutečně smazat daný tvar, když už našel tlačítko.

Zde si můžeme prohlédnout poznatky o testování druhým uživatelem, který studoval informatiku a nyní pracuje jako tester.

1. Našel a provedl bez problémů.
2. Našel a provedl bez problémů.
3. Našel a provedl. V menu je vykreslit obdélník, čímž lze dosáhnout vykreslení čtverce, ale už to není čtverec, proto by byl vhodnější nějaký univerzálnější název spojující oboje. Taky transparentní barva je matoucí, protože to není možné nijak ověřit, až vytvořením tvaru. Rovněž by prý bylo vhodné implementovat nějaký nástroj pro výběr barev.
4. Našel a provedl bez problémů.
5. Našel a provedl bez problémů.
6. Našel a provedl. Časovou osu by bylo lepší nějakým způsobem popsat a zvýraznit.
7. Našel a provedl bez problémů.

8. Našel a provedl bez problémů.
9. Našel a provedl bez problémů. Blok v pravé části aplikace by neměl mít název Log, protože se tam nachází i prvky menu a název nevystihuje skutečnou podstatu.
10. Našel a provedl bez problémů.