VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ PRO GRAFICKÝ NÁVRH HTML5 APLIKACÍ

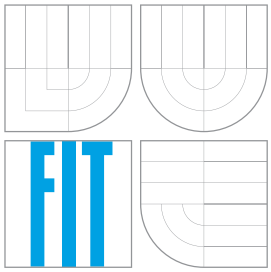
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK SABOVČIK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ PRO GRAFICKÝ NÁVRH HTML5 APLIKACÍ

GRAPHICAL DESIGN TOOL FOR HTML5 APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FRANTIŠEK SABOVČIK
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Bakalářská práce se zabývá návrhem a implementací nástroje pro grafický návrh HTML5 aplikací vykreslovaných na Canvas. Tento editor umožnuje vizuální způsobem vytvářet uživatelské rozhraní pomocí pozicování, transformace a sdružování grafických prvků s následným exportem do cílové aplikace. V práci je popsán proces vývoje tohoto nástroje od popisu stávajícího problému, přes průzkum teoretických poznatků nutných pro vývoj, návrh, až k následné implementaci výsledné aplikace. V závěru práce je popsána zpětná vazba potenciálních uživatelů, která vyhodnocuje úspěšnost plnění zadaných požadavků.

Abstract

This bachelor thesis is focused on design and implementation of a tool for graphical design of HTML5 aplications drawn on Canvas. The editor helps to create user interface visually with positioning, transformation and grouping of image sprites and export to target aplication. The work describes process of development of this tool, involving description of solved problem, exploration of theoretical informations, design and implementation of final product. Feedback of potential users is described in the end of the work, evaluating success of solving defined requirements.

Klíčová slova

HTML5, Grafický nástroj, JavaScript, HTML5 Canvas, Uživatelské rozhraní, Webové aplikace

Keywords

HTML5, Graphical Tool, JavaScript, HTML5 Canvas, User Interface, Web Applications

Citace

František Sabovčik: Nástroj pro grafický návrh HTML5 aplikací, bakalářská práce, Brno, FIT VUT v Brně, 2015

Nástroj pro grafický návrh HTML5 aplikací

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatné pod vedením pana Ing. Vítězslava Berana, Ph.D

.....
František Sabovčik
19. 5. 2015

Poděkování

Chtěl bych poděkovat panu Ing. Vítězslavu Beranovi, Ph.D. za odbornou pomoc a dále všem, kteří mě v mém snažení podporovali.

Obsah

Úvod

V současné době je možné sledovat výrazný rozvoj webových technologií, jak na straně serveru, tak na straně klienta. Klientské části interaktivních aplikací v minulosti dominovala platforma Adobe Flash, která je nahrazována novějším HTML5.

Na rozdíl od Adobe Flash, pro HTML5 neexistuje obecný editor umožnující návrh grafiky a uživatelského rozhraní, jehož výstup by byl přímo spustitelný ve vyvíjené aplikaci. Skutečnost, že tento nástroj chybí, snižuje efektivitu vývojového procesu (viz 2.1). Cílem bakalářské práce je zaplnit tuto mezeru a **poskytnout univerzální nástroj ke grafickému návrhu HTML5 aplikací**. Účelem této aplikace je **zrychlit workflow**, a to zvýšením efektivity procesu **vytváření layoutu** a pozicování jednotlivých prvků na obrazovku.

Práce se zaměřuje na podmnožinu HTML5, která je charakterizována vykreslováním grafiky na element Canvas a používáním vykreslovacích knihoven (např. Pixi.js) nebo herních frameworků, které vykreslování obsahují v sobě. Tato technologie je určena především pro vývoj her a interaktivních prezentací, kde je požadován vysoký výkon nebo vizuální efekty. Zavedení navrhovaného nástroje je vhodné právě pro tyto aplikace.

Teoretickou část pokrývá kapitola 1, jsou zde probrány koncepty **vykreslování na Canvas**, existence **knihoven** zprostředkující jeho efektivnější využití a dále také **návrhové vzory** a jiné techniky důležité pro tvorbu kvalitních aplikací. V kapitole 2 je poté proveden návrh vycházející z požadavků a průzkumu trhu. v rámci nějž jsou navrženy základní **koncepty** aplikace, její **architektura**, **algoritmy** a v neposlední řadě efektivní **uživatelské rozhraní**. Konkrétním detailů v **implementaci** výsledného produktu, které stojí za zmínku, se věnuje kapitola 3. **Vyhodnocení** úspěšnosti řešení požadavků je provedeno na základě zpětné vazby od potenciálních uživatelů v podobě testovacího využití výsledné aplikace a odpovědí na zadané otázky.

Kapitola 1

HTM5 aplikace založené na Canvasu

Tato kapitola obsahuje potřebné teoretické znalosti nutné k tvorbě návrhu a řešení cíle bakalářské práce.

1.1 Vykreslování grafiky na Canvas

Bakalářská práce se zaměřuje na aplikace vykreslované na Canvas, proto je nutné pochopení fungování tohoto principu. Canvas je element ze specifikace HTML5, představuje rastrové plátno, na které je možné za pomocí API vykreslovat grafické prvky, v HTML dokumentu je reprezentován tagem <canvas>[8].

Canvas představuje nízko-úrovňové API umožnující vykreslení geometrických primitivů, rastrových obrázků a textu. Příkladem může být tento kód vykreslující trojúhelník:

```
1  var context = document.getElementById("canv").getContext("2d");
2  var width = 125;  // Triangle Width
3  var height = 105; // Triangle Height
4  var padding = 20;
5
6  // Draw a path
7  context.beginPath();
8  context.moveTo(padding + width/2, padding);          // Top Corner
9  context.lineTo(padding + width, height + padding); // Bottom
       Right
10 context.lineTo(padding, height + padding);          // Bottom
       Left
11 context.closePath();
12
13 // Fill the path
14 context.fillStyle = "#ffc821";
15 context.fill();
```

Po vykreslení požadovaného prvku jsou body „trvale“ zapsány na canvasu a nelze je dále měnit, jistým ústupkem jsou metody context.save() a context.restore() umožnující zá-

sobníkově pracovat s aktuálním vykreslovacím kontextem, tzn. aktuální barvy, transformce atd. V současnosti je podstatná část vykreslování na Canvas akcelerována pomocí GPU [10] [11], což má za následek podstatné zvýšení výkonu.

1.2 Použití knihoven pro vykreslování

Pro použití v reálných aplikacích je žádoucí využití tzv. 2D rendererů, které zapouzdřují nízkoúrovňové vykreslování do objektového modelu, řada z nich je propojena s herními enginy (např. impactjs, cocos2d-x) nebo zaopatřují pouze vykreslování (EaselJS, Pixi.js). Nad rámec vykreslování primitivů na Canvas 2D renderery nabízejí abstraktní objektovou vrstvu, ve které je možné grafické objekty sdružovat do hierarchického modelu, kdy grafická transformace rodiče je přenášena na potomka, lze tedy pracovat se zanořenými skupinami objektů. Příklad práce s nimi v API knihovny Pixi.js:

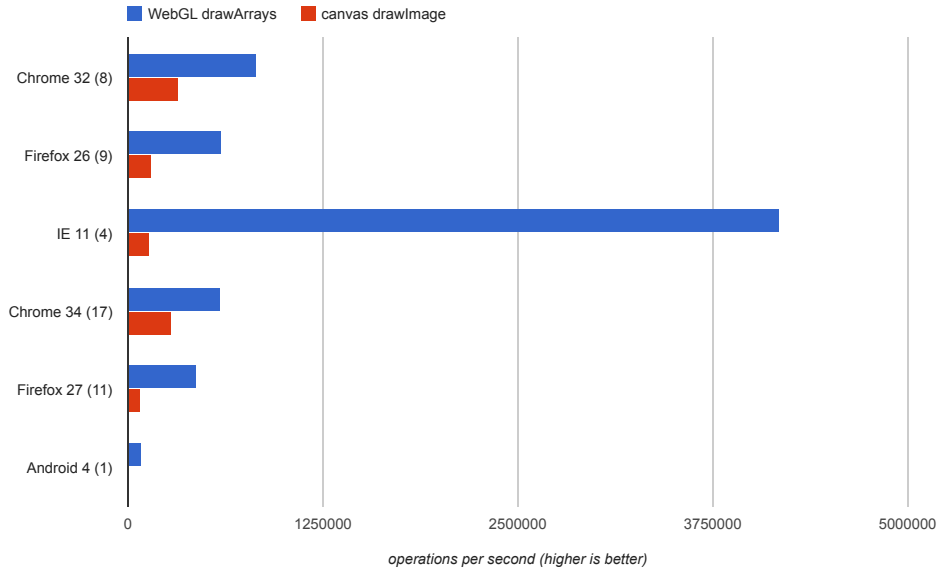
```
...
var container = new PIXI.DisplayObjectContainer();
cotnainer.addChild(triangle);
cotnainer.addChild(rectangle);
container.rotation = Math.PI/4;
...
```

Z abstrakce také vychází možnost měnit parametry jako jsou souřadnice nebo natočení grafickým objektům jednotlivě, o samotné překreslení na Canvas se stará renderer, tímto způsobem lze docílit animací:

```
setTimeout(function(){ sprite.x +=5; }, 100);
```

Dále zaopatřují interaktivitu s uživatelem. Nad jednotlivými grafickými objekty lze zavěsit události jako kliknutí myši nebo dotyk na dotykové obrazovce, které nemohou být zprostředkovány samotným Canvasem, který je pouze rastrem jednotlivým pixelů.

Některé 2D renderery (např. Pixi.js) jsou schopné pro vykreslování použít nízkoúrovňové WebGL API (s fallbackem na Canvas), které vykresluje s plnou podporou GPU a zvyšuje tak výkon [12].



Obrázek 1.1: Srovnání výkonu při vykreslování na Canvas a WebGL [12]

1.3 Objektové programování v jazyce JavaScript

Protože je vyvíjený nástroj psaný v JavaScriptu, stejně jako aplikace na které je cílen, je na místě pochopení principům OOP v tomto jazyku, neboť kvalitní objektový návrh zvyšuje celkovou kvalitu výsledného produktu.

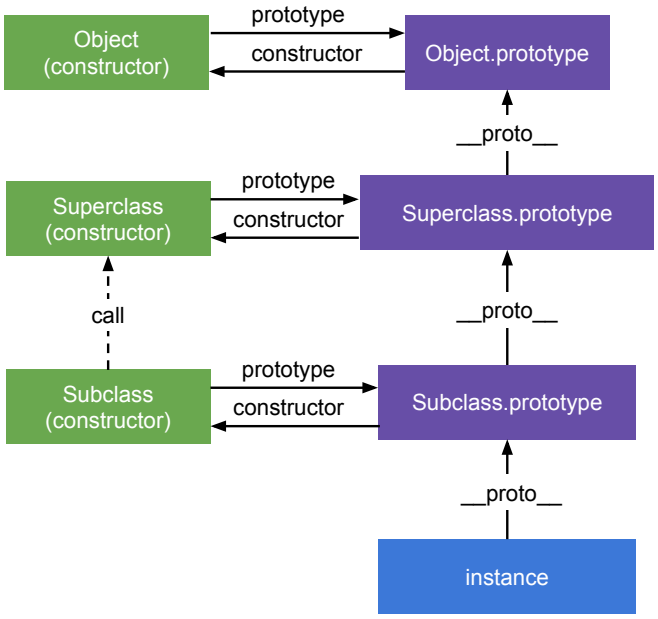
JavaScript se řadí mezi tzv. prototype-based dynamické programovací jazyky. Každému objektu je možné přiřadit prototyp (prototype), odkaz na tzv. prototypový objekt (prototype object), na který jsou delegovány všechny zprávy, které není schopen sám objekt obsloužit [13].

Nové objekty jsou tvořeny operátorem new, který tzv. konstruktorem iniciuje jeho hodnoty a nastavuje mu odpovídající prototyp, který získává z použitého konstrukturu [14].

```
1 var MyConstructor = function(x, y){
2   this.x = x;
3   this.y = y;
4 }
5
6 MyConstructor.prototype.inheritedFunction = function(){ this.x =
      150; };
7
8 var object1 = new MyConstructor(125, 135);
9 var object2 = new MyConstructor(140, 150);
```

Prototyp je vhodné použít pro sdílených funkcí všemi instancemi, především z hlediska výkonnosti, tato varianta je dokonce až 15× rychlejší oproti přiřazení funkce objektu v konstruktoru [15].

Stejně tak i prototypový objekt může mít vlastnost prototype, čímž se vytváří tzv. prototypový řetězec (prototype chain), který je možné využít k implementaci dědičnosti.



Obrázek 1.2: Diagram ukazující příklad prototypové dědičnosti v JavaScriptu

Popisované jazykové konstrukce jsou platné pro specifikaci EcmaScript 5 a starší, EcmaScript 6 zavádí klíčové slovo class, které zapouzdřuje definici tříd a dědičnosti [16], ačkoliv není prozatím implementována v současných prohlížečích.

1.4 Optimalizace kódu

Tato sekce se věnuje technikám, aplikovatelným při vývoji nástroje, které pomohou zvýšit rychlost provádění kódu. Přestože se zvyšuje výkon výpočetních systému a běhových prostředí, je na místě snaha o optimalizaci, také kvůli zvýšené režii, kterou si kvůli své povaze s sebou nesou interpretované vysokoúrovňové jazyky.

1.4.1 Chybné uvolňování paměti

Garbage collector je mechanismus, používaný převážně v interpretovaných jazycích, uvolňující dynamicky alokovanou paměť, která již není dále využita. Garbage collector prochází reference na objekty a paměť, na niž již není referencováno, je dealokována. [17]

V JavaScriptu je chybou snaha o „manuální dealokování“ pomocí operátoru delete, který není pro tyto účely určen a naopak způsobuje zpomalení kódu v mechanismus hidden classes v enginu V8 [19]. Nejlepší způsobem dealokace je zamezení používání globálních proměnných, využití implicitní platnosti proměnných (např. lokální proměnné) a ponechání dealokaci garbage collectoru [20].

1.4.2 Komunikace s hostitelskými objekty

Při komunikaci vně prostředí JavaScriptu, jako je práce s DOM objekty renderovanými na obrazovku, dochází k výraznému zpoždění. Řešením je omezení zbytečných interakcí s DOM a složitých selektorů při výběru elementů [18].

1.4.3 Zanořování closure a dědičnost

Při zanořování tzv. closure dochází při při prohledávání nadřazených jmenných prostorů a vyhodnocování proměnných ke zpomalení. Ačkoliv patří closures k silným stránkám JavaScriptu, je vhodné se vyhnout jejich přehnanému používání. Podobný problém nastává při prototype chain, kdy je nutné při získání proměnné projít řetez dědičnosti až k objektu, který danou proměnnou obsahuje (v případě neexistence až ke konci).

1.4.4 Úniky paměti (memory leaks)

Paměť, která nemůže být uvolněna garbage collectorem, nejčastěji z důvodu cyklických závislostí, například při zavěšení události na DOM element s listenerem, který má (např. pres closure) referenci zpět na daný element.[21] Řešením je využívat osvědčené systémy pro připojování události (např. jQuery), které těmito neduhy netrpí. [22]

1.5 Návrhové vzory a architektura aplikace

Tato kapitola probírá různé návrhové vzory a principy použitelné pro návrh architektury aplikace vyvíjené v rámci této bakalářské práce.

Použití návrhových vzorů je vhodné z několika důvodů, především se jedná o způsob, jak využít nejlepší postupy při vývoji, které byly v minulosti získány, s cílem zrychlit vývoj a také zlepšit kvalitu samotného kódu [23].

Speciální skupinou jsou architekturální návrhové vzory, těm je věnována pozornost jako prvním. Norma IEEE 1471 2000 architekturu definuje takto [24]:

„*Základní organize systému, daná jeho komponentami, vzájemnými vztahy těchto komponent a jejich vztahy k okolnímu prostředí a principy řídícími jejich návrh a evoluci.*“

1.5.1 Model-View-Controller (MVC)

Jedním z nejpopulárnějších návrhových vzorů pro návrh uživatelského rozhraní [25] je tzv. MVC, rozděluje aplikaci do tří základních částí Model, View (Pohled) a Controller:

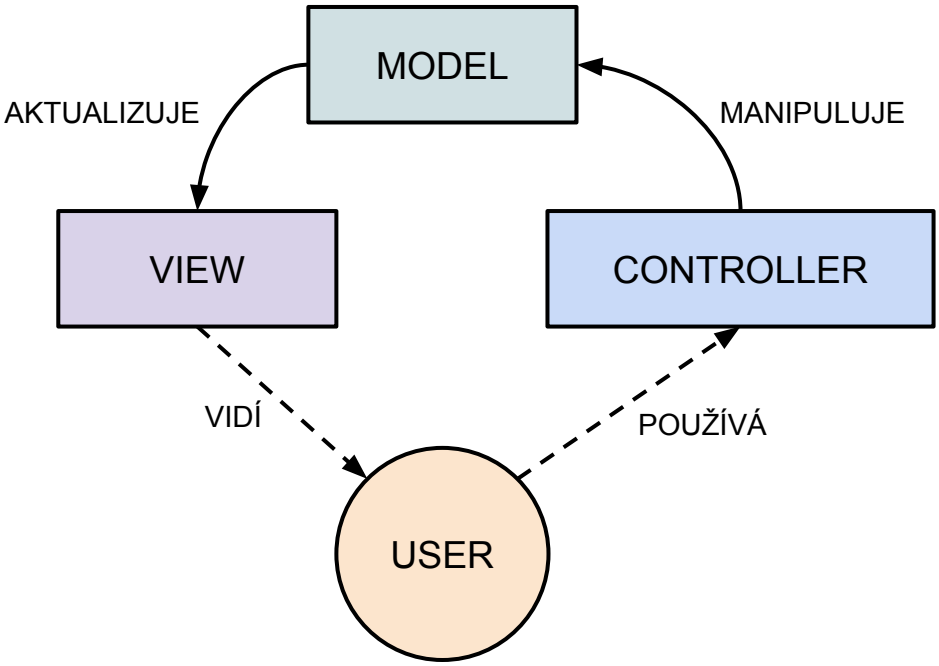
Model: Obsahuje data pro běh aplikace, reprezentuje její aktuální stav a veškerou business logiku. Model neví o existenci View ani Controlleru. Po změně stavu je View notifikován pomocí určitého rozhraní, např. použitím návrhového vzoru Observer-Observable.

View: Tato vrstva reprezentuje vizuální výstup, který vidí uživatel. Smyslem view je oddělit logiku aplikace od její vizuální reprezentace, díky tomu je možné mít více různých View pro

stejná data, vykreslit tak např. XML, tabulku nebo graf bez nutnosti zasahovat do zbytku aplikace.

Controller: Přijímá uživatelský vstup (např. kliknutí myši) a aktualizuje stav modelu, který poté zpětně notifikuje view. Z pohledu webových prezentací v určitých případech vzor MVC plně neodpovídá všem požadavkům, které jsou na něj kladeny, proto vzniklo několik variant (viz níže).

Obrázek 1.3: Typická spolupráce komponent v MVC [26]



1.5.2 Model-View-Presenter (MVP)

Jedná se o obdobu vzoru MVC, také zde vystupuje Model a View, místo Controlleru je zde ovšem Presenter, který plní roli prostředníka mezi View a Modelem. Hlavním rozdílem oproti MVC je prohození pozice Modelu a Presenteru, kdy v MVP roli hraje roli ústředního prvku spojujícího Model a View.

View – Je pasivním hráčem vizualizující data, uživatel komunikuje s View, který předává požadavky (uživatelské události) na presenter.

Presenter – Obsahuje logiku zpracování událostí, po obdržení komunikuje odpovídajícím způsobem s Modelem a zpětně předává data, které může upravit odpovídajícím způsobem. Model - Obsahuje doménovou logiku a další služby, neví o existenci View a Presenteru, model v MVP je totožný jako v MVC.

Vzor MVP byl rozdělen na dva odvozené návrhové vzory, a to Passive View a Supervising Controller [27].

Passive View – varianta, kdy je pohled pasivní a neobsahuje naprosto žádnou logiku. Data k vykreslení do view předává pouze Presenter. Výhodou tohoto přístupu je vyšší

testovatelnost, ale za cenu vyšší komplexnosti při vývoji.

Supervising Controller – Jde o určitý kompromis mezi MVC a MVP, data mohou být „nabindována“ z Modelu přímo do View, na komplexnějších operacích se podílí presenter. Nevýhodou tohoto přístupu je horší testovatelnost a menší nezávislost jednotlivých částí.

1.5.3 Ostatní varianty MVC

S postupem času se s různorodými požadavky utvořila motivace k tvorbě dalších variant MVC, jsou to například **HMVC** (Hiearchical MVC) nebo **PAC** (Presentation-Abstraction-Control), které vycházející vstříc návrhu uživatelského rozhraní pomocí widgetů tím, že vytvářejí určitou hierarchii.

Jiný přístup ukazuje architektura **Model-View-ViewModel** z dílny Microsoftu, je založená na vzoru **PresentationModel** Martina Fowlera [29], která opět obsahuje View a Model, View je ovšem propojen s tzv. ViewModelem pomocí data bindingu, který transformuje data z Modelu do podoby přijatelné pro mapování [30]. Je používán např. na platformě .NET nebo v některých JavaScriptovým frameworcích (např. Knockout) [33].

1.5.4 S.O.L.I.D.

S.O.L.I.D. je zkratka zaštiťující pět základních principů objektově orientovaného návrhu definovaných Robertem C. Martinem, jejichž dodržování pomáhá docílit kvalitního návrhu a dosáhnout kvalit jako je velká soudržnost (high cohesion) a nízká provázanost (low coupling), což znamená, že komponenty nejsou staticky „zadrátované“ do systému (je možné je vyměňovat), zároveň ovšem spolu dokáží, pokud možno, co nejlépe spolupracovat.

S	Single responsibility principle	Třída by měla mít jen jednu odpovědnost, tzn. třídu by měla být schopná ovlivnit pouze jediná změna specifikace.
O	Open/closed principle	Softwarové entity by měly být otevřeny pro rozšíření, ale uzavřeny pro modifikaci.
L	Liskov substitution principle	Objekty by mely být nahraditelné instancemi jejich podtypu bez úprav zbytku programu.
I	Interface segregation principle	Rozhraní by měla být co nejspecifičtější. Pokud třída implementuje dané rozhraní, měla by zároveň plně vyhovovat všem vstupům, které toto rozhraní vyžadují.
D	Dependency inversion principle	Konkrétní by mělo záviset na abstraktním, z tohoto principu vychází např. Dependency Injection

1.5.5 Publish/Subscribe

Jedná se o návrhový vzor zabývající se komunikaci mezi objekty, jehož prostřednictvím lze docílit **škálovatelnosti** (scalability) aplikace a **nízké provázanosti** (low coupling) jednotlivých komponent. [31]

Návrhový vzor popisuje dva typy objektů: **vydavatele** (publisher) a **odběratele** (subscriber). Vydavatel při zaslání zprávy přímo neurčuje, komu má být určena, ale charakterizuje ji určitou třídou bez vědomí, zda nějaký příjemce poslouchá na daný typ zprávy [32]. Podobným způsobem odběratel vyjadřuje zájem o jeden nebo více typů zprávy bez vědomí, od kterého objektu tyto zprávy obdrží.

1.6 Běhová prostředí

Tato sekce se zabývá možnostmi prostředí pro běh HTML5 aplikací, a tak navrhuje způsoby, které se nabízejí pro realizaci vyvíjeného nástroje. HTML5 aplikace není nutné spouštět pouze v prohlížeči s backendem na vzdáleném serveru, ale je zde také možnost spuštění čistě na klientovi v jednom z běhových prostředí [34].

Brackets Shell

Toto běhové prostředí je backendem pro projekt Brackets IDE společnosti Adobe, je však možné jej použít i pro vlastní projekt, z toho pramení výhody i nevýhody tohoto produktu. Toto prostředí je předmětem intenzivního vývoje, za kterým stojí velká společnost, je dobře integrované s Gruntem a používá Node.js pro rozšíření API. Nevýhodou je nepřívětivost pro vývojáře, kdy je nutné kompilovat prostředí ze zdrojových souborů a nedostatečná dokumentace.

AppJS

Běhové prostřední využívající Node.js k doplnění práce se souborovým systémem a jiných funkcí potřebných pro vývoj na desktopu. Projekt není plně odladěn a bohužel se zdá, že je tento projekt již mrtvý, tudíž není vhodný pro nasazení.

NW.js

Projekt využívající Chromium a Node.js, které běží v jednom vlákně a komunikace je proto výkonější, je sponzorován společností Intel a je intenzivně vyvíjen. Má kvalitní dokumentaci a je snadné začít nový projekt. K prostředí existují nástroje, které projekt automaticky zabalí pro různé platformy, tyto nástroje jsou ovšem vyvíjeny třetími stranami a ne vždy jsou odladěné.

TideSDK

V rámci průzkumu bylo zkoumáno i toto prostředí, kromě JavaScriptu je v něm možné vyvíjet i v další jazycích (Python, Ruby, PHP), ovšem bylo jištěno, že i tento projekt již není nadále vyvíjen, proto není vhodný pro další využití.

1.7 Pravidla pro návrh uživatelského rozhraní

Při vývoji aplikace je vhodné správným způsobem navrhnout uživatelské rozhraní, aby se zrychlil proces učení, zvýšila se efektivita práce, snížila chybovost a aby se samotné používání aplikace stalo pro uživatele zážitkem. V této sekci je nastíněno několik pravidel, jejichž dodržování přispívá k dosažení zmiňovaných kvalit.

Především je nutné si uvědomit **cílovou skupinu**, pro kterou je uživatelské rozhraní navrhováno, to určuje řadu parametrů, jako je např. přípustná komplexnost, která bude pro expertního uživatele vyšší. Také je třeba přizpůsobit rozložení a jiné atributy tomu, na co je daný uživatel zvyklý z jiných prostředí podobného zaměření [35].

Důležitým znakem je **konzistence**, tzn. že podobné postupy se provádějí podobně nebo stejně. Dále je vhodné uživatele odpovídajícím způsobem **notifikovat** o prováděných akcích, aby byl obeznámen o jejich výsledku. Akce je vhodné rozdělit do **jednotlivých kroků**, kterými jej uživatelské rozhraní provede, ty které není v daný moment možné vykonat, by měli být neaktivní, aby se předcházelo chybnému chování. Uživatelské rozhraní by mělo být **předvídatelné** a uživatel by měl být iniciátorem provádějící akce, v případě, že dojde k nechtěné akci, mělo by být možné se **vracet zpět** nebo ji opětovně provést. V neposlední řadě by mělo být uživatelské rozhraní přehledné a uživatel by neměl být nucen pamatovat si jednotlivé části rozhraní, platí pravidlo 7 ± 2, které říká, že člověk je schopen si krátkodobě zapamatovat 5–9 skutečností [35].

Pro rozložení uživatelského rozhraní je doporučováno řídit se těmito zásadami [35]:

- **Vyváženost**: uživatelské rozhraní by mělo být rovnoměrně zaplněno, neměla vznikat prázdná místa, kde se nenachází žádné ovládací prvky.
- **Souměrnost**: levá i pravá strana rozložení by měla být souměrná, nesouměrné rozložení působí rozměrněji.
- **Pravidelnost**: rozměry, vzhled a vzdálenost totožných ovládacích prvků by měly být stejné, aby se zabránilo zmatení uživatele, který se takto nebude moct orientovat dle stejných znaků.
- **Předvídatelnost**: uživatel by měl chápat funkčnost všech částí aplikace stejně, toho je nutné docílit totožným stylem a logikou.
- **Následnost**: ovládací prvky je vhodné rozmístit podle logiky věci z hlediska zásad pro tok ovládacím prvků, který je shora–dolů a zleva–doprava.
- **Proporce**: je kladen důraz na celek, poměry jednotlivých délek, jejich vzdálenost apod.

1.8 Knihovny pro uživatelské rozhraní

Tato sekce se zabývá nástroji zrychlující vytváření uživatelského rozhraní, mezi ně můžeme řadit CSS frameworky [36] nebo Javascriptové UI knihovny, které obsahují sady ovládacích prvků, vytvořených v HTML, CSS a JavaScriptu, které lze přímo použít pro projekt. Při vykreslování na Canvas žádné z těchto knihoven sice nemohou být použity, ale HTML a CSS může být použito jako doplněk uživatelského rozhraní mimo samotný Canvas. Byly zkoumány následující:

w2ui

Jedná se o jednoduchou sadu widgetů využívajících jQuery, výhodou je licence MIT, která umožňuje použití i v nesvobodných projektech. Nedostatkem je menší počet widgetů a ačkoliv se jedná o subjektivní dojem, vizuální vzhled není natolik vyzrálý.

webix

Pokročilá knihovna obsahující velký počet widgetů s propracovaným vzhledem, umožňuje také propojení s angular.js. Další výhodou je možnost použití motivů [38] a deklarativní definice vzhledu.

Nevýhodou je cena, která začíná okolo 469$, v případě, že vývojář nechce platit, je nutné vydat výslednou aplikaci pod licencí GNU GPL. Faktem snižujícím přínos pro využití v této aplikaci je skutečnost, že bude potřeba pouze několika základních widgetů.

Bootstrap

Bootstrap je CSS framework distribuovaný pod licencí MIT, který umožňuje zjednodušení definice layoutu stránky s podporou responzivního designu [43], který však pro účely tohoto projektu není nutný.

Mimo to obsahuje především sadu různých ovládacích prvků, které lze použít v projektu. Výhodou Boostrapu je jeho upravitelnost, naprostá většina widgetů je definována primárně v CSS, tudíž lze jednoduchým zásahem do stylů přizpůsobovat vzhled [43].

Za zmínku stojí také velké množství motivů dostupných na internetu. Nevýhodou je poněkud menší výběr widgetů, což je způsobeno zaměřením Bootstrapu [41], to lze ovšem překonat použitím doplňků vývojářů třetích stran [42].

1.9 Existující nástroje pro grafickým návrh

Je nutné podrobné zkoumání existujících produktů, aby výsledná aplikace neduplikovala již existující řešení, ale také se inspirovala již existujícím softwarem, překonávala jeho možnosti a zaplňovala tím mezeru na trhu. [1]

Google Web Designer a Adobe Edge

Animační nástroje vytvořený společnostmi Google a Adobe. Jsou spojeny v tomto jednom popisu, neboť se jedná o velmi podobné produkty. Dokáží exportovat animace do CSS, v současné době jsou určené především pro vytváření webových bannerů. Tyto aplikace mají sice pokročilé uživatelské rozhraní a silné hráče, kteří stojí za vývojem, ale nejsou určeny pro vývoj aplikací založených na Canvasu, tudíž se míjejí s cílem této práce [2] [3].

Cocos Studio

Jedná se o grafický editor k hernímu frameworku Cocos2D-X, obsahuje řadu pokročilých funkcí, zvládá pozicovat sprity, používat pokročilé editační funkce a vytvářet animace, i s použitím tzv. kostí [4]. Nevýhodou tohoto nástroje je chybějící podpora pro GNU/Linux. Z hlediska požadavků na univerzálnost je nevýhodou jeho omezení na jeden herní framework a jeho primární zaměření na vývoj her.

Adobe Flash CC

Pokročilý nástroj pro tvorbu interaktivních aplikací a animací, původně určený pro výstup na platformě Flash (SWF), v posledních verzích podporuje i export do HTML5 s knihovnou EaselJS [5]. Jedná se o vyzrálý nástroj a je inspirací i pro vyvíjený nástroj v rámci této bakalářské práce, ovšem v exportu do HTML5 trpí nedostatečnou možností práce s kódem (např. nemožnost provázat grafický prvek s určitou programovou třídou), což činí vývoj aplikací značně problematický, zároveň programátor nemá možnost vybrat vlastní vykreslovací knihovnu nebo herní engine. Nevýhodou může být vyšší pořizovací cena.

Unity 3D a Unreal Engine

Tyto herní enginy, dalo by se říct herní platformy, byly zařazeny do toho přehledu, protože v nedávné době byly započaty snahy o export do HTML5 [6][7] a také umožňují vývoj 2D her, nicméně primárně se jedná o komplexní nástroje pro vývoj 3D her. Pro reálné použití s HTML5 prozatím nejsou vhodné, protože se jedná o prvotní verze. Z hlediska obecného vývoje HTML5 aplikací jsou nevhodné z důvodu zaměřenosti na vývoj 3D her. Nevýhodou Unity je také nepodpora systému GNU/Linux.

Animatron, Mixeek, HTML5Maker

Tyto nástroje jsou v této analýze spojeny dohromady kvůli svým společným vlastnostem, není nutné se jimi zabývat zvlášť. Jsou to webové aplikace obsahující jednoduché rozhraní a jednoduchý způsob publikování. Jejich účel je pouze tvorba jednoduchých animací, z hlediska požadavků na aplikaci jsou proto nevhodné.

Shrnutí

Na poli existujících řešení existuje několik nástrojů, které by bylo možné použít, ovšem každý z nich trpí jedním nebo více nedostatky. U řady z nich je se možné inspirovat kvalitním uživatelským rozhraním.

Kapitola 2

Návrh grafického nástroje

V přecházejících sekcích byl zadán cíl aplikace a požadavky, které jsou na na ni kladeny, dále pak možnosti, které nabízí platforma webových aplikací. Tato sekce si klade za úkol **připravit návrh**, který bude odpovídajícím způsobem reagovat na zadané požadavky a zjištěné skutečnosti z teoretické části.

2.1 Analýza problému

Aby bylo možné odpovídajícím způsobem řešit problém chybějícího grafického nástroje pro HTML5, věnuje se tato sekce podrobnějšímu seznámení s řešeným problémem. V rámci průzkumu byly dotazovány dvě role v procesu vývoje HTML5 aplikací — „front-end vývojář“ a „grafický designér UI“. Toto jsou jejich pracovní postupy při vytváření grafického uživatelského rozhraní (GUI):

Grafický designér UI - postup návrhu uživatelského rozhraní

1. Vytvoření grafiky v grafickém editoru (např. Adobe Photoshop nebo Adobe Ilustrator)
2. Export jednotlivých spritů (např. ve formátu png) se slovním popisem
3. Programátor musí ručně napozicovat dané sprity a vytvořit animace

Neefektivita spočívá ve **znovuvytváření layoutu programátorem**, zároveň vzniká zpoždění způsobené čekáním na zapracování změn, což se stává kritické u verifikace výsledné aplikace. Grafický designér není schopen při tomto pracovním postupu interaktivně revidovat změny, které provedl.

Front-end vývojář - implmenetace UI ve vyvíjené aplikaci

1. K dispozici grafické prvky ve formě spritů (např. ve formátu png)
2. Otevření textového editoru
3. Ruční zadávání požadovaných hodnot do kódu
4. Refresh aplikace a kontrola výsledku
5. Přepnutí do editoru a korekce zadaných hodnot

6. Skok na krok 3

Všechny hodnoty jsou napsány přímo do kódu a je nutný následný refresh celé aplikace, aby bylo možné výsledek zobrazit a ověřit. Tento proces může být však poměrně **neefektivní** s přihlédnutím k následujícím detailům:

- Vývojář nemá přehled o výsledném celkovém rozložení
- Nepřesné pozicování na úrovni pixelů (nutný mnohonásobný refresh)
- Refresh rozsáhlejší aplikace trvá příliš dlouho
- Při refreshi je nutné vyvolat akci, kterou chceme kontrolovat

Příklad kódu:

```
this.priceCoin = this.getSprite("base/options-popup-button-coin.png");
this._priceCoin.x = 50;
this._priceCoin.y = 7;
this._priceGem = this.getSprite("base/options-popup-button-gem.png");
this._priceGem.x = 50;
this._priceGem.y = 9;
this._priceGem.rotation = 0.15;
this._priceGem.visible = false;
```

2.2 Analýza cílové skupiny

Cílovou skupinu projektu tvoří **vývojáří front-endu** webových aplikací tvořených pomocí tzv. 2D rendererů a herních enginů **vykreslovaných na Canvas**. Můžeme mluvit především o vývojářích interaktivních her umístitelných na web, pro desktop nebo mobilní zařízení či různé interaktivní prezentace, využívající vykreslování na Canvas.

Bakalářská práce se z důvodu výhod specializace na určitý druh aplikací primárně nezaměřuje na aplikace využívající SVG, HTML, DOM a stylování přes CSS.

2.3 Tvorba editoru

Řešením výše popsaného problému je zavést do výše popsaných postupů **grafický editor** umožnující návrh, pozicování a organizaci grafických prvků, který umožní automatický export kódu, který bude obstarávat vše potřebné k vykreslování zvolených grafických prvků na obrazovku za pomocí 2D rendereru.

Následný export se poté připojí do výsledné aplikace a vývojář doplní kód obstarávající funkčnost aplikace, např. připojí eventy, které umožní interaktivitu celé aplikace.

Nástroj je zamýšlen především pro střední až velké projekty, kde je nutné použít větší počet grafických objektů (spritů), tzn. je problematičtější a časově náročnější je organizovat.

Tento nástroj by měl být multiplatformní a umožňovat rozšiřitelnost pomocí modulů tak, aby umožnoval export do více 2D rendererů a herních frameworků a také, aby bylo možné aplikaci jednoduše rozšiřovat a přidávat novou funkcionalitu.

2.4 Typický případ užití

Následující postup je předpokládaný nejčastější případ užití:

- Import spritů (externí obrázkových souborů) do knihovny
- Práce s grafickými objekty
- Vkládání jednotlivých spritů na plochu
- Jejich pozicování a změna atributů
- Seskupování
- Export do vyvíjené aplikace

2.5 Požadované funkce

Je navrženo, aby obsahovala tyto části (více v 2.6):

- **Kreslící plátno** na které budou umisťovány grafické objekty
- **Knihovna** do které se načtou z disku sprity ve formátu png
- **Widgety** sloužící k manipulaci s objekty na platně.

Aplikace by měla zvládat tyto úkony:

1. Načítání seznamu spritů
V aplikaci je nutné specifikovat seznam spritů, se kterými se bude pracovat, ty budou nejčastěji ve formátu png, program by měl umět přidat např. celou složku nebo jednotlivé soubory.

2. Přetahování spritů na plochu z knihovny
Načtený sprite by měl jít vložit na plochu, kde se zobrazí jeho grafická podoba a bude dál upravovatelný, jeden sprite bude moct být vložen několikrát.

3. Změna atributů spritu
Cílem aplikace je především určovat atributy spritů, v programu by mělo být co nejjednodušeji možné měnit polohu spritu (např. přetahováním), rotaci atd., tyto atributy se samozřejmě musí okamžitě promítnout do grafického náhledu. Mělo by být možné zadávat atributy i číselně s okamžitou odezvou ve vizuální podobě.

4. Seskupování spritů do skupin
Podstatnou výhodou vykreslovacích knihoven nad Canvasem je to, že umožňují sdružovat sprity do skupin, je to tím pádem i základní funkcionalita aplikace. Je nutné vytvářet hierarchii grafických objektů a jednoduše navigovat mezi jednotlivými úrovněmi.

5. Relativní pozicování
Objekt bude možné nastavit jako relativní vůči jinému objektu. Tak bude možné realizovat např. zarovnání, po změně hodnot jednoho objektu (pozice x a y) se změní i hodnota objektu navázaného.

6. Zarovnání „středu“
Zarovnání „středu objektu“ (tzv. pivotu), od kterého se počítají souřadnice objektu k hraně nebo doprostřed. Po změně velikosti se obsah objektu posune tak, aby odpovídal zarovnání.

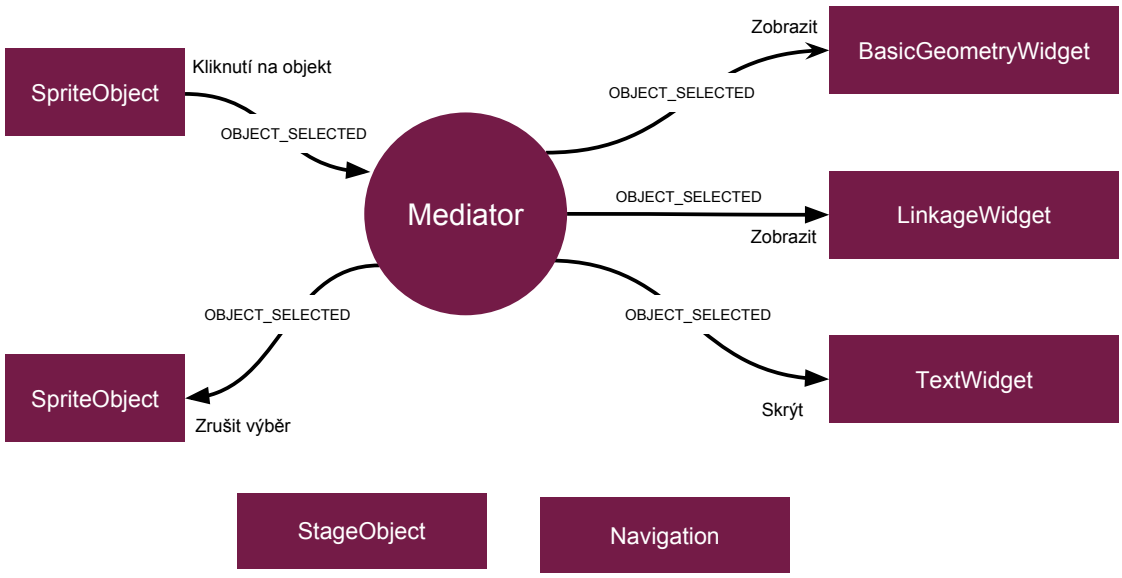
7. Připojení do kódu
Cílem aplikace je sice grafický návrh vyvíjené aplikace, je nutné ale propojit vizuální stránku věci s funkcionalitou, tudíž kódem.

Z těchto požadovaných funkcí vyplývá potřeba množiny několika základních typů objektů, ze kterých se bude skládat výsledná grafická podobna (viz 2.8):

- Sprite
- Skupina (Group)
- Text

2.6 Struktura aplikace

Aby bylo možné aplikaci jednoduše rozšiřovat, je hlavním požadavkem při návrhu architektury modularita, proto je navrženo použití návrhové vzoru **Publish/Subscribe** (viz 1.5.5).



Obrázek 2.1: Příklad komunikace v aplikaci pomocí Publish/Subscribe

Aplikace se bude skládat z jednotlivých komponent a služeb, které o sobě „nebudou vědět“, tím je snížená jejich provázanost (viz 1.5.4). Navzájem budou komunikovat pomocí událostí, které budou nastávat v systému, jsou navrhovány tyto základní události:

- `APPLICATION_INITIALIZED`

- `OBJECT_SELECTED`
- `OBJECT_DESELECTED`
- `BEFORE_SAVE`
- `BEFORE_OPEN`

Grafické komponenty využívají návrhového vzoru **Presentation Model**, odpovídající komponenta je tedy vždy rozdělena do dvou tříd, samotný PresentationModel a View.

2.6.1 Plátno

Platno reprezentuje třída **StageObject** a **StageObjectView**, z knihovny jsou zde přetahovány grafické objekty. Aby nemusela být provázána komponenta plátna a knihovny, je vytvořena univerzální služba typu DOMDraggableManager, která se stará o systém přetahování, je zavedeno rozhraní IDroppable, které implementují komponenty, na které je možné přetáhnout objekty:

- **wantsAcceptDraggable(object)** - Vrací, zda je daná komponenta ochotna akceptovat přetahovaný objekt
- **dropDraggable(object)** - V případě, že dojde k „upuštění“, je zavolána tato funkce, ve které je obsloužena tato akce

Komponenty, na které se má přetahovat (IDroppable), se registrují přes DOMDraggableManager.register, přetahované objekty se poté dotazují voláním DOMDraggableManager.check.

2.6.2 Knihovna

Komponenta, která načítá a obsahuje grafické objekty k přidání na plátno, reprezentují jej třídy **Library** a **LibraryView**. Tato komponenta se automaticky serializuje při ukládání (viz 2.8.2).

2.6.3 Navigace

Hlavní nabídku programu představuje služba typu **Navigation**. Nabídka je navržena jako služba, ke které nezávisle přistupují komponenty, které chtějí přidat do nabídky novou položku, která je definována popiskem a callbackem, který se vyvolá po stisknutí.

Díky tomu je možné efektivně snížit provázanost systému a pohodlně rozšiřovat program o nové funkce, které po připojení do programu autonomně aktualizují menu.

2.6.4 Widgety

Jedná se o grafické komponenty, které zpřístupňují určitou funkcionalitu k manipulaci s grafickým objektem na ploše. Widgety využívají události OBJECT_SELECTED a OBJECT_DESELECTED k tomu, aby aktivovali nebo deaktivovali svoji funkci.

Aby se zamezilo k provázání grafických objektů a widgetů, je pro každý typ funkcionality zavedeno rozhraní informující, zda daný objekt může být manipulován tímto způsobem. Na základě toho widget vyhodnotí, zda daný objekt podporuje a aktivuje nebo deaktivuje svoji funkci.

Tak je možné nezávisle přidávat nové widgety podporující jistý druh objektu a naopak nové typy objektů, které implementují určitou množinu rozhraní. Jsou navrhovaný tyto základní rozhraní:

- **IBasicGeometry** - vlastnosti x, y, rotation, scale
- **ILinkable** - vlastnosti name, linkage
- **IText** - vlastnosti font, text, size

Odpovídající widgety v základní verzi programu jsou **BasicGeometryWidget**, **LinkageWidget** a **TextWidget**.

Příkladem rozšíření může být zavedení nového druhu grafického objektu, např. prvku zobrazující grafy, který bude implementovat IBasicGeometry a ILinkable, a tím pádem automaticky využít možnosti existujících widgetů.

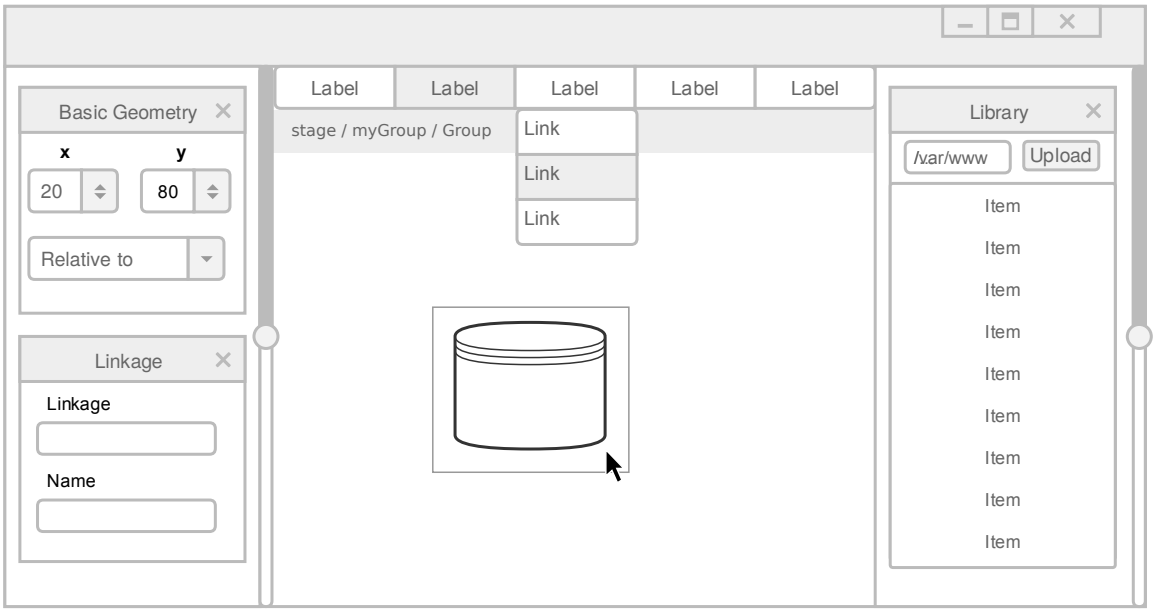
2.6.5 Drobečková navigace

Tato komponenta zobrazuje aktuální úroveň zanoření, aby bylo jasné, kde v hierarchii grafických objektů se uživatel momentálně nachází. Tato komponenta poslouchá na událost OBJECT_SELECTED a iterováním přes vlastnost parent se vyhodnotí cesta k aktuální úrovni.

2.7 Uživatelské rozhraní

Při návrhu uživatelského rozhraní bylo vycházeno z jednotlivých požadavků definovaných výše. Výsledný wireframe je možné vidět na obrázku 2.2.

Plátno je umístěno doprostřed uživatelského rozhraní, neboť se jedná o nejdůležitější komponentu aplikace a musí být tím pádem nejzřetelnější. **Drobečková navigace** se nachází nad plátnem, protože představuje informaci o aktuální poloze v hierarchii grafických objektů a tím pádem s ním úzce souvisí. Aby ovládácí prvky ve formě **widgetů** nepůsobily rušivě, jsou umístěny na stranách plátna.

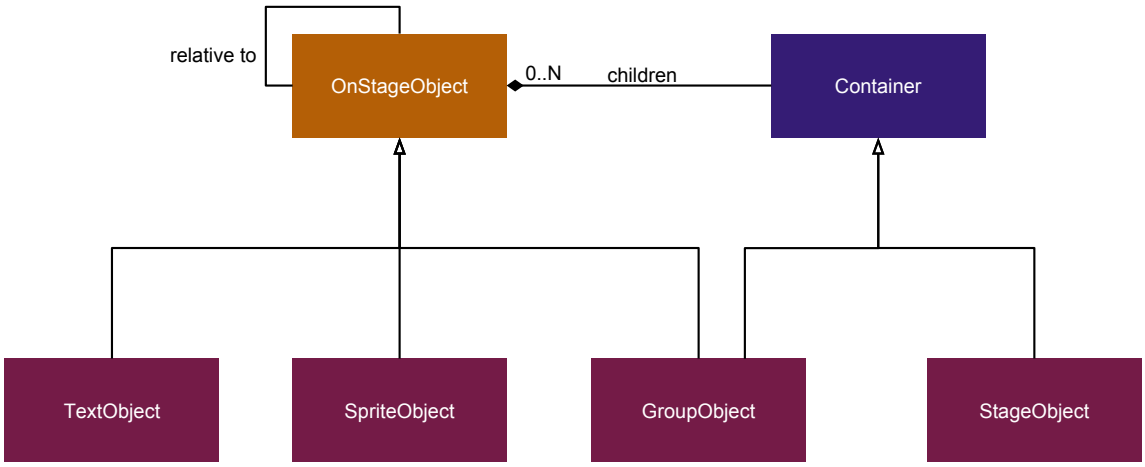


Obrázek 2.2: Wireframe uživatelského rozhraní

Každý widget představuje určitou logicky ohraničenou sadu funkcí, která je vizuálně sdružena, tím je dosaženo **zásady seskupování**. V rámci widgetu jsou rozmístěny ovládací prvky ve směru shora dolů a zleva doprava, čímž je docíleno **zásady následnosti**. Jednotlivé widgety se skrývají a zobrazují v závislosti na vybraném objektu na ploše, takže uživatel není rušen ovládacími prvky, které nepoužívá. Díky rozdělené do widgetů je celé pracovní prostředí přehlednější a zároveň je jej možné lépe rozšířit jednoduše přidáním dalšího widgetu.

2.8 Datový model

Datová vrstva aplikace bude reprezentována objektovým datovým modelem, který bude tvořit hiearchická struktura objektů představujících grafické prvky. Jsou reprezentovány jednotlivé grafické objekty umístěné na ploše, v základní verzi jsou to Text (TextObject), Skupina (GroupObject), Obrázek (SpriteObject) a Stage (StageObject), který představuje nejvyšší prvek v hierarchii. Každý typ grafického prvku umístitelného na plochu je reprezentován svou třídou dědící z OnStageObject. Instance **GroupObject** a **StageObject** mohou obsahovat další objekty, a tak tvořit hierarchii. Instance StageObject tvoří kořen stromu.



Obrázek 2.3: Třídní diagram

OnStageObject má tyto vlastnosti:

- **x a y** - pozice v rámci nadřazené položky
- **rotation** - otočení
- **relativeTo** - pozice je relativní k tomuto objektu (viz 2.5)
- **relativeAlign** - výběr hrany k zarovnání k relativeTo
- **pivotAlign** - zarovnání podle „středu“ (viz 2.5)
- **name** - identifikátor pro adresaci
- **linkage** - název třídy pro export
- **parent** - rodič

2.8.1 Perzistence (serializace)

Základním požadavkem je uchování aktuálního stavu rozvržení prvků na ploše. Proto je nutné uchovat stav aplikace v perzistentním uložišti (např. pevný disk). Nejpřímějším způsobem je serializace:

Díky tomu je možné automaticky převést objektový model na textovou reprezentaci. Nejlépe se k tomu hodí pro JavaScript typický JSON, který ovšem neuchovává informaci o typu objektu nebo referenci, dokáže pouze uchovávat základní typy jako je řetězec, číslo nebo pole [40].

Řešením je vytvořit speciální modul serializéru a deserializéru, který bude ukládat pomocí metadat i typ objektu a referenci (použitou např. v relativním pozicování), která bude realizována adresací založenou na primitivních datových typech.

2.8.2 Serializátor/Deserializátor

Serializátor musi kromě samotných primitivních datových typů do vyslédného datového souboru ve formátu JSON uchovávat i metadata.

Metadata musí uchovat jednak datový typ, v JavaScriptu jej reprezentuje konstruktor použitý při inicializaci a také informaci o referencích na jiné objekty. Členské proměnné obsahující reference musí být "doinicializovány"deserializátorem po vytvoření všech objektů. Dále jsou uvedeny navržené algoritmy pro serializaci a deserializaci:

Proces serializace:

1. Uložení informace o použitých konstruktorech
2. Předání objektu k serializaci
3. Zpracování objektu X
   (a) Existuje tento objekt v seznamu serializovanych objektů? Vrať ID, jinak pokračuj
   (b) Vytvoř nový serializovaný objekt a ulož jeho ID a konstuktor
   (c) Procházej členské proměnné objektu
      i. Narazíme-li na primitivní datový typ, přidej ho do serializovaného objektu
      ii. Narazíme-li na neprimitivní datový typ (objekt, pole), rekurzivně se zanoř do bodu 3 s argumentem aktuální zpracovávané proměnné a ulož do aktuálního objektu ID
   (d) Vrať ID z bodu 3.b
4. Ulož do souboru seznam serialiazovných objektů

Proces deserializace:

1. Načtení seznamu serializovaných objektů ze souboru
2. Uložení informace o použitých konstruktorech
3. Předání seznamu objektů k deserializaci
4. Zpracování objektu X
   (a) Existuje tento objekt v seznamu deserializovaných objektů? Vrať jej, jinak pokračuj
   (b) Vytvoř objekt pomocí definovaného konstuktoru a ulož do seznamu deserializovaných objektů
   (c) Přidej členské proměnné primitivních datových typů
   (d) Procházej neprimitvní členské proměnné
      i. Spusť rekurzivně bod 4 s parametrem členské proměnné a ulož do objektu vrácenou referenci
   (e) Vrať objekt

2.9 Export

Z aplikace je nutný export, aby bylo možné navržené grafické rozložení **použít v cílové aplikaci**. Tato sekce se věnuje způsobům, jak této funkce dosáhnout. V návrhu je uvažován export do aplikací založených na Pixi.js, ačkoliv díky modulární architektuře je možné připravit export i pro jiné knihovny nebo frameworky.

2.9.1 Adaptér

Uložení stavu aplikace do datového souboru může být totožný s procesem serializace (viz výše). Aby bylo možné realizovat funkci exportu do více různých knihoven a frameworků, je vhodné umístit mezi program a cílovou aplikaci **další vrstvu**, nazvanou adaptér, která bude zpřístupňovat data konkrétní knihovně.

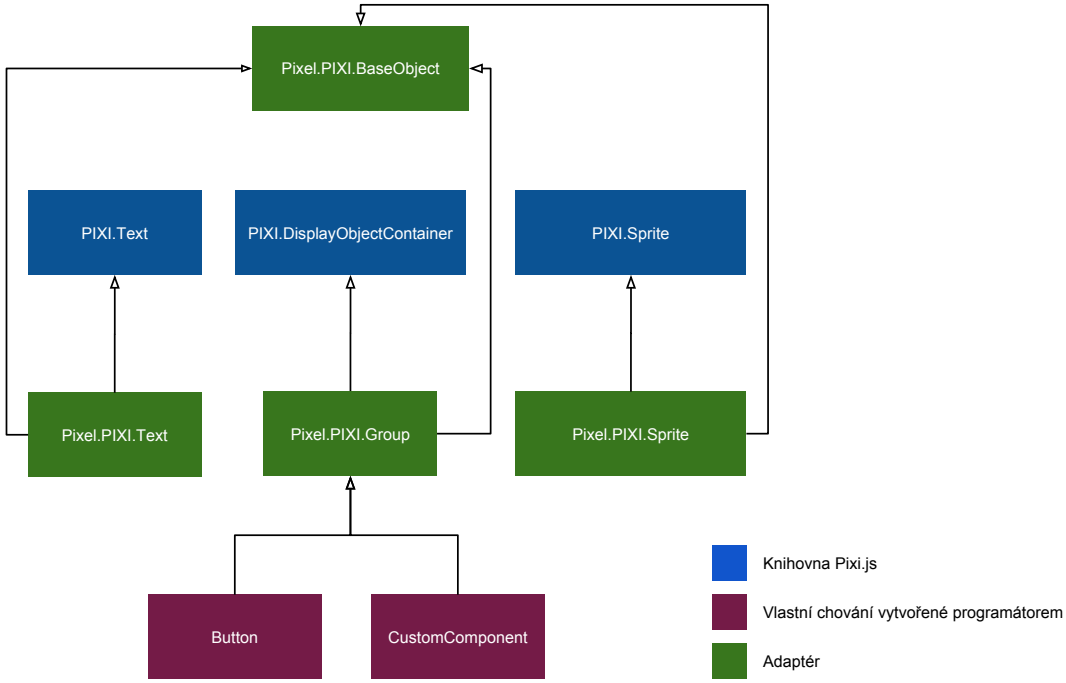
Adaptér obsahuje podpůrné třídy realizující funkce (nyní především zarovnávání) pro konkrétní knihovnu (viz 2.9.2) a deserializátor, který vytvoří ze vstupního datového souboru grafické objekty vykreslitelné v dané knihovně. Pro každou knihovnu nebo framework musí existovat zvláštní adaptér.

2.9.2 Reprezentace v cílovém programu

V cílové aplikaci vzniká potřeba reprezentace dat a funkčnosti (relativní pozicování, zarovnání) exportované z editoru a zároveň potřeba doplnění vlastní funkčnosti programátorem (např. chování tlačítka).

Byl zvolen přístup, kdy je funkčnost definovaná v editoru zajišťována třídou, která dědí z třídy grafického objektu knihovny, pro kterou je export určen. Zároveň je zde možnost pro programátora dále rozšířit tuto třídu o své vlastní chování. Právě parametr **linkage** popsaný výše vybírá tuto **uživatelem definovanou třídu**.

Je navrženo, aby v případě, že v cílovém programu není uživatelská třída definována, adaptér tuto třídu nadefinoval, aby bylo možné z jiných míst programu inicializovat tento grafický prvek, tzn. v editoru grafický prvek s parametrem linkage nastaveným např. na hodnotu „Button“ bude po exportu přístupný přes „new Button()“.



Obrázek 2.4: Hiearchie tříd v cílovém programu

2.9.3 Verzování datového souboru

Zdrojový soubor je automaticky serializovaný datový model aplikace, tudíž je náchylný na změny z důvodu vývoje nových verzí aplikace. Proto je vhodné verzovat schéma zdrojového souboru, navrhované je označení ve tvaru X.Y.Z (př. 1.2.14):

X - základní číslo verze (měnit se bude po úplném přepracování formátu)
Y - verze kompatibility (změní se pokud se provede zpětně nekompatibilní změna)
Z - revize (např. po přidání nových atributů, formát je zpětně kompatibilní)

Kapitola 3

Tvorba aplikace Pixel Studio

Tato kapitola popisuje konkrétní implementační detaily při tvorbě nástroje, který je cílem této bakalářské práce.

3.1 Platforma realizace a použité technologie

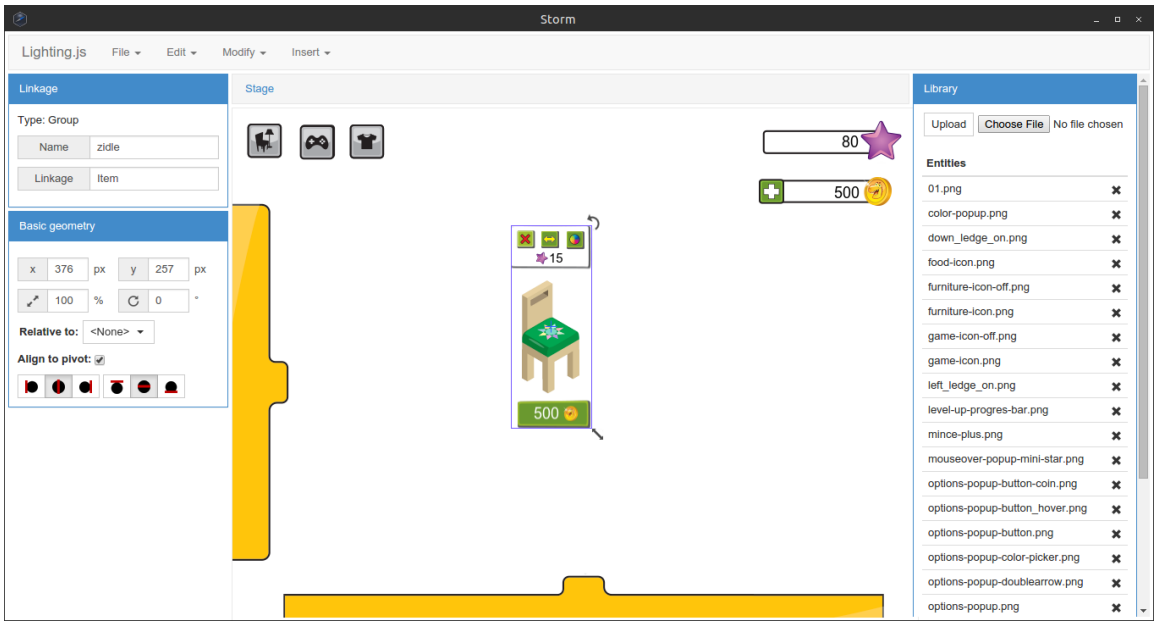
Pro implementaci byla zvolena platforma HTML5, tzn. použití technologií HTML, CSS a JavaScriptu. Tato volba byla proveda z těchto důvodů:

- **Multiplatformost** – HTML5 aplikace mohou běžet na jakémkoliv systému s odpovídajícím moderním webovým prohlížečem nebo ekvivalentním běhovým prostředím.
- **Cílení na stejnou platformu** – Cílem aplikace je podporovat vývoj pravě HTML5 aplikací za pomocí rendererů, které jsou napsány v JavaScriptu a používají Canvas z HTML5 pro vykreslování rastrové grafiky, tudíž samotná aplikace může pro vykreslování použít právě ty stejné 2D renderery, bez psání duplicitního mechanismu pro vykreslování náhledu v editoru.
- **Cílová skupina** – Protože cílovou skupinou jsou vývojáři HTML5 aplikací a sami budou mít zájem dále vyvíjet a rozšiřovat tento nástroj, tudíž mají předpoklady (znalosti) pro JavaScript a souvisejících technologie.
- **Pohodlný vývoj** – Kombinace vysokoúrovňového dynamického jazyku JavaScript a velmi flexibilní prezenční vrstva ve formě HTML dělají z této platformy kvalitní vývojový nástroj

Ačkoliv je zřejmé, že online webové aplikace mají řadu nesporných výhod, byla zvolena cesta aplikace spustitelné na desktopu, protože výhody online přístupu nejsou v této situaci příliš využitelné a naopak výhody desktopové varianty:

- **Práce se soubory** – Desktopová varianta může pracovat se soubory uloženými na disku, může tak bez problému ukládat zdrojové soubory, načítat obrázky atd.
- **Rychlost** – Ačkoliv je připojení k internetu stále rychlejší [39], desktopová aplikace bude vždy rychlejší bez nutnosti načítaní jakýchkoliv dat z internetu a s menší latencí bez čekání na odezvu ze vzdáleného serveru.

- **Offline běh** – Aplikace načítaní z pevného disku bude k dispozici vždy, zatímco vzdálený server může vypovědět službu (stejně jako výpadek internetu), následkem čehož vznikají ztráty ve vývojovém procesu.
- **Úplná kontrola** – Protože je aplikace, stejně tak jako např. soubory popisující rozložení grafických prvků, uložena na pevném disku vývojáře, má úplnou kontrolu nad těmito zdroji, viz souvislost s následujícím bodem.
- **Lepší forkování** – Aplikace spuštěná z pevného disku je fyzicky přítomná na počítači vývojáře, může tedy ihned začít upravovat zdrojový kód bez nutnosti složitějšího nastavování a instalace webového serveru a databázového softwaru.



Obrázek 3.1: Snímek obrazovky implementované aplikace, ve větším rozlišení je dostupný v příloze D

Aplikace byla realizována v programovacím jazyku **JavaScript**, pohledová vrstva byla implementována za pomoci značkovacího jazyka **HTML** a **CSS**. Jako běhové prostředí bylo použito **NW.js**.

Protože **plátno**, které zobrazuje rozložení v editoru vlastně odpovídá zobrazení v cílové aplikace, byla pro jeho realizaci kvůli zjednodušení použita stejná technologie, tedy **vykreslování na Canvas.** Byla vybrána knihovna **Pixi.js**, protože se jedná o čistě vykreslovací knihovnu, která neobsahuje žádné další (a pro účely tohoto nástroje nepotřebné) funkce.

Pro zbytek uživatelského rozhraní byla vybrána realizace skrze **HTML** a **CSS**, respektive pomocí CSS frameworku **Bootstrap**, který obsahuje kvalitně zpracované ovládací prvky, které lze v případě potřeby upravit nebo doplnit z jiného zdroje.

Další knihovny nebo frameworky nebyly použity, a to z důvodu udržení jednoduchého a srozumitelného návrhu aplikace, která by měla být jednoduše upravitelná uživatelem a případná nutnost učení se další knihovny nebo frameworku by tomu stavěla překážky.

V sekci 3.1 byla jako požadavek definována možnost spuštění na desktopu. Jako běhové prostředí bylo vybráno NW.js, a to z důvodu odladěnosti prostředí, snadného spouštění a propracované dokumentace.

3.2 Implementace chybějících jazykových konstrukcí

Protože implementace JavaScript dle verze EcmaScript 5, která je přítomna v současných moderních prohlížečích, neobsahuje jazykové konstrukce k **definici tříd**, bylo přistoupeno k realizace této syntaxe pomocí současných možností. Tato sekce vychází z teoretických znalostí získaných v sekci 1.3. Kód, který tuto funkčnost umožňuje je uveden v příloze B.

Definice třídy v této implementací má možnost určit **rodiče**, libovolný počet **mixinů** (jedná se o formu vícenásobné dědičnosti), definovat libovolné množství **metod** a dále tzv. **properitesObject**, který slouží k určení zvláštních vlastností proměnných, především **getterů a setterů**.

Příklad zápisu je následující:

```
var MyCustomClass = Class([ExtendedClass, Mixin1, Mixin2], {
  constructor: function(parametr1){

  },
  method1: function(){ ... },
  method2: function(){ ... },
  propertiesObject: {
    myVariable: {
      get: function(){ ... },
      set: function(){ ... }
    }
  }
}
```

Tato definice vytváří třídu MyCustomClass, která rozšiřuje rodičovskou třídu ExtendedClass a připojuje mixiny Mixin1 a Mixin2, dále je definován konstruktor, který se automaticky volá při instanciaci a metody této třídy. V propertiesObject je definována proměnná myVariable, ke které se přistupuje pomocí setteru a getteru.

Následně je možné pracovat s třídou následujícím způsobem:

```
var instance = new MyCustomClass(123);
instance.method1();
instance.myVariable = 155;
```

Tato implementace je vlastně „syntactic sugar“ nad prototypovou dědičností, funkce `Class` vezme funkci constructor ze vstupního parametru a přiřadí jí prototype třídy, kterou rozšiřuje a následně tento prototype rozšíří o metody (funkce) definované ve vstupním parametru, tak je docíleno toho, že jsou v instanciovaném objektu přítomny, jak metody rozšiřovanému objektu, tak metody konkrétní třídy.

Dále byla implementována jednoduchým způsobem možnost **rozhraní**. JavaScript patří mezi slabě typované jazyky, je zde využíván tzv. „duck typing“, z toho vyplývá, že jazyk neobsahuje explicitní definici rozhraní.

Přesto je vhodné v objektovém modelu tuto skutečnost nějakým způsobem reflektovat, proto byla implementována jednoduchá třída reprezentující interface. Příklad použití je následující:

```
1 var IBasicGeometry = new Interface(["x", "y", "rotation"]);
```

Nyní je možné se dotazovat, zda daný objekt implemetuje dané rozhraní pomocí volání `IBasiGeometry.isImplemented(object)`. Metoda je postavané tak, že projde jednotlivé atributy a otestujde, zda je zadaný objekt obsahuje, v případě, že nějaký chybí, zahlásí chybu.

3.3 Implementace systému událostí

Byla vytvořena třída **EventDispatcher**, která zaštiťuje chování objektu, který je schopen emitovat zprávy (události) a také umožňovat ostatním objektům, aby se přihlásily k odebírání určitému typu zprávy. EventDispatcher má tyto metody:

- `EventDispatcher.on(event, callback, context);`
- `EventDispatcher.off(event, callback, context);`
- `EventDispatcher.trigger(event, [parameter1, parameter2, ... ]);`

Pomocí metody **on** je možné se přihlásit k odebírání určitého druhu události, specifikovaném řetězcem **event**, pomocí **off** je možné jej odhlásit. Metoda **trigger** se obvykle volá z vnitřku objektu, tato metoda projde všechny zaregistrované callbacky a zavolají s předanými parametry.

Výměna zpráv přes Publish/Subscribe je realizována pomocí globálního objektu **mediator**, který je instancí právě EventDispatcher-u, skrze který si komponenty v aplikace nezávisle na sobě vyměňují globální zprávy (viz 2.6).

3.4 Realizace vzoru Presentation Model

Většina komponent v aplikace byly realizovány za pomoci tohoto návrhového vzoru. Například základní grafické prvky na ploše — SpriteObject, GroupObject a TextObject — mají svůj komplement v podobě SpriteObjectView, GroupObjectView a TextObjectView.

Základní třída reprezentuje **model**, který obsahuje logiku komponenty a data a **view** slouží ke grafickému zobrazení.

Problém synchronizace mezi modelem a view je řešen za pomocí getteru a setterů. Je zaveden pomocný princip implementovaný funkcí **utils.bindProperties(from, properties, to)**. Tato funkce dynamicky zavede gettery a settery, které způsobí, že pří přístupu k proměnné v objektu from je tento požadavek přesměrován do objektu to, příklad použití je možné vidět zde:

```
1 app.utils.bindProperties(this, ["visible", "relativeToVisible",
    "relativeTo", ... ], this.view);
```

3.5 Distribuce

Finální produkt je nutné distribuovat směrem k uživateli, HTM5 aplikace tvořící jádro aplikace je sice multiplatformní, je nutné ovšem distribuovat také spustitelné binární soubory běhového prostředí pro konkrétní platformu.

Byl otestován nástroj **node-webkit-builder** a s tím související rozšíření pro grunt **grunt-node-webkit-builder**. Tento nástroj spojí binárku běhového prostřední a samotné jádro aplikace do jednoho souboru, po vyzkoušení bylo ovšem zjištěno, že tato varianta není funkční pro všechny platformy.

Byla tedy vybrána varianta, kdy je aplikace zabalena do zipu, přípona přejmenována na .nw a umístěna do složky s binárkou běhového prostředí. Pro každou platformu je poté celá aplikace distribuována zvlášť.

3.6 Vyhodnocení

Aby bylo možné vyhodnotit správnost návrhu a plnění cíle bakalářské práce, tzn. tvorbu nástroje pro grafický návrh, který zvyšuje efektivitu vývoje HTML5 aplikací, byla navržena uživatelská zpětná vazba, která je rozdělena do dvou částí.

3.6.1 Zadání testu

Část 1

Uživateli je předložena hotová aplikace spolu se stručným popisem funkcí, je mu sděleno, aby vyzkoušel, zda pro něj má nástroj smysl, či odstraňuje nedostatek v jeho pracovním procesu.

Smyslem této první části je vyhodnocení uživatelem nezávisle na záměru vývojáře, otestování, k čemu a zda uživatel dokáže využít funkce obsažené v editoru, bez informace o jejich účelu, tak aby zpětná vazba dokázala lépe vyhodnotit přínos a smysl obsažených funkcí pro uživatele.

Scénář:

1. Zaslání archivu se spustitelným programem
2. Zaslání odkazu na odpovědní formulář obsahují i přehled funkcí
3. „Potřebuji, abyste otestoval/a tento nástroj k podpoře procesu vývoje HTML5 aplikací, vykoušejte, zda a jaký přínos vám může editor přinést, čas testování je libovolný, poté budete dotázán/a na 3 otázky v odpovědním formuláři.“
4. Uživatel spustí a testuje program
5. Uživatel odpovídá na 3 otázky v odpovědním formuláři

Otázky:

1. Co vnímáte jako přínos tohoto programu?
2. 3 postřehy, které vás k programu napadnou
3. Jaké jsou nedostatky, co chybí?

Jako doplněk k uživatelem vyplňovaným otázkám je vyhodnocení, zda uživatel v první části dokázal využít funkcí obsažených v editoru (tzn. pochopil jejich smysl a funkčnost), ke každé je mnou vyplněno, zda ano, ne nebo částečně. Jsou vyhodnocovaný tyto funkce:

1. Vkládání objektů na plochu
2. Manipulace se skupinami
3. Manipulace s objekty
4. Zarovnávání podle středu
5. Relativní pozicování („zavěšení“ na jiný objekt)

Část2

V druhé části je uživateli předvedeno použití funkcí, jejich zamýšlený účel a zamýšlený smysl aplikace, následně je uživatel vyzván k vyplnění dodatečných 3 otázek reflektující skutečnosti vycházející z této části.

Scénář:

1. V první části jste vyzkoušeli používání programu bez jakýchkoliv instrukcí, nyní vám předvedu zamýšlené použití funkcí a popíšu předpokládaný pracovní postup.
2. Uživateli je seznámen
3. Uživatel odpovídá na 3 otázky v odpovědním formuláři

Otázky:

1. Použil/a byste program ve svém workflow? Kde?
2. Proč byste (ne)doporučili program kolegovi?
3. 3 funkce, které by editor měl umět

3.6.2 Interpretace výsledků

Výsledky testů jsou k nahlédnutí v příloze A. V rámci průzkumu bylo zjišťováno především:

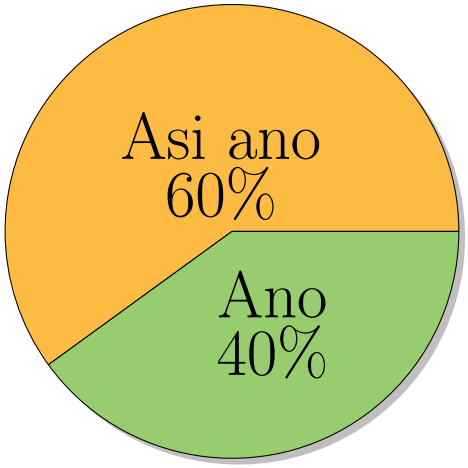
a) Zda tento nástroj řeší zadaný problém

b) Jak jsou uživateli dané funkce srozumitelné

c) Zpětná vazba v podobě chybějících funkcí a nápadů

Řeší nástroj zadaný problém?

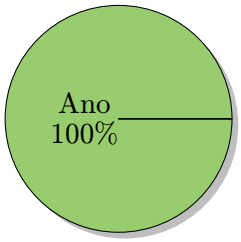
Vyhodnocení bylo provedeno podle otázky „Použil/a byste program ve svém workflow?“, zadané odpovědi byly klasifikovány jako „Ano“, „Asi ano“, „Ne“ a „Asi ne“.



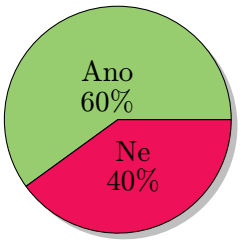
Obrázek 3.2: Odpovědi na otázku „Použil/a byste program ve svém workflow?“

Jak jsou dané funkce srozumitelné?

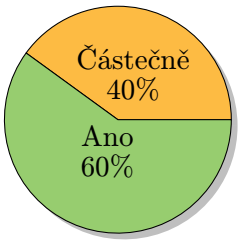
Bylo provedeno procentuální hodnocení podle výsledků využívání daných funkcí, které byly získány z provedeného testu.



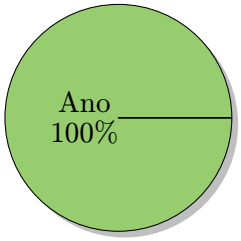
(a) Vkládání na plochu



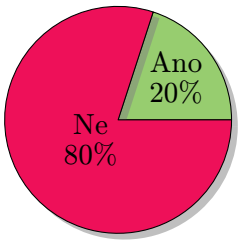
(b) Manipulace se skupinami



(c) Manipulace s objekty



(a) Zarovnání podle středu



(b) Relativní pozicování

Obrázek 3.4: Pictures of animals

Slovní interpretace

Ze získaných výsledků lze vyvozovat, že vyvinutý nástroj může řešit problém zadaný v úvodu této práce, většina jeho funkcí je uživateli srozumitelná a je možné jeho využití v pracovním postupu. Problematická byla především funkčnost „relativního pozicování“, která pro většinu uživatelů nebyla srozumitelná.

Přípomínky a nápady

V této části jsou shrnuty připomínky k chybám a nápady na novou funkcionalitu získané z odpovědí ve formuláři.

Nedostatky:

- Chybějící tooltipy a nápověda
- Obrázky (Sprite) se vkládají jinak než ostatní objekty (Skupina, Text)
- Nejasnost funkce „zavěšení“, problematické vybírání cílového objektu

Nápady na nové funkce:

- Editor kódu přimo v programu
- Toolbar
- Vkládání geometrických tvarů
- Animace

Závěr

Cílem bakalářské práce bylo vypracovat nástroj, který by umožnil zlepšení pracovního postupu při vývoji HTML5 aplikací vykreslovaných na Canvas.

Aplikace byla naprogramována v jazyce JavaScript, o zobrazení uživatelského rozhraní se postaralo HTML a CSS prostřednictvím frameworku Bootstrap. Jako běhové prostředí bylo použito NW.js, které umožňuje aplikaci pracovat se soubory a distribuovat nástroj jako desktopovou aplikaci.

Bylo ověřeno, že HTML5 aplikace spouštěné na desktopu, mezi které vyvíjený nástroj patří, mohou být plnohodnotnou alternativou k nativním aplikacím kompilovaným do strojového kódu a určeným pro konkrétní platformu, zároveň bylo vypozorováno, že u tohoto druhu aplikací je nutné věnovat zvláštní pozornost optimalizaci, neboť s sebou nese zvýšenou režii a při komplexnějších úlohách může docházet k výraznému zpomalení.

Výsledný produkt, hotová aplikace, je dostupná na přiloženém CD. Aplikace byla podrobena zpětné vazbě, která vyhodnocuje úspěšnost řešení zadaného problému. Z jejího výsledku je možné usuzovat, že nástroj může plnit svůj účel, ačkoliv se jedná o pilotní verzi, která neobsahuje mnoho funkcí. Ze zpětné vazby také vyplývá, že by bylo vhodné vylepšit intuitivnost některých funkcí (především relativní pozicování) a doplnit dokumentaci.

Aplikace byla navrhována s cílem snadné rozšiřitelnosti, aby bylo možné do aplikace přidávat další funkčnost, z nejdůležitějších je možné jmenovat podpora animací, práce s projektem nebo integrovaný editor kódu.

Literatura

[1] KNOWLES, Michael. How to Write a Competitive Analysis. *Michael Knowles Consulting* [online]. 2002 [cit. 11. 5. 2015]. Dostupné z: http://www.mwknowles.com/free_articles/companalysis/companalysis.html

[2] Features. *Google Web Designer* [online]. [cit. 11. 5. 2015]. Dostupné z: http://www.google.com/webdesigner/features/

[3] *Adobe Edge* [online]. [cit. 11. 5. 2015]. Dostupné z: https://creative.adobe.com/products/animate

[4] Cocos Studio. *Cocos2d-x* [online]. [cit. 11. 5. 2015]. Dostupné z: http://www.cocos2d-x.org/wiki/Cocos_Studio

[5] TRANI, Paul. Targeting HTML5 Canvas in Adobe Flash Professional CC. *Adobe Inspire Magazine* [online]. Únor 2014 [cit. 11. 5. 2015]. Dostupné z: http://www.adobe.com/inspire/2014/02/flash-html5-canvas.html

[6] ECHTERHOFF, Jonas. On the Future of Web Publishing in Unity. *Unity Blog* [online]. 29. 4. 2014 [cit. 11. 5. 2015]. Dostupné z: http://blogs.unity3d.com/2014/04/29/on-the-future-of-web-publishing-in-unity/

[7] GamePix. Unreal Engine 4 Gets Native HTML5 Exporting. *HTML5 Games Blog* [online]. 25. 2. 2015 [cit. 11. 5. 2015]. Dostupné z: http://www.gamepix.com/blog/unreal-engine-4-gets-native-html5-exporting/web-publishing-in-unity/

[8] HICKSON, Ian a HYATT David. HTML 5: A vocabulary and associated APIs for HTML and XHTML. *World Wide Web Consorcium* [online]. Poslední změna 25. 8. 2009 [cit. 8. 5. 2015]. Dostupné z: http://www.w3.org/TR/2009/WD-html5-20090825/the-canvas-element.html

[9] LAIRD, Cameron. Unleash the Power of Hardware-Accelerated HTML5 Canvas. *Microsoft Developer Network* [online]. [cit. 8. 5. 2015]. Dostupné z: https://msdn.microsoft.com/en-us/hh562071.aspx

[10] LAIRD, Cameron. Unleash the Power of Hardware-Accelerated HTML5 Canvas. *Microsoft Developer Network* [online]. [cit. 8. 5. 2015]. Dostupné z: https://msdn.microsoft.com/en-us/hh562071.aspx

[11] HEIKINNEN, Ilmari.Taking advantage of GPU acceleration in the 2D canvas. *HTML5 Rocks* [online]. 5. 7. 2012 [cit. 11. 5. 2015]. Dostupné z: http://updates.html5rocks.com/2012/07/Taking-advantage-of-GPU-acceleration-in-the-2D-canvas

[12] Canvas drawImage vs. WebGL drawArrays. *jsPerf* [online]. [cit. 8. 5. 2015]. Dostupné z: http://jsperf.com/canvas-drawimage-vs-webgl-drawarrays

[13] Details of the object model. *Mozilla Developer Network* [online]. Poslední změna 26 .4. 2015 6:07 [cit. 8. 5. 2015]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

[14] FLANAGAN, David. JavaScript: kompletní průvodce. 2. aktualiz. vyd. Praha: Computer Press, 2002, xiv, 825 s. Všechny cesty k informacím. ISBN 80-7226-626-8.

[15] Prototype vs. this. *jsPerf* [online]. [cit. 8. 5. 2015]. Dostupné z: https://jsperf.com/prototype-vs-this/8

[16] Classes. *Mozilla Developer Network* [online]. Poslední změna 28 .4. 2015 7:14 [cit. 8. 5. 2015]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

[17] Garbage collection. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-08]. Dostupné z: http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29

[18] SALES DE, Stevens. 25 Techniques for Javascript Performance. *desalasworks* [online]. [cit. 8. 5. 2015]. Dostupné z: http://desalasworks.com/article/javascript-performance-techniques/

[19] Design Elements. *Google Developers* [online]. Poslední změna 17.9. 2012 [cit. 8. 5. 2015]. Dostupné z: https://developers.google.com/v8/design

[20] OSMANI, Addy. Writing Fast, Memory-Efficient JavaScript. In: *Smashing Magazine* [online]. 5. 11. 2012 [cit. 8. 5. 2015]. Dostupné z: http://www.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript/

[21] LIPPERT, Erick. What are closures?. In: *Erick Lippert's Blog* [online]. 17. 11. 2003 [cit. 8. 5. 2015]. Dostupné z: http://blogs.msdn.com/b/ericlippert/archive/2003/09/17/53028.aspx

[22] Optimizing JavaScript code. *Google Developers* [online]. Poslední změna 17.9. 2012 [cit. 8. 5. 2015]. Dostupné z: https://developers.google.com/speed/articles/optimizing-javascript

[23] PECINOVSKÝ, Rudolf. Návrhové vzory: [33 vzorových postupů pro objektové programování]. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-80-251-1582-4.

[24] IEEE 1471-2000. *Úvod do architektury MVC. In* [online]. Dostupné z: http://standards.ieee.org/findstds/standard/1471-2000.html

[25] BERNARD, Borek. Úvod do architektury MVC. In: *Zdroják* [online]. 7. 5. 2009 [cit. 8. 5. 2015]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[26] RegisFrey. The model, view, and controller (MVC) pattern relative to the user. *Wikimedia commons* [online]. 1. 5. 2010 [cit. 12. 5. 2015]. Dostupné z: http://commons.wikimedia.org/wiki/File:MVC-Process.svg

[27] FOWLER, Marting. Retirement note for Model View Presenter Pattern. *Martin Fowler* [online]. 11. 6. 2006 [cit. 8. 5. 2015]. Dostupné z: http://martinfowler.com/eaaDev/ModelViewPresenter.html

[28] Diagram that depicts the Model View Presenter (MVP) GUI design pattern. *Wikimedia commons* [online]. 27. 7. 2006 [cit. 11. 5. 2015]. Dostupné z: http://commons.wikimedia.org/wiki/File:Model_View_Presenter_GUI_Design_Pattern.png

[29] FOWLER, Marting. Presentation Model. *Martin Fowler* [online]. [cit. 8. 5. 2015]. Dostupné z: http://martinfowler.com/eaaDev/PresentationModel.html

[30] DAJBYCH, Václav. MVVM: Model-View-Viewmodel. In: *DOT NET PORTÁL*. 21. 4. 2009 [online]. [cit. 8. 5. 2015]. Dostupné z: http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[31] Publish/Subscribe. *Microsoft Developer Network*. [cit. 12. 5. 2015]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff649664.aspx

[32] FREEMAN, Eric, Elisabeth FREEMAN, Kathy SIERRA a Bert BATES. Head first design patterns. Sebastopol, CA: O'Reilly, c2004, xxxvi, 638 p. ISBN 0596007124.

[33] Observables. *Knockout Documentation*. [cit. 8. 5. 2015]. Dostupné z: http://knockoutjs.com/documentation/observables.html

[34] BERRY, Clint. HTML5 Apps on Desktop in 2013. *Blog — Client Berry* [online]. 11. 5. 2013 [cit. 8. 5. 2015]. Dostupné z: http://clintberry.com/2013/html5-apps-desktop-2013/

[35] DOSTÁL, Martin. *Základy tvorby uživatelského rozhraní* [online]. Olomouc: Univerzita Palackého, 2007. Dostupné z: http://phoenix.inf.upol.cz/esf/ucebni/gui-dostal.pdf

[36] SMITH, Grace. 20 Exceptional CSS Boilerplates and Frameworks. In: *Mashable* [online]. 26. 4. 2013 [cit. 8. 5. 2015]. Dostupné z: http://mashable.com/2013/04/26/css-boilerplates-frameworks

[37] SMITH, Grace. 20 Exceptional CSS Boilerplates and Frameworks. In: *Mashable* [online]. 26. 4. 2013 [cit. 8. 5. 2015]. Dostupné z: http://mashable.com/2013/04/26/css-boilerplates-frameworks

[38] NARAYANASWAMY, Anand. Webix 1.3 Adds New Skins, HTML5 Video Element and Updated Charts Widget. In: *InfoQ* [online]. 14. 12. 2013 [cit. 8. 5. 2015]. Dostupné z: http://www.infoq.com/news/2013/12/webix-1-3

[39] SOUKUP, Tomáš. Patříme k zemím s nejrychlejším průměrným připojením k internetu. In: *Živě.cz* [online]. 26. 8. 2013 [cit. 8. 5. 2015]. Dostupné z: http://www.zive.cz/bleskovky/patrime-k-zemim-s-nejrychlejsim-prumernym-pripojenim-k-internetu/sc-4-a-170273/default.aspx

[40] ECMA-404. *The JSON Data Interchange Format* [online]. Říjen 2013 [cit. 11. 5. 2015]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[41] MICHÁLEK, Martin. K čemu je dobrý Bootstrap a frontend frameworky?. In: *Zdroják* [online]. 6. 12. 2013 [cit. 11. 5. 2015]. Dostupné z: http://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/

[42] PHUONG, Phan. 10+ Bootstrap components from third party developers. In: *YoArts* [online]. 30. 11. 2013 [cit. 11. 5. 2015]. Dostupné z: http://www.yoarts.com/third-party-bootstrap-components/

[43] *Bootstrap* [online]. 30. 11. 2013 [cit. 11. 5. 2015]. Dostupné z: http://getbootstrap.com/

Příloha A

Výsledky zpětné vazby

A.1 Test 1 (Front-end vývojář)

Co vnímáte jako přínos tohoto programu?	Zjednodusenie vytvarania UI pre rozne druhy JS aplikacii.
3 postřehy, které vás k programu napadnou	Jednoduche UI.
Jaké jsou nedostatky, co chybí?	Napoveda, tooltipy, moznost dat objekt na top/bottom/up/down.

Použil/a byste program ve svém workflow? Kde?	Ak by som robil nejake UI pre JS hru, tak ano.
Proč byste (ne)doporučili program kolegovi?	Neviem.
3 funkce, které by editor měl umět	Mat okno s layermi/objektami (ako photoshop).

Vkládání objektů na plochu	Ano
Manipulace se skupinami	Ne
Manipulace s objekty	Částečně
Zarovnávání podle středu	Ano
Relativní pozicování	Ne

A.2 Test 2 (Front-end vývojář)

Co vnímáte jako přínos tohoto programu?	Po 10 minutách zkoušení, jsem nebyl schopen vyzkoušet všechny funkce tohoto programu. Tato aplikace může sloužit jako základ pro programy zaměřené na tvorbu uživatelského prostředí.
3 postřehy, které vás k programu napadnou	- v aplikaci chybí toolbar, volby jsou dostupné pouze z menu - bylo by vhodné přidávání geometrických tvarů - "vzdušné"uživatelské prostředí
Jaké jsou nedostatky, co chybí?	Nedostatky korenspodují s některými postřehy

---

Použil/a byste program ve svém workflow? Kde?	Pokud bych měl řešit situaci ve které bych potřeboval urgentně navrhnout uživatelské rozhraní, pravděpdobně by mě napadla tato aplikace, jelikož je jediná toho typu, kterou jsem doposavad vyzkoušel. Ovšem před zařazní do workflow bych si udělal analýzu dostupných aplikací a následně zhodnotil výhody a nevýhody.
Proč byste (ne)doporučili program kolegovi?	Dle infromací od autora tento program zatím není zařazen do praxe, tudíž může potencionálě osahovat chyby. Toto ovšem nevylučuje, že se časem může jednat o velice kvalitní program, který bych doporučil kolegovi.
3 funkce, které by editor měl umět	Nevím, bylo by třeba použití v praxi.

Vkládání objektů na plochu	Ano
Manipulace se skupinami	Ne
Manipulace s objekty	Ano
Zarovnávání podle středu	Ano
Relativní pozicování	Ano

A.3 Test 3 (Grafik)

Co vnímáte jako přínos tohoto programu?	Developer si moze rychlo nasadit grafiku a rovno to vizuálne vidiet. v programe su bezne tooly - otočenie, zarovnanie atd.
3 postřehy, které vás k programu napadnou	Nevím
Jaké jsou nedostatky, co chybí?	Asi nic.

Použil/a byste program ve svém workflow? Kde?	Asi áno, urychlilo by to pracu medzi grafikom a rogramatorom.
Proč byste (ne)doporučili program kolegovi?	Doporučila, z dôvodov výše.
3 funkce, které by editor měl umět	V podstate na to comu sluzi myslim ze má vsetko potrebne co ma mat, najma dolezite tooly na pozicovanie jednotlivych itemov, ich otáčanie, posuvanie vrstiev hore a dole, zvacsovanie a zmensovanie, praca z textom atd.

Vkládání objektů na plochu	Ano
Manipulace se skupinami	Ne
Manipulace s objekty	Ano
Zarovnávání podle středu	Ano
Relativní pozicování	Ne

A.4 Test 4 (Front end vývojář)

Co vnímáte jako přínos tohoto programu?	Efektivní tvorba a správa spritů. Člověk si to hezky nakliká..
3 postřehy, které vás k programu napadnou	- Funkci zavěšení jsem nepochopila. Člověk čeká, že se na ploše "něco pohne"stejně tak jako to dělá zarovnání podle pivota. Takže buď bych navhla nějakou interakci na ploše, jako je to u všech ostatních nástrojů popřípadě alespoň dobrá dokumentace. Velký podíl má na tomhle nepochopení i to, že člověku se v tomto dotazníku nejdřív představí zarovnání a až potom zavěšení, které zdaleka není po UX stránce tak odladěné. - Vložení obrázku z knihovy jsem nejdříve hledala v horním menu, kde jsem pod záložkou Insert čekala vedle Group a Text něco jako Image. - Nepochopila jsem jak funguje provázání se třídou, to by asi taky chtělo přívětivěji. - Chybí mi tu možnost pojmenovat objekty, vůči kterým se zavěšuje. Takhle je tam pod sebou 20x Text a nevím, který je který.
Jaké jsou nedostatky, co chybí?	Viz předchozí odpověď.

Použil/a byste program ve svém workflow? Kde?	Rozhodně ano. Zatím jsem nenarazila na lepší program s touto funkcionalitou. Na vzájemé pozicovaní je dokonalý.
Proč byste (ne)doporučili program kolegovi?	- Doporučila bych ho, protože tvorba spritů je o mnoho rychlejší a jednodušší. - Rozhodně bych ale kolegovi nejdříve představila všechny funkce programu, protože za daného stavu bych si nemohla být jistá, že odhalí a bude umět použit všechny funkce, což by byla škoda.
3 funkce, které by editor měl umět	- Do budoucna by měl mít určitě interní konzoli, aby si člověk nemusel upravovat třídy v externím editoru. - Měl by být uživatelsky přívětivější.

Vkládání objektů na plochu	Ano
Manipulace se skupinami	Ano
Manipulace s objekty	Ano
Zarovnávání podle středu	Ano
Relativní pozicování	Ne

A.5 Test 5 (Front end vývojář)

Co vnímáte jako přínos tohoto programu?	Umožní webovým vývojářům efektivněji a jednodušeji vývijet webové aplikace a hry.
3 postřehy, které vás k programu napadnou	Má přehledné uživatelské rozhraní, velmi snadno se ovládá.
Jaké jsou nedostatky, co chybí?	Při přetahování snippetu se neukazuje jeho náhled.

Použil/a byste program ve svém workflow? Kde?	Vývoj programu sleduji, jakmile bude produkční verze, začnu jej testovat a možná nasadím do produkce.
Proč byste (ne)doporučili program kolegovi?	Doporučil bych mu to, kvůli funkcím obsaženým v programu.
3 funkce, které by editor měl umět	Bylo by dobré doplnit i nadstandardní funkce jako podporu relativního pozicování ke škálování, animací a vestavěného editoru kódu, i když to nejsou zásadní problémy a už se stávající verzí se dají obejít (místo vestavěného editoru můžu například pořád exportovat a importovat).

Vkládání objektů na plochu	Ano
Manipulace se skupinami	Ano
Manipulace s objekty	Částečně
Zarovnávání podle středu	Ano
Relativní pozicování	Ne

Příloha B

Implementace tříd a rozhraní v JavaScripti

```
1  var Class = function(param1, param2) {
2      var extend, mixins, definition;
3      if(param2){     //two parameters passed, first is extends, second
           definition object
4          extend = Array.isArray(param1)?param1[0]:param1;
5          mixins = Array.isArray(param1)?param1.slice(1):null;
6          definition = param2;
7      }else{      //only one parameter passed => no extend, only
           definition
8          extend = null;
9          definition = param1;
10     }
11
12
13     var Definition = definition.hasOwnProperty("constructor")?definition
           .constructor:function(){ };
14
15     Definition.prototype = Object.create(extend?extend.prototype:null);
16     var propertiesObject = definition.propertiesObject?definition.
           propertiesObject:{};
17     if(mixins){
18         var i,i2;
19         for(i in mixins){
20             for(i2 in mixins[i].prototype){
21                 Definition.prototype[i2] = mixins[i].prototype[i2];
22             }
23             for(var i2 in mixins[i].prototype.propertiesObject){
24                 propertiesObject[i2] = mixins[i].prototype.
                       propertiesObject[i2];
25             }
26         }
27     }
28
29     Definition.prototype.propertiesObject = propertiesObject;
30
31     Object.defineProperties(Definition.prototype, propertiesObject);
32
33     for(var key in definition){
```

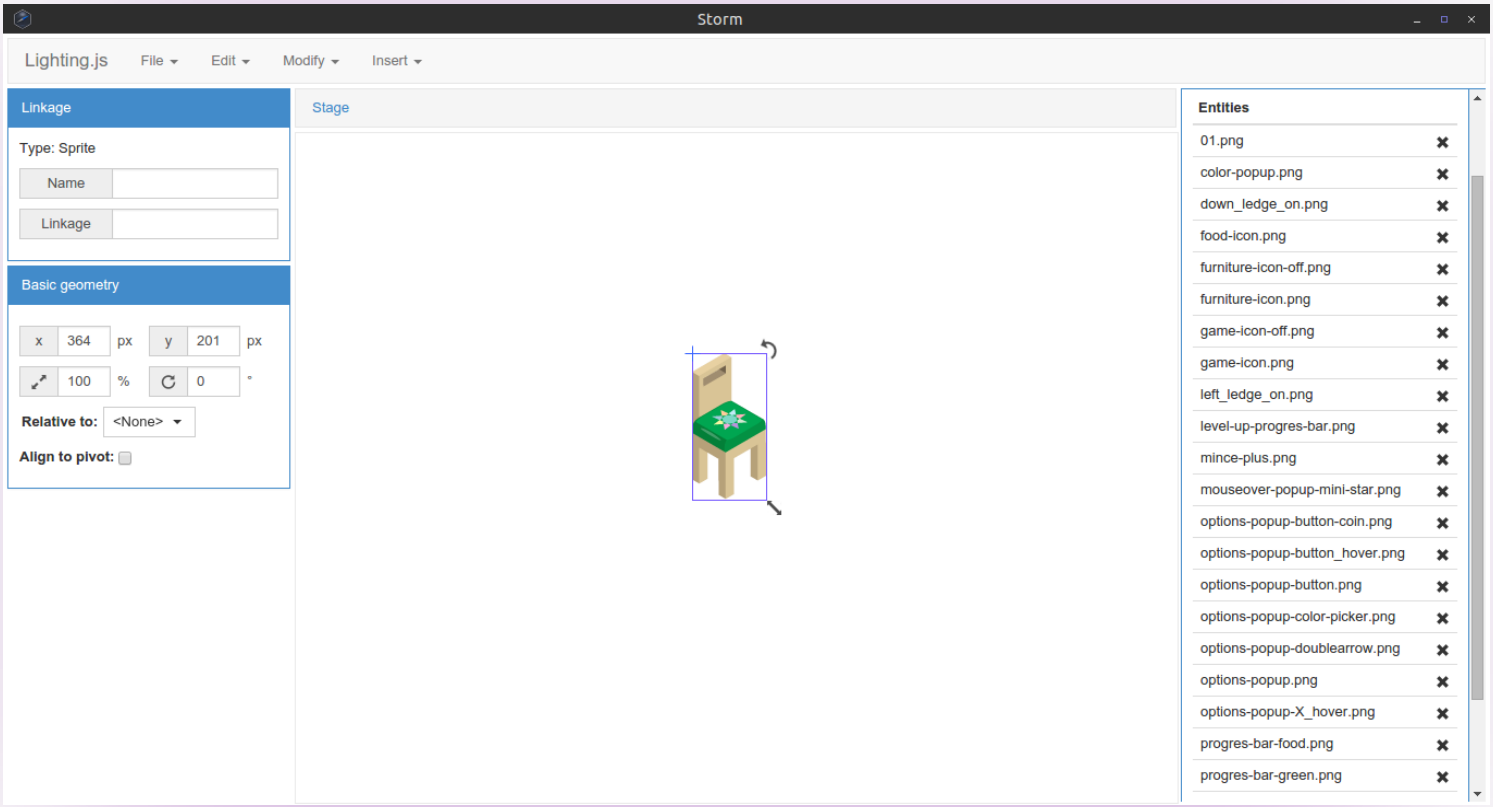
```
34          if(definition.hasOwnProperty(key)){
35              Definition.prototype[key] = definition[key];
36          }
37      }
38
39      Definition.prototype.constructor = Definition;
40
41      return Definition ;
42  }
43
44
45  var Interface = function(properties){
46      this.properties = properties;
47  }
48
49  var InterfaceException = function(message){
50      this.name = "InterfaceException";
51      this.message = message || "";
52  }
53
54  InterfaceException.prototype = new Error();
55
56  Interface.prototype.implements = function(target){
57      for(var i in this.properties){
58          if(target[this.properties[i]] == undefined){
59              throw new InterfaceException("Missing property " + this.
                    properties[i] );
60          }
61      }
62      return true;
63  }
64
65  Interface.prototype.doesImplement = function(target){
66    for(var i in this.properties){
67          if(target[this.properties[i]] === undefined){
68              return false;
69          }
70      }
71      return true;
72  }
```

Příloha C

Plakát

PIXEL EDITOR

Nástroj pro grafický návrh HTML5 aplikací



Import spritů

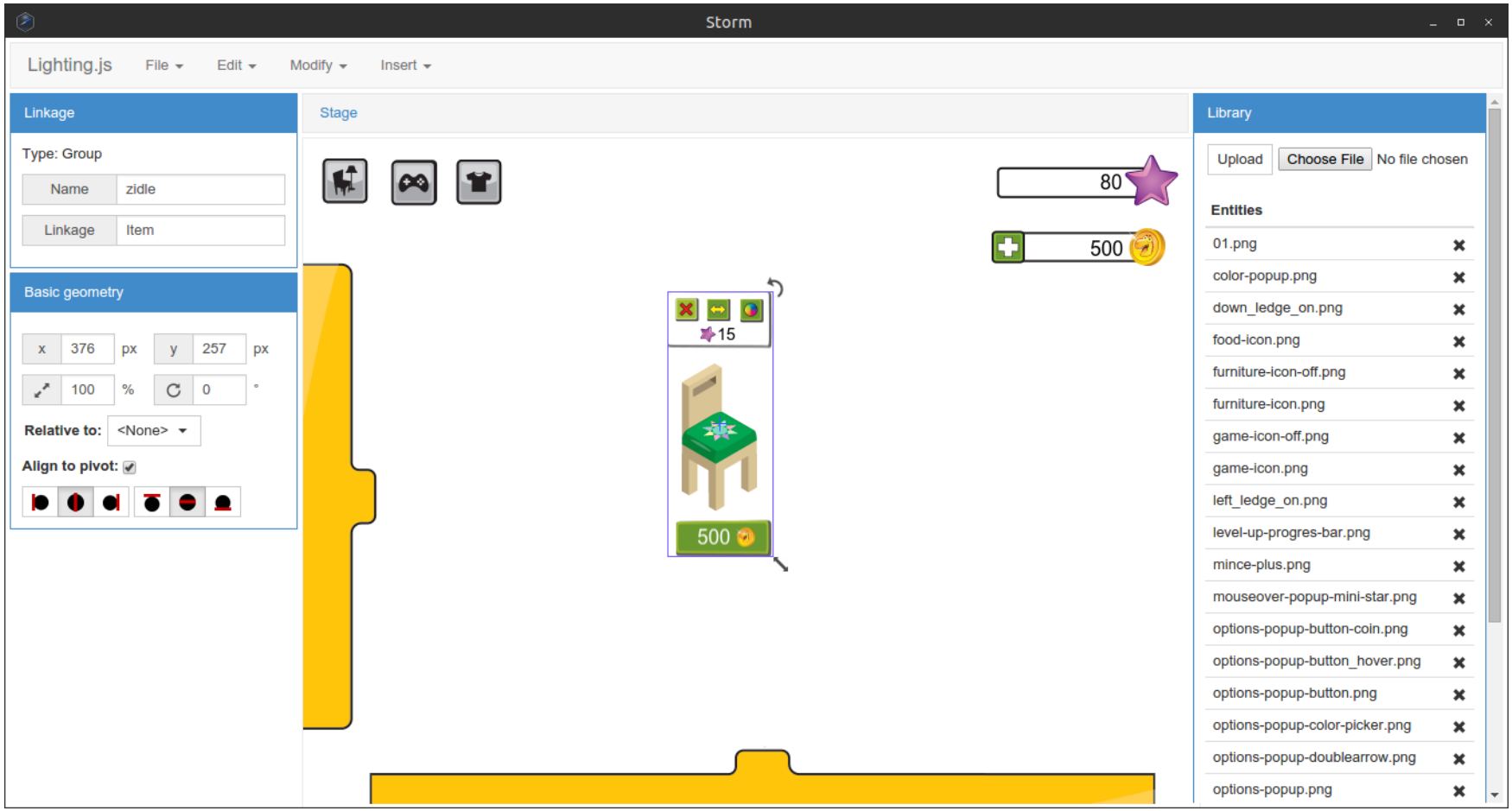
Pozicování, otáčení, škálování

Modulární architektura

Pokročilé funkce zarovnávání

Příloha D

Snímek obrazovky


Storm
Lighting.js
File
Edit
Modify
Insert
Linkage
Type: Group
Name
zidle
Linkage
Item
Basic geometry
x
376
px
y
257
px
100
%
0
°
Relative to:
<None>
Align to pivot:
Stage
80
500
15
500
Library
Upload
Choose File
No file chosen
Entities
01.png
color-popup.png
down_ledge_on.png
food-icon.png
furniture-icon-off.png
furniture-icon.png
game-icon-off.png
game-icon.png
left_ledge_on.png
level-up-progres-bar.png
mince-plus.png
mouseover-popup-mini-star.png
options-popup-button-coin.png
options-popup-button_hover.png
options-popup-button.png
options-popup-color-picker.png
options-popup-doublearrow.png
options-popup.png

Příloha E

Obsah CD

Z důvodu omezené kapacity CD není přiloženo běhové prostředí pro systém Mac OS X, to lze ale stáhnout z webu http://nwjs.io. Program je možné spustit pomocí spustitelného souboru **nw** pro GNU/Linux nebo **nw.exe** pro Microsoft Windows.