Mendelova univerzita v Brně
Provozně ekonomická fakulta

Tvorba arkádové hry v prostředí webového prohlížeče

Bakalářská práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Gustav Chládek

Brno 2015

Stránka se zadáním práce

Děkuji Ing. Jiřímu Lýskovi, Ph.D., za vedení práce a pomoc při její tvorbě.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Tvorba arkádové hry v prostředí webového prohlížeče** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 21. května 2015 _____

Abstract

Chládek, G. Developing arcade game in the web browser environment. Bachelor thesis. Brno: Mendel University, 2015.
This thesis focuses on the development of arcade game which runs in the web browser environment. It focuses on the usage of the JavaScript language in parallel with HTML and CSS. These technologies are used to develop a sample arcade game for the web browser.

Keywords

Arcade games, JavaScript, web browser, game development

Abstrakt

Chládek, G. Tvorba arkádové hry v prostředí webového prohlížeče. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2015.
Tato práce se zabývá tvorbou arkádové hry v prostředí webového prohlížeče. Zaměřuje se na využití jazyka JavaScript a s ním souběžně používaných technologií HTML a CSS. Tyto technologie jsou následně použity pro vytvoření ukázkové arkádové hry pro webové prohlížeče.

Klíčová slova

Arkádové hry, JavaScript, webový prohlížeč, vývoj hry

Obsah

1 Úvod a cíl práce

1.1 Úvod

V dnešní době rozvoje moderních technologií se mění mnoho aspektů lidského života. Zábava potažmo hraní her je jednou z nich. Arkádové hry při svém vzniku byly často dostupné jen jako statické herní terminály. Dnes díky stále se zvyšující podpoře nových webových technologií na mobilních platformách jsou právě webové aplikace nejjednodušším způsobem distribuce na rozdílných mobilních i klasických elektronických zařízení různých výrobců.

Arkádové hry, které kladou důraz na jednoduchost ovládání a jednoduchost konceptu samotné hry, jsou proto vhodným žánrem právě i pro mobilní platformy.

1.2 Cíl

Cílem práce je seznámit čtenáře s historií i vývojem her jako takových. Definovat pojmy jako počítačová hra a arkádová hra. Porovnat jazyk JavaScript s podobně zaměřenými programovacími jazyky. Popsat technologie spojené s tvorbou webové aplikace jako je JavaScript, CSS a HTML. Jejich historii i současný stav a za jejich pomoci vytvořit konkrétní arkádovou hru. Objasnit tak postup implementace a možná úskalí tvorby webových aplikací.

2 Teoretická část

2.1 Hra

Definice pojmu hra se liší podle oboru autora definice.

Jak konstatují Průcha a kol. (2013, s. 75) je hra forma činnosti, která má řadu aspektů: aspekt poznávací, procvičovací, emocionální, pohybový, motivační, tvořivostní, fantazijní, sociální, rekreační, diagnostický, terapeutický. Zahrnuje činnosti jednotlivce, dvojice, malé skupiny i velké skupiny. Podle uvedených autorů existují hry, k jejichž provozování jsou nutné speciální pomůcky (hračky, herní pomůcky, sportovní náčiní, nástroje, přístroje). Většina her má podobu sociální interakce s jasně formulovanými pravidly (danými dohodou aktérů nebo společenskými konvencemi). Ve hře se mnoho pozornosti věnuje jejímu průběhu (hry s převahou spolupráce, s převahou soutěžení).

Naproti tomu Salen a Zimmerman (2003, s. 80) definují hru jako systém, ve kterém hráči soupeří v umělém souboji, definovaném pravidly, který vyprodukuje měřitelný výsledek. Odlišnou definici nabízí Suits (2005, s. 55), který definuje hraní hry jako dobrovolný pokus překonat jinak nadbytečné překážky.

2.2 Počítačová hra

Obecně je počítačová hra druh softwaru, který pomocí lidské interakce s uživatelským rozhraním, vytváří vizuální případně i zvukovou zpětnou vazbu. Podle Carra a kol. (2006, s. 4) se můžeme v anglicky mluvících zemích kromě pojmu computer games (počítačové hry), který se používá pro označení her na platformě osobních počítačů, setkat i s pojmem video games (videohry), častěji používaném pro hry hrané na herních konzolích a herních terminálech. V České republice se pojem počítačová hra používá pro oba termíny.

2.2.1 Historie

Za jednu z prvních počítačových her se považuje hra OXO vytvořená Alexandrem S. Douglasem roku 1952 v rámci disertační práce na téma problematiky interakce člověka s počítačem. Jednalo se variantu piškvorek hranou na ploše 3x3 políčka. Pro ovládání se používal otočný telefonní volič. O grafický výstup se starala upravená obrazovka osciloskopu, která byla připojena k počítači EDSAC a umožňovala zobrazení bitmapy o velikosti 35×16 pixelů (Tišnovský, 2011).

V roce 1958 William Higinbotham vytvořil hru Tennis For Two (Tenis pro dva), aby zabavil návštěvníky v Národní laboratoři Brookhaven v New Yorku. Podobně jako OXO využívala pro grafický výstup obrazovku osciloskopu. Hra se ovládala pomocí dvou joysticků.

Jendou z prvních známějších her se stala Spacewar!, vytvořená v roce 1962 Stephanem Russellem společně s kolegy z laboratoře umělé inteligence v Massachusetts Institute of Technology. Hra byla určena pro demonstraci schopností počí-

tače PDP-1. Ve hře proti sobě stáli dva hráči pilotující vesmírné lodě s cílem zničit jeden druhého.

2.2.2 Arkádové počítačové hry

Arkádové počítačové hry neboli zkráceně arkády jsou žánrem počítačových her. Jsou charakteristické krátkými úrovněmi, jednoduchým ovládáním a stupňující se obtížností. První arkádová hra inspirována Spacewar!, Computer Space, vznikla v roce 1971. Autory byli budoucí zakladatelé Atari, Nolan Bushnell a Ted Dabney.

Někteří odborníci však považují za první arkádovou hru tu, která byla provozována, už v roce 1947 a byla určena pro hraní na katodových trubicích. Tato hra, vyvinutá Thomasem Goldsmithem a Estlem Mannem (patentovaná v roce 1948), bývá popisována jako velmi jednoduchá. Šlo o to, že dohromady osm katodových trubic bylo používáno k simulaci projektilu vyslaného na cíl. Na obrazovce byl malý svítící bod, jehož dráhu a rychlost ovlivňoval hráč několika tlačítky.

Jednou z nejznámějších a nejúspěšnějších arkádových her se v roce 1972 stal Pong. Pong byl také první komerčně úspěšnou hrou. Jednalo se o dvojrozměrnou hru, která simulovala stolní tenis. Hru bylo možné hrát ve dvou hráčích. Cílem hry bylo získat více bodů než oponent. Hráči získávali body pokaždé, když se protihráči nepodařilo odrazit míček zpět.

Trh s arkádovými hrami se dále rozrůstal a to až do krachu trhu roku 1977. Atari krach přežilo díky úspěšné hře Space invaders (1978). Vývoj dál pokračoval, známým je například Pac-Man (1980) od Namco. S rozšířením levných počítačů jako Commodore 64 a později PC rostl i zájem o další platformy a vývoj tak pokračoval a trvá dodnes.

Podle mého názoru je ale negativní stránkou jednoduchého konceptu hrozba přílišné triviality a nedostatku variace. Zdá se tedy, že jsou arkádové hry již za vrcholem svého využití a nemohou obstát v konkurenci moderních her, které je převyšují ve všech myslitelných parametrech (grafické zpracování, úroveň interakce, atd.). Nicméně i v současné době mají své zastoupení například na mobilních zařízeních. Objevují se však i zcela nové možnosti jejich využití. Například Jensen a kol. (2013) uvádějí ve svém článku „Monkeys would rather see and do: preference for agentic control in rhesus macaques“ možnost využití arkádové hry pro zkoumání sociálních vztahů u skupiny primátů druhu Makak rhesus (*Macaca mulatta*).

Další možností využití arkádových her představuje péče o seniory zlepšováním kvality jejich života. Například v časopise „Nursing older people“ vyšel v roce 2014 článek s názvem „Arcade games bring fun and fitness to care home residents in Japan.“, který se tímto využitím arkádových her zabývá.

2.3 JavaScript

JavaScript je objektově orientovaný, interpretovaný programovací jazyk. Jedná se o jazyk interpretovaný ve webovém prohlížeči na straně klienta. Syntaxe JavaScript byla ovlivněna jazykem C a Java. JavaScript se používá i mimo webové prohlížeče, v souborech PDF nebo rozšířeních do aplikací jako je například Mozilla Thunderbird.

2.3.1 Historie JavaScript

Brendand Eich, který toho času pracoval pro Netscape, začal s vývojem skriptovacího jazyka s názvem Mocha, později LiveScript, pro vydání Netscape Navigator 2 roku 1995. Těsně před vydáním Netscape Navigator byl jazyk přejmenován na JavaScript, aby bylo využito zájmu veřejnosti o jazyk Java.

Javascript se stal úspěšným natolik, že společnost Microsoft jej pod jiným názvem (JScript) implementovala do svého webového prohlížeče Internet Explorer. Z důvodu množství různých verzí bylo nakonec rozhodnuto o standardizaci jazyka organizací European Computer Manufacturers Association (ECMA). ECMA standardem ECMA-262 vytvořila ECMAScript, který se stal základem pro další implementace jazyka JavaScript (Zakas, 2012, s. 2).

2.3.2 ECMAScript

ECMAScript je skriptovací jazyk standardizovaný Ecma international v ECMA-262. ECMScript tvoří základ JavaScriptu. ECMAScript je specifikace jazyka, zatím co JavaScript je jeho konkrétní implementace. ECMAScript je používán jako základ pro další jazyky jako je například ActionScript. Vývoj ECMAScriptu probíhal v následujících edicích (Mozilla Developer Network, 2014).

- ECMAScript 1: První edice vyšla v červnu 1997.
- ECMAScript 2: Změny ve specifikaci, aby byla v souladu s mezinárodním standardem ISO/IEC 16262 byly zveřejněny v srpnu 1998.
- ECMAScript 3: Podpora například do-while, regulárních výrazů, nové metody pro práci s řetězci (concat, match, replace, slice, split s regulárními výrazy atd.), práce s výjimkami. Byl vydán v prosinci 1999.
- ECMAScript 4 : Čtvrtá verze byla nakonec opuštěna v červenci 2008, z důvodu rozdílných pohledů na směřování jazyka a navrhovaných novinek a otázek zpětné kompatibility s předchozími verzemi. Komise se dohodla na vývoji vylepšení ECMAScriptu 3, (kterým se později stal ECMAScript 5). Vývoj nové verze, která zavádí méně změn, než navrhovala edice ECMAScript 4, ale více než předpokládané vylepšení ECMAScriptu 3. Kódové označené této nové verze bylo Harmony. Mezi novinky plánované pro edici ECMASript 4, od kterých bylo nakonec upuštěno, patřily například podpora name-space a packages. Dohoda měla za cíl vytvoření méně radikální nové verze.

- ECMAScript 5: Kromě ostatních novinek byla v prosinci 2009 zveřejněna podpora pro striktní režim (režim s přísnější kontrolou chyb a vynucení některých zásad, například povinnost explicitně deklarovat proměnné), nové metody pro pole a podporu JSON.
- ECMAScript 5.1: V červnu 2011 uveřejněny změny ve specifikaci, aby byly v souladu s třetí verzí mezinárodního standardu ISO/IEC 16262:2011 (ES Wiki 2014).
- ECMAScript 6: Momentálně ve vývoji (ECMAScript 6, 2014).

2.3.3 Document Object Model

Document Object Model (DOM) je programovací rozhraní pro HTML a XML dokumenty (Webplatform, 2015). Poskytuje strukturovanou reprezentaci dokumentu a definuje způsoby, jak mohou být tyto struktury zpřístupněny pro programy. Ty pak mohou měnit obsah, styl a rozložení dokumentu. Document Object Model poskytuje reprezentaci dokumentu jako strukturovanou skupinu uzlů a objektů, které mají vlastní metody a vlastnosti.

Webová stránka je dokument. Dokument můžeme zobrazit jak v okně webového prohlížeče, tak jako zdrojový kód. Jedná se však stále o stejný dokument pouze jinak zobrazen. Document Object Model (DOM) poskytuje další způsob zobrazení, zapsání a manipulace s dokumentem. DOM je plně objektově orientovaná reprezentace webové stránky, která může být modifikována skriptovacím jazykem jako je JavaScript.

JavaScript používá DOM k přístupu k dokumentu a jeho prvkům. Bez DOM by JavaScript neměl žádný model ani referenci k webovým stránkám, XML stránkám a prvkům, které ovlivňuje. Každý prvek dokumentu od hlavičky dokumentu přes tabulky a obrázky po dokument samotný je součástí DOM tohoto dokumentu.

2.3.4 Syntaxe JavaScript

JavaScript je case-sensitive tedy například proměnná s názvem „hodnota“ a proměnná s názvem „Hodnota“ je z pohledu JavaScriptu rozdílný objekt. JavaScript ignoruje mezery, jestliže nejsou součástí řetězce nebo textu uvedeného v uvozovkách.

2.3.5 Datové typy

Všechny programovací jazyk mají zabudovány datové struktury. Tyto datové struktury se mohou měnit jazyk od jazyka. ECMAScript definuje 6 datových typů.

- String: Řetězec je hodnota, která obsahuje nula nebo více znaků, běžně je používán jako reprezentace textu. Jednotlivé znaky jsou zapisovány mezi jednoduché nebo dvojité uvozovky. Do řetězce je možno zahrnout také speciální řídící sekvence jako například \n, označující nový řádek.

- Number: Číslo, JavaScript, nerozlišuje mezi celými čísly a hodnotami s plovoucí desetinou čárkou (interně jsou reprezentována všechna čísla jako čísla s plovoucí desetinou čárkou).
- Boolean: Datový typ boolean může nabývat pouze dvou logických hodnot true a false (pravda a nepravda).
- Null: Proměnná, která obsahuje hodnotu null, neobsahuje žádnou platnou hodnotu typu Number, String, Boolean, Array či Object. Datový typ null má pouze jednu hodnotu a to null.
- Undefined: Jedná se o hodnotu, která je vrácena pro proměnné, které byly deklarovány, ale nebyla jim přiřazena žádna hodnota. Undefined je zpětná hodnota pří využití vlastností objektu, který neexistuje.
- Object: V JavaScriptu je objekt kolekcí vlastností a metod. Metoda je funkce, která je členem objektu.

2.4 Java

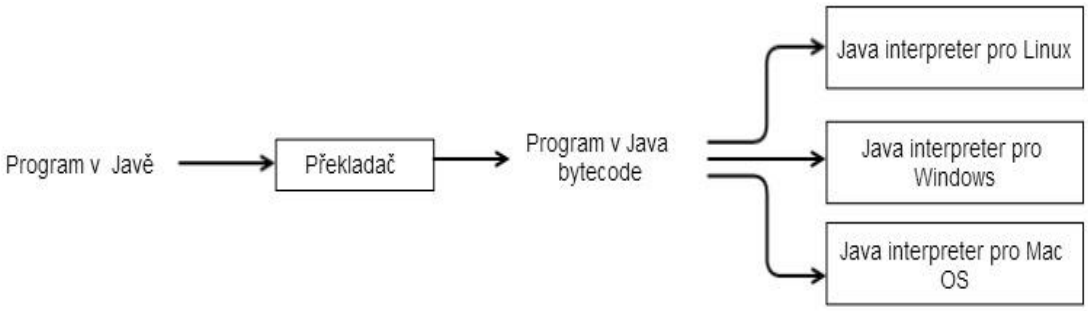
Java je plně objektově orientovaný programovací jazyk. Vytvořen společností Sun Microsystems a v současnosti vyvíjen společností Oracle Corporation. Java může být použita k tvorbě samostatných aplikací a speciálních mini aplikací zvaných Java applet.

2.4.1 Java Applet

Java applet je malý program vytvořen v Javě a vložen nejčastěji na webovou stránku. Spuštěn je Javovým interpretorem, v případě webových stránek se zpravidla jedná o doplněk (plug-in) do prohlížeče. Vložení na stránku se provádí pomocí HTML tagu <object>, někdy se můžeme setkat s dnes již zastaralým tagem <applet>

2.4.2 Java Virtual Machine

Vývojáři Javy se rozhodli pro kombinaci překladu a interpretace. Programy psané v Javě jsou překládané do strojového kódu zařízení, které fyzicky neexistuje. Tento takzvaný virtuální počítač se nazývá Java Virtual Machine. Strojový kód pro Java Virtual Machine se jmenuje Java bytecode(viz Obr.1). To umožňuje Javě nebýt tolik závislý na platformě jako ostatní programovací jazyky.



Obr. 1 Znázornění životního cyklu Java programu. Zdrojový kód je zkompilován překladačem, do podoby „mezikódu" Java bytecode, který je pak interpretován pro procesory jednotlivých platforem.

2.4.3 Porovnání s JavaScript

JavaScript a Java jsou dva odlišné jazyky byť s podobným stylem zápisu. Programy v Javě jsou přeloženy před spuštěním, zatímco JavaScript je interpretován klientem. Javascriptový kód je většinou integrován do webové stránky. Java umožňuje vytvářet samostatné aplikace a applety jsou oddělené od HTML. Javascript používá dynamickou typovou kontrolu, kdy je datový typ proměnné určen až za běhu programu. Java používá statickou typovou kontrolu, datový typ proměnné musí být definován již před překladem. Podobnost je tedy jen ve jméně jazyků.

2.5 ActionScript

ActionScript je objektově orientovaný programovací jazyk pro Adobe Flash Player a Adobe AIR runtime enviromment. Vyvinut společností Macromedia (nyní Adobe Systems). Syntakticky je podobný Javě a C#. Poslední verze ActionScript 3.0 je založena na 4. edici specifikace ECMAScript. Primárně je používán pro vývoj webových stránek a softwaru na platformě Adobe Flash Player za pomocí souborů SWF (Small Web Format) vložených na stránku. Specifikace jazyka ActionScript je veřejně dostupná.

2.5.1 ActionScript Virtual Machine

Podobně jako Java Virtual Machine, jsou programy psané v ActionScriptu překládány do strojového kódu virtuálního zařízení (bytecode). Bytový kód je vnořen do souborů SWF, které jsou spustitelné programem AIR nebo přehrávačem Flash Player.

2.5.2 Adobe Flash Player

Adobe Flash Player je bezplatný multiplatformní software, který slouží ke sledování multimediálních souborů, streamování audia a videa a spouštěni Rich Internet Application pomocí platformy Adobe Flash. Flash Player je dostupný jako

doplněk (plug-in) webových prohlížečů. Je dostupný i na některých mobilních zařízeních nebo jako samostatná aplikace na Windows, Linux a Mac OS.

2.5.3 Porovnání s JavaScript

ActionScript stejně jako JavaScript je založen na specifikaci jazyka ECMASript, proto jádro jejich syntaxe je podobné. Actionscript podporuje jak statickou, tak dynamickou typovou kontrolu. Oproti JavaScriptu obsahuje také některé další datové typy jako je unsigned integer (uint). ActionScript má zabudovanou relativně bohatou knihovnu s třídami pro podporu přehrávání multimediálních souborů a je konzistentní na všech platformách. Actionscript umožňuje tvůrcům vytvářet své vlastní třídy.

ActionScript obsahuje balíčky, které obsahují třídy pro specifickou činnost. Například flash.filesystem package obsahuje třídy pro práci se soubory. ActionScript umožňuje definovat přístupnost k vlastnostem a metodám tříd. Nastavit u funkcí povinné parametry. V ActionScritpu jsou všechny události (event), zpracovávány pomocí funkcí zvaných event listeners. Když objekt vyvolá event, pak event listener na tuto událost odpoví. Event, což je object ActionScriptu, je předán jako parametr funkci event listener. Používání event objektů se liší od modelu eventů DOM, který používá JavaScript.

2.6 World Wide Web

World Wide Web je síť informačních zdrojů. Web je sbírka webových stránek, spojených mezi sebou pomocí hypertextů (HTML a CSS). Web využívá tří mechanismů pro poskytnutí informací co nejširšímu spektru uživatelů.

1. Jednotné schéma pojmenování pro získání zdroje na Webu (např. URI).
2. Protokoly pro přístup k pojmenovaným zdrojům na Webu (např. http).
3. Hypertext pro jednoduchou navigaci mezi zdroji (např. HTML).

2.6.1 URI

Každý zdroj dostupný na internetu HTML dokument, obrázek, video soubor, program atd., má adresu, která může být zaznamenána pomocí Universal Resource Identifier (URI).

URI se typicky skládá ze tří částí:

1. Název takzvaného schématu, používaném k přístupu ke zdroji.
2. Identifikátor zařízení poskytující daný zdroj.
3. Identifikátor samotného zdroje jako cesta.

Například: `http://www.sfinance.cz/pojisteni`

Toto URI se dá číst následujícím způsobem: Existuje dokument, přístupný pomocí protokolu http, na serveru identifikovaném jako `www.sfinance.cz`, dostupný pod cestou /pojisteni. Další schémata, která se často používají v HTML dokumentech, zahrnují „mailto“ pro email a „ftp“ pro File Transfer Protocol.

Některé URI odkazují na konkrétní umístění ve zdroji. Tyto typy URI končí znakem „#“ následovaném označením fragmentu. Například, toto URI odkazuje na fragment dokumentu pojmenovaný „odkazy“:
`http://en.wikipedia.org/wiki/Dopyera#References`
Jednou z variant použiti URI jsou relativní URI. Relativní URI neobsahují žádné názvy schémat. Obecně odkazují na zdroje na stejném zařízení jako je stávající dokument. Relativní URI mohou obsahovat cesty ke zdrojům zapsaným pomocí specifických symbolů (například „..“znamená o jednu úroveň výš v hierarchii definované cestou) a mohou obsahovat označení fragmentů.

Relativní URI jsou převedena na úplná URI s využitím základního URI. Například jestliže jako základní URI budeme uvažovat následující webovou adresu http://en.wikipedia.org/wiki/Ophiuchus, pak relativní URI použita v tomto dokumentu jako například „/wiki/Alpha_Ophiuchi“ je převedena na:
`http://en.wikipedia.org/wiki/Alpha_Ophiuchi`.

V HTML dokumentech se URI používají mimo jiné k:

- Odeslání formuláře.
- Odkazu na jiný dokument nebo zdroj .
- Odkazu na externí style sheet či skript.
- Vložení obrázků a dalších objektů na stránku.
- Odkaz na informace o metadatech v hlavičce souboru.

2.6.2 URL a URN

Uniform Resource Locator (URL) je podskupinou Uniform Resource Identifier (URI). URL obsahuje jak název protokolu používán k získání zdroje, tak jeho umístění. Například http://www.mendelu.cz/cz

Uniform Resource Name (URN) je stejně jako URL podskupinou Uniform Resource Identifier. URN je řetězec znaků sloužící k identifikaci jména webového zdroje, neobsahuje informaci o jeho umístění. Například urn:isbn:0451450523

2.7 HTML

HyperText Markup Language je značkovací jazyk, který popisuje sémantiku a strukturu webového dokumentu. Je využíván globálně pro publikování informací díky jeho všeobecné podpoře.

HTML využívá série tagů (značek), které říkají prohlížeči, co dělat s informacemi na stránce. Tento zápis byl inspirován SGML (Standard Generalized Markup Language) a dále modifikován podle vývoje na poli webových technologií. Mezi značky se uzavírají části textu dokumentů a tím se určuje význam (sémantika) obsaženého textu. Názvy jednotlivých značek a jejich vlastnosti se uzavírají mezi úhlové závorky <> (Duckett, 2011 s. 20). Značky mohou být jak párové, kdy počáteční značka má i svou koncovou značku například <table> (tag pro tabulku) nebo nepárové, které koncové značky nemají například
 značící konec řádku.

2.7.1 Historie

HTML bylo původně vyvinuto v roce 1990 Timem Bernersem-Lee, když působil v Evropské organizaci pro jaderný výzkum (CERN). Zpopularizováno bylo díky prohlížeči Mosaic vyvinutém v Americkém National Center for Supercomputing Applications (NCSA). Během devadesátých let rostlo rozšíření HTML díky rostoucímu zájmu o Web. Během tohoto období bylo HTML několikrát rozšířeno. Z důvodu zájmu o zachování konzistence a kompatibility vznikla myšlenka vytvoření specifikací pro HTML.

HTML 2.0 (Brensen-Lee a Connolly, 1995), bylo vytvořeno pod dozorem Internet Engineering Task Force (IETF) za účelem zformalizování praktik běžných koncem roku 1994. HTML+ roku 1993 a HTML 3.0 roku 1995 slibovaly mnohem bohatší verze HTML. Tyto návrhy vedly k zavedení nových vlastností. Snaha pracovní skupiny World Wide Web Consortium (W3C) vedla ke kodifikování používaných metod a k vytvoření HTML 3.2 v lednu 1997.

Většina lidí se shodovala na tom, že HTML dokumenty by měly fungovat v prostředí různých prohlížečů a platforem stejně. Dosažení kompatibility snižuje náklady pro tvůrce obsahu, protože jim stačí vytvořit jen jednu verzi dokumentu. Bez snahy o standardizaci vzniká mnohem větší riziko, že se z internetu stane prostředí různých soukromých navzájem nekompatibilních formátů, což by v konečném důsledku mělo za následek snížení komerčního potenciálu pro všechny zúčastněné.

Každá nová verze HTML se snažila odrážet konsenzus mezi klíčovými hráči na trhu informačních technologií, aby tvůrci obsahu nebyli nuceni mrhat časem na tvorbu dokumentů ve standardech, které se stanou překonanými v krátkém časovém období. HTML bylo vytvořeno s vizí, že všechny možné druhy platforem by měly být schopny zprostředkovat informaci z internetu, PC různých technických specifikací i rozličná mobilní zařízení s různými rychlostmi připojení.

HTML 4 rozšířil HTML o podporu pro style sheet, skripty, framy, vložené objekty, zvýšenou podporu pro písma psaná zprava do leva, vylepšení funkce tabulek

a vylepšení formulářů se záměrem zlepšení dostupnosti pro uživatele s postiženími.

2.7.2 XHTML

Extensible HyperText Markup Language (XHTML) je modifikovaná verze jazyka HTML vyvinutá organizací World Wide Web Consortium (W3C). Snahou nového standardu bylo učinit HTML více rozšiřitelné a zlepšit spolupráci s jinými datovými formáty. Základ si bere ze syntaxe HTML 4.0, ale je rozšířen o prvky XML.

XHTML zavedl oproti HTML striktnější způsob zápisu syntaxe. Všechny XHTML prvky musí být uzavřeny a to včetně nepárových. XML je též case-sensitive, všechny hodnoty atributů musí být v uvozovkách (Duckett, 2009, 5s.). Nejběžnější rozdíl pro uživatele byl ve způsobu zpracování chyb. Při nalezení libovolné syntaktické chyby prohlížeč zobrazil chybové hlášení a přestal s vykreslováním stránky. Rozdíly byly i ve zpracovávání JavaScriptu a CSS z důvodu case-sensivity XHTML.

Původně byl standard XHTML plánován jako nástupce jazyka HTML, jehož vývoj měl být ukončen verzí HTML 4.01. Proti tomu se ovšem postavili zástupci tvůrců některých webových prohlížečů a založili vlastní skupinu WHATWG (Web Hypertext Application Technology Working Group) a začali pracovat na HTML5. Skupina byla složena z lidí z Apple, Mozilla Foundation a Opera Software. W3C pracoval na XHTML dál publikoval dvě verze XHTML 1.0 a XHTML 1.1 a pracoval na XHTML 2.0. V roce 2007 se ale W3C rozhodla oficiálně uznat HTML5 jako další generaci standardu HTML a v roce 2009 upustila od podpory XHTML 2.0.

2.7.3 Struktura dokumentu HTML

Mezi základní prvky, které tvoří strukturu každého HTML dokumentu, patří: <html>, <head> a <body>. Jedním z důležitých prvků HTML dokumentu je i prvek DOCTYPE. Doctype určuje podle jaké definice má být HTML dokument zpracován. Tag <html> určuje, že následuje dokument typu html, čí xhtml, <head> uvozuje hlavičku dokumentu, kde jsou uloženy metadata jako například název dokumentu, informace o tom jakou znakovou sadu dokument používá, také může obsahovat URL odkazy na jiné dokumenty. Tag <body> uvozuje tělo dokumentu (Larsen, 2013, s. 6). Je místem, které obsahuje veškerý zobrazovaný obsah. Znázornění struktury dokumentu HTML je uvedeno na Obr 2.

```
01 <!DOCTYPE HTML>
02 <html>
03 <head>
04 <title>
05 Ukázková stránka </title>
06 <link href="css/styles.css" rel="stylesheet" type="text/css">
07 <meta charset="UTF-8" />
08 </head>
09 <body>
10  Hello world!
11 </body>
12 </html>
```

Obr. 2 Ukázka základní struktury HTML dokumentu

2.7.4 HTML5

HTML5 je poslední verzí standardu HTML. Finální specifikace byla vydána 28. října 2014 (wikipedia, 2015). HTML5 vznikl jako projekt spolupráce mezi World WIde Web Consortium (W3C) a Web Hypertext Application Technology Working Group (WHATWG) (Sarris, 2013, s. 11). Nový standard přináší novinky jako nativní přehrávání videa, drag-and-drop. Věci, které byly předtím možné jen za použití externích doplňku jako Adobe Flash a Microsoft Silverlight. Podpora nových programovacích rozhraní (API) jako Geolocation API, která dovoluje prohlížeči získat informace o poloze uživatele (s jeho souhlasem), podobně jako některé mobilní aplikace.

Dalším cílem bylo zjednodušení a zkrácení kódu. Jedním z běžných vylepšení u stránek obsahující formuláře je zjednodušení vyplňovaní formuláře pomocí například předpřipraveného textu, validace dat na straně klienta čí možnost vybrat datum a čas pomocí kalendáře. Tato běžně používaná vylepšení, ale před příchodem HTML5 nebylo možné vytvořit bez pomoci JavaScriptu. Kvůli tomuto konceptu je obvyklé, že každá stránka používá jinou implementaci, což vede k nesladěnosti zpracování a někdy i k nefunkčnosti. HTML5 proto zjednodušuje tvorbu těchto běžných formulářových vylepšení, zavedením způsobu jak jich docílit za použití samotného HTML. Toto ulehčuje údržbu kódu a kompatibilitu mezi platformami, protože jednotný způsob řešení může být lépe odladěn i v samotných prohlížečích.

HTML5 také přináší nové možnosti jak usnadnit práci internetovým robotům, známým též jako boti. Za pomoci takzvaných microdat (je možné, například říci internetovým robotům, za pomocí tagu, že následující odkaz na fotografii je foto-

grafií osoby. Tím získávají tvůrci webových stránek možnost označovat obsah tak, aby byl strojově lépe zpracovatelný a dostupnější prohlížečům a vyhledávačům.(MacDonald, 2013, s. 6)

HTML5 také zavádí nový prvek Canvas (Hickson, Ian a kol., 2014). Aplikace, kterou vytvářím v praktické části bakalářské práce, je vykreslována za pomoci prvku <canvas>. Canvas lze použít k vykreslování grafiky skrz skripty v JavaScriptu, lze jej použít i pro přehrávání videa či renderování 2d objektů (Cameron, 2013, s. 19). Canvas vytvoří v HTML dokumentu oblast specifikované výšky a šířky, do které lze vykreslovat grafiku.

2.8 CSS

Cascadining Style Sheets (Kaskádové styly), umožňují úpravu rozložení a designu webové stránky. Zatím co HTML popisuje strukturu CSS prezentaci (Teague, 2011, s. 1). CSS umožňuje, krom ostatního, měnit velikost a barvu textu, pozici prvků jako jsou obrázky na stránce, přidat prvkům ohraničení, či je udělat neviditelné. CSS popisuje, jak má být prvek vykreslen na obrazovku.

Kaskádové styly umožňují mnohem více úprav formátování, než klasické HTML. CSS umožňuje mít oddělenu strukturu a obsah dokumentu od jeho prezentace. Způsob zobrazení dokumentu se nazývá jeho styl. Oddělení stylu od obsahu umožňuje:

- Zjednodušení úprav a údržby jak stylu, tak obsahu a struktury.
- Možnost použití stejného obsahu s různými styly zobrazení.
- Zmenšení velikosti stahovaného zdroje, soubor CSS je stažen jen jednou a poté může být použit pro několik dalších stránek na stejném serveru.
- Návštěvníkům personalizaci stránky, např. vypnutím CSS.
- Konzistenci grafické podoby stránek na celém serveru.

2.8.1 Historie

HTML měla historicky velmi omezené možnosti jak upravovat prezentaci webového dokumentu. Pro úpravu prezentace používala kombinaci prvků a atributů, což mělo za následek dosti nepřehledný kód. HTML se tak starala jak o obsah, tak o prezentaci. Proto byly World Wide Web Consortium (W3C) prezentovány návrhy různých jazyků upravující prezentaci dokumentů. CHSS (Cascading HTML Style Sheets) jazyk, ze kterého CSS nakonec vznikl, byl prvně navržen v roce 1994. V té době například populární prohlížeč Mosaic, který později posloužil jako základ pro prohlížeče Internet Explorer a Netscape Navigator, neumožňoval tvůrcům měnit prezentaci dokumentů s výjimkou několika málo základních HTML tagů. Cílem CSS bylo poskytnout jednoduchý jazyk pro úpravu prezentace dokumentů, který by umožňoval jak tvůrcům, tak uživatelům měnit vzhled jejich dokumentů.

CSS 1 bylo první verzí kaskádový stylů vydanou v prosinci roku 1996. Zavedlo mimo jiné podporu pro změnu písma (Font), barvu textu a pozadí, zarovnání textu, obrázků a tabulek.

CSS 2 druhá verze kaskádových stylů. Rozšiřuje vlastnosti pro formátování písma, absolutní, relativní a fixní polohu.

CSS 2.1 je vylepšenou verzí CSS 2, opravuje chyby a odstraňuje některé nefunkční prvky. CSS 2.1 je momentálně verzi doporučenou W3C.

2.8.2 Implementace

Při zobrazení dokumentu musí prohlížeč spojit informace o stylu s obsahem a strukturou dokumentu. Zpracovává dokument ve dvou stupních:

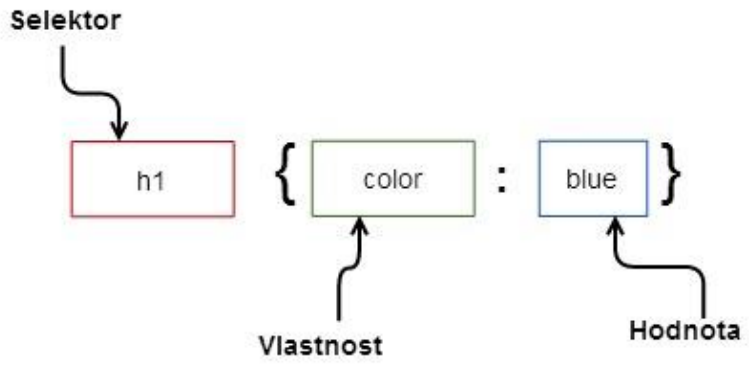
1. Prohlížeč převede prvky značkovacího jazyka CSS do DOM (Document Object Model).
2. Prohlížeč zobrazí obsah DOM.

Značkovacím jazykem nemusí být jen HTML, ale může být například i XML.

2.8.3 Syntaxe

Základní stavební prvky syntaxe(viz Obr3) CSS jsou:

- Selektor: Identifikátor prvku, který bude modifikován.
- Blok deklarací .
    - Vlastnost: Identifikátor vlastnosti, která bude modifikována. Vlastnosti jsou specifikovány ve standardu CSS.
    - Hodnota: Každá vlastnost má množinu přípustných hodnot, které jí mohou být přiděleny.



Obr. 3 Znázornění syntaxe jazyka CSS. Nadpis první úrovně (h1), mění vlastnost color (barva) na hodnotu blue (modrá). (McFarland, 2013, s. 38)

Deklarace jsou uskupeny do bloků, které jsou uzavřeny do složených závorek, které označují začátek a konec bloku. Dvojtečka, odděluje měněnou vlastnost od hod-

noty. V případě více jak jedné deklarace jsou mezi sebou deklarace odděleny středníky. Prvek na stránce může být označen několika selektory současně (Lie, 2014).

2.8.4 CSS3

CSS3 je nejnovější verzí kaskádových stylů, která dále rozšiřuje poslední verzi CSS2.1. Přináší novinky jako stíny, zaoblené hrany, animace či podporu nových rozvržení stránek pomocí zavedení více sloupců textu.

CSS2 trvalo 9 let, od roku 2002 až do 2011, než mu bylo uděleno W3C Recommendation. Tento proces trval tak dlouhou dobu, protože několik málo relativně bezvýznamných podpůrných metod drželo zpět celou specifikaci. Aby bylo urychleno zavedení bezproblémových nových prvků, tak na rozdíl od CSS2, které bylo uvedeno jako jedna velká specifikace s mnoha novými prvky, CSS3 je vyvíjeno v několika samostatných dokumentech zvaných moduly.

Každý tento modul je nyní nezávislá součást jazyka, se svoji vlastní pracovní skupinou, která jej vyvíjí a postupně aktualizuje jeho specifikaci až do konečné podoby. Každý modul má vlastní vývojový cyklus, který se skládá z následujících fází:

- Working draft (pracovní návrh).
- Last Call (poslední výzva).
- Candidate Recommendation (kandidát k doporučení).
- Proposed Recommendation (navržený k doporučení).
- Recommendation (doporučení).

Některé tyto moduly jsou už nyní ve stabilní fázi a jsou již podporovány v některých prohlížečích a dokonce některým už bylo uděleno W3C Recommendation.

3 Metodika

Jedním z úkolů v zadání mé bakalářské práce, bylo vytvořit vlastní jednoduchou arkádovou hru v prostředí webového prohlížeče za pomoci vybrané technologie, v mém případě jazyka JavaScript. Nejdříve bylo nutné rozhodnout, o jaký typ arkádové hry se bude jednat. Rozhodoval jsem se podle následujících kritérií:

- Popularita, dané hry.
- Náročnost provedení.

Mou snahou bylo vytvořit hru, která je známá do té míry, že většina potenciálních hráčů, buď podobnou hru někdy hrála, nebo viděla hrát někoho jiného. Jednou z výhod arkádových her je i jejich vrozený jednoduchý koncept, který usnadňuje pochopení hry. Hra je tak více dostupná širšímu a různorodějšímu počtu hráčů. Dle mého názoru, ale negativní stránkou jednoduchého konceptu je hrozba přílišné triviality a nedostatku variace.

Při výběru jsem též zvažoval technickou náročnost provedení různých typů arkádových her tak, abych našel typ hry, který by svou náročností odpovídal rozsahu bakalářské práce.

Rozhodoval jsem se mezi arkádovou hrou typu had (viz. Obr4) a arkanoid. Ačkoliv považuji arkádovou hru typu had za známější, rozhodl jsem se nakonec zvolit hru typu arkanoid, jelikož koncept hry umožňuje větší možnosti ve způsobu zpracování a provedení.

Obr. 4 Ukázka arkádové hry typu Had

Při výběru použitých technologií pro tvorbu samotné aplikace jsem se rozhodoval mezi několika JavaScriptovými knihovnami a frameworky. Nakonec jsem zvolil framework Phaser, z důvodu bezplatnosti použití, podpory práce s prvkem HTML 5 Canvas a propracovanou dokumentací dostupnou online. Díky tomu, že je Phaser zaměřen na tvorbu her v prostředí webového prohlížeče a mobilních zařízení, disponuje množstvím předdefinovaných funkcí a prvků pro jejich tvorbu.

Pro využití HTML Canvas jsem se rozhodl také proto, že umožňuje zobrazení složitých grafických útvarů a animací v prostředí webového prohlížeče bez nutnosti použití externích doplňků. Hra bude obsahovat grafické komponenty, vytvořené za pomoci grafického editoru. Bude zobrazovat informaci o dosaženém skóre tak, aby motivovala hráče pokračovat ve hře. Hra bude postupně zvyšovat obtížnost v závislosti na pokroku uživatele. Dále bude zobrazovat zbývající počet životů, aby měl hráč přehled v jaké se nachází situaci.

4 Praktická část

Tato část práce se věnuje tvorbě konkrétní jednoduché arkádové hry typu arkanoid, vývojovému prostředí a technologiím použitým při její tvorbě. Popisuje způsob implementace zvoleného řešení.

4.1 Arkanoid

Arkanoid je arkádová hra. Ve hře je na horní třetině obrazovky několik vrstev cihel (viz. Obr.5). Ve hře putuje míč po obrazovce odrážejíc se od horních a bočních hran obrazovky. Když míč zasáhne cihly, míč se odrazí a cihla zmizí, ve hře hráč ovládá malou plošinu na spodní hraně obrazovky, od které se míč odráží. Hráč ztrácí život, pokud se míč dotkne spodní hrany. Jakmile klesnou životy na nulu, hra skončí. Cílem hry je rozbít co nejvíc cihel a neztratit při tom všechny životy.

4.1.1 Historie

Arkanoid a jemu podobné hry mají své kořeny sahající až do roku 1967, když Ralph Baer navrhl herní systém Magnavox Odysses. Jedna z her, která na něm vznikla, byla hra, která jako první zavedla koncept míče a pálky.

O několik let později se Nolan Bushnell a Steve Bristow, společně se Stevem Jobsem a Stevem Wozniak rozhodli posunout koncept míče a pálky o kus dál a navrhli a vytvořili Breakout Atari v roce 1976. To byla první hra, která ke konceptu pálky a míče přidala kostku. Roku 1986 Sega vydala Gigas a Gigas Mark II, hry, které inovovaly tento žánr dále přidáním nových grafických a herních prvků.

Stejného roku vydalo Taito Arkanoid, ačkoliv byl vydán později než Gigas. Stal se jednou z nejpopulárnějších her tohoto žánru.

Obr. 5 Ukázka ze hry Arkanoid

4.2 Vývojové prostředí a nástroje

Pro tvorbu aplikace, byl použit textový editor Sublime Text v kombinaci s dalšími nástroji jako je JavaScriptový task runner GRUNT. Jako webový server jsem použil XAMPP, debuggr v podobě Google Chrome Developer Tools a grafický editor Gimp.

4.2.1 Sublime Text

Sublime Text je dílo Jona Skinnera. Jedná se o textový editor napsaný v jazyce C++. Sublime Text se liší od běžných textových editorů svojí modifikovatelností díky zabudovanému správci balíčků, který umožňuje instalaci velkého množství doplňků tvořených komunitou.

Sublime Text podporuje zvýrazňování syntaxe pro jazyk JavaScript a za pomocí doplňků i podporu statické analýzy JavaScriptové kódu založeném na nástroji JSHint. Podobné doplňky jsou k dispozici i pro ostatní jazyky jako HTML a CSS.

Sublime Text umí zaznamenat a používat vlastní makra. Automaticky dokončovat kód, zvýraznit párové závorky pro snadnější orientaci v kódu a možnost uložení často používaného kódu do takzvaných „Snippets“ pro jeho opakované použití.

4.2.2 Grunt

Grunt je JavaScriptový task runner (spouštěč úloh), který funguje jako doplněk na platformě Node.js. Grunt umožňuje automatizovat některé činnosti, které je jinak tvůrce nucen opakovaně provádět při práci na JavaScriptovém projektu. Grunt se instaluje lokálně. Úkolem globálního příkazu „grunt-cli“ je nainstalovat lokální ver-

zi Gruntu do adresáře projektu. Je proto možné mít souběžně několik projektů, každý s jinou verzí Gruntu.

Grunt pro svou práci využívá dvou souborů package.json (viz Obr. 6) a Gruntfile.js (viz. Obr. 7). Package.json obsahuje dobrovolná metadata (jméno autora, verze, název projektu), názvy doplňků a jejich verze. Gruntfile.js je JavaScriptový soubor, který Grunt používá, aby byl upravitelný a použitelný pro specifické požadavky konkrétního projektu. Při spuštění „grunt“ z příkazového řádku, jej Grunt rekurzivně hledá v adresáři projektu. Gruntfile.js umožňuje registrovat vlastní příkazy pro použití v projektu.

Při tvorbě svého projektu jsem využíval doplněk contrib-watch, který umožňuje sledovat změny v souborovém systému a při změně, vytvoření nebo smazání souboru, který odpovídá masce (například jen soubory s příponou js), spouštět předdefinované úlohy. Contrib-watch má zabudovanou podporu pro komunikaci s doplňkem pro Google Chrome, LiveReload. LiveReload umožňuje automaticky obnovit stránku v prohlížeči. Při změně souboru tak Grunt komunikoval s LiveReload a obnovoval stránky v mém prohlížeči při práci na JavaScriiptových souborech.

Mezi oblíbené doplňky patří ještě validátor JSHint, zmenšovač souborů JavaScipt UglifyJS čí spojovač souborů textových souborů Concat a další.

```
{
  "name": "Arkanoid",
  "version": "0.2.0",
  "description": "Grunt na livereload",
  "main": "gruntfile.js",
  "dependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-watch": "~0.6.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "test"
  },
  "author": "Gustav Chladek"
}
```

Obr. 6 Obsah souboru package.json. V sekci „dependencies“ ohraničenou složenými závorkami jsou vypsány použité prvky

```
module.exports = function(grunt) {
grunt.loadNpmTasks('grunt-contrib-watch');

    grunt.initConfig({


        pkg: grunt.file.readJSON('package.json'),

        watch: {
            css: {
                files: 'js/**/*.js',

                options: {
                    livereload: true,
                },
            },
        },

    });


};
```

Obr. 7 Obsah souboru gruntfile.js. Při běhu „grunt watch“, sleduje grunt lokální změny v souborech uložených v adresáři js s příponou JavaScript(js). V případě změny informuje prohlížeč Google Chrome pomocí rozšíření Google Chrome livereload.

4.2.3 Git

Git je distribuovaný systém správy verzí. Jedná se o systém, který zaznamenává změny v souborech projektu tak, aby bylo možné se později vrátit k předcházejícím verzím. Git pracuje na principu mezi snímků (snapshoots). Při uložení stavu projektu (git commit) udělá Git snímek stavu souborů, které má určeny ke sledování. Pokud se od posledního commitu soubory nezměnily, Git je znova neukládá a místo toho jen odkazuje na předešlou verzi souborů. Mezi těmito commity lze procházet

a vracet se tak k jednotlivým verzím souborů, které commit tvoří. Git podporuje i práci více tvůrců na jednom projektu, ukládání na vzdálené úložiště čí větvení projektů na jednotlivé nezávislé větve.

4.2.4 Google Chrome Developer Tools

Developer Tools prohlížeče Google Chrome jsou skupinou nástrojů sloužících k ladění webových aplikací, prohlížení komunikace mezi uživatelem a vzdáleným hostem, sledování výkonu jednotlivých komponentů webové stránky. Pro tvorbu mé aplikaci byla ovšem nejužitečnější konzole JavaScriptu, která umožňuje zjistit hodnotu proměnných při běhu aplikace. Stejně tak jako zobrazit chyby v běhu programu i s odkazem na umístění ve zdrojovém kódu.

4.2.5 Gimp

Gimp je svobodný bezplatný rastrový grafický editor. Ve svém projektu jsem jej využil pro tvorbu grafických podkladů a úpravu obrázků, které jsem použil ve své hře. Při tvorbě aplikace byla použita volně dostupná grafika z `http://opengameart.org/content/puzzle-game-art`.

4.3 Použité technologie

Pro tvorbu aplikace jsem využil framework Phaser. Jedná se o framework určený pro tvorbu her za pomoci HTML5 napsaný v jazyce JavaScirpt. Hra je vložena do HTML dokumentu, který je stylizován pomocí CSS.

4.4 Phaser

Phaser je herní JavaScriptový framework založený na JavaScirptové renderovacím enginu Pixi.

4.4.1 Koncept

Hra je spuštěna v abstraktním světě, kde všechny herní objekty jsou „naživu“. Jelikož herní svět může být větší než velikost okna hry, hráč se na svět dívá přes koncept kamery. Kamera je zobrazení do herního světa viditelného v okně prohlížeče.

4.4.2 Objekt

Pro vytvoření hry v Phaseru, musí byt nejdřív vytvořen objekt Game. Při vytvoření instance objektu Game konstruktorem, jsou první dva parametry právě výška a šířka canvasu. Canvas je poté vytvořen pomocí DOM v HTML dokumentu.

Poté je hra vykreslena na tento prvek Canvas, který je překreslován frekvencí 60 snímku za sekundu. Tento canvas prvek je vložen pod tag HTML body, pokud není určeno jinak pomocí parametru „parent“ při vytváření objektu Game.

4.4.3 Stavy

Hra vytvořena v Phaseru je posloupností stavů (states). Každá hra musí obsahovat alespoň jeden stav. Stavy mohou být přidány jak při tvorbě objektu game, tak kdekoliv jinde v kódu, jelikož objekt hry je přístupný odkudkoliv v programu. Stav je sám objektem.

Každý stav může implementovat některé rezervované funkce, které jsou spouštěny v následujícím pořadí:

- Init: je zavolán jako první, používá se pro zpracování parametrů přijatých při změně stavu hry.
- Preload: se používá nejčastěji pro načítání externích podkladů.
- Create: je zavolán po ukončení preload. Tato část slouží k vytvoření instancí všech trvalých objektů ve hře.
- Update: je zavolán po ukončení jednoho průběhu herní smyčky. Jeho cílem je aktualizace hry, například zvýšení skóre, detekce kolize, změna polohy hráče.
- Render: je zavolán po dokončení vykreslení hry.
- Paused: je zavolán, pokud byla herní smyčka pozastavena.
- Shutdown: je zavolán při destrukci objektu stav.

4.5 Node.js

Serverovou část aplikace má na starosti Node.js, JavaScriptové běhové prostředí určené pro použití na serverové a síťové aplikace. Node.js používá V8 JavaScript Engine od společnosti Google. Node, podporuje instalaci modulů rozšiřujících jeho možnosti, pomoci node package manager.

4.6 Redis

Redis je key-value (klíčová), NoSQL databáze. Podporuje několik datových struktur:

- Řetězce: pod jedním klíčem najdeme jeden řetězec.
- Seznamy: kolekce řetězců řazených podle pořadí vložení.
- Sady: kolekce unikátních neřazených řetězců.
- Seřazené sady: sada unikátních řazených prvků podle číselné hodnoty skóre.
- Hashe: implementace asociativního pole.

Pro uchování žebříčku hráčů, je použita datová struktura Seřazené sady. Výhodou tohoto řešení je jednoduchost implementace, jelikož databáze je již podle skóre seřazena. Nevýhodou tohoto řešení je vynucená unikátnost řetězců (jmen), proto

ačkoliv je možné mít více hráčů se stejným skóre, není možné mít více hráčů používající stejné jméno a tak dochází k přepisování záznamů se stejným jménem.

4.7 jQuery

jQuery je multi-platformní JavaScriptová knihovna, používána na více jak polovině z 10 000 nejvíce navštěvovaných stránek. Jedná se tak o nejpoužívanější JavaScriptovou knihovnu na světě. Je dostupná zdarma pod licencí MIT. Knihovna usnadňuje navigaci v dokumentu a výběr DOM prvkům tvorbu animací a AJax aplikací.

jQuery() funkce (zkráceně $()) je nejdůležitější funkcí v jQuery knihovně. Je často přetěžována a existují 4 způsoby jejího vyvolání:

1. Zavolání s parametrem CSS selektoru (řetězec). Takto zavolaná funkce vrátí pole prvků aktuálního dokumentu odpovídající danému selektoru.
2. Zavolání s parametrem prvku, dokumentu nebo objektu okna. Takto zavolaná funkce vrátí tento objekt vložený do jQuery objektu, což umožňuje s ním manipulovat použitím jQuery metod.
3. Zavolání s parametrem řetězce HTML textu. jQuery vytvoří prvky popsané v parametru a vrátí objekt jQuery obsahující tyto prvky. jQuery automaticky nevloží tyto prvky do dokumentu.
4. Zavolání s parametrem funkce. Tato funkce bude zavolána po dokončení načítání dokumentu a DOM je připraven k manipulaci.

jQuery také obsahuje mnoho metod pro prácí s událostmi a univerzálních užitečných funkci (např. each()).

5 Implementace

5.1 Klientská část

Jak již bylo zmíněno, framework Phaser, pracuje se stavy. Aplikace je rozdělena na samostatné JavaScriptové soubory. Každý odpovídající jednomu konkrétnímu stavu. Jak bylo taktéž zmiňováno v předcházející kapitole, každý stav může využít předdefinovaných funkcí, které umožňují spouštět kód v závislosti na aktuální situaci aplikace. Ve své práci využívám čtyři z těchto funkci: init, preload, create, update. Při spuštění aplikace jsou stavy volány v následujícím pořadí:

1. Boot
2. Načti
3. Menu
4. Hra
5. Score

5.2 Boot

Boot je prvním volaným a také obsahově nejkratším stavem hry. Ve funkci init jsou zpravidla nastaveny parametry specifické pro konkrétní zařízení. Zde je pomoci parametru input.maxPointers zakázáno používání multi-touch u dotykových obrazovek, jelikož nemá v dané aplikaci žádné využití. V následující funkci preload je pomoci Phaser.loader načtena do cache grafika tvořící progress bar. Ve funkci create je následně volán další stav hry Načti.

5.3 Načti

Hra pro svou činnost používá externích grafických souborů. Soubory jsou uloženy ve vlastní složce v projektovém adresáři. Jsou zde uloženy společně s ostatními grafickými podklady. Načítaní těchto podkladů má za úkol JavaScriptový soubor nacti.js.

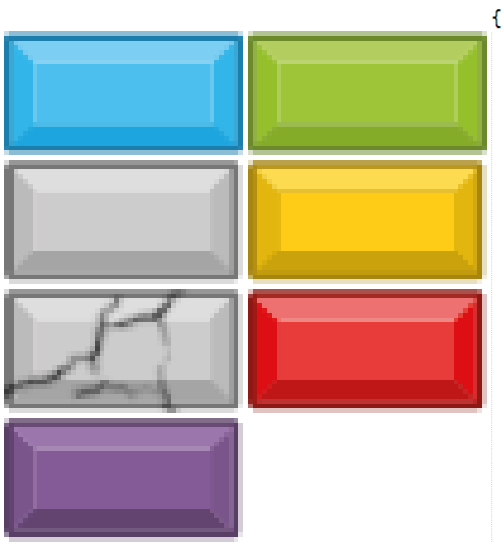
Ve funkci preload probíhá načtení podkladů pomocí třídy Loader. Postup načítaní je graficky zobrazován pomocí metody setPreloadSprite, která jako parametr přijímá obrázek (načtený v Boot) a jeho šířku následně ořezává podle stupně dokončení. Loader se také stará o zpracování všech externích prvků jako jsou obrázky a datové soubory. Načteny jsou do třídy Cache, která je používaná pro skladování těchto souborů. Každá hra má jen jednu instanci Cache. V aplikaci jsou použity 3 způsoby uchování grafiky načítané metodami třídy Loader:

- Image: Dvojrozměrný bitmapový obrázek, který je možno vložit do scény. Metoda má jako povinné parametry. V aplikaci jej používám pro vkládání jednoduché grafiky jako je například míč a plošina:

- Klíč (string) sloužící k referenci obrázku.
- Url (string) adresu obrázku.

- Sprite sheet: Obrázek skládající se ze dvou a více dvojrozměrných bitmapových obrázků zpravidla stejné velikosti. V aplikaci jej používám pro tvorbu tlačítek reagujících na pozici kurzoru. Metoda má jako povinné parametry:
  - Klíč (string) sloužící k referenci obrázku.
  - Url (string) adresu obrázku.
  - frameWidth (number) šířka jednotlivého obrázku.
  - frameHeight (number) výška jednotlivého obrázku.

- Atlas: Obdobně jako Sprite sheet se jedná o jeden větší obrázek skládající se z více jednotlivých obrázků (viz. Obr. 8). Na rozdíl od něj ovšem může obsahovat i obrázky různého typu a velikosti. Umístění, názvy a rozměry jednotlivých obrázků jsou uloženy v příslušném JSON nebo XML dokumentu, který je načten společně s obrázkem. V aplikaci jej používám pro uložení různobarevných cihel s použitím JSON dokumentu (viz. Obr. 8). Metoda má jako povinné parametry:
  - Klíč (string) sloužící k referenci obrázku.
  - Url (string) adresu obrázku.
  - atlasURL nebo atlasData odkaz na JSON nebo XML specifikující jednotlivé obrázky.



```
{
"frames": [{
    "filename": "modra",
    "frame": {
        "x": 0,
        "y": 0,
        "w": 50,
        "h": 26
    },
    "rotated": false,
    "trimmed": false,
    "spriteSourceSize": {
        "x": 0,
        "y": 0,
        "w": 50,
        "h": 26
    },
    "sourceSize": {
        "w": 50,
        "h": 26
    }
```

Obr. 8 Atlas cihel s ukázkou části souvisejícího JSON dokumentu

Po ukončení funkce preload následuje funkce create, která změní barvu pozadí, změnou vlastnosti backgroundColor třídy stage a zavolá následující stav Menu.

5.4 Menu

V tomto stavu je vytvořeno menu, za pomoci třídy button s využitím grafických souborů načtených v předchozím stavu Načti. Button je speciální typ grafiky, který je nastaven tak, aby automaticky zpracovával události třídy Pointer. Pointer má za úkol zpracování vstupů myši. Button registruje 4 stavy korespondující s pozicí tlačítka a myši a umožňuje nastavit pro každý z nich odlišnou grafiku. V aplikaci používám stav „Over“, který je zavolán při překrytí kurzoru a tlačítka. Při němž měním vykreslovaný obrázek ve sprite sheetu grafiky tlačítka. Při poklepání na tlačítko hrát, zavolá tato třída přiřazenou funkci start_Klik, která naopak zavolá další stav Hra. Obdobně při poklepání na tlačítko score se změní stav hry na stav score. Speciální případ je tlačítko Myš/Klávesnice, které slouží k možnosti zvolení preferovaného druhu ovládání hry.

5.5 Hra

Stav Hra je stavem, ve kterém hráč stráví nejdéle času. Obdobně stejnojmenný JavaScriptový soubor obsahuje hlavní nejdelší část zdrojového kódu aplikace.

5.5.1 Create

V této části programu, je prvně pomocí třídy Physics nastaveno, chování fyzikálního modelu hry. Hra, dále generuje dynamické pozadí pomocí náhodné funkce a opakujícího se časového intervalu. Při každém průběhu intervalu je náhodně generován jeden ze čtyř možných oblaků a taktéž náhodně je mu přidělena velikost a průsvitnost. Velikost je pak přímo úměrná rychlosti pohybu oblaku po obrazovce.

Úrovně hry jsou generovány z předlohy ve formátu JavaScriptového pole řetězců. Každý druh cihly je označen písmenem. Pomocí for cyklu je pole procházeno až do konce, každý znak odpovídá jedné vygenerované cihle na hrací ploše. Výjimkou je znak pomlčky, která značí prázdné místo.

Při generaci cihel je za pomoci číselné hodnoty za znakem inicializováno i jméno cihly. Jméno cihly později slouží ke generování powerUpů při zničení cihly. Tudíž například „z1“ vytvoří zelenou cihlu, která při zničení vytvoří padající powerUp ve tvaru srdce, který hráč může zachytit a doplnit si tak ztracený život. Funkce obsahuje ještě dvě konstanty určující odsazení cihel od levého a horního okraje hry.

HUD (head-up display) hry je tvořen textem zobrazujícím dosažené skóre a úroveň, ve které se nacházíte. Dále obsahuje ikony srdcí indikující počet zbývajících životů a konečně tlačítko umožňujícím pozastavit hru.

Hra poté vytvoří objekt míč a objekt plošina (všechny objekty jsou tvořeny pomocí grafických souborů načtených v předcházejícím stavu hry). Podobně jako plošina je nastaven úchytný bod uprostřed míče a nastaveny parametry chování v herním světě (nastaven odraz od hran hry s výjimkou spodní hrany).

Pozastavení hry je možné pomocí tlačítka umístěném v levém horním rohu vykreslovaného okna hry. Phaser podporuje metodu game.pause, bohužel při použití této metody hra nejenom, že přestane vykreslovat hru, ale také přestane poslouchat signály vstupu, což dosti komplikuje zpracování signálu pro opětovné spuštění hry. Proto je pauza ve hře řešena pomocí vlastní funkce. Při stisku tlačítka je uloženo, zda byla hra právě pozastavena, do booleovské proměnné. Následně je rychlost míče uložena do vlastní proměnné a společně s ostatními předměty je jeho pohyb zastaven přiřazením nuly do hodnoty rychlosti. Následně je selektivně zablokováno sledování směrových kláves, které pohybuji plošinou případně změna polohy plošiny při pohybu myši. Na vykreslovací ploše hry je zobrazena průhledná informační grafika a tlačítko pauzy je zvýrazněno. Pro opětovné pokračování hry je možno kdykoliv stisknout klávesu P případně kliknout na tlačítko pauzy. Poté je nastavena rychlost míče zpět pomocí uložené hodnoty. Obdobně je přiřazena rychlost pro padající powerUpy (jejichž rychlost je konstantní a tak není potřeba ji ukládat) a vstup na kurzorových klávesách je opět povolen, stejně jako registrace pohybu myši a podle ní pohyb plošiny.

5.5.2 Update (Herní smyčka)

Herní smyčka je podobně jako u filmu vizuální reprezentace každé hry tvořena posloupností snímků (frame). Framy tvoří obraz herního stavu v danou chvíli. Tato funkce je volána frekvencí 60 snímků za sekundu.

V této smyčce je neustále upravována poloha plošiny podle vstupu z klávesnice, či myši. Nejdůležitějšími funkcemi v této části hry jsou funkce, které jsou volány při kolizích herních objektu a to:

- ballHitPlatform

Je funkce, která si jako parametr bere instanci objektu míč a plošina. V závislosti na úhlu dopadu, míče na plošinu určuje horizontální zrychlení, které je míči uděleno. Zrychlení je závislé na místě dopadu míče na plošinu. Okraje plošiny odrážejí míč s větším zrychlením než střed plošiny.

- ballHit

Je funkce volaná při zásahů míče a cihly. Jako parametr si bere instanci míče a zasažené cihly. Cihla, která byla zasažena, proběhne animací mizení a poté je poškozena předdefinovanou metodou damage, která jako parametr používá číselnou hodnotu, o kterou se sníží vlastnost health cihly. Pokud je vlastnost health cihly rovná nule je cihla automaticky zničena metodou kill a hra zvýší hráči skóre. Jestliže již neexistují na hrací ploše cihly, hra načte další úroveň, ke skóre přidá bonus a vrátí míč na svoji startovací polohu.

- powerUpHit

Tato funkce je volána při zásahu platformy a powerUpu. Objekty powerUpu mají definováno jméno při svém vytvoření určující jejich efekt na hru. Přepínač switch pak podle tohoto identifikátoru určí a provede příslušný efekt powerUpu.

5.6 Score

Stav score je posledním stavem hry. V tomto stavu se odehrává komunikace mezi hrou a databází s nevyššími skóre. Jelikož Phaser jakožto herní framework nepodporuje odesílání dat ani zpracování vstupního textu používá aplikace JavaSciriptovou knihovnu jQuery a nástavbu jQuery UI.
V úvodní funkci init se ukládá skóre odeslané jako parametr metody start třídy Phaser.Statemanager volané při změně stavu hry. Poté je vytvořena grafika a jsou připraveny textové objekty pro zobrazení přijatých dat. Samotné zpracování a odeslání jména hráče společně s dosaženým skóre probíhá pomocí HTML form. Formulář je běžně skryt použitím CSS vlastnosti visibility. Vzhled formuláře je pak upraven pomocí widgetu Dialog ze jQuery UI do podoby dialogového okna se vstupním polem pro jméno. Před odesláním formuláře je u zadaného jména zkontrolována délka a pomoci funkce využívající regulárních výrazů je zkontrolována přítomnost nepovolených znaků. Případné možné chybové situace vizuálně zobrazuje funkce tips v hlavičce dialogového okna.

Samotný proces odeslání má na starosti metoda jQuery.ajax, která odešle hodnotu z formuláře převedenou na JSON objekt použitím metody JSON.stringify a specifikuje cílové url, očekávány typ dat v odpovědi (application/json) a dotazovací metodu (POST). Při úspěšném přijetí odpovědi zavolá Callback metodu, která přijatými daty naplní připravené textové objekty.

Pokud hráč vyplní a odešle své jméno, zobrazí se mu jména a skóre prvních 4 hráčů v žebříčku a také jeho pořadí a skóre. Jestliže jméno neodešle, zobrazí se mu jen pořadí prvních 4 hráčů, stejné je to pokud hráč přejde do stavu skóre přímo z úvodního menu hry.

5.7 Serverová část

Serverová část běží na platformě Node.js. Aplikace využívá moduly express, underscore, redis, body-parser. Potřebné moduly jsou zapsány a uloženy v souboru package.json v sekci dependencies (závislosti). Tento soubor obsahuje dále metadata související s projektem (název projektu, verze, popis a další). Instalace závislostí probíhá pomocí příkazu npm install (balíčkovací systém pro Node.js) spuštěného z kořenové složky projektu. Ten se podle instrukcí nalezených uvnitř package.json postará o jejich korektní instalaci.

5.7.1 Express

Express je webový Framework pro Node.js. Express v aplikaci umožňuje nastavit základní směrování a nastavit tak chovaní aplikace při požadavku na konkrétní endpoint (URI nebo cesta) a druh dotazovací metody. Express pro generování HTML souborů využívá šablonovací engine Jade (viz. Obr. 9). Jade je založena na jiném populárním enginu Haml.

```
#formular
  form
    fieldset
      p.Score
      p.validateTips Zadeje jméno hráče
      input#name.text.ui-widget-content.ui-corner-all(type='text', name='name', value='')
      input(type='submit', tabindex='-1', style='position:absolute; top:-1000px; p:-1000px')
```

```
<div id="formular">
  <form>
    <fieldset>
      <p class="Score">
      </p>
      <p class="validateTips">
        Zadeje jméno hráče
      </p>
      <input id="name" type="text" name="name" value="" class="text ui-widget-content ui-corner-all">
      <input type="submit" tabindex="-1" style="position:absolute; top:-1000px; p:-1000px">
    </fieldset>
  </form>
</div>
```

Obr. 9 Ukázka převodu prvku formulář zapsaného v Jade do HTML.

5.7.2 Underscore

Underscore je JavaSciriptová knihovna, která poskytuje užitečné funkce pro práci s poli, objekty a daty. V aplikaci je používána pro převod odpovědi odesílané serverem do klientské aplikace z datového typu pole do datového typu JavaScriptový objekt.

5.7.3 Redis

Redis je modul, který umožňuje Node.js komunikovat s Redis databází. Je používán pro navázání spojení s databází a autorizaci. Pomocí tohoto modulu jsou také přidávány do databáze nové klíče a zpracování požadavků na databázi, konkrétně pořadí hráče v žebříčku.

5.7.4 Body-parser

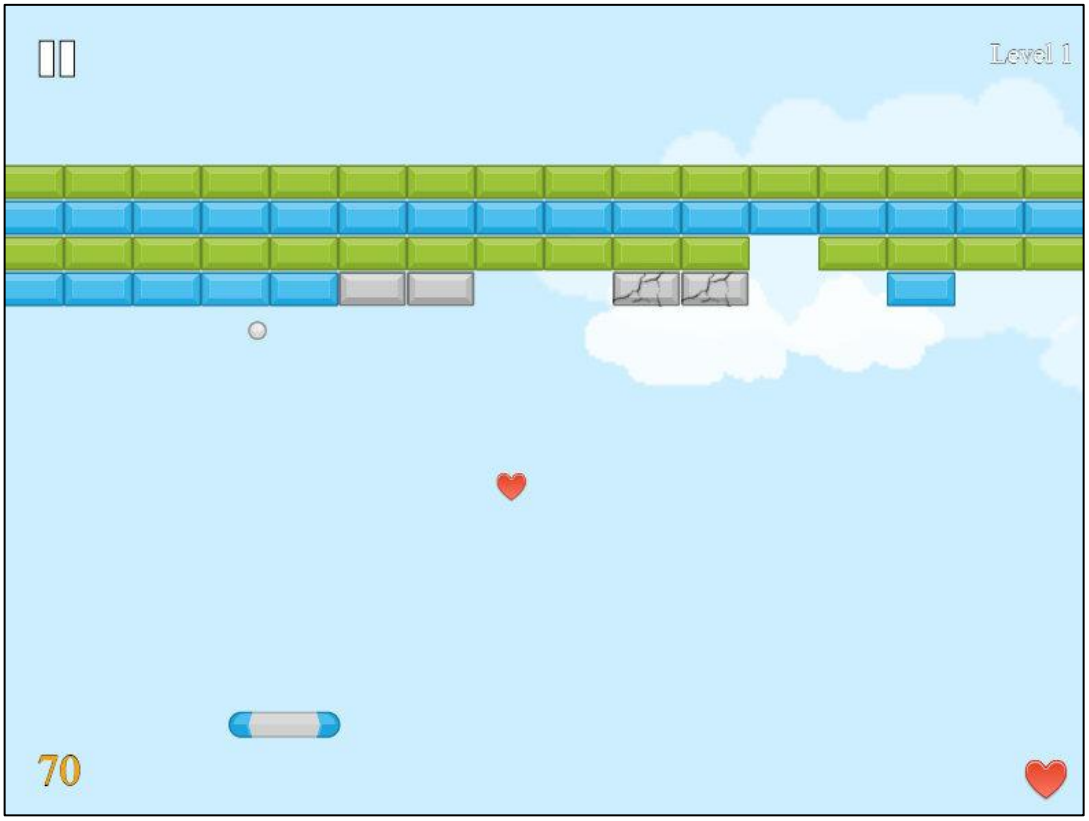
Body-parser je modul pro převod dat přijatých z těla požadavku do JSON objektu.

Statické soubory jsou zpracovány pomoci express.static, která definuje veřejně přístupnou složku ve struktuře aplikace. Index.html, který obsahuje tělo hry je odeslán jako odpověď na HTTP požadavek GET na kořenový adresář.

V další části serverová aplikace naslouchá na HTTP požadavek typu POST na relativní cestě /formulář. Při zaznamenání přijetí požadavku se aplikace připojí k databázi a autorizuje uživatele. Body-parser převede přijatá data na JSON objekt.

Pokud bylo přijato jméno, uloží jej aplikace společně se skóre do databáze. Dále aplikace pošle požadavek do databáze o vypsání 4 nejlepších hráčů. Pokud bylo v požadavku přijato jméno a skóre, zjistí jeho umístění.

Odpověď uložená jako pole je v následující časti kódu převedena na objekt pomocí stejnojmenné metody knihovny underscore. Takto zpracovaný objekt je následně odeslán jako odpověď zpět uživateli.



Obr. 10 Ukázka ze hry

6 Závěr

Hra, za kterou můžeme považovat systém, ve kterém hráči soupeří v umělém souboji, definovaném pravidly, který vyprodukuje měřitelný výsledek, je nedílnou součástí vývoje lidské společnosti. S nástupem informačních technologií se objevily nové možnosti jejich využití také v oblasti her. Svébytným žánrem počítačových her jsou arkádové hry (arkády). Jsou charakteristické krátkými úrovněmi, jednoduchým ovládáním a stupňující se obtížností.

Jako součást této bakalářské práce jsem vytvořil jednoduchou arkádovou hru (viz. Obr. 10) v prostředí webového prohlížeče za pomoci jazyka JavaScript. Druh arkádové hry jsem vybíral podle popularity a náročnosti provedení. Zvažoval jsem arkádovou hru typu had nebo arkanoid. Ačkoliv považuji arkádovou hru typu had za známější, rozhodl jsem se nakonec zvolit arkanoid, jelikož mi poskytnul větší možnosti při způsobu zpracování a provedení.

Z možných technologií pro tvorbu samotné aplikace jsem z JavaScriptových knihoven a frameworků vybral framework Phaser. Bylo to především z důvodu bezplatnosti použití, podpory práce s prvkem HTML 5 Canvas a propracovanou dokumentací dostupnou online. Díky tomu, že je Phaser zaměřen na tvorbu her v prostředí webového prohlížeče a mobilních zařízení, disponuje množstvím předdefinovaných funkcí a prvků pro jejich tvorbu.

Pro využití HTML 5 Canvas jsem se rozhodl také proto, že umožňuje zobrazení složitých grafických útvarů a animací v prostředí webového prohlížeče bez nutnosti použití externích doplňků. Hra proto obsahuje grafické komponenty, vytvořené za pomoci grafického editoru. Zobrazuje informaci o dosaženém skóre tak, aby motivovala hráče pokračovat ve hře. Rovněž zobrazuje počet zbývajících životů, díky tomu má hráč přehled o situaci, ve které se nachází. Hráči umožňuje sbírat předměty padající ze zasažených cihel, které ovlivňují herní prvky hry. Po skončení hry si hráč může také porovnat svoje dosažené skóre s ostatními.

Při tvorbě hry jsem narazil také na některá úskalí, na která bych chtěl upozornit. Týkají se obecně moderních webových aplikací. Korektní běh aplikací závisí na prohlížeči uživatele. Implementace a podpora novějších prvků HTML5 a CSS3 se liší u jednotlivých prohlížečů. Dalším problémem použití JavaScriptu jako interpretovaného jazyka je přístupnost zdrojového kódu a nutnost použití interpreta (webový prohlížeč), což už v principu s sebou přináší větší požadavky na systémové zdroje a nižší rychlost.

Stanovený cíl práce byl splněn. Vytvořená hra je funkční a zdrojové kódy jsou uvedeny v příloze. Aplikace je dostupná také online na adrese `https://shielded-taiga-1294.herokuapp.com/`

7 Literatura

BERNERS-LEE, T. A CONNOLLY, D., 1995, *Hypertext Markup Language - 2.0*. Rfc-editor.org [online]. 1995. [cit. 31. prosince 2014]. Dostupné na http://www.rfc-editor.org/rfc/rfc1866.txt

CAMERON, DANE, *A software engineer learns HTML5, JavaScript & jQuery*. 1. vyd. Cisdal Publishing, 2014, 206 s. ISBN 1493692615.

CARR, D., Buckingham, D., Burn A., Schott, G. *Computer Games: Text, Narrative and Play*. 1. vyd. Cambridge: Polity, 2006. ISBN 07-456-3401-X.

Docs.webplatform.org, 2014, *WPD* · WebPlatform.org. [online]. 2014. [cit. 31. prosince 2014]. Dostupné na

https://docs.webplatform.org/wiki/dom

DUCKETT, JON, *HTML & CSS.* 1. vydání Indianapolis, IN: Wiley, 2011, 512 s. ISBN 1118008189.

HICKSON, IAN a kol., 2014, *HTML5*. W3.org [online]. 2014. [cit. 31. prosince 2014]. Dostupné na http://www.w3.org/TR/html5/

JENSEN, G., ALTSCHUL, D., ERRACE, H. *Monkeys would rather see and do: preference for agentic control in rhesus macaques. Experimental Brain Research.* 2013, vol. 229, č. 3, s. 429-442. ISSN 0014-4819.

LARSEN, ROB, *Beginning HTML & CSS.* Indianapolis, Ind. : Wiley 2013, 864 s. ISBN 0470540702.

LIE, HÅKON WIUM A BOS, BERT, 2014, *Cascading Style Sheets, designing for the Web – Chapter 2: CSS.* W3.org [online]. 2015. [cit. 31. prosince 2014]. Dostupné na http://www.w3.org/Style/LieBos2e/enter/ Lunn, Ian, 2013, CSS3 foundations. Chichester, West Sussex : Wiley.

MACDONALD, M. *HTML5: The Missing Manual* 2. Vyd. Sebastopol, CA: O'Reilly, 2013. ISBN 978-144-9363-260.

MCFARLAND, DAVID SAWYER, *CSS3: The Missing Manua*l. Sebastopol, CA : O'Reilly, 2013, 650 **s.** ISBN 1449325947

MOZILLA DEVELOPER NETWORK, 2014, *ECMAScript 6 support in Mozilla*. [online]. 2014. [cit. 31. prosince 2014]. Dostupné na https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/ECMAScript_6_support_in_Mozilla

PRŮCHA, J., WALTEROVÁ, E., MAREŠ, J. *Pedagogický slovník*. 4. aktualiz. vyd. Praha: Portal, 2003, 322 s., ISBN 80-7178-772-8.

SALEN, K., ZIMMERMAN, E. *Rules of Play: Game Design Fundamentals*. Cambridge: The MIT Press, 2003. ISBN 0-262-24045-9.

SARRIS, SIMON, *HTML5 unleashed*. 1. vyd. USA, Pearson Education Inc., 2013, 432 s. ISBN 0672336278

SUITS, B. *The Grasshopper: Games, Life and Utopia*. 1. vyd. Peterborough, Ont.: Broadview Press, 2005. ISBN 1-55111-772-X.

TEAGUE, C. *CSS3: Visual QuickStart Guide*. 3. vyd. Berkeley, CA: Peachpit Press, 2011. 433 s. ISBN 03-217-1963-8.

TIŠNOVSKÝ, P. *Historie vývoje počítačových her.* [online] 10. 11 2011. [cit. 4. dubna 2014]. Dostupné na http://www.root.cz/clanky/historie-vyvoje-pocitacovych-her-1-cast-prvni-milniky/#ic=serial-box&icc=text-title.

WIKI.ECMASCRIPT.ORG, 2014, [*ES Wiki*]. [online]. 2014. [cit. 31. prosince 2014]. Dostupné na http://wiki.ecmascript.org/doku.php

WIKIPEDIA, 2015, *HTML5*. [online]. 2015. [cit. 1. ledna 2015]. Dostupné na http://cs.wikipedia.org/wiki/HTML5#cite_note-1

ZAKAS, N. *Professional JavaScript for Web Developers.* 3. vyd. indianapolis: Wrox, 2012. ISBN 1-118-02669-1.

Přílohy

A Obsah CD

Součástí této práce je také kompaktní disk obsahující následující přílohy v elektronické podobě:

- Zdrojové soubory arkádové hry
- Zdrojové soubory serverové aplikace