Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

HTML5 hra

2016

Petr Němeček

Vedoucí práce:
Mgr. Martin Trnečka

Studijní obor: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Petr Němeček
Název práce:	HTML5 hra
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2016
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Martin Trnečka
Počet stran:	34
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Petr Němeček
Title:	HTML5 canvas game
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2016
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Martin Trnečka
Page count:	34
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Výsledným produktem práce je akční, arkádová a webová 2D hra s prvky RPG a vesmírnou tématikou vytvořená za použití HTML5 Canvas technologie. Hráč řídí svého robota, ničí nepřátele a za získané zkušenosti si robota může vylepšit pomocí předmětů z herního obchodu. V každé misi se seznamuje se speciálním poselstvím, které mu hra předkládá.

Synopsis

The resulting product of this work is action, arcade and web-based 2D RPG game with space theme created using HTML5 Canvas technology. Player controls his robot, destroys its enemies and can upgrade through items in game store for experiences he acquires. Each level delivers some kind of message.

Klíčová slova: HTML5, canvas, hra

Keywords: HTML5, canvas, game

Děkuji vedoucímu bakalářské práce Mgr. Martinu Trnečkovi za úžasnou spolupráci a poskytnuté rady.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

1.1 Hra

Vždy jsem měl aspirace na to vytvořit hru. Bakalářská práce pro mě byla ideální příležitostí, jak se o to pokusit. Celý proces je nesmírně zábavný a je v něm velká svoboda. Jakou konkrétní podobu moje práce získá záleželo pouze na mě. Zároveň jsem se nespokojil s tím, že by hra měla mít jen povrchní charakter, bylo třeba ji dát nějaké silné charisma, originální příběh, díky kterému ji někdo bude chtít hrát. Koncept hry představím na začátku následující sekce 2.1.

1.2 Engine

Po ustanovení zásad pro vypracování jsem měl možnost psát hru v již existujícím herním frameworku. Tu jsem nevyužil, vydal jsem se vlastní cestou a všechno, co běžně tyto frameworky v základu nabízejí – přednačítání, manipulace se sprites, animace, ovládání, načítání map, pohyb, fyziku a systém pro detekci kolizí – jsem si vytvořil a implementoval sám. Získal jsem tímto přístupem absolutní ponětí o tom, co všechno je k takovému typu softwaru potřeba. Současně jsem během celého vývoje narážel na spoustu překážek a nepříjemností, výsledkem je však nezávislý projekt jehož potenciál jde rozšířit ještě dál, nejenom díky tomu, že lze hru z velké části upravit a rozšířit bez zásahu do zdrojového kódu. Všechny technické postřehy rozebírám více v sekci 4.

2 O hře

2.1 Koncept

The Journer of Karel neboli Karlova Cesta, jak jsem hru pojmenoval podle spisovatele Karla Čapka, který poprvé použil a popularizoval slovo robot ve své divadelní hře R.U.R., je zasazena do postapokalyptické doby po zániku lidstva, zničení naší planety, dosažení technologické singularity a tématicky se dotéká aktuálně stále více diskutovaného fenoménu umělé inteligence, kterým jsem osobně velmi fascinován.

Hlavním hrdinou příběhu je vesmírná sonda Karel, která byla vyslána ze své domovské planety autonomních robotů na planetu Zemi původně za účelem těžby, co však Karel nalézá nejsou ložiska vzácného prvku užívaného při výrobě samoopravovacího materiálu pro nové série robotů, ale artefakty, pozůstatky našeho světa v podobě autentických záznamů z reálných zdrojů, jakými jsou novinové články, knihy, hudba, dokumenty a další zmínky o událostech, které svým způsobem změnily svět navždy. Hra se tímto stává restrospekcí naší existence, poskytuje hráči nestereotypní gameplay a je přinejmenším nesnadné zařadit ji do nějakého žánru.

2.2 Artefakty

Artefakty pojednávají zejména o temných momentech lidstva (například počátek II. světové války, vývoj první atomové bomby, teroristické útoky z 11. září a na redakci satiristického časopisu Charlie Hebdo), rasových poměrech v naší společnosti (například zvolení prvního Afroamerického prezidenta Spojených států Baracka Obamy a proslov Martina Luthera Kinga), vývoji v informatice (například von Neumannova koncepce počítače, třídící algoritmus Quicksort Tonyho Hoare, představení operačního systému MS-DOS, Linuxu a počítače Macintosh), vědeckofantastických věcech (například divadelní hra Karla Čapka R.U.R. a film Matrix), myšlenek zabývajících se umělou inteligencí (například vytvoření disciplíny AI Johnem McCarthym, Turingův test, komentáře Stephena Hawkinga a Elona Muska), umění (například poezie Edgara Allana Poea, Shakespearova hra Romeo a Julie, obraz Leonarda da Vinci), hudbě (například skupina Queen, Beatles, Justin Bieber a Kurt Kobain) zmínky o politice a státních zřízení (například dokument Magna Carta alias základní kámen demokracie, Günther Schabowski a pád Berlínské zdi) a další (například první let bratří Wrightů, vypuštění Sputniku I na oběžnou dráhu, závěť Alfreda Nobela a popravení Saddáma Husseina).

2.3 Motivace

V každém z deseti levelů se cílem nyní stává nejen zničit pozůstalé nepřátelské existence, které Karlovi stojí v cestě, ale posbírat hlavně všechny artefakty na mapě. Artefakty jsou seskupeny podle určité souvislosti a každému levelu je přiřazen příběh, jehož znění se hráč dozvídá před jeho začátkem pomocí anotace. Každý artefakt má přiřazen jakými emocemi může svým příběhem na hráče působit. Získáváním těchto artefaktů se zvyšuje Karlův emoční level, což zároveň reflektuje emoční dojem samotného hráče. Herní databáze navíc disponuje hláškami, u nichž jsou uvedena kritéria na Karlův emoční level, jestli se Karel nachází na požadovaném levelu, smí tuto hlášku použít. V zájmu hráče je postupně zvyšovat tyto emoce, čím vyšší emoční úroveň Karel má, tím říká méně strojové a více lidské či dokonce slangové hlášky. Hra využívá model kategorizace emocí amerického psychologa Paula Ekmana do 6 základních.

Zabíjením nepřátel získává hráč odměny ve formě zkušeností, za které lze v herním obchodě zakoupit předměty zvyšující základní atributy jako zdraví, sílu a rychlost střelby a obranu.

2.4 Platforma

Hra běží ve webovém prohlížeči a je určena pro desktopová zařízení. K jejímu ovládání je zapotřebí klávesnice pro pohyb robota na mapě a myš pro střelbu.

3 Vizuální podoba

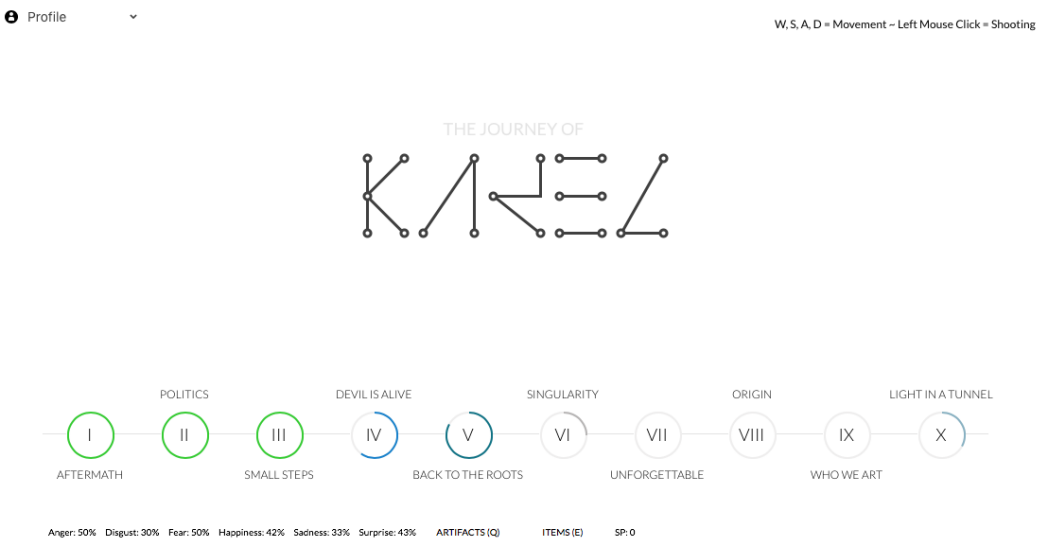
Veškerá herní grafická podoba je definována v ortogonálním 2D stylu. Všechny prvky, včetně map, obrázků robotů, předmětů z herního obchodu a artefaktů, jsou složeny z malých čtverečků o velikosti 32 pixelů, kterým se říká dláždice, respektive sprites v případě robotů a zmíněných předmětů.

Ačkoliv se kompozice mapy z čtverců může zdát velmi limitující, opak je pravdou. S trochou šikovnosti lze mapu uvést do podoby, kdy hráč nebude mít nejmenší tušení, že je hra tvořena ze čtverců skládaných pravidelně vedle sebe. Různé kombinace několika pár dláždic můžou ve výsledku znázornit stejný terén z několika různých perspektiv. Skrz tento fakt považuji grafiku za největší výzvu, která mi paradoxně zabrala nejvíc času. Nakonec jsem ovšem musel udělat kompromis a přenést se přes to, s jakým výsledkem to celé dopadlo.

Grafický návrh jsem realizoval ve vektorovém editoru Sketch na OS X. Výsledný produkt jsem mohl následně rasterizovat i do větších rozlišení pro zařízení, které disponují s vysokou hustotou pixelů na palec (dpi). U zařízení Apple je tato vlastnost známa pod komerčním názvem Retina.

3.1 Menu

Menu je první věc, se kterou se hráč setká. Po jeho načtení je mu nabídnut absolutní přehled na jedné obrazovce.



Obrázek 1: Herní menu.

Vlevo nahoře se nachází rozbalovací nabídka spojená s akcemi, které lze provádět s uživatelským profilem. Aktuálně se zde nachází možnost resetování profilu ve smyslu vynulování nasbíraných zkušeností, zakoupených předmětů a nasbíraných artefaktů.

Vpravo nahoře je hráč seznámen s herním ovládáním.

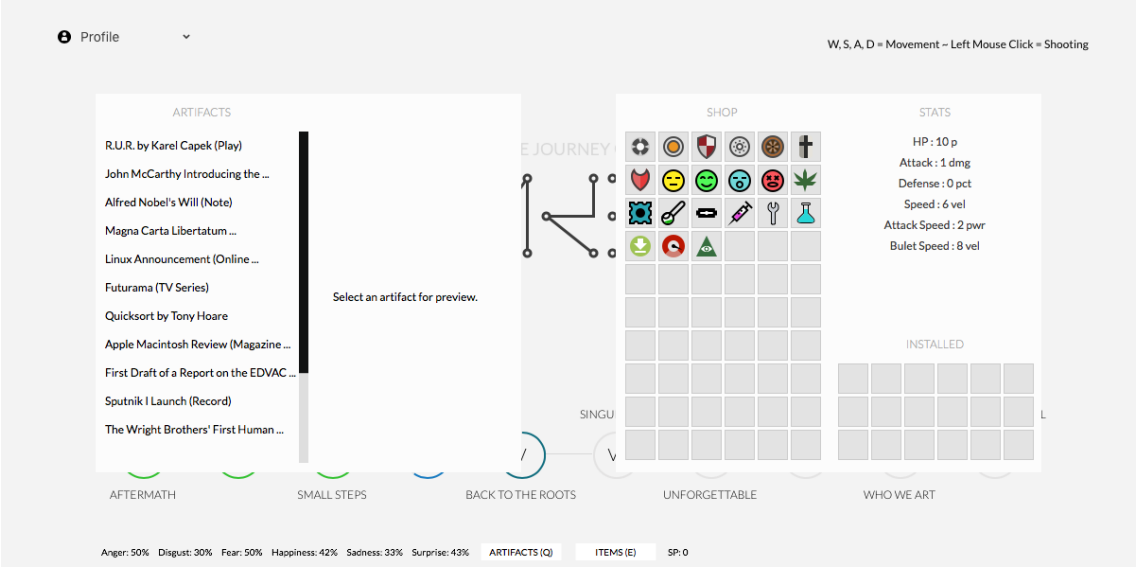
Uprostřed menu je umístěno jednoduché typografické logo. Slovo Karel je nápadně nastylováno, aby připomínalo elektrický obvod.

O trochu níže lze spatřit vylistovaných 10 základních misí, jejichž splnění je hráčovým cílem. Mise jsou rovněž vybaveny efektní a přehlednou vizualizací, radiálním pruhem, který prezentuje procentuální postup v dané misi, přesněji řečeno, kolik artefaktů hráč na dané mapě už posbíral. Každá mise má svůj název napovídající její charakter, který je naplno odkryt po najetí myší na vybranou misi, v tomto okamžiku se hráči zobrazí celý příběh této mise, kterou může po kliknutí začít hrát.

Na dolním okraji obrazovky je situovaný přehled o postupu hráče ve hře. Vedle tlačítek pro zobrazení artefaktů nebo herního obchodu jsou zobrazené informace o emočním stavu Karla a počet získaných zkušeností. Tento pruh překrývá nejen herní menu, ale i mapu, tím je zajištěn přístup k přehledu nad postupem ve hře odkudkoliv.

3.2 Inventář

Přístup ke všemu, co se herního postupu týče, lze přes herní komponentu, kterou jsem nazval inventář, buď pomocí tlačítek v přehledu herního postupu na spodním okraji obrazovky nebo vyznačenými klávesovými zkratkami.

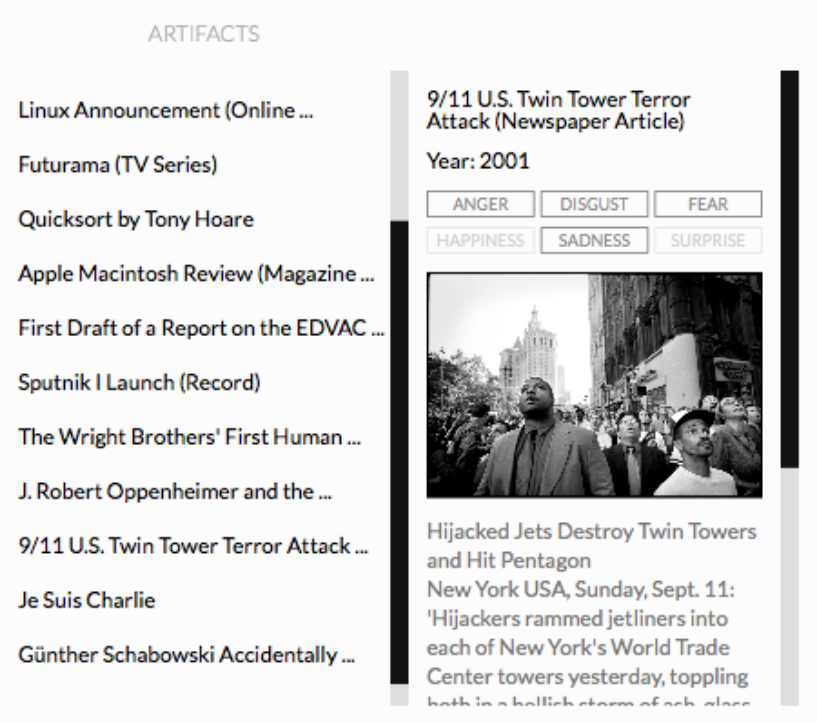


Obrázek 2: Herní inventář s přehledem získaných artefaktů (vlevo) a herním obchodem (vpravo).

Inventář je dostupný k zobrazení odkudkoliv, hlavním záměrem bylo totiž mít možnost prohlížet si nasbírané artefakty nebo nakupovat bez nutnosti být případně rozptýlen aktuálně probíhající misí na pozadí.

3.2.1 Artefakty

Levou stranu inventáře vyplňuje přehled získaných artefaktů. Jednoduché a kontrastní dvouslopcové rozvržení nejprve dává na výběr detail kterého artefaktu si hráč chce pohlédnout a poté jsou jeho informace zobrazeny vedle tohoto výběru.



Obrázek 3: Přehled získaných artefaktů s možností náhledu.

Titulek artefaktu doplňuje rok jeho události. Jestliže je u artefaktu i příloha ve formě obrázku, bude zobrazen její náhled. Náhled je možné zvětšit kliknutím (viz. obrázek 4). Nedílnou součástí jsou ovšem štítky, které podtrhují, jaký konkrétní emoční zážitek může hráč z této události mít. Štítky emocí, kterých se to týká jsou viditelně více výrazné než ostatní.

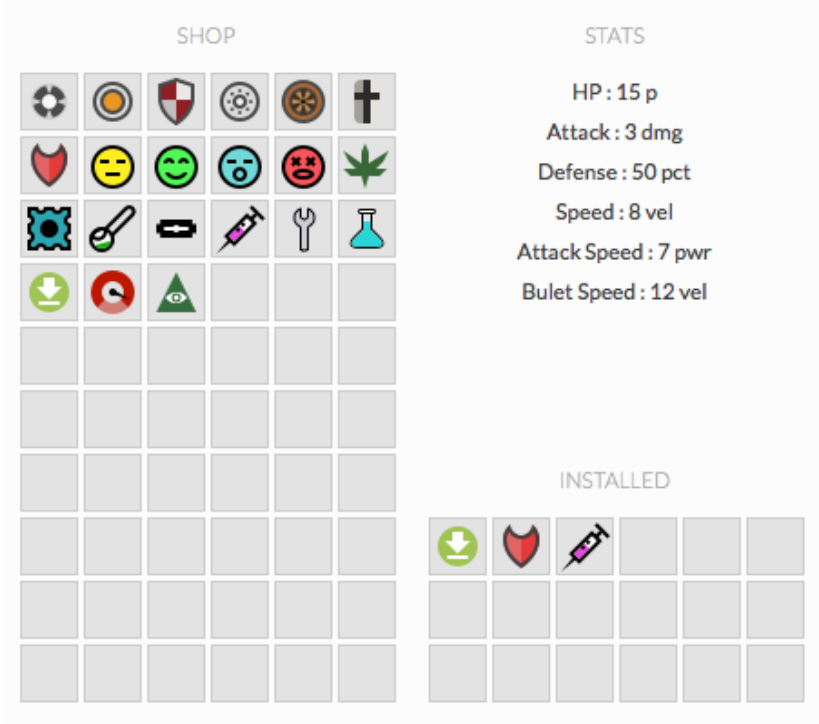
Pod štítky následuje krátká autentická zmínka spojená s událostí. Způsob, jakým jsem úryvky z různých zdrojů získával je spíše náhodný. Buď se jedná o událost, která je obecně známá, ale například už ne, jak se stala nebo je to událost, na kterou se už zapomnělo, přesto je však zásadní kvůli tomu, že ovlivnila společnost natolik, že se změnilo mnohé.

Výběr samotných artefaktů je rovněž tématicky přímo spojen s tématem hry.

3.2.2 Obchod

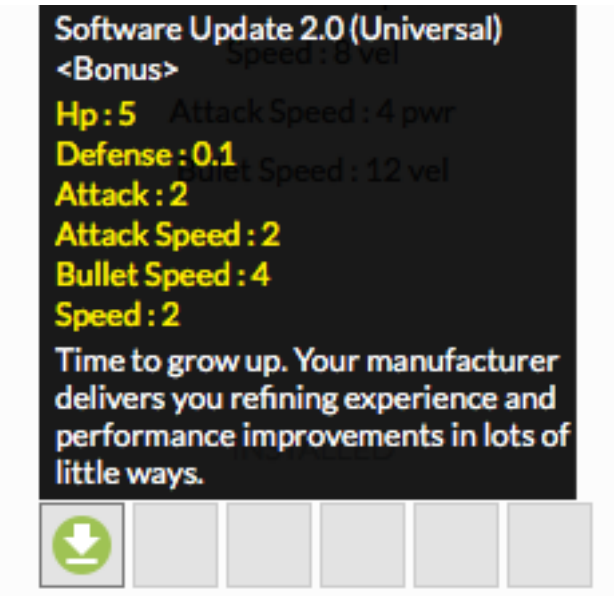
Jedním z elementů, které si hra osvojuje je možnost vylepšení robota Karla díky předmětům z herního obchodu, který je k nalezení na pravé straně v inventáři. Všechny dostupné předměty jsou přehledně vyskládány vedle sebe v dláždicovém uspořádání.

Obrázek 4: Zobrazení většího rozlišení obrázkové přílohy artefaktu.



Obrázek 5: Herní obchod s přehledem nainstalovaných předmětů a statistikou atributů robota.

Informace o žádaném předmětu může hráč získat najetím myší, tím se mu zobrazí box (viz. obrázek 6) s názvem předmětu, typem, cenou, vlastnostmi, které předmět vylepší a relevantním popiskem. Dvojím poklepáním myší si předmět zakoupí, v případě neúspěchu se hráči zobrazí hláška, která upozorní na nedostatek zkušeností nebo na to, že hráč už tento předmět nebo předmět stejného typu zakoupil.



Obrázek 6: Informace o předmětu z herního obchodu.

Podporovány jsou všechny předměty, které zvyšují robotův život, obranu, útok a rychlost pohybu, střelby a projektilu. Buďto předmět zvyšuje pouze jeden z těchto atributů anebo jsou podporovány i speciální předměty univerzálního typu, které zvýšují i více atributů najednou.

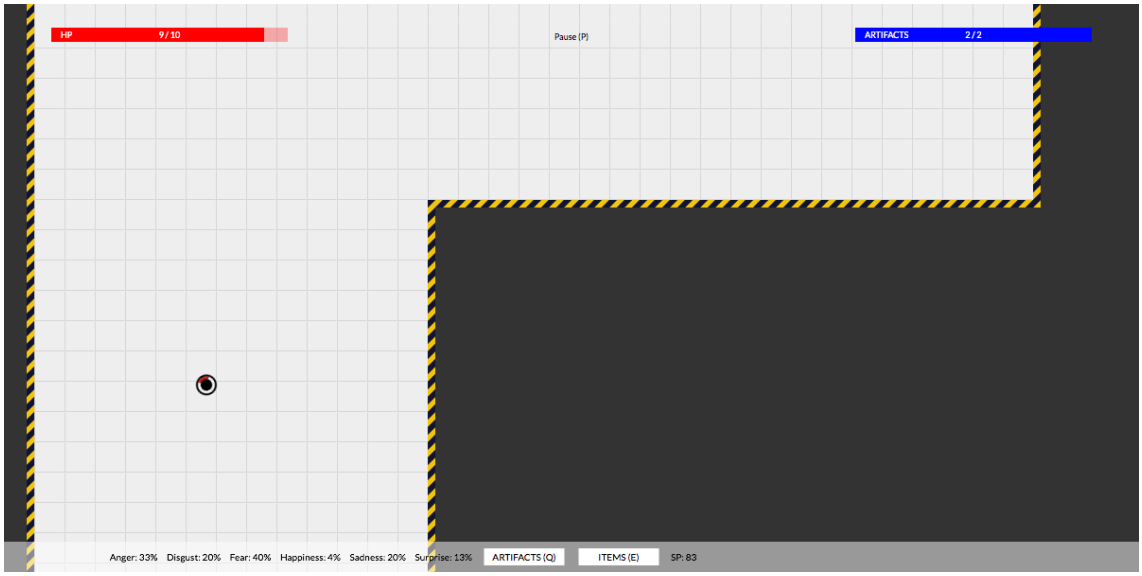
3.3 Mapa

Po vybrání mise, kteru chce hráč hrát a jejím načtení se zobrazí herní obrazovka. Z obrázku 7 lze vypozorovat, že kromě dolního pruhu inventáře a samotné mapy skládající se z vykreslených dláždic budu věnovat pozornost prvkům v horní části, stavovému řádku. Na levé straně je indikátor zdraví robota a na straně pravé je indikátor nasbíraných artefaktů, které jsou na aktuální mapě.

4 Technologie

4.1 Databáze

Srdce mé hry tvoří databáze. Obsahuje data o umístění tile a spritesheetů, mapách, artefaktech, robotech a hlášek. Soubor s těmito daty je ve formátu JSON a



Obrázek 7: Herní mapa.

nachází se v `/src/js/data.json`. Díky němu lze hru libovolně modifikovat či rozšířit bez větších znalostí a nutnosti zásahu do zdrojového kódu v JavaScriptu. V následující sekci je nastíněna jeho hierarchická struktura a popis vlastností.

tileset_src (cesty ke spritesheetům)

maps:

name – název mise

src – cesta k souboru

artifacts – artefakty k nalezení, tento výčet načítá herní menu pro zobrazení herního postupu

annotation – příběh vztahující se k misi a jejím artefaktům

artifacts:

name – titulní název události

year – rok, kdy se událost stala

emotions – výčet 6 emocí s nastaveným příznakem pravdivostní hodnoty, zda událost na hráče působí určitou emocí

message – příběh události

icon – orientační název pro přiřazení ikonu

sprite

offset_y – vertikální počáteční pozice spritu

offset_x – horizontální počáteční pozice spritu

frames

offset – výčet sprite framů od počáteční pozice

source – seznam zdrojů, ze kterých pocházejí informace

robots:

name – název robota

hp – počet životů

attack – kolik životů ubírá střela nepříteli při zásahu

defense – procentuální šance na vykrytí střely

speed – rychlost

bullet_speed – rychlost projektilu

attack_speed – počet projektilů, které může robot vystřelit za dobu 1s

sprite

offset_y – vertikální počáteční pozice spritu

offset_x – horizontální počáteční pozice spritu

frames

offset – výčet sprite framů od počáteční pozice

durations – trvání jednotlivých framů pro účely animace

collision

type – geometrický charakter kolizního objektu

bounds – rozměry a meze kolizního objektu

description – popis robota

items:

name – název předmětu

type – typ předmětu

id – identifikátor předmětu

bonus – výčet vlastností, které vylepšuje

price – cena předmětu

sprite

offset_y – vertikální počáteční pozice spritu

offset_x – horizontální počáteční pozice spritu

frames

offset – výčet sprite framů od počáteční pozice

description – popis předmětu

messages:

text – text hlášky

occassion – událost, při které se hláška vyvolá

emotions – procentuální výčet požadavků na určitý emocionální level, aby se hláška mohla vyvolat

4.2 JavaScript

Následující subsekce se věnuje výhradně úloze a funkcionalitě JavaScriptových zdrojových kódů. Nejdřív poskytne přehled všech vyskytujících se skriptů a jejich popis, poté bude následovat ukázkový výpis prototypů dvou hlavních objektů, mapy a inventáře. Poslání ostatních skriptů vysvětluje subsekce 4.5, která kód pro změnu popisuje z perspektivy řešených komplexních problémů při vytváření hry.

4.2.1 Zdrojové soubory

Tabulka 1: Rejstřík JavaScriptových zdrojových souborů.

Soubor	Funkce
asset_manager.js	Objekt: AssetManager(). Přednačítá a spravuje obrázky, zejména spritesheety a přílohy artefaktů.
AsyncIterator.js [**asynciterator**]	Asynchronní iterátor, který zrychluje proces načítání mapy a umožňuje provádět intenzivní smyčky jako neblokující operace, aby při jejich vykonávání nedošlo k zamrznutí prohlížeče a pádu aplikace.
data.json	Herní data.

entities.js	Objekty: AABB(), Obstackle(), Artifact(), Bullet() a Item().
game.js	Inicializační skript, který renderuje menu, spouští přednačítání obrázků, profilu, vytváří sprity robotů, předmětů a artefaktů.
inventory.js	Objekt: Inventory(). Obsluha inventáře, která renderuje přehled artefaktů a herní obchod.
jquery-2.2.3.min.js [jquery]	Závislost, kterou vyžaduje AsyncIterator.js. Jiné použití knihovna v projektu nemá, vše je psáno v čistém JavaScriptu.
map.js	Objekt: Map(). Načtení mapy a správa běhu celé hry.
ocanvas-2.8.3.min.js [ocanvas]	Knihovna pro objektovou nádstavbu nad HTML5 Canvas.
pako.min.js [pako]	Knihovna umožňující práci se zlib kompresí.
pako_inflate.min.js	Dekompresní metody rozšiřující knihovnu Pako.
pathfinding-browser.min.js [pathfinding]	Knihovna pro vyhledání nejkratší cesty k cíli.
profile.js	Objekt: Profile(). Spravuje, načítá a ukládá herní průběh do prohlížeče pomocí technologie HTML5 localStorage.
quadtree.js	Objekt: Quadtree(). Abstraktní stromová struktura sloužící pro určení, jaké objekty se mohou kolidovat.
radialIndicator.min.js [radialindicator]	Knihovna pro vytvoření radiálního ukazatele postupu daného levelu v menu.
robot.js	Objekty: Robot(), Player() a Enemy().
sat.min.js [sat]	Knihovna pro detekci kolizí.
utilities.js	Pomocné funkce a rozšiřující prototypy.
vector.js	Knihovna pro práci s vektory.

4.2.2 Map.js

Map.prototype.start()

Spouští herní smyčky. Nejdříve nastaví smyčku, ve které se vypočítává pohyb objektů a testují se kolize za pomocí volání `setTimeout()`, které zajišťuje neblokující průběh i pro ostatní část kódu a poté spustí renderovací smyčku

knihovny oCanvas, která se pro změnu ve vhodný čas stará o překreslení plátna s aktualizovanými objekty.

Map.prototype.update()

Provádí výpočet a aktualizuje polohy robotů, jejich projektilů a volá metodu pro kontrolu kolizí mezi nimi.

Map.prototype.render()

Volá překreslovací modul knihovny oCanvas.

Map.prototype.load()

Načítá data z mapy a inicializuje herní kameru, kvadrantový strom a mapu pro hledání cest o rozměrech odpovídajících načítané mapě. Rovněž spouští načítání tilesetů a dláždicové vrstvy.

Map.prototype.restart()

Při restartu mapy ji tato metoda vrací do původního stavu, odstraňuje vykreslené sprity i objekty vystřelených projektilů a zničených nepřátel, které vrací do tzv. fronty, kde čekají na vykreslení, až když se objeví na mapě. Navrací hráče na počáteční pozici a resetuje herní kameru.

Map.prototype.pauseGame()

Pozastaví hru, herní smyčku a popřípadě zobrazí menu s důvodem pozastavení.

Map.prototype.speaking()

Výběr a aktualizace pozice bubliny s hláškou. Robot může mluvit v určitých intervalech, poté se náhodně vybere hláška z databáze, kterou robot ještě neřekl. Testuje se i to, zdá má robot co říct ve smyslu vyčerpání zásoby hlášek a zda je emočně kvalifikován vybranou hlášku pronést.

Map.prototype.new()

Jakmile hráč hru zapne, vybere misi a začne hrát, vytvoří se prvně instance objektu Map, poté co hráč přepne nebo přejde na další misi, objekt mapy v paměti se zrecykluje pro nová data. Ostraní všechny vykreslené sprity a nastaví všem objektům hodnotu **`null`**, aby garbage collector vyčistil všechny instance z předchozí mapy a spouští metodu `Map.load()` pro načtení mapy z nových dat.

Map.prototype.showMenu()

Když se hra pozastaví, je možné v určitých případech zobrazit menu, o což se stará tato metoda. Stane se takto například, když si hráč vyžádá pozastavení herního průběhu, dojde ke zničení jeho robota nebo k nasbírání všech artefaktů na mapě. Menu je univerzální a podle důvodu pozastavení se zobrazují jen akce, které je možné v dané situaci provést.

Map.prototype.hideMenu()

Odstraní vykreslené menu.

Map.prototype.resetMenu()

Zobrazení všech možností zmíněného menu. Potřeba ho resetovat vzniká při restartu mapy nebo načtení nové mapy.

Map.prototype.createMenu()

Vytvoření grafických objektů herního menu.

Map.prototype.createStatusBar()

Vytvoření grafických objektů stavového panelu hry umístěného v horní části obrazovky, skládá se z indikátoru života na levé straně a indikátoru počtu nasbíraných a zbývajících artefaktů napravo.

Map.prototype.refreshStatusBar()

Při zásahu robota hráče, kolizi s jiným robotem nebo získáním artefaktu je potřeba aktualizovat indikátory statového panelu, o to se stará tato metoda.

Map.prototype.registerControls()

Registrace detekce událostí klávesnice a myši určené pro pohyb hráče na mapě a jeho střelbu.

Map.prototype.rotatePlayer()

Otáčí sprite robota hráče směrem ke kurzoru myši.

Map.prototype.moveBullets()

Aktualizace pohybu projektilů. Metoda je univerzální pro projektily hráče i nepřátel.

Map.prototype.createRobotCollider()

V herní databází jsou u robotů vepsány kolizní meze a geometrický charakter kolidujícího objektu. Tato metoda vytváří tento objekt při inicializaci každého robota na mapě.

Map.prototype.updatePlayerColliders()

Aktualizace pozice kolidujících objektů robota hráče.

Map.prototype.populateCollidersPool()

Vkládání kolidujících objektů do kvadrantového stromu.

Map.prototype.checkCollisions()

Kontrola kolizí různého typu. Hráče se stěnami, jeho projektilů se stěnami a testování, jestli hráč narazil na artefakt. Nepřátel s hráčem a jeho projektily a jejich projektilů se stěnami. Dochází i k testování, jestli má smysl, aby nepřítel střílel.

Map.prototype.loadTilesets()

Načtení tilesetů z dat mapy.

Map.prototype.loadTileLayer()

Načtení dláždicové vrstvy z dat mapy a vytváření instancí jejich spritů.

Map.prototype.loadObjectLayers()

Načtení objektových vrstev z dat mapy.

Map.prototype.loadArtifacts()

Načtení artefaktů z dat mapy a vytváření instancí jejich spritů.

Map.prototype.loadWalls()

Načtení stěn z dat mapy a vytváření kolidujích objektů.

Map.prototype.loadEnemies()

Načtení nepřátel z dat mapy, vytváření instancí jejich spritů a kolidujích objektů.

Map.prototype.removeVisibleTiles()

Odstranění vykreslených dláždic. Využívá se při restartu mapy.

Map.prototype.addInitTiles()

Po vypočtu, které dláždice jsou viditelné a má smysl je vykreslovat, se nakonec vykreslují touto metodou.

Map.prototype.loadPlayer()

Načtení informace o počáteční pozici hráče pro pozdější nastavení kamery a vytváření kolidujích objektů.

Map.prototype.calculateVisibleTiles()

Nastavení indexů mezí viditelných dláždic do objektu herní kamery.

Map.prototype.adjustCamera()

Po načtení hráče je známa jeho počáteční pozice, poté je spuštěna tato metoda, který doupraví herní kameru a jeho pozici.

Map.prototype.addTiles()

Odstraňuje resp. přidává sprity sloupce resp. řádky dláždic.

Map.prototype.updateTiles()

V případě, že dojde k posunutí mapy je metoda, aby zjistila, jestli je nutné přidat resp. odebrat sloupce resp. řádky dláždic.

Map.prototype.updateArtifactsPosition()

Aktualizuje polohu spritu artefaktů.

Map.prototype.updateTilesPosition()

Aktualizuje polohu spritu viditelných dláždic.

Map.prototype.updateEnemiesPosition()

Aktualizuje polohu spritu nepřátel.

Map.prototype.isWithinBounds()

Sprity veškerých objektů se vykreslují až tehdy, jsou-li na mapě viditelné čili jsou v záběru herní kamery, jedině tehdy se aktualizuje jejich relativní pozice na plátně všemi výše zmíněnými metodami.

Map.prototype.addEnemies()

Všichni nepřátelé čekají na vykreslení v tzv. frontě do té doby, než tato metoda zjistí, že jsou v záběru herní kamery a měli by být vykresleni.

Map.prototype.moveEnemies()

Aktivním útočícím nepřátelům viditelných na mapě je aktualizována pozice spritu a kolidujích objektů. Metoda se mimi to stará o výpočet trasy nepřítele k hráči a jejich střelbu.

Map.prototype.shoot()

Vytváření instance projektilu, spritu, nastavení jeho dráhy a rychlosti.

Map.prototype.move()

Pohyb robota hráče o aktuální vektor, případně celé mapy, jestliže se hráč nachází na hranici zadaných mezí. To způsobí přenastavení kamery, aktualizaci dláždic, nepřátel, artefaktů a kolidujících objektů.

Map.prototype.getTilePacket()

Při vytváření spritu dláždice je nutné touto metodou na určité pozici na tile nebo spritesheetu vyhledat výřez pro zadanou dláždici touto metodou.

4.2.3 Inventory.js

Inventory.prototype.toggleInventory()

Způsob, jakým je inventář proveden je velmi zajímavý. Technicky se jedná o samostatné Canvasové plátno o výšce jen několik desítek pixelů umístěné v herní obrazovce dole. Dvojice tlačítek, která jsou na něm zobrazena jsou zodpovědná za zobrazení tohoto inventáře. V momentu, kdy se inventář zobrazí, zaplní celé okno herní obrazovky, jedná se o takový efekt maximalizace. Tato metoda má na starost přepínínání jeho zobrazení a skrytí.

Inventory.prototype.shrinkShopInventory()

Při skrytí resp. minimalizaci inventáře dojde technicky ke zmenšení Canvasového plátna inventáře. Plní-li hráč misi, hra bude pokračovat.

Inventory.prototype.expandShopInventory()

Při zobrazení resp. maximalizaci inventáře dojde technicky ke zvětšení Canvasového plátna inventáře. Plní-li hráč misi, hra se pozastaví.

Inventory.prototype.refreshStatusBar()

Aktualizace informací umístěných na informačním pruhu inventáře.

Inventory.prototype.createControls()

Vytvoření stavového pruhu inventáře.

Inventory.prototype.createCaption()

Pomocná funkce k vytvoření nadpisu sekcí inventářových oken.

Inventory.prototype.createContainer()

Pomocná funkce k vytvoření inventářových oken.

Inventory.prototype.createScrollbar()

Canvasové plátno jsou samé pixely. Mám-li nějaký obsah, který vyžaduje efekt skrolování, musím si ho v Canvasu implementovat sám. Tato metoda mi vytváří jeho grafický podklad.

Inventory.prototype.updateScrollbar()

Efekt skrolování je vytvořen za pomocí simulace odsazení skrolovacího elementu, kterým udávám odsazení obsahu, který má být skrolován.

Inventory.prototype.refreshStats()

Dojde-li v herním obchodě k nákupu, je třeba aktualizovat atributy hráče zahrnující bonus z právě nakoupeného předmětu.

Inventory.prototype.createArtifactEntity()

Pomocná funkce pro vytvoření položky seznamu artefaktů.

Inventory.prototype.refreshArtifactsList()

Aktualizuje list artefaktů po přidání nového či při skrolování listem.

Inventory.prototype.previewArtifact()

Metoda zodpovědná za zobrazení informací o artefaktu.

Inventory.prototype.createArtifactsPreview()

Vytvoří okno pro přehled artefaktů.

Inventory.prototype.createTileFrame()

Pomocná metoda při vytváření mřížky herního obchodu a nainstalovaných předmětů.

Inventory.prototype.createShop()

Vytvoří okno herního obchodu.

4.3 Tiled Map Editor

Když jsem přemýšlel o tom, jakým způsobem budu sestavovat mapu, věděl jsem, že bude potřeba nějaký editor, nakonec jsem sáhnul po Tiled. Jednoduchý a účinný open-source multiplatformní software pro tvorbu dláždicových map izometrického, hexagonálního a v mém případě i ortogonálního typu. Umožňuje vyvtvořit libovolně velkou mapu s libovolně velkými dláždicemi.

Tiled vytvořené mapy dovoluje ukládat v různých formátech, jakými jsou JSON, XML nebo TMX. Poslední z nich, jak jsem si všiml, používají ve velkém herní frameworky, které ho umí automaticky parsovat. Pro mé účely jsem zvolil formát JSON, protože používá stejnou syntaxi jako JavaScript, ve kterém je hra napsána.

Tiled ještě nabízí v souboru mapy dláždicové vrstvy zakódovat do base64 a zkomprimovat pomocí gzip nebo zlib (tu jsem si vybral), tyto vrstvy jsou v základu ukládány jako CSV (hodnoty oddělené čárkami), vytvářím-li mapu o rozměrech několika set dláždic, velikost souboru vzrostě i na několik MB, což mi přijde nepřijatelné. V takovém případě vůbec nevadí, že jsou dláždicové vrstvy v souboru po kompresi nečitelné, kdokoliv by mapu chtěl editovat, měl by to provést v samotném editoru.

Mapa každé mise si uchovává svou specifickou strukturu, kterou jsem jim přidělil a kterou popisuji v tabulce č. 2.

Tabulka 2: Struktura vrstev map.

Vrstva	`Typ`	Popis
player	`objectgroup`	objekt playerspawn pro určení počáteční pozice hráče na mapě
enemies	`objectgroup`	objekty s id nepřátel z herní databáze
artifacts	`objectgroup`	objekty s id artefaktů z herní databáze
walls	`objectgroup`	objekty pro kolizní systém
floor	`tilelayer`	dláždice terénu mapy

4.4 HTML5 Canvas

Hra je vytvořena za použití HTML5 Canvas technologie a JavaScriptu. Je možné ji hrát v jakémkoliv webovém prohlížeči podporující Canvas API, které má velkou podporu ve všech hlavních webových prohlížečích už několik dlouhých let, zejména i v nepopulárním Internet Exploreru a to již od verze 9, který byl vydán před pěti lety. V takovém případě mi Canvas nabízí možnost udělat plně multiplatformní hru bez žádných kompromisů.

4.4.1 oCanvas

Na začátku tohoto textu jsem zmínil, že jsem hru nepsal v žádném herním frameworku. U kreslení na Canvasové plátno jsem udělal vyjímku, využil jsem knihovny oCanvas a mám pro to tuto technologickou volbu řádný důvod. Jednak je taková knihovna potřeba, jedná se totiž o objektovou nádstavbu nad plátnem – místo pixelů se pracuje s objekty a to se využívá například pro rozpoznávání událostí prováděných myší a druhým podstatným důvodem proč jsem si dovolil použít tuto knihovnu bylo to, že jsem přímo s HTML5 Canvas API již pracoval. Mám už zkušenosti s tím, jakým způsobem se s plátnem pracuje a jak se na něj kreslí, použitím knihovny tedy o nic nepřicházím. oCanvas je velice

příjemná knihovna se spoustou funkcionality. Zvolil jsem si ji na základě doporučení mého vedoucího práce. Jejím autorem je Švéd Johannes Koggdal, webový vývojař, který aktuálně pracuje pro hudební službu Spotify ve Stockholmu. Jako uživatele jeho knihovny mě zajímalo, jakým způsobem s ním správně pracovat. Neváhal jsem ho konktaktovat skrze email, který uvádí na stránce projektu oCanvas. Naše komunikace se datuje od října minulého roku, kdy jsem začal zjišťoval, jak lze korektně grafiku vykreslovat na zařízeních s větší hodnotou ppi (pixely na palec). Odpověděl obratem, byl velmi potěšen, že učitelé o jeho projektu mluví a poradil mi se vším, na co jsem se ho zeptal. Nicméně naše emailová komunikace pokračovala dále a během celého mého vývoje jsme si vyměnili více než tři desítky zpráv.

Zpětná vazba

Johannes byl neuvěřitelně vstřícný a zatímco on mi poskytoval rady, jak oCanvas používat, já mu poskytoval zpětnou vazbu. Na moje podněty byla dokonce knihovna dvakrát aktualizována a opravena její dokumentace. První problém, na který jsem narazil, bylo při vytváření a klonování sprite instancí. V této třídě šlo vlastností `image` předat pouze cestu k obrázku jako řetězec a předání obrázku jako `HTMLImageElement` nefungovalo. V prvním případě se oCanvas nedíval do cache prohlížeče, zda už je obrázek načten, ovšem já si všechny obrázky přednačítám sám a navíc je mám k dispozici už jako zmíněný `HTMLImageElement`. Johannes kýženou chybu opravil v nové verzi oCanvas 2.8.2. Další věc, na kterou jsem ho upozornil, bylo nepotřebné překresování celého plátna při načítání obrázku po vytvoření instance sprite, který dramaticky zpomaloval výkon celé hry, Johannes to opravil a vydal verzi 2.8.3. K uvedené opravě dokumentace došlo v momentu, kdy jsem náhodou přes vyhledávání v Googlu našel, že oCanvas umožňuje vykreslovaným entitám nastavit vlastnost `clipChildren`, díky které se nezobrazují přetékající části podřízených objektů. Jedinná zmínka o této funkci byla na stránce seznamu změn u vydání verze 2.8.0, nikoliv v dokumentaci. Jsem rád, že jsem to odhalil, jelikož jsem tuto vlastnost potřeboval. Poté jsem napsal Johannesovi další email a ten ji přidal do dokumentace.

4.5 Postupy

Vedle programátorské dokumentace, která popisuje úlohu jednotlivých objektů a jejich metod, je potřeba věnovat se některým postupům ve větším rozsahu a tomu se takto stane v této kapitole. Vysvětlím celý problém i s uvedením řešení, které jsem na ně aplikoval. Kapitola bude uspořádána chronologicky, postupně se dostanu od načítání hry a map až ke koliznímu systému.

4.5.1 Načítání hry

Základní kámen hry jsem položil napsáním konstruktu, který hru načte a zobrazí. O tyto věci se výhradně stará obsluha zavedená v skriptu `map.js`. Ze všeho

nejdříve se musí načíst databázový soubor s herními daty `data.json`, po jeho načtení se z něj zjistí, jaké obrázky se musí přednačíst, aby se nemuseli načítat v průběhu hry. Takové obrázky zahrnují všechny spritesheety a přílohy všech artefaktů. Pomáhá mi při tom objekt `AssetManager`, který vytvořím a naplním jeho frontu požadavky na stažení. Kromě toho funguje i jako poskytovatel těchto obrázků, uchovává si jejich reference.

Až je vše načteno, callback vyvolá inicializační funkci `init()`, která připraví Canvasové plátno k použití a postará se o vytvoření tzv. sprite prototypů robotů, předmětů z herního obchodu a artefaktů. Díky dobře a univerzálně navržené databázi může sprity všech typů načíst jedinná funkce `loadSprites()`.

Důvodem, proč tyto prototypy, jak jsem je nazval, vytvářím již při načítání hry je, že už následně při načítání mapy mám u spritu každého typu vytvořené jeho framy, které slouží k případné animaci, už mi jej stačí pouze naklonovat v potřebném počtu. Tyto framy mi generuje šikovná funkce `genSpriteFrames()`.

Profil

Nesmím zapomenout na načtení profilu hráče. Pomocí technologie HTML5 localStorage API jsou přímo v datech webového prohlížeče ukládány informace o jeho herním průběhu, tedy nasbíraných artefaktech, zkušenostech, nainstalovaných předmětů z herního obchodu a vyřčené hlášky. Vše je ukládáno v textové podobě jako klíč a hodnota. API je velmi jednoduché a účinné, obsluhu profilu zaobstarává objekt `Profile`.

4.5.2 Načítání mapy

Herní mapu jsem skládal v editoru Tiled, kterému jsem se věnoval v subsekci 4.3. Záležitostí, se kterou jsem se musel vypořádat bylo, jak tuto mapu v kódu načíst a vykreslovat z ní dláždice.

Jakmile si hráč vybere misi, kterou chce hrát, spustí se načtení relevantního JSON souboru a dojde funkcí `createMap()` k vytvoření instance mapy, jejíž konstruktor automaticky začne načítat data a sestavovat z nich mapu. V případě, že instance mapy už v paměti existuje, nevytváří se nová, nýbrž se recykluje a nahrávají se do ni data nová. Od tohoto momentu se vše odehrává v souboru `map.js`. Prototyp objektu mapy je detailně popsán v subsekci **??**.

Velké mapy a intenzivní iterace

Proces načítání mapy bylo ovšem nutné později přehodnotit, kvůli zjištění, jak některé prohlížeče reagují na to, když se zpracovává velká mapa. Drtivou část vývoje hry jsem strávil na testovací mapě o rozměrech pouhých 50x50, tj. pro zjednodušení 2 500 iterací pro jednu dláždicovou vrstvu. K mému nevyhnutelnému překvapení došlo při zvětšení na 500x500, tj. najednou 250 000 iterací, což se může zdát příliš, pro počítač by to však problém neměl být. Ve skutečnosti

je problém ukryt jinde. Prohlížeč Safari na OS X zvládal tyto situace bravůrně, avšak testováním na jiných prohlížečích a zejména na systému Windows se projevila nedostatečně provedená optimalizace načítání. Nejenom, že v ostatních prohlížečích byl průběh dramaticky pomalejší, ale po delší době, zhruba půl minutě, prohlížeč zamrzne a stránka s hrou spadne. Je to způsobeno tím, že bylo načítání blokující operací, které plně zaměstnalo hlavní JavaScriptový UI vlákno, a protože mělo dlouhého trvání, prohlížeč si myslel, že stránka neodpovídá a vyvolal její pád.

Tuto nepříjemnost jsem se pokusil vyřešit zavedením asynchronního iterátoru, který vnitřně využívá metodu `requestAnimationFrame()`, ta je zodpovědná za opakované volání zadané události. Přesně v tom tkví klíč. Iterátor díky němu simuluju asynchronní vykonávání a odlehčuje tím hlavnímu vláknu. Další výhodou je, že lze iterace rozdělit na dávky, v mém případě jsem vytváření dláždic rozdělil po 5000 průchodech. Než se navíc začne vykonávat další dávka, nestanu se tomu okamžitě, ale čeká se na prohlížeč, kdy k tomu bude vhodná příležitost a tím by nemělo dojít k tomu, že prohlížeč stránku shodí.

Každopádně jsem vysledoval u prohlížeče Google Chrome, že ze záhadného důvodu nezvládne více než 80 tisíc iterací, což je v případě příliš velkých map nedostatečné. Inspirací pro tuto hru byly prvotně tituly, které rozdělovaly mapu na místnosti. Každá místnost je přesně stejně velká jako herní okno a jak hráč přechází do jiných místností, načítá se jejich prostředí, až při vstupu do nich. Tento přístup je jednodušší a dá se ním těmto problémum lehce předejít, jenže já se rozhodl pro zavedení volně průchozích map, zdá se mi to lepší.

4.5.3 Render mapy

Vykreslování grafiky zajišťuje oCanvas pomocí smyčky. Objekty, které má aktuálně ve svém kontejneru pro překreslení kreslí do plátna při 60fps. Takový moment nastane pouze jednou za celý průchod, při vkládání objektu do překreslovacího kontejneru je vypnuta výchozí značka pro překreslení celého plátna.

V kódu je zavedena široká řada optimalizací různého druhu pro co nejplynulejší herní průběh. Neustále se kontroluje, zda nepřítele, artefakt nebo dláždici zabírá herní kamera a má smysl se pokusit o vykreslení jejich spritů. Není samozřejmě problém se o to u všech objektů neustále pokoušet, dochází ale ke značným ztrátám ve výkonu hry.

Příznačný podíl na optimalizaci v běhu má technika vykreslování dláždic, jakožto objektů, kterých je vždy na mapě zobrazováno nejvíce. Využívá se k tomu záběr herní kamery. Jestliže je znám obdelníkový výběr kamery a rozměry dláždic, lze z toho metodou `Map.calculateVisibleTiles()` snadno a s dokonalou přesností vypočítat index počátečního resp. koncového sloupce resp. řádku dláždic. Vykreslovat a aktualizovat souřadnice se budou jenom těmto, ani o jednu dláždici, kterou není vidět navíc. Jakmile se dojde k závěru, že se herní mapa posunula, metoda `Map.updateTiles()` zkontroluje, jestli je nutné odstranit resp. přidat nové sloupce resp. řádky dláždic.

4.5.4 Systém boje

Hráč posunuje svého robota po mapě standartními kombinacemi kláves W, S, A a D. Jeho nepřátelům se k němu vypočítává nejkratší cesta pomocí algoritmu A* (A star) v knihovně `PathFinding.js` [**pathfinding**].

Kliknutím na Canvasové plátno se střílí, popřípadě lze stisktnuté levé tlačítko myši držet zmáčknuté pro kontinuální střelbu, která vysílá projektily jen tolikrát za sekundu, kolikrát udává součet základní rychlosti střelby a bonusy z předmětů z herního obchodu.

Hráč i jeho nepřátelé mají zavedenou hodnotu pro obranu. Ta udává procentuální šanci, že projektil, který robota zasáhne nezpůsobí žádné poškození.

Projektily nejsou jedinou možností, jak může být hráčův robot poškozen. Dojde-li k přímému střetu mezi hráčovým a nepřátelským robotem, bude hráči způsobeno poškození ve stejné míře, jako by ho zasáhl nepřítelův projektil. Zároveň bude nepřítel efektně odhozen opačným vektorem jeho pohybu od hráče.

4.5.5 Kolizní systém

Systém kolizí ve dvourozměrném prostoru se dá provést nejrůznějšími metodami. Separating Axis Theorem (SAT) [**sat_theorem**] byl pro mou hru nejvhodnějším kandidátem. Využívá vektorové matematiky a zjišťuje, zda je mezi vybranými dvěma objekty oddělující mezera, hned jakmile zjistí její existenci, prohlásí objekty za nekolidující, v opačném případě na konci testování prohlásí objekty za kolidující. SAT funguje pouze pro konvexní polygony, což v mém případě nevadí. U testování mezi polygonem a kruhem je se využívá Voronoi regionů [**voronoi**]. Původně jsem se snažil o hlubší pochopení této problematiky a vlastní implementaci algoritmů, řešení bylo nakonec mírně nespolehlivé a proto jsem sáhnul po knihovně `sat.js`.

Po této technologické volbě bylo ještě nutné systém dotáhnout do konce. Přemýšlel jsem nad tím, jakým způsobem se bude kontrola kolizí provádět. Ve hře se budou vyskytovat roboti, projektily a zdi, jenže ne všechny typy kolizí dávají smysl, tudiž je nemá cenu testovat. Například nebudu hru zatěžovat kontrolou kolize mezi dvěma stěnami, mezi dvěma napřáteli nebo jejich dvěma projektily, když navíc prohlásím, že jsou navzájem imunní. Ani nebudu testovat střet objektů, které jsou od sebe vzdálené půl mapy. V tom mi pomohl kvadrantový strom implementovaný ve skriptu `quadtree.js`. Slouží k rekurzivnímu rozdělení dvourozměrného prostoru na čtyři kvadranty. V každém takovém kvadrantu nebo-li regionu se nachází určitý počet objektů, u nichž má smysl navzájem kontrolovat jejich kolize [**quadtree**]. Kvadrantový strom bylo potřeba přinutit spolupracovat se SAT z přechozího odstavce. SAT sice umožňuje vytvářet polygony a kruhy, ale to bylo z hlediska pro pozdější práci s kvadrantovým stromem nedostatečné, musel jsem si tedy ještě napsat nad SAT objekty abstrakci, kterou obsluhuje skript `collider.js`, v němž jsou objekty, které pracují s objekty, které vytváří knihovna SAT. Hlavním důvodem bylo nad objekty SAT zavést například metodu `Collider.getAABB()`, ta vytváří obdelníkový obal objektu, aby se

podle jeho rozměrů a pozice dal objekt korektně zařadit do správného regionu v kvadrantovém stromu. Musel jsem se dále postarat o posun, umístění a rotaci objektů, SAT knihovna mi zajišťuje už jen samotnou detekci. Kvadrantový strom se konstruuje v každém aktualizačním průchodu, přesto nedochází ke spomalení zabýváním se nesmyslných kolizí.

Závěr

Výsledkem práce je akční, vesmírná a vzdělávací hra The Journey of Karel a vlastní herní engine, který se zrodil při její výrobě. Hra zaujme originálním konceptem a poutavým příběhem, díky kterému není hráčovým cílem pouze stereotypní ničení nepřátel.

Hra je z velké části modifikovatelná. Herní databáze dovoluje bez zásahu do zdrojového kódu změnit a přidávat roboty, artefakty a hlášky. V mapovém editoru Tiled lze upravit i herní mise, například po přidání nových dláždic do setu i grafický vzhled. Bylo zajímavé setkat se s novými postupy a řešit netriviální problémy při její tvorbě. Jednu z věcí, kterou bych si přál na práci zlepšit je grafický vzhled a mít více času na to udělat lepší mapy.

Vyzkoušel jsem si celý proces programování hry. Ať už se jedná o navržení konceptu, designu grafické části nebo level-designu, což je úžasné z toho hlediska, že v praxi na těchto jednotlivých částech pracují i celé týmy. Prohloubil jsem své znalosti programovacího jazyka JavaScript, který je velmi perspektivní a vzniká pro něj stále více pracovních pozic, protože na web, jako platformu, se kterou je nejvíc spojován, se pomalu přesouvá veškeré dění.

Conclusions

The result is an action, space and educational game The Journey of Karel and custom game engine I made at the same time. The game attracts players thanks to its original concept and an engaging storyline so it's not only about stereotypical destroying enemies.

The game is largely modifiable. It allows to add robots, artifacts and one-liners easily via game database. Even maps can be edited in the Tiled Map Editor. It was very interesting to meet new procedures and solve nontrivial problems while creating it. One of the things that I wanted to work on more is to improve the graphical appearance and have more time to make better maps.

I've tried the whole process of programming the game. Whether it is making a concept and graphic design or level-design which is amazing because in reality there are groups of people specialized on each part mentioned. I improved my knowledge of JavaScript programming language which is very perspective and there are more and more jobs being created for it thanks to Web, its main domain.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové kódy hry se všemi potřebnými (příp. převzatými) zdrojovými kódy, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu / adresářové struktury pro zkopírování na webový server.

src/css
: Soubor STYLE.CSS, který styluje vzhled a pozice herního okna v HTML stránce.

src/index.html
: Stránka s hrou. Otevřením tohoto HTML souboru se uživateli hra načte.

src/img
: Grafické prvky hry včetně ikon herního menu, setů dláždic, obrázkových příloh artefaktů a soubor k jejich úpravě ve formátu Sketch.

src/js
: Zdrojové JavaScriptové texty včetně všech použitých knihoven.

src/maps
: Mapy ve formátu JSON upravitelné v editoru Tiled.

readme.txt
: Instrukce pro lokální spuštění hry.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Bibliografie

[1] HORN, Benjamin: GitHub – beije/AsyncIterator: An asynchronous iterator that uses requestAnimationFrame to batch process large datasets without locking up the UI. [online]. 2016. [cit. 2016-04-28]. Dostupné z: https://github.com/beije/AsyncIterator

[2] The jQuery Foundation: jQuery [online]. 2005-2016. [cit. 2016-04-28]. Dostupné z: https://jquery.com

[3] KOGGDAL, Johannes: oCanvas – Object-based canvas dawing [online]. 2016. [cit. 2016-04-28]. Dostupné z: http://ocanvas.org

[4] NODECA: GitHub – nodeca/pako: high speed zlib port to javascript, works in browser & node.js [online]. 2016. [cit. 2016-04-28]. Dostupné z: https://github.com/nodeca/pako

[5] XU, Xueqiao: GitHub – qiao/PathFinding.js: A comprehensive path-finding library for grid based games [online]. 2011-2012. [cit. 2016-04-28]. Dostupné z: https://github.com/qiao/PathFinding.js

[6] YADAV, Sudhanshu: GitHub – s-yadav/radialIndicator: A simple and light weight circular indicator / progressbar plugin. [online]. 2015. [cit. 2016-04-28]. Dostupné z: https://github.com/s-yadav/radialIndicator

[7] RIECKEN, Jim: GitHub – jriecken/sat-js: A simple JavaScript library for performing 2D collision detection [online]. 2012-2015. [cit. 2016-04-28]. Dostupné z: https://github.com/jriecken/sat-js

[8] WIKIPEDIA: Hyperplane separation theorem [online]. 2015. [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Hyperplane$_{s}eparation_{t}heorem$

[9] WIKIPEDIA: Voronoi diagram [online]. 2016. [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Voronoi$_{d}iagram$

[10] WIKIPEDIA: Quadtree [online]. 2016. [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Quadtree