VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D HRA WEBGL S FYZIKOU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE DAVID MAIXNER
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D HRA WEBGL S FYZIKOU

3D WEBGL GAME WITH PHYSICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE DAVID MAIXNER
AUTHOR

VEDOUCÍ PRÁCE Ing. HORVÁTH ZSOLT
SUPERVISOR

BRNO 2013

Abstrakt

Tato bakalářská práce diskutuje možnosti vytváření her pro webové prohlížeče. Zaměřuje se především na tvorbu 3D webové grafiky pomocí WebGL. Výbírá frameworky pro zjednodušení práce s WebGL a fyzikou. Dále se zabývá návrhem a implementací 3D webové hry s fyzikou s využitím vybraných frameworků. Nakonec se zabývá testováním aplikace.

Abstract

This bachlor thessis discusses the possibility of creating games for web browsers. It focuses on making of 3D web graphics using WebGL. This thessis chooses frameworks for easing up the work. The thessis also describes design and implementation of 3D web game with physics with use of selected frameworks. Finally it discusses testing of the application.

Klíčová slova

WebGL, Webová hra, Fyzika, Physi.js JavaScript, HTML5

Keywords

WebGL, Web game, Physics, Physi.js JavaScript, HTML5

Citace

David Maixner: 3D hra webGL s fyzikou, bakalářská práce, Brno, FIT VUT v Brně, 2013

3D hra webGL s fyzikou

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Horvátha Zsolta

. . . . . .
David Maixner
15. května 2013

Poděkování

Děkuji vedoucímu mé práce panu Ing. Horváthovi Zsoltovi za jeho čas, pomoc a odborné rady, které mi věnoval při řešení této práce.

Obsah

Kapitola 1

Úvod

S rozvojem internetu a jeho specifikací se webový obsah stává poutavějším a interaktivnějším. Původní dvourozměný obsah začíná být pomalu obohacován o 3D doplňky. Jednou z možností tvorby 3D webového obsahu je WebGL. Jde o multiplatormní JavaScriptové API odvozené z OpenGL. Tato technologie nám umožňuje mimo jiné i tvorbu grafů matematickýh funkcí, zobrazování 3D map a samozřejmě i tvorbu her. To vše v prostředí webového prohlížeče. WebGL vyniká harwarovou akcelerací aplikací na grafické kartě, což nám dovoluje zobrazovat ve webovém prohlížeči náročnou grafiku, dosud používanou pouze v instalovaných aplikacích. A to vše bez jakéhokoliv zásuvného modulu.

Cílem této práce je navrhout a implemetovat 3D hru, postavenou na fyzikálních základech, která bude využívat právě WebGL. Vzhledem k tomu, že WebGL poskytuje nízkoúrovňové API, bude k usnadnění práce použit nějaký z mnoha existujících frameworků. Ve hře bude muset hráč dostavět rozestavěnou úroveň a odpálit kuličku, tak aby se dostala co cílové jamky.

Tato práce v 2. kapitole rozebírá způsoby vytváření her ve webovém prohlížeči. Speciálně se pak zaměří na stěžejní WebGL, které bude rozebráno podrobněji. Čtenář se seznámí jak s jeho vlastnostmi, tak s principy jeho fungování. V 3. kapitole bude čtenáři představen návrh hry. Dočte se zde o výběru frameworku, pro tvorbu hry, samotném navrhu a způsobu řešení fyziky. Ve 4. kapitole nazvané Implementace představím řešení a použité postupy. 5. kapitola se bude zabívat testováním jak výkonu, tak uživatelskými názory. Poslední 6. kapitola pak zhodnotí dosažené výsledky a navrhne možná rozšíření.

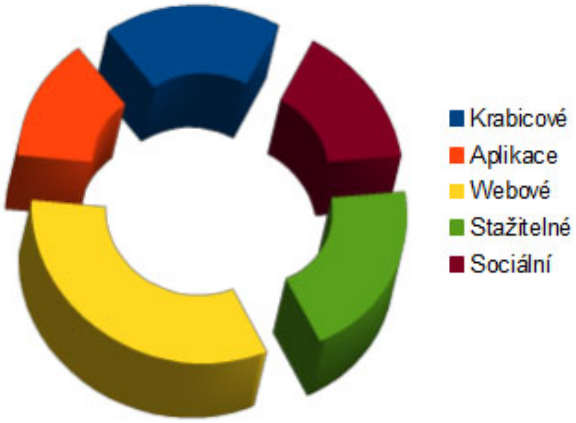
Kapitola 2

Možnosti vytváření her ve webovém prohlížeči

Hry ve webovém prohlížeči jsou velmi oblíbené a přestože je v dnešní době stále více vytlačují hry pro smartphony, jsou webové hry s přehledem nejpopulárnější. Mají totiž několik velikých výhod:

- **Velmi jednoduché spuštění** - Od otevření webové stánky s hrou po stamotného hraní nás dělí pouze pár sekund načítání. Některé technologie (Adobe Flash, Microsoft Sirverlight) sice vyžadují stažení zásuvného modulu do našeho prohlížeče, ale tato akce je pouze jednorázová s následnou občasnou aktualizací modulu.
- **Levné hraní** - Většina webových her je k mání zdarma. Autoři vydělávají pouze na reklamách a popřípadě mikrotransakcích. Hráč proto může vyzkoušet velké množství her, něž si vybere „tu svoji“.
- **Rychlé šíření her** - Vzhledem k tomu, že ke spuštění hry stačí kliknout na odkaz, je šíření hry, v dnešním světě sociálních sítí, snadná záležitost. Vývojáři stačí za „like“ poskytnout hráčí nějaké drobné benefity. Pokud je hra dobrá, může takto během pár dní obletět zeměkouli.
- **Sociální funkce** - Lidé jsou od přírody soutěživí a najdou se i tací, pro které je Twitter či Facebook téměř domovskou stánkou. Těmto hráčům mohou webové hry nabídnout snadné sdílení skóre nebo předhánění se s kamarády.
- **Nenáročnost na HW** - Webové hry jsou oproti svým desktopovým protějškům mnohem méně náročné na výpočetní výkon. To se samozřejmě odráží na kvalitě visualního provedení, ale zároveň to zpřístupňuje webové hraní téměř každému.

Zřejmě díky těmto výhodám se webové hraní umístilo na špici v evropském průzkumu zahrnujícím přes 15 000 respondentů z 15 zemí včetně České republiky [4]. Oblíbenost různých typů her je znázorněna na obrázku 2.1. Ale jaké jsou možnosti webového hraní? Ty vycházejí z použitých technologií. Všechny mají společný fakt, že používají aplikace na straně klienta. Pouze u serverových her, se klientská aplikace musí dotazovat serveru, zda může provést požadovanou akci a jaký je její výsledek. Nyní Vám ty nepoužívanější technologie představím. O samotném WebGL, na kterém bude postavena tato práce, si ale povíme až v kapitole 3. Jde totiž o důležité informace, které si zaslouží svou vlastní kapitolu.



Obrázek 2.1: Rozložení hranosti her podle typu

2.1 Jednoduché hry

Tyto hry využívají pouze standardní webové technologie jako je HTML, CSS, JavaScript a v případě serverové hry i PHP. Většinou se jedná o textové hry, u nichž se vyskytutí maximálně jednoduché 2D animace. Typická hra, která těží maximum z této technologie vypadá zhruba takto : Hráč má k dispozici několik statických obrazovek, přičmž každá mu zpřístupňuje jiné možnosti. Hra propojuje jednotlivé klienty pomocí serveru, který obstarává veškerou funkčnost hry a klientovi pouze posílá data k zobrazení a příjímá od něj požadavky. Typickým příkladem je hra Travian.

Tato technologie má jistě své místo. Vyniká nenáročností jak na výbavu na straně klienta (postačí jakýkoliv počítač se zapnutým JavaScriptem), tak na programátorské dovednosti (nijak se nelyší od programování složitějších webových stránek). Po té co se HTML5 a CSS3 plně zaběhne, získají tyto hry více vizualizačních možností a budou o to poutavější a akčnější.

2.2 Adobe Flash

Tato technologie se používá především pro zobrazovnání inteaktivních reklam, her ale i pro streamování audia nebo videa. Do svého webového prohlížeče musíte nejdříve nainstalovat zásuvný modul, který obsahuje Adobe Flash Player. Dnes ho ovšem má většina prohlížečů již predinstalovaný. Ten nám pak umožní bezpečné zobrazování souborů s koncovkou .swf.

Výhodou Flashe je, že pracuje především s vektorovou grafikou, která má oproti rastrové malou datovou velikost, navíc je neomezeně škálovatelná. Další výhodou je podpora videa ve formatech On2 VP6, Sorenson Spark a od verze 9 H.264. Ve verzi 10.2 byla dokonce přidána hardwarová akcelerace pro vykreslování videa. Flash samozřejmě umí pracovat i s audiem - podporuje více kódování, ale mezi nejpoužívanější patří MP3 a AAC. Další výhodou je i podpora hardwarových zařízení jako mikrofon a webkamera. To z Flashe dělá mocný nástroj, který lze využít tvorbu velkého spektra aplikací. Největší výhodou z pohledu vývojáře je, že Flashový obsah je distribouván v kompilovaném souboru, a tudíž je nemožné získat jeho zdrojový kód a velmi náročné soubor upravit.

Prvním negativem může být fakt, že pro vývoj Flashových aplikací se používá speciální programovací jazyk ActionScript. Jde o objektově-orientovaný jazyk, který naštěstí moc

nevybočuje z klasických syntaxí, jelikož je založen na jazyku ECMAScript, na které je založen například i JavaScript. Flash také nemůže nijak ovlivnit stránku a naopak. Další a mnohem větší mínus je chybějící podpora na mobilních platformách. Apple nepodporoval na iOS Flash nikdy, kvůli velké hardwarové náročnosti. Z Android marketu byl Flash stažen 15.srpna 2012. Microsoft sice dodává IE 10 s předinstalovaným Flash Playerem, ale v metro verzi jde zobrazit flashový obsah jen u stránek, které Microsoft explicitně povolí.

Dle mého názoru je technologie Flash sice pořád jedna z nejpoužívanějších, nicméně je na ústupu a je jen otázkou času než zmizí úplně.[1]

2.3 Silverlight

Technologie Silverlight je konkurencí Flashe od Microsoftu. Její cíle jsou naprosto stejné, ovšem v možnostech se liší. Stejné jako u Flashe je potřeba stáhnout zásuvný modul do prohlížeče. Také se zaměřuje především na vektorovou grafiku, ale zvládá i rastrovou. Verze 1, vydaná v roce 2007, umožňovala popis vzhledu pomocí XAML (známe z Visual Studia) a popis funkcionality pomoci JavaScriptu. Z tohoto důvodu moc nevyniká oproti standardnímu řešení HTML + Javascript. Proto budu rozebírat funkcionalitu aktuální verze 5 z prosince roku 2011.

Specializuje se na přehrávání videa ve vysokém rozlišení (až HD). Samozřejmostí je proto jeho hardwarová akcelerace. Oproti Flashi navíc podporuje formát .wmv. Umožňuje komunikovat s webovou stránkou - Silverlight aplikaci je například možné spustit pomocí tlačítka umístěného jinde na stránce, nebo můžete pomocí aplikace upravit elementy na stránce. Využívá .NET Framework, ale velmi příjemná vlastnost je, že framework nemusí být na počítači nainstalován. Aplikace pro Silverlight se mohou programovat v jakémkoliv .NET jazyce například C#, Visual Basic .NET, F#, J# a jiné. Jejich vyzuální stránku popisujeme pomocí XAML souboru. Díky tomu můžeme použít jakýkoliv editor, ovšem nejlepší je Visaul Studio, kde si můžeme vzhled „naklikat“. Další zajímavou vlastností, která byla přidána ve verzi 5, je hardwarová akcelerace 3D grafiky.

Nevýhodou Silverlightu je jeho malá rozšířenost. Pro Unixové systémy byla vyvýjena implementace Moonlight, ta ovšem za schopnostmi Silverlightu zaostává. Její vývoj byl ukončen, podle vyjádření vedoucího projektu [2], protože Silverlight není vhodná technologie pro desktopové hraní a Microsoft do ní přidal mnoho omezení.

Vzhledem k malému využití, už se dalším vývojem nepočítá ani Microsoft a aktuální 5. verze je nejspíš poslední. Konec podpory je plánován na rok 2021, ale velké služby plánují přechod již teď. Například Netflix - služba pro streamování filmů plánuje přechod na HTML5 video. Mým názorem je, že Silverlight předčil Flash ve funkčnosti, ale přišel na trh v době největší slávy Flashe, kdy se prostě nemohl uchytit. Proto byl odsouzen k zániku, a i přes své nesporné výhody - především v oblasti streamování videa, se nejedná o technologii, která bude tvořit budoucnost webu.

2.4 Ostatní

Internet ovlivňuje více než třetinu obyvatel naší planety [3]. Proto vzniklo mnoho projektů zabývajících se oživením webových stránek. Ten nejzajímavější - WebGL rozebereme až

[1] Adobe je na tuto variantu připraveno, jelikož v posledním SW pro tvorbu flashových aplikací Adobe Flash Professional CS6 zavedlo export v HTML5 doplněným o JavaScript

v kapitole 3. Teď se podíváme na ty ostatní. Nejsou sice masově rozšířené, přesto stojí za povšimnutí.

- Haxe

  Haxe není vývojová platforma, jen programovací jazyk. Ale má velmi zajímavou vlastnost - jeho překladač dokáže exportovat kód pro mnoho platform. Umí vytvořit JavaScriptový soubor pro použití na webu. Pro Flash Player vytvoří .swf soubor. Vygeneruje zdrojové soubory C++ s makefile pro použití na iOS a dále ve verzi 3.0 přinese podporu generování kódu v C# a Javě. Programátorovi poskytuje jak univerzální knihovny jako Math, Date, Xml a jiné, tak i platformově-specifické knihovny pro cílové API. Je to objektově-orientovaný open-source jazyk. Díky tomu má velký potenciál stát se univerzálním nástrojem v rukou programátora.

- Adobe Shockwave

  Podobná technologie jako Flash, která se ovšem trochu liší. Největší výhodou Shockwave je jiný, propracovanější rendrovací engine, který využíva hardwareovou akceleraci pro zobrazení 3D grafiky. Shockwave je určen především pro tvorbu filmů a animací - aplikace jsou pro něj vytvářeny v programu Adobe Director, který používá skriptovací jazyk Lingo. Díky tomu lze vytářet hry, ale je to komplikovanější, než u jiných technologií. Adobe Director se proto používá především pro tvorbu menu u DVD a Shockwave pak především pro přehrávání animací na webu.

- Java Applet

  Tato technolonologie je velmi nyní málo používaná, ale před příchodem WebGL byla nejlepší pro zobrazení výpočetně náročných scén, a to především díky podpoře hardwarové akcelerace 3D grafiky. Bohužel nyní je na ústupu a nyní ji z milionu nejnavštěvovanějších stánek používá pouze 0,28% z top 10 000 dokonce jen 0,05% [5].Pro spuštení appletu musíme nainstalovat RTE[2] a zásuvný modul do prohlížeče. Applet běží na většině prohlížečů v sandboxu[3], aby se zamezilo napadnutí počítače škodlivým kódem. Applet přehraje přeložený binární soubor napsaný v javě. Tato technologie je multiplatformní - běží na Linuxu, Windowsu i OS X. Bohužel na mobilních platformách je zakázána.

- Curl Applet

  Posledním vybraným projektem jsou applety jazyka Curl. Jako u Javy je potřeba stáhnout RTE a zásuvný modul. Stejně jako Java jsou Curl applety podporovány na všech velkých platformách, s vyjímkou mobilních (Linux, Windows, Mac OS X). Curl kombinuje značkovací, skriptovací a oběktově orientovaný jazyk v jednom sjednoceném frameworku.

2.5 WebGL

WebGL je mladá technologie, v mnohém odlišná od výše uvedené konkurence. V této kapitole se jí pokusím představit více do hloubky, než ostatní technologie. Je totiž stavebním kamenem této práce, která si vzala za cíl vytvořit hru založenou na fyzikálních principech právě pomocí technologie WebGL.

[2] RTE - Runtime environment

[3] Sandbox poskytuje appletu pouze přísně kontrolovanou množinu zdrojů

2.5.1 Historie

V roce 2006 začal Vladimir Vukićević z Mozzily pracovat na projektu Canvas 3D, který umožnoval pomocí HTML5 elementu canvas, zobrazovat hardwarově akcelerovanou 3D grafiku. K adresování GPU používal z počátku OpenGL ES 1.1, později OpenGL ES 2.0. Před koncem roku 2007 již měla jak Mozzila tak Opera své vlastní implementace Canvas 3D.

Vukićević požádal neziskové technologické konzorcium Khronos Group[4] o standartizaci API, to vedlo k založení skupiny WebGL Working Group, která se stará o vývoj. Mezi členy skupiny patří mimo jiné i velké firmy vyvýjející webové prohlížeče - Google (Chrome), Apple (Safari), Mozzila (Firefox), Opera (Opera). 3.3.2011 byla konzorciem Khronos Group vydáná finální specifikace WebGL 1.0. V té době ho již podporovali všechy výše zmíněné prohlížeče. Posledním krůčkem by bylo uznání za standard konzorciem W3C. Ta se ho ovšem zdráhá uznat kvůli bezpečnostním rizikům.

2.5.2 Základní fakta

WebGL je v mnoha směrech velmi pozoruhodná technologie. Nevyžaduje žádný zásuvný modul, pouze kompatibilní prohlížeč viz. 2.5.3. Je nezávislá na platfomě. Jak již bylo řečeno, je založena na OpenGL ES 2.0, díky tomu má API známe pro tvůrce OpenGL aplikací. Stejně jako OpenGL využívá hardwarové akcelerace na grafické kartě. Kvůli tomu je nutné napsat programy pro grafickou kartu, takzvané shadery viz. 2.5.5. Je to jen malá daň za možnost zobrazovat velmi složitou 3D grafiku. Navíc pomocí shaderů se dají naprogramovat spektakulární speciální efekty.

WebGL se váže k HTML5, protože pro jeho zobrazení je potřeba nový HTML5 element canvas (plátno). Je zasazeno do DOM stránky, takže s ní může komunikovat. Díky tomu můžou vzniknout velmi zajímavé efekty. Stejně jako zbytek funčnosti webové stránky se WebGL kód, s vyjímkou shaderů viz. 2.5.5, píše pomocí JavaScriptu. Ten přináší pozitiva i negativa. Velkým plusem pro vývojáře je, že nemusí spravovat pamět. JavaScript totiž alokaci a uvolnění paměti řeší sám. Negativem je, že JavaScript je interpretován a nijak se nekompiluje. To z něj dělá velmi nevhodnou technologii pro komerční užití. Řešením jsou speciální programy, které z kódu smažou veškeré komentáře, bíle znaky a přejmenují proměnné náhodně vygenerovanými jmény. Nicméně kód je pořád dostupný. Druhým jazykem, který WebGL používá je GLSL. Ten slouží k popisu shaderů pro grafickou kartu a je stejný jako se používá u OpenGL.

WebGL provází i bezpečnostní rizika. Nepomůžou sice k šíření škodlivého kódu, nebo získání osobních informací, ale i tak jsou trnem v patě konsorcia W3C a společnosti Microsoft. Vzhledem k tomu, že přistupujeme přímo na grafickou kartu, může útočník vylepšit geometrii, nebo texturu o kód pro grafickou kartu a způsobit tak její zamrznutí nebo pád systému [1]. Proto je potřeba se ujistit, že je ovladač vaší grafické karty aktuální.

2.5.3 Prohlížeče s podporou WebGL

WebGL se, narozdíl od většiny technologií popsaných v kapitole 2, tlačí i na mobilní platformy. Většina velkých webových prohlížečů pro desktopy tuto technologii již podporuje a proto přicházejí pomalu na řadu i prohlížeče na mobilních zařízeních.

- Chrome V prohlížeči od Googlu je WebGL podporováno od verze 9. Podpora je standardně zapnutá. Také zastřešuje Chrome Experiments, které mimo jiné sdru-

[4] Khronos Group v současné bodě spravuje mimo jiné standardy OpenGL, OpenGL ES, WebGL

žují i WebGL projekty. Od verze 13 používá Chrome Cross-origin resource sharing (CORS)[5].

- Firefox Mozilla podporuje WebGL ve svém prohlížeči od verze 4. Podpora je také standardně zapnutá. Od verze 8 používá CORS. Firefox for mobile zpřístupňuje WebGL pro platformu Android.
- Safari Safari podporuje WebGL od verze 5.1 na verzích operačního systému Mac OS X Leopard, Snow Leopard, Lion. Podporu je třeba explicitně zapnout.
- Opera Opera podporuje WebGL od ve verzích 11 a Next(12). Také její mobilní prohlížeč pro Andriod Opera Mobile [6] s touto technologií umí pracovat.
- Internet Explorer Microsoft prohlásil, že jeho prohlížeč WebGL nepodporuje a v budoucnu na tom neplánuje nic měnit. Shledává totiž technologii nebezpečnou[1]. Ovšem již existují zásuvné moduly, které do Internet Exporeru přidávají podporu WebGL. Toto je oficiální stanovisko Microsoftu, ale vypadá to, že ho microsoft v tichosti změnil. Podle zatím neoficiálních zpráv od testerů nového updatu Windows 8, známého pod označením Blue, který bude obsahovat nový Explorer 11, to vypadá, že Microsoft přidá nativní podporu WebGL[**?**].

2.5.4 Rozdíly oproti OpenGL

Jak už bylo zmíněno WebGL vychází z OpenGL ES 2.0, což speciální grafické API speciálně upravená pro vestavěné systémy - odtud ES (Embedded systems). Používá se především v mobilních zařízeních - telefonech, tabletech, satelitních navigacích, set-top boxech a podobně. V podstatě jde o klasické OpenGL 2.0 ořezané o náročnější operace. Cílem je totiž dosáhnout co nejmenší spotřeby zařízení a co největší rychlosti zobrazení i na pomalém hardwaru. OpenGL ES podporuje pouze 3 základní primitiva, ze kterých se mohou skládat objekty - body, úsečky, trojúhelníky. Na rozdíl od klasického OpenGL nepodporuje čtyřúhelníky. Také nejsou podporovány 3D textury - ty lze ale emulovat přidáním normálové mapy. Dále není podporován typ GL_DOUBLE.

2.5.5 Pipeline

Pipeline je série kroků, které WebGL dělá při renderování scény. Jde o velmi náročný proces, který probíhá neustále. Pro každý obrázek je potřeba vypočítat barvu až několika milionů pixelů - toto číslo záleží na rozlišení canvasu, do kterého je scéna promítána. Například pro rozlišení VGA 640x480 pixelů jde o 307 tisíc pixelů, ale u nejpoužívanějšího rozlišní[8] 1366x768 už jde o více než milion vykreslovaných bodů. To je ale pouze jeden obrázek, pro kvalitní 3D zážitek můsí být aplikace schopka vykreslit minimálně 30, ideálně 60 fps (frames per second / obrázků za vteřinu). To dělá i 60 milionů pixelů za vteřinu.

To je softwarově neproveditelné, proto se většina práce odehrává na grafické kartě, které je schopna paraelního zpracovávání. Samotná pipeline je pak zřetězené zpracování, které musí proběhnout, za účelem vykresení jednotilivých bodů. Nyní si popíšeme jeho zjednodušenou variantu[6]. Jak na sebe jednotlivé akce navazují demonstuje obrázek 2.5.5. A teď rozebereme jednotlivé akce.

[5] CORS ve WebGL slouží k řešení bezpečnostních rizik. Až na vyjímky povolené pomocí CORS brání načítání textur z jiného webu.

[6] Pouze Opera Mobile, populární Opera Mini WebGL nepodporuje

- Buffer vrcholů

  Obsahuje data, která WebGL potřebuje k posání geometrie jednotlivých objektů.

- Atributy

  Jsou to vstupní proměnné pro vertex shader. Obsahují souřadnice vrcholu, jeho barvu atd.

- Vertex shader

  Zkompilovaný program napsaný v jazyce GLSL a uložený na grafické kartě. Tento program je volán na každý jednotlivý vrchol (vertex) a manipuluje s jeho souřadnicemi, barvou, normálou. Jeho cílem je v podstatě přepočítat souřadnice bodů na nativní souřadnice. Nativní souřadnice udávají kde na canvasu se bod objeví.

- Fragment shader

  Typem se jedná o stejný program jako vertex shader, ovšem úkolem fragment shaderu je vypočítat barvu fragmentu (pixelu). To se děje nejčastěji pomocí interpolace - podle tří daných barev vrcholů trojúhelníka, jehož je fragment součástí, se spočítá barva fragmentu. Daší možností je určení barvy fragmentu z textury, kterou je objekt potažen, nebo vyplnění všech fragmentů trojúhelníku stejnou barvou.

- Konstanty - Uniforms

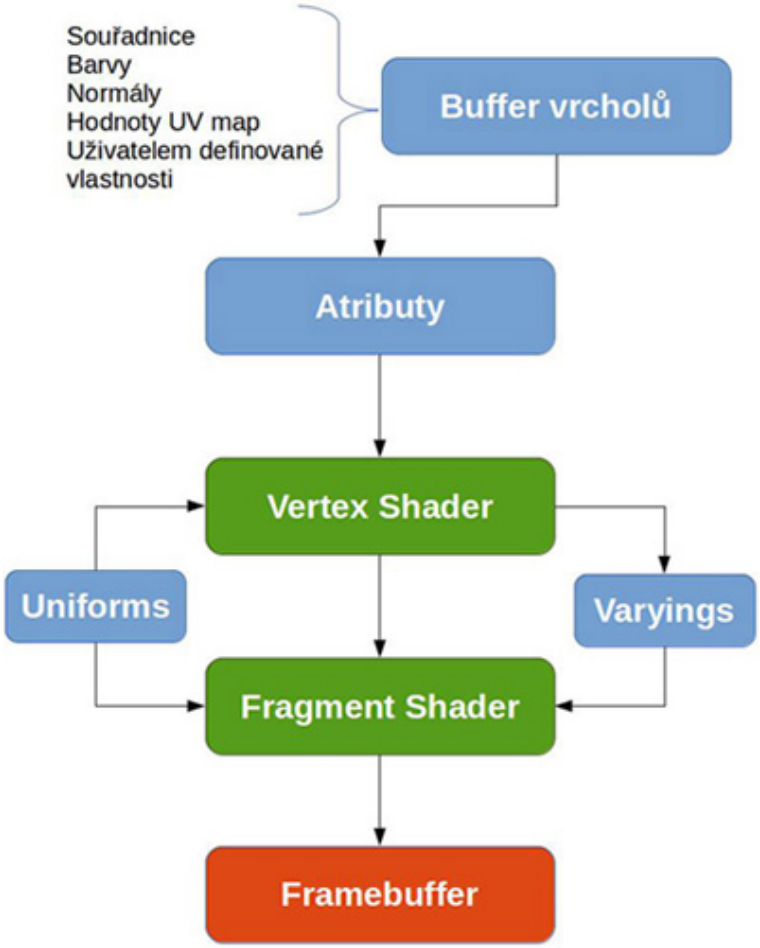
  Jde o hodnoty, které jsou po dobu jednoho rendovacího cyklu jednotné a má k nim přístup jak vertex, tak fragment shader. Typickou konstantou je poloha světla.

- Proměnné - Varyings

  Tyto proměnné předává vertex shader fragment shaderu.

- Frame buffer

  Sem se ukládají barvy jednotlivých pixelů a výsledý obraz je posílán do monitoru.



Obrázek 2.2: WebGL pipeline

Kapitola 3

Návrh hry

Tvorba 3D hry s fyzikou je velmi komplexní a náročná činnost, na které běžně pracují celé týmy designérů a kodérů. Ovšem díky tomu, že je hra určena pro webové prohlížeče, nebude tak náročná, tudíž i její vývoj bude jednodušší. Ale i přesto bylo nutno, vymyslet hru, která bude reálně implementovatelná v daných termínech. Rozhodl jsem se proto vyvynout hru, která bude využívat možností, které mi WebGL a jeho frameworky nabízejí a zároveň půjde o projekt přimněřeného rozsahu.

3.1 Základní představa

Hra bude určena pro jednoho hráče a je zaměřena především na logické uvažování. Vše se bude odehrávat v 3D prostoru s funkčními fyzikálními principy. Cílem bude dopravit 1 - 5 kuliček ze startovního místa do cílové jamky. Level bude z části postaven a z části ho bude muset vytvořit hráč z nabídnutých součástí. Hráč bude moct kuličce udělit počáteční rychlost - odpálit jí a pak záleží pouze na tom, jak hráč postavil bludiště.

3.2 Výběr frameworku

WebGL poskytuje opravdu nízkoúrovňové API. To znamená, že programátor má velmi volnou ruku a spoustu možností. Na druhé straně se i základní činnosti, jako např sestavení scény s jedním 2D objektem, se stává problémem na desítky řádků kódu. Dále se setkáme s nepříjemným zjištěním, že většinu operací děláme stále dokola. Proto vzniklo relativně hodně projektů, které si vzali za cíl odstínit programátora od matematicky složitých konstrukcí a opakující se operace zastřešit pod parametrizovatelné metody. Výsledkem těchto pojektů jsou frameworky, které se zde pokusím zhodnotit. V rozhodovnání mi také pomáhal nástroj na srovnávání projektů[1].

- Scene.js První, čím mě tento framework zaujal, je skutečnost, že se při programování využívá velmi podobných syntaktických konstrukcí jako u JQuery. Z mého pohledu to ovšem činí programování trochu nepřehledné. Framework zvládá pokročilé animační techniky včetně key-frame animace[2] a morphování[3]. Dále zvládá multi-texturování a

[1] https://www.ohloh.net/p/compare?project_0=three.js&project_1=scenejs&project_2=GLGE

[2] Způsob animace, kdy se určí kdy bude předmět v určitém místě a pohyb předmětu se dopočítá. Díky tomu mohou vznikat komplexní nelineární animace.

[3] Změna tvaru objektu. Nepostradatelné například při mimických pohybech, které by jinak vyžadovali extrémě složitý model.

použití videa jako textury. Dokáže do scény importovat objekty ve formátu JSON a po přidání plug-inu i OBJ a COLLADA. Nástroj prohlížeče Chrome - správce úloh, však ukázal při podobných scénách o dost méně fps, než u ostatních porovnávaných frameworků. Dalším negativem je zřejmě zastavení práce na projektu, jelikož podle srovnávače Ohloh, nebyl projekt aktualizován více jak tři čtvrtě roku. Projekt nemá vůbec žádnou dokumetaci a je k němu nejmíň příkladů ze všech srovnávaných.

- GLGE Zajímavý framework, jehož největší předností je dobře zakometovaný kód, což s odráží na velké kvalitě automaticky generované dokumentace. Oproti Scene.js navíc podporuje základní fyziku, částicový systém (důležitý pro tvorbu kouře, mlhy atd.) a více druhů osvětlení. Také při testu fps pomocí správce úloh v prohlížeči Chrome obstál při rendrování podobné scény lépe než Scene.js. Bohužel je zde málo tutoriálů a příkladů a podle Ohlohu projekt ustává a poslední změny byli provedeny před půl rokem.

- Three.js Nejrozšířenější WebGL framework nejen z výše uvedených. API je velmi jednoduché, ale dokáže poskytnout i pokročilé funkce. Kromě základní funkčnosti nabízí podporu odrazů, stínování, video-textur, multi-texturování a mnoho dalšího. Největší předností je ale jeho rozšířenost. Většina WebGL projektů, na které jsem narazil je tvořena právě pomocí Three.js. Díky tomu je k dispozici mnoho tutoriálů, nespočetné množství příkladů a hlavně jde o projekt, který se stále rozvíjí. Kromě autora, který je znám pod jménem Mr.doob, přispívá do tohoto open-source projektu i mnoho jiných vývojářů. Bohužel Three.js nepodporuje fyziku a ta je pro tuto práci kritická. Nicméně díky rozšířenosti frameworku, existuje velká množina nadstaveb, které lze spolu s Three.js použít. Více o tom v sekci 3.5.

  Po úvaze jsem si pro práci na projektu zvolil právě tento framework a to především kvůli jeho rozšířenosti, velému množství příkladů, jeho rozšiřitelnosti o fyzikání frameworky a kvůli knize WebGL Up and Running[7], která se zabývá programováním právě za pomoci Three.js

3.3 Návrh UI

3.3.1 Menu

Po spuštění hry uvidí hráč hlavní menu. Původní plán byl, že menu bude tvořeno plovoucím 3D textem, vytvořeným pomocí WebGL a doplněno zajímavým efektem v podobě 2 světelných bodů, pohybujících se po spirále okolo textu. Pro zintenzivnění 3D pocitu bude na menu náhled trochu ze strany a kamera se bude jemně pohybovat. Ovšem v průběhu práce jsem ovšen návrh změnil. Nyní se menu skládá z 2 částí. 2D část, která slouží k ovládání hry a poskutuje veškerou funkčnost menu. 3D část se stará o dodatečnou vizualizaci a dělá menu zajímavějším. Při označení položky se kamera začne pohybovat směrem k ní a zároveň se rozpohybuje i objekt symbolizující danou položku menu. Budou na výběr 3 možnosti:

- Nová hra - Spustí novou hru od 1. úrovně.

- Výběr úrovně - Zobrazí dodatečnou nabídku pro výběr úrovně. Po kliknutí na číslo bude hra stuštěna od vybrané úrovně.

- Nastavení - Zobrazí 2D menu, které bude umožnovat nastavit některé (především technické) prvky hry. Mezi nastavitelné možnosti patří robrazení FPS, FPS fyzikáního frameworku, nastavení velikosti textur, permanentí zobrazení tipu a nastavení síly gravitace.

Výběr jakékoliv z možností bude mít za následek odstranění menu a zobrazení příslušného obsahu.

3.3.2 Postraní nabídka

Postraní nabídka bude nabízet ovládání v prostředí samotné hry. V menu nebude tato nabídka dostupná. Uživateli bude zprostředkovávat následující volby:

- Spustit
- Restartovat
- Hlavní menu
- Nabídka s díly pro stavbu úrovně Pro snížení hardwarových nároků bude tato nabídka obsahovat pouze 2D obrázky dílů z perspektivího pohledu.

3.3.3 Navigace v 3D prostoru

Hráč bude pohybovat kamerou za pomocí kliknutí a tažení myši. Kamera se bude pohybovat po myšleném tubusu okolo hrací plochy. Kvůli tomu není možné na hrací plochu pohlédnout přímo z prostoru nad plochou, jako by bylo možné u pohledu z myšlené polokoule. Nevzniká však problém s přetočením pohledu na nejvyšším místě koule a při dostatečné výšce tubusu je pohled i tak velmi přehledný. Navíc bylo přidáno přibližování a oddalování od/ke středu scény za pomocí kolečka myši. Díky tomu, si může hráč úroveň lépe prohlédnout.

3.4 Herní principy

3.4.1 Stavba úrovně

Každá úrověň bude tvořena pevnou podložkou a neviditelnou 3D krychlovou mřížkou, která bude podstavou kopírovat podložku a bude růst směrem do výšky. V určité výšce bude limitováná, aby nevznikali nebetyčné konstrukce. Do této mřižky se budou umisťovat jednotlivé herní díly, které budou zabírat jednu nebo více krychliček. Díly bude možné stavět pouze tak aby navazovali na již existující díly nebo podložku. Hráč tedy po spuštění levelu uvidí podložku s rozestavěným levelem. Z postraní nabídky vybere díl, který bude chtít umístit a při myši nad hrací plochou se budou zvýrazňovat styčná plocha na původní kostičce respektive podložce. Při kliknutí myši se nová kostička naváže na původní kostičku ve vyznačeném místě. Kostičky přidané hráčem budou od původních kostiček rozlišeny barvou, aby hráč na první pohled poznal, které kostičky může odebírat a přesouvat.

3.4.2 Pohyb kuličky

Kulička se bude řídit zákony fyziky, které zprostředkuje fyzikální framework viz. 3.5. Hráč bude moc ovlivnit její pohyb pomocí dvou aspektů.

- Stavba levelu Stavbou levelu bude moct hráč ovlivnit trasu kuličky. Rozmístěním herních dílů hráč určí po kterých se bude kulička pohybovat, do kterých narazí a změní směr.

- Odpálení kuličky Druhou možností jak ovlivnit pohyb kuličky je její odpal. Síla odpalu se určí pomocí kliknutí na automaticky se pohybující stupnici. V okamžiku kliknutí se stupnice zastaví a určí sílu odpalu kuličky. Rychlost pohybu kuličky bude určovat, zda kulička nevyletí z dráhy, nebo zda se vyhne pohyblivým překážkám na trase. Tím přibyde do hry element náhody a hra se stane akčnější.

3.4.3 Cíle hry

Jak už bylo řečeno cílem hry bude vždy dopravit všechy kuličky do správné jamky. To je nutná podmínka pro splnění úrovně. Jednotivé dvojice - kulička a jamka budou barevně rozlišeny. Díky tomu hráč jednoduše zjistí, kam má kterou kuličku dopravit. Skóre se bude určovat podle času, který bude třeba k úspěšnému splnění levelu. To znamená, že bude potřeba ideálně sestavit úroveň z nabídnutých dílů a k tomu odpálit kuličku co největší rychlostí, při které ještě nevybočí (například při klopené zatáčce) z trasy a zároveň se vyhne všem překážkám.

3.4.4 Herní díly

Jedna z nejdůležitějších součástí hry jsou díly, ze kterých se tvoří úroveň. Hra bude obsahovat mnoho dílů, které budou sloužit pro vedení kuličky napříč úrovní. Díly je možno rozdělit do dvou skupin :

- Pasivní díly

  Sem patří rovná cesta, zatáčka, klopená zatáčka, stoupání (které může sloužit i jako klesání). Tyto díly se nebudou s postupem času měnit. Z fyzikáního hlediska budou vyrobeny z tvrdého nerozbytného a neposunutelného materiálu.

- Překážky

  Tyto díly jsou také pohyblivé, nicméně je hráč nemůže modifikovat a stavět. Jejich cílem je většinou vystrčit nebo zničit kuličku. O to se pokoušejí periodicky - to znamená, že hráč musí nastavit zprávnou rychlost aby se těmto překážkám vyhnul. Jde třeba o kyvadlo, které kuličku při střetu vychýlí z dráhy.

3.5 Fyzikální framework

Jak již bylo řečeno framework Three.js nepodporuje fyziku. Proto bylo nutné najít framework, který je kompatibilní s Three.js a dokáže vytvořit scénu, která se řídí fyzikáními zákony.

Po dlouhém uvažování jsem vybral framework Physi.js, který je postaven na Ammo.js. Jeho programování je totiž téměř stejné jako u Three.js. Navíc je relativně hodně rozšířen.

Stejně jako všechny fyzikální frameworky podporuje detekci kolizí. To je nesmírně důležité, jelikož Three.js i přes počáteční snahy nemá podporu detekce kolizí a tuto záležitost umožňuje řešit pouze pomocí paprsků. Podobným způsobem řeším klikání na objekty ve scéně. Řešení spočívá v tom, že se z předmětu vyšlou paprsky (objekt `THREE.Ray`) do různých směrů. Paprsek po zavolání metody `THREE.Ray.intersectObjects` vrátí seřazené

pole objektů, které protíná spolu se vzdáleností v jaké je protíná. Díky tomu můžeme zjistit, zda došlo ke kolizi. Bohužel čím složitější tvary tím více musíme mít paprsků. Protože u složitějších objektů bychom pro dokonalou detekci museli mít velké množsví paprsků (v řádu stovek), jsou objekty velmi zjednodušovány. Například pro lidskou postavu používá framework Physi.js válec s polokoulemi místo podstav, ale i tak se používají desítky paprsků. Z popisu vyplívá, že se nejedná o jednoduchou záležitost, proto byla detekce kolizí prvním požadavkem, který musí framework zvádat.

Další předností je podpora materiálů, které řeší například tření a odrážlivost předmětů). Tyto hodnoty jsou plně nastavitelné, takže může programátor vytvářet nové materiály se zajímavým chováním. Dále nabízí framework jednoduché řešení částečně uchycených předmětů, které mohu využít u překážek typu pohybující se kladivo.

Problémem tohoto frameworku je značné snížení fps při větším počtu pohyblivých předmětů ve scéně. Ovšem scény v této hře se budou skládat z maximálně 10 pohyblivých předmětů - až 5 kuliček + překážky. Proto mi tato skutečnost příliš nevadí.

Použití Physi.js je velmi jednoduché, jelikož se práce s ním velmi podobá práci s Three.js. Pro spuštění fyzikální simulace je potřeba zavolat metodu `scene.simulate()`, která scénu rozpohybuje. Simulace probíhá ve zvláštním vlákně, takže neovlivňuje tok chodu hlavního programu.

3.6 Modelování dílů

Knihovna Three.js sice umožňuje tvorbu základních 3D objektů, ale to pro tvorbu graficky zajímavé aplikace jakou je hra nestačí. Je třeba vytvořit vlastní předměty a ty potáhnout texturami. Jednou možností je vytvořit objekty definování vrcholů do bufferu. Toto je však velmi pracné a u složitějších modelů časově extrémě náročné. Navíc pro takovýto model je nemožné vytvořit texturu. Herní díly pro to budou vytvořeny pomocí programu určeného k modelování 3D objektů. K tomuto účelu jsem zvolil freewarový Blender. Je sice velmi složité s ním začít, především protože pro většinu fukncí je třeba znát speciální klávesovou zkratku, ale po čase je to velmi účinný nástroj. Tým okolo THREE.js navíc vytvořil exporter do speciálního THREE.js JSON formátu pro popis modelů. Ten mi velmi ulehčil práci a hlavně zajistil 100% kompatibilitu a správnost exportovaného modelu.

3.7 Návrh úrovní

Při navrhování úrovní jsem postupoval tak, aby se hráč seznamoval s možnostmi hry postupně. Z vlastních zkušeností vím, že když uživatele uvrhneme do nepřehnedné aplikace a nedáme mu nápovědu, tak aplikaci velmi brzy přestane používat. Náhrad je vždy více než dost. V návrhu jsem ovšem počítal s díly, které se později nepovedlo použít pomocí fyzikálního frameworku, více o tom v přístí kapitole.

Kapitola 4

Implementace

Tato kapitola bude pojednávat o samotné implementaci hry a řešení problému, které při ní nastali. Herní jádro tvoří aplikace napsaná v JavaScriptu. V tom jsou také napsané veškeré knihovny využívané aplikací. Vzhledem k tomu, že jde o webovou aplikaci, je její 2D část (především menu a nastavení) popsaná pomocí HTML5 a CSS3. Posledním použitým jazykem GLSL pro programování shaderů.

4.1 Struktura aplikace

Jak již bylo uvedeno, tak aplikace jde rozdělit na dvě části. HTML a CSS část, která obstarává 2D vzhled a JavaScriptovou aplikaci, která se stará o funkčnost a s pomocí frameworků o 3D vykreslování a celou funkčnost 3D scény. JavaScriptový objekt(jedináček), který tvoří samotnou aplikaci je pojmenovaný `puzzleBall`. Ten pod sebou združuje mnoho jiných objektů, které zastřešuje pod jeden jmenný prostor. Obrázek 4.1 rozbrazuje veškeré objekty i s jejich metodami. Komentář a popis v obrázku není, ale některé významější objekty a metody budou popsány v dalších sekcích.

Pro zápis zdrojového kódu jsem použil návrhový vzor singleton. Původně jsem chtěl použít modulární přístup s přednačtenými funkcemi. Jeden článek totiž uváděl, že oproti klasickému modulárnímu přístupu poskytuje až 20% zrychlení při interpretaci. Moje první testy s jednoduchým objektem s jedinou metodou vykazovali dokonce poloviční časy ve prospěch vzoru s přednačtenými funkcemi. Ovšem pokročilejší testy pro objekt s 2 metodami a vnořenými objekty, se kterými metody pracovali (což se více podobá struktuře mé aplikace), dopadli naprosto opačně 4.1. Proto jsem od tohoto přístupu odstoupil a metody definoval přímo v objektech. Následující kód ukazuje, jak vypadal testovací objekt s předdefinovanými metodami.

```
 precachedCompute = function () {
  Math.log(this.num);
};

precachedRand = function () {
  this.num = Math.random();
};

PrecachedModule = {
  num: 0,
```

```
  compute: precachedCompute,
  getNum: precachedRand,
};
```

Typ objektu	Návrhový vzor	Průměrný čas[ms]
Jednoduchý objekt	Modularní	154ms
Jednoduchý objekt	Přednačtené funkce	73ms
Složitější objekt	Modularní	92ms
Složitější objekt	Přednačtené funkce	157ms

Tabulka 4.1: Porovnání rychlosti interpretace návrhových vzorů

4.2 Popis objektů

Pro lepší orientaci v aplikaci popíši velmi jednoduše funkci jednotlivých hlavních objektů.

- puzzleBall

  Je to hlavní objekt, který slouží jako jmenný prostor a zastřešuje celé Javascriptové jádro aplikace. Obsahuje objekty z knihovny THREE.js, které slouží k renderování obsahu. Dále obsahuje objekty, které jsou využívány jak v menu tak při hře. Jde například o pole objektů ve scéně nebo proměnné pro pohyb kamery. Nejdůležitější metodou zde je `puzzleBall.setUpApp`, která spustí celou aplikaci.

- puzzleBall.settings

  Obsahuje nastavení aplikace. Odtud se například zjisťuje v jakém rozlišení se mají načíst textury, nebo jak velká gravitace se má použít.

- puzzleBall.maps

  Obsahuje objekty do kterých jsou při načítání aplikace načteny textury pro pokrytí objektů scény. Načtení provádí metoda `puzzleBall.maps.load`, která volá metodu load u každého dílčího objektu.

- puzzleBall.materials

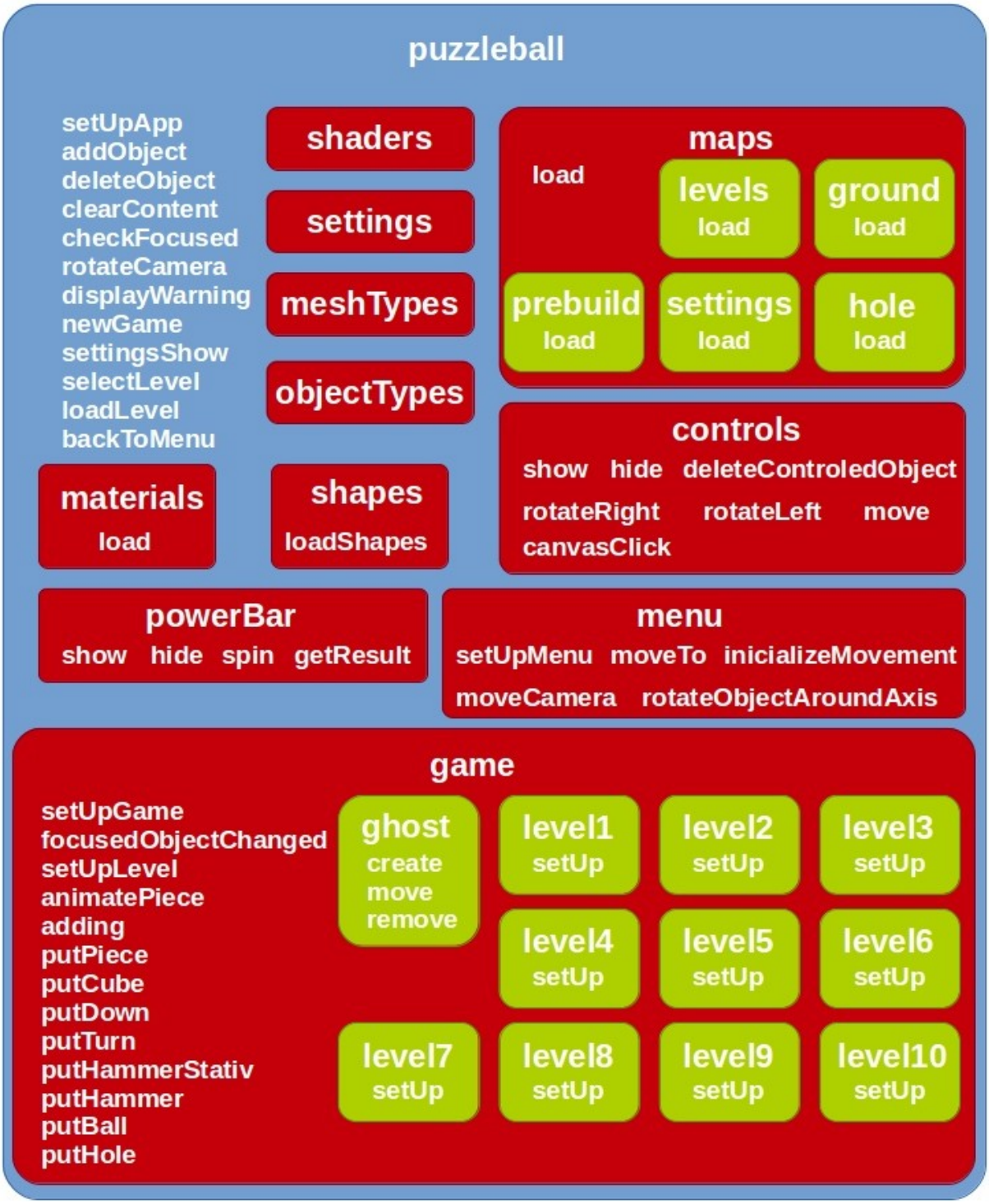
  Obsahuje objekty s materiály ze kterých jsou tvořeny herní 3D meshe. Každý materiál má 2 verze. Klasickou a focused - ta se použije pro aktuálně označený objekt. Tyto 2 verze se liší vždy pouze použitými texturami. Nejdůležitější metodou je opět `puzzleBall.materials.load`. Ta je volána vždy po načtení textur, takže například při změně rozlišení textur dojde okamžitě ke znovuvytvoření materiálů.

- puzzleBall.shapes

  Obsahuje geometrie použité v aplikaci. Některé jsou vytvořeny pomocí THREE.js konstruktoru, jiné jsou načteny z JSON souboru vytvořeného v Blenderu. Metoda `puzzlBall.shapes.loadShapes` načte právě tyto JSON modely.

- puzzleBall.controls

  Obsahuje metody volané z 2D ovládacího prvku pro změnu hráčem přidaných dílků. Metody zajišťují zobrazení/schování 2D rozhraní, otočení dílku, odstranění dílku a povolení pohybu s dílkem.



Obrázek 4.1: Struktura aplikace - objekty s metodami

- puzzleBall.powerBar

  Spravuje 2D rozhraní pro odpal kuličky. Metody točí s prvkem a následně odečítají uživatelem zvolenou rychlost odpalu a nastavují ji kuličce.

- puzzleBall.menu

  Obsahuje metody pro rozpohybování 3D scény v menu - ta totiž není řízená fyzikou. Spuštění a nastavení obstarává `puzzlBall.menu.setUpMenu`. Dále obsahuje sadu metod pro pohyb kamery ve scéně s menu.

- puzzleBall.game

  Spravuje vše co se týká herní části. Jde o nejrozsáhlejší objekt. Hru spouští metoda `puzzlBall.game.setUpGame`. Jednotlivé levely jsou pak nastaveny a načteny pomocí `puzzlBall.game.setUpLevel`. Dále obsahuje sadu metod pro přidávání dílů do hry. Ty jsou tak parametrizovatelné, že jsou používány jak při programovém vytváření úrovní, tak při vkládání dílů hráčem. Dále obsahuje objekty jednotlivých úrovní.

- puzzleBall.game.ghost

  Zvláštním objektem je takzvaný duch. Stará se o to, aby hráč viděl kam zrovna umisťuje dílek. Pohybuje se spolu s kurzorem a přilepuje s k objektům, ke kterým lze dílek umístit. To znamená, že tento objekt obsahuje hodně kritických metod, které ovládají přidávání dílů.

- run

  Tato funkce není kvůli rychlosti začleněna do objektu a to kvůli rychlosti. Jde o hlavní vykreslovací smyčku, která je volána i 60x za vteřinu.

4.3 Grafika

Grafiku ve hře bych mohl rozdělit na 2 části. První část tvoří uživatelské rozhraní, které má hráče informovat o stavu hry a umožnit mu ji efektivně ovládat. A druhou částí jsou pak 3D objekty a jejich textury. Do této části spadají i objekty vykreslované vlastním shaderem, ale pro ten je vytvořená samostatná sekce.

4.3.1 2D

Pro tvorbu 2D ovládacích prvků jsem použil jednoduchý černo-bílý design. Jde celkem o 8 skupin ovládacích prvků, které slouží k různým účelům. V následujícím seznamu je uvedeno kde ve hře se vyskytují, k jakému účelu slouží a jak jsou vytvořeny.

- Položky hlavního menu

  Hlavní menu 4.3.1 je hráči zobrazeno po spuštení hry. Jsou zde 3 hlavní ovládací prvky vytvořené pomocí HTML a vystylované pomocí CSS. Umožňují začít novou hru (od 1. úrovně), výběr úrovně a nastavení hry. Na jejich `onmouseover` a `onmouseout` události jsou nabindované funkce pro pohyb v 3D menu na pozadí. To ovšem plní jen dekorativní funkci.

Obrázek 4.2: Screenshot hlavního menu hry

- Položky menu Vybrat úroveň

  Tato nabídka 4.3.1 je hráči dostupná po zvolení v hlavním menu. Obsahuje tlačítka pro spuštění hry od určité úrovně a pro návrat zpět do hlavního menu. Opět je vyvořená pouze pomocí HTML a CSS.

- Nastavení

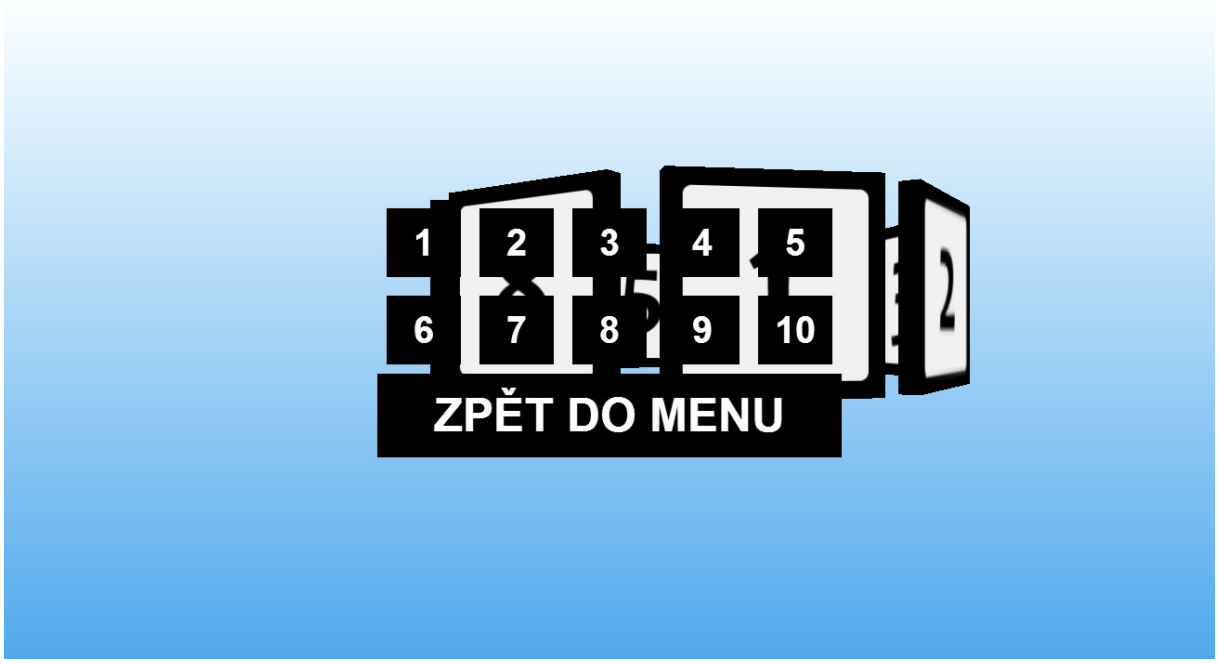
  Tato nabídka 4.3.1 lze zpřístupnit opět z hlavního menu. Ovládací prvky jsou zde vytvořeny pomocí knihovny jQueryUI a slouží k nastavení různých prvků hry. Lze zde například zobrazit fps rendrovacího a fps fyzikáního frameworku. Na to je využita knihovna stats.js, jejíž použití je velmi jednoduché. Stačí vytvořit objekt měřiče, přidat ho do DOM stránky. Pro zaznamenání vydendrovaného snímku pak stačí zavolat metodu `Stats.update`. To provádím v hlavní rendrovací smyčce a pro fyzikální FPS při jejím výpočtu. Nutno si uvědomit, že fyzikální simulace neprobíhá celou dobu, ale pouze po dobu spuštěné simulace, takže v menu nebo při stavění úrovně se může měřič jevit jako nečinný. Nastavení dále nabízí možnost nastavení síly gravitace, trvalé zobrazení nápovědního tipu, nebo nastavení velikosi textur. Ty jsou pro hru vyrobeny ve 3 rozlišeních.

- Postraní nabídka ve hře

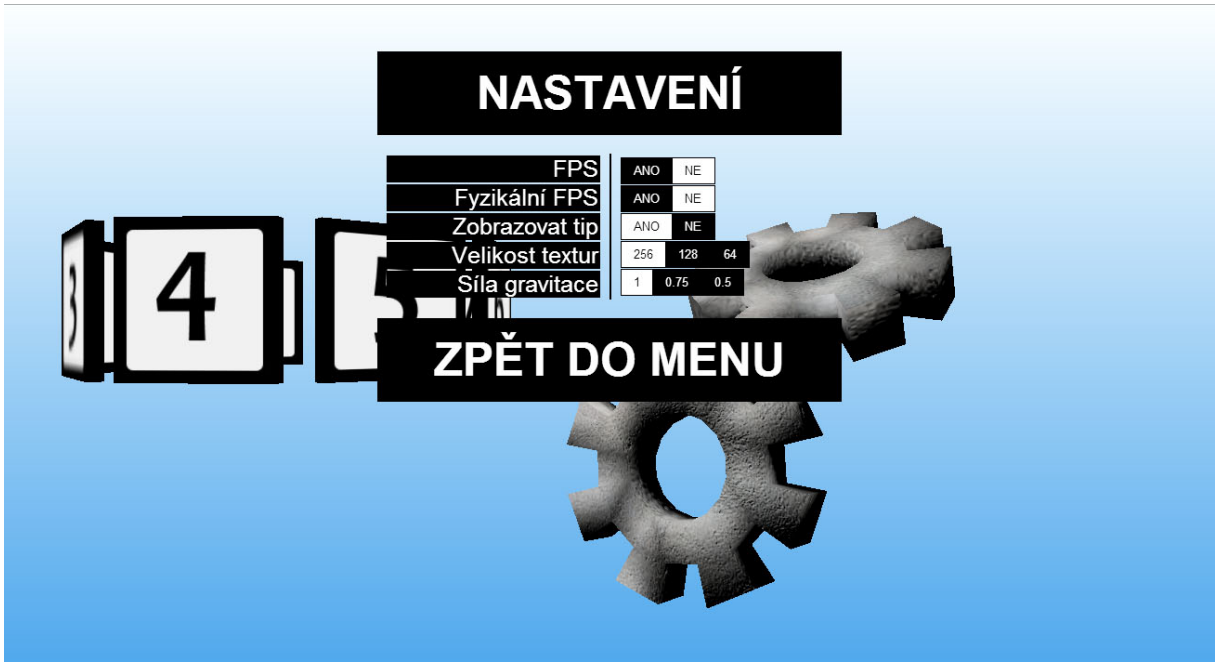
  Tato nabídka 4.3.1 se nachází v pravém horním rohu při samotném hraní. Slouží pro návrat do hlavního menu, pro restartování úrovně a pro zahájení simulace. Také je zde nabídka s dílky, odkud si hráč vybírá, jaké dílky do hry umístí. Ta se s každou úrovní liší.

- Tlačítka pro editaci 3D obsahu

  Po kliknutí na uživatelem přidaný objekt se zobrazí tato nabídka 4.3.1. Umožňuje objekt otočit o 90° stupňů oběma směry, přesunout ho, nebo ho odstranit ze scény.

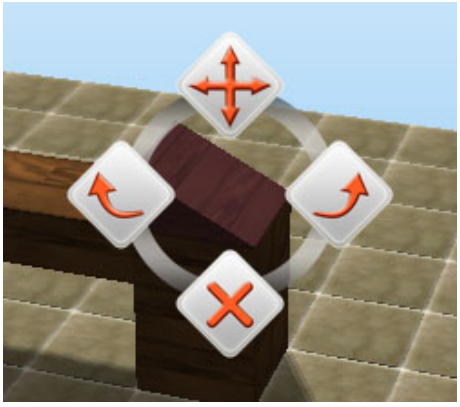


Obrázek 4.3: Screenshot nabídky pro výběr úrovně



Obrázek 4.4: Screenshot nastavení

Obrázek 4.5: Screenshoty postraního menu s různě rozbalenými nabídkami



Obrázek 4.6: Tlačítka pro editaci 3D obsahu

Otočení není zobrazeno pro základní kostičku, kde otočení o 90° nedává smysl. Nabídka je tedy tvořena ze 4 tlačítek a poloprůhledného podkladu. Tlačítka jsou tvořeny obrázky i s efektem pro `:hover` událost. Tlačítka jsou vystylována pomocí CSS Sprites. To znamená, že jsou všecha uložena v jednom souboru a pouze měníme pozici tohoto pozadí tak, aby bylo vždy vidět požadované tlačítko. Tím odbouráme problikávání tlačítek a snížíme počet HTTP požadavků na webový server, což se projeví ve svižnějším načtení stránky. Tlačítka jsem nakreslil v grafikém editoru. Jejich funkčnost řeší objekt `puzzleBall.controls`

- Posuvník pro nastavení síly odpalu kuličky

  Tento posuvník 4.3.1 se zobrazí po kliknutí na kuličku. Je tvořen 2mi přes sebe položenými obrázky - pozadím a výplní. Výška výplně je periodicky měněna, takže dostáváme efekt naplňujícího se prvku. Funkčnost obstarává objekt `puzzleBall.powerBar`.

- Tip

  Je zobrazen při hraní v levém dolním rohu. Jeho cílem je napovědět hráči co a jak v úrovni dělat a provést ho hrou. Po kliknutí na text se tip animovaně schová a zůstane pouze jeho nadpis. Po kliknutí na nadpis se opět zobrazí. Defaultní nastavení/schování tipu lze nastavit v nastavení. Tip je vytvořen pomocí HTML a CSS a k animaci využívá knihovnu jQuery.

- Nápisy v centru obrazovky

Obrázek 4.7: Posuvník pro nastavení síly odpalu kuličky

Zobrazují různé události přes střed obrazovky, kterých by si uživatel měl všimnout. Jde především o číslo úrovně a varování, že nelze spustit simulaci, dokud uživatel neodpálí veškeré kuličky. Napisy jsou vytvořeny pomocí HTML a CSS a animovány pomocí jQuery. Na obrazovce se zobrazí po dobu 2,5 vteřin.

4.3.2 3D

Dalším typem grafiky je ta stěžejní - 3D. Každý objekt ve scéně je definován pomocí geometrie, materiálu a fyzikálních vlastností. Tuto množinu jsem ještě rozšířil o další vlastnosti specifické pro tuto aplikaci.

- Geometrie

  Geometrie objektu je pro jednodušší objekty jako podložka vytvořená pomocí knihovy THREE, pro herní díly se ale načítá ze souborů ve formátu JSON. Tento formát je specifický pro THREE.js a jeho autoři napsali také addon pro Blender, díky kterému se dají modely snadno exportovat. U modelů bylo nutno definovat UV souřadnice, aby se poté dali otexturovat. UV souřadnice pro každý vrchol značí na jakou pozici v textuře se mapuje. Oproti plánu se musela pozměnit jamka. Fyzikální framework bohužel neumí moc dobře pracovat s konkávními útvary, takže jamka jako kostička s dírou naprosto neprošla testováním. Její funkčnost naprosto neodpovídala jejímu vzhledu. Proto jsem se rozhodl vymodelovat jamku jako takovou čenou díru, která kuličku pohltí. Jejím geometrickým základem je proto koule. Nefunkčnost konkávních geometrií také znamená velmi špatné chování pro zatáčku. Ta nefunguje úplně podle plánů, ale je použitelná.

- Materiály

  Materiály popisují jak barvu objektu, tak jeho fyzikální vlastnosti. Prvním použitým typem materiálu je `THREE.MeshPhongMaterial`. Ten je upraven pouze barvami - jsou nastaveny difusní, ambientní a spekulární barvy a odrazovost. Tento materiál je vytvořen v 6 odstínech (3 klasické a 3 pro označený objekt). Používá se pro kuličky a to proto, že fyzikální framework s kuličkou sice pohybuje, ale neotáčí s ní. Když je kulička potažena texturou, tak je tahle skutečnost vidět a kazí dojem ze hry. Veškeré ostatní materiály jsou pak tvořeny několika texturami. Základní difusní texturou a dále normálovou, spekulární a rozptylovou texturou pro lepší vizuální zážitek. Tyto textury nejsou mým výtvorem, jsou nabízeny volně k dispozici na serveru www.turbosquid.com. Licence omezující použití těchto textur umožňuje jejich použití ve hře, pokud se nebudou dál šířit. Dalším typem materiálu je pak speciální shader

materiál pro jamku, ale tomu bude věnována celá příští sekce. Veškeré materiály musí mít také nastavené tření a skákavost.

- Meshe

  Každý jeden mesh je vytvořen z geometrie a materiálu. Frameworky THREE a Physijs nevyžadují vytvoření nové instance geometrie a materiálu, ale stačí reference na ně. Díky tomu se ušetří spousta paměti, ale zároveň nelze označenému objektu změnit texturu materiálu, protože se změní celý materiál. Ten využívá více meshů ve scéně a proto se změní textura u všech. Proto je třeba vytvořit jiný materiál používaný pro označený objekt a ten objektu přiřadit.

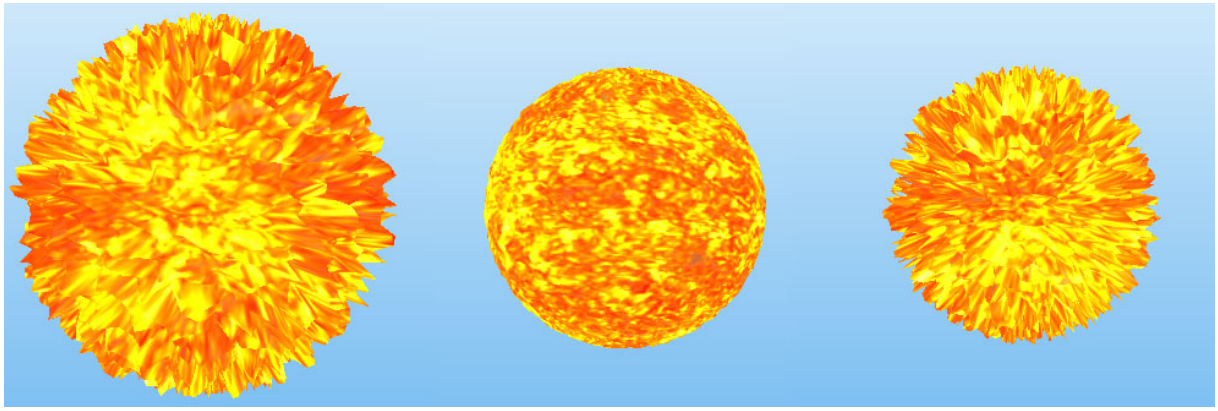
4.4 Tvorba shaderů

Shadery jsou velmi pokročilou a zároveň velmi mocnou zbraní v rukách grafika. Ten totiž může ovlivnit celý proces vykreslování pixelů na obsazovce. Knihovna THREE má již pár vlastních shaderů předpřipravených a jejich použití je velmi snadné. Bohužel na jamku jsem potřeboval vytvořit něco velmi speciálního, abych dostal efekt alespoň vzáleně podobný černé díře. Nakonec jsem se rozhodl modifikovat geometrii pomocí vertex shaderu, tak aby se periodicky pohybovala dovnitř směrem do jamky a pak ven z ní. Pomocí fragment shaderu jsem docílil vířícího efektu. Použil jsem 2 textury - jednu spodní pro barvu a jednu vrchní, se kterou se v závislosti na čase pohybuje a ta ovlivňuje barvu spodní textury. Zde je okomentovaný GLSL kód. Celá animace je řízena z hlavní renderovací smyčky, kde měníme hodnoty time a amplitude. Obrázek 4.4 zobrazuje krajní stavy animace.

```
uniform float amplitude; //Aktuální amplituda - cyklicky ji upravujeme
attribute float displacement;  //Obsahuje pole náhodných hodnot pro každý vrchol
varying vec2 texCoord;
void main() {
texCoord = uv; // souřadnice pro mapování textur budeme potřebovat v FS
//Chceme aby se vrchol vytahoval ve směru normály
vec3 newPosition = position + normal *  vec3(displacement * amplitude);
//Vynásobíme MV a Projection maticí
vec4 mvPosition = modelViewMatrix * vec4(newPosition, 1.0);
gl_Position = projectionMatrix * mvPosition;
}

uniform float time; //Předávame čas
uniform sampler2D texture1; //Poloprůhledná textura
uniform sampler2D texture2; //Spodní textura
varying vec2 texCoord;
void main(void) {
vec4 noise = texture2D(texture1, texCoord);
// Vectory směru pohybu textury
vec2 T1 = texCoord + vec2(1.5,-1.5) * time * 0.01;
vec2 T2 = texCoord + vec2(-0.5, 2.0) * time * 0.01;
// Vynásobíme je barvou poloprůhledné textury abychom dostali
// zajímavý nahodný efekt
T1.x -= noise.r * 2.0;
```

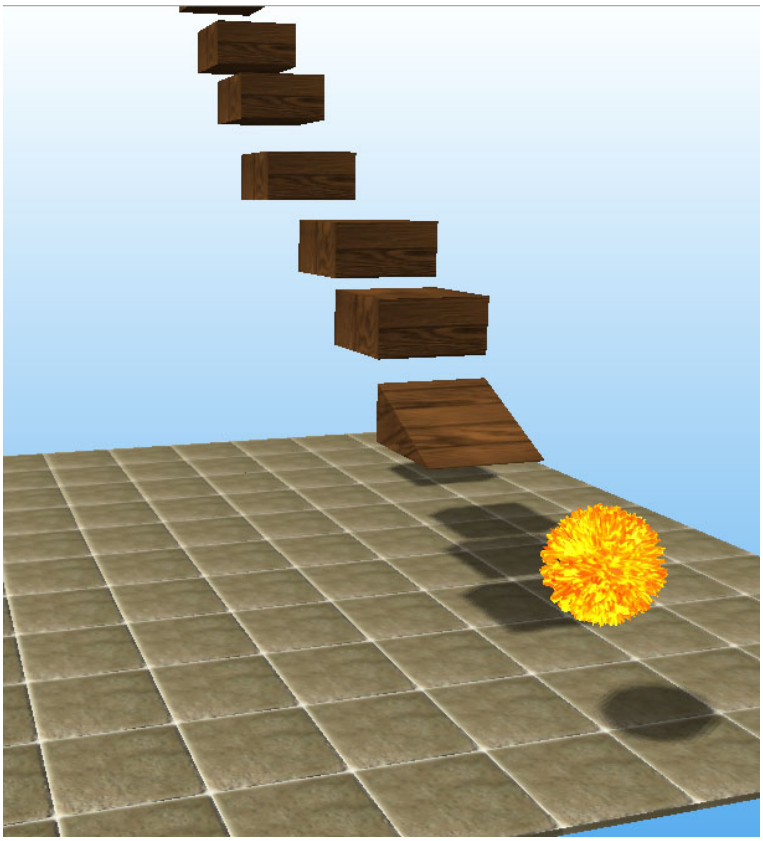
```
T1.y += noise.g * 4.0;
T2.x += noise.g * 0.2;
T2.y += noise.b * 0.2;
// Z průhledné textury nás zajímá jen její průhlednost
float p = texture2D(texture1, T1 * 2.0).a +0.25;
// Barvu vezmeme z barevné textury
vec4 color = texture2D(texture2, T2);
// A smícháme výslednou barvu
gl_FragColor = color * 2.0 * (vec4(p,p,p,p))+(color * color);
}
```



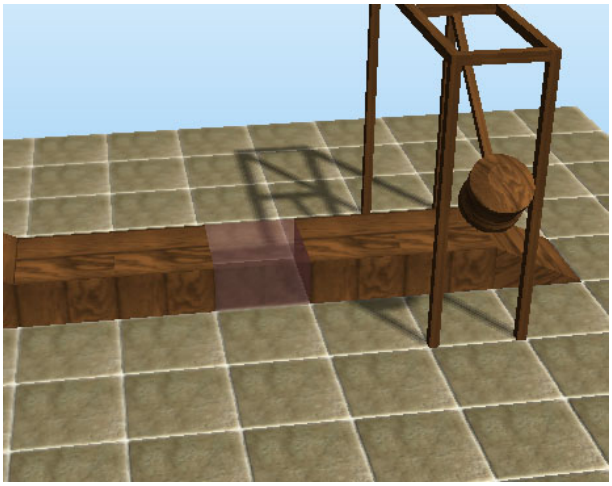
Obrázek 4.8: Objekt vykreslený pomocí vlastního shaderu v různých časech

4.5 Načítání úrovní

O načtení úrovně se stará metoda `puzzleBall.game.setUpLevel`. Ta podle stavové proměnné pozná, jakou úroveň má načíst. U objektu reprezentujícího tuto úroveň spustí metodu `setUp`. Ta přidá objektu úrovně do pole `objects` veškeré meshe. Poté pokračuje metoda `puzzleBall.game.setUpLevel`, po určitém intervalu vezme mesh z pole a umístí ho do scény mnohem víš než by měl být a přidá do pole `puzzleBall.game.putting`. V hlavní smyčce zjistíme jestli je hra ve stavu načítání. Pokud ano, tak projdeme pole `puzzleBall.game.putting` a každý dílek posuneme o něco níž, dokud se nedostane na svou pozici. Toto je spíše ústupek než plánovaný efekt. V průběhu experimentů jsem totiž zjistil, že si fyzikální framework nárokuje obrovské množství paměti (250MB) a renderovací framework společně s mou aplikací cca 150MB. To v kombinaci s faktem, že JavaScript neumožňuje programově uvolňovat alokovanou pamět, způsobovalo problémy. Než totiž zabral garbage-collector a smazal objekty na které nejsou žádné reference, tak uplynulo 5 - 10 vteřin. Mezitím stihl uživatel několikrát restartovat úroveň nebo postoupit do další. To způsobilo, že v paměti bylo naalokováno místo pro několik úrovní s fyzikální scénou. V určitém momentě pak operační systém nedovolil naalokovat další paměť a aplikace spadla. Zajímavé je, že tato situace nastávala pouze v Google Chrome, ve Firefoxu tento problém nenastal - ten naalokoval pro aplikaci až 1,3GB paměti a celkově stíhal paměť uvolňovat rychleji.



Obrázek 4.9: Animace skládání úrovně



Obrázek 4.10: Screenshot přidávání dílku

4.6 Interakce s 3D objekty

Hráčovým úkolem ve hře je dostavět úroveň a následně odpálit kuličku. Proto bylo potřeba zavést nějaký druh interakce s 3D prostorem. Hráč potřebuje určit místo kam dílek přidá, jakým směrem ho natočí a určit jakou kuličku bude odpalovat. Základem tohoto mechanismu je metoda `puzzleBall.checkFocused`. Pokaždé, když hráč pohne myší nad canvasem, tak je zavolána tato metoda. Ta vyšle z bodu pod kurzorem do scény paprsek `THREE.Raycaster`, který protne objekty ve scéně a vratí je uspořádné do pole. To postačuje pro označení objektu. Pro vizualizaci zavoláme metodu `puzzleBall.game.focusedObjectChanged`, která změní materiál označeného objektu na mírně tmavší a díky tomu hráč bezpečně pozná, na jaký objekt ukazuje. Po kliknutí se pak podle typu označeného objektu zobrazí 2D ovládací prvek.

Při přidávání objektu do scény je vytvořen takzvaný duch. Duch je průhledný a umisťuje se ve scéně podle toho, kam ukazuje kurzor. Po kliknutí se pak duch změní v plnohodnotný objekt. Ovšem pro pohyb ducha potřebujeme znát i stěnu objektu na kterou hráč ukazuje. Naštěstí `THREE.Raycaster` nám vrací i souřadnice ve scéně, kde byl objekt paprskem protnut. Z nich vypočítáme na jakou stěnu hráč ukazuje a můžeme k ní přesunout ducha.

Přidávání dílků bylo nutné omezit tak, aby hráč nemohl stavět tam, kde už stojí něco jiného například kulička. Proto se objekty rozšiřují o pole `isBlockingCubes`, které obsahuje 3D souřadnice krychliček, které tento objekt zabírá. Kontrolu tohoto pravidla má na starosti metoda `puzzleBall.game.ghost.move`. Ta je volána při každém pohybu myši, když hráč zrovna přidává dílek do scény.

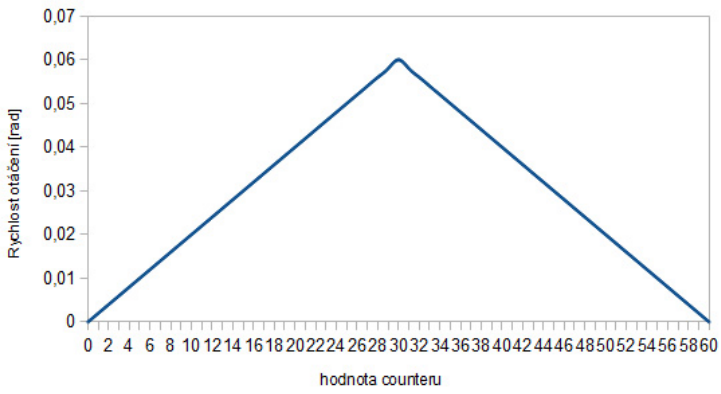
4.7 Pohyb objektů ve scéně

Hlavními pohyblivými objekty ve scéně jsou kuličky, kterým hráč udělí počáteční impuls a pak už se řídí zákonitostmi fyziky. A jak je to v kódu? Při vkládání kuličky do scény pomocí metody `puzzleBall.game.putBall`, předávám jako parametr vektor směru pohybu kuličky. Poté, když hráč klikne ve scéně na kuličku a zvolí rychlost odpalu se tato rychlost uloží do skalární proměnné power. Při spuštení sminulace pak vynásobíme vektor směru skalárem síly a výsledný vektor použijeme pro odpálení kuličky pomocí metody z knihovny Physi.js

-`Physijs.mesh.applyCentralImpulse`. Po spuštění simulace pak v každém kroku hlavní smyčky voláme metodu z knihovní metodu `Physijs.scene.simulate`. Ta upraví polohu kuliček.

Ve scéně se ale nachází i překážky jako kladivo, se kterými musíme pohybovat bez pomoci frameworku. Je nutné s nimi pohnout pouze pokud framework upadatoval scénu. Každá takto pohyblivá část se registruje do pole `puzzleBall.movements`. Musí mít také metodu `myMoveFunction`, která vypočítává pohyb objektu. V hlavní smyčce při simulaci pak projdeme pole movements a u každého objektu zavoláme jeho metodu myMoveFunction. Následující kód ukazuje metodu myMoveFunction u pohyblivého kladiva. Graf 4.7 pak znázorňuje rychlost rotace v závislosti na pootočení, které reprezentuje proměnná counter.

```
hammer.myMoveFunction = function () {
   // counter, jehož definice zde není, se stará o změnu směru
   // a také o krokování pohybu
   // jeden švih kladiva v jednom směru je rozdělen do 60 kroků
   if (this.counter < 1) {
      this.direction = !this.direction;
      this.counter = 60;
   }
   // vlastní rotace kladivem
   // tento vzorec je vymyšlen tak aby kladivo působilo přirozeně
   if (this.direction == true){
      this.rotation.z += Math.abs(Math.abs(this.counter - 30) -30) / 500;
   }else{
      this.rotation.z -= Math.abs(Math.abs(this.counter - 30) - 30) / 500;
   }
   // řekneme frameworku, že má počítat s námi vypočtenou polohou
   this.__dirtyRotation = true;
   this.__dirtyPosition = true;
   // krok animace
   this.counter--;
}
```



Obrázek 4.11: Závislost rychlosti rotace na aktuálním natočení

Kapitola 5

Testování

Tato kapitola se bude zabívat testováním aplikace a to jak z pohledu jejího výkonu v různých prohlížečích a na různých počítačích. Druhá sekce se pak bude zabívat názory uživatelů na aplikaci a jejich hodnocením.

5.1 Výkon aplikace

K měření výkonu aplikace jsem použil výše zmíněné měřiče FPS. Veškeré testování probíhalo na 3 různých strojích z nichž 2 měli jak dedikovanou tak integrovanou grafickou kartu. Na všech počítačích byl nainstalován operační systém Windows 8. Testování také probíhalo ve 2 různých prohlížečích - Google Chrome a Mozzila Firefox. Tyto prohlížeče využívají jiný JavaScriptový interpret a výsledky porovnání byly až zarážející. Nyní už samotná tabulka s daty. Hodnoty v buňkách značí naměřené FPS.

Z tabulky je možné vyčíst, že Chrome má mnohem rychlejší JavaScriptový interpret. Proto důrazně doporučuji spouštět jí právě pod Chrome. Další zajímavostí je, že papírově slabší počítač s Core I3 má lepší výsledky než ten s Core I5, pro ten hovoří mírně i výkon grafické karty. Jenže tento počítač musí vykreslovat mnohem více pixelů, proto má slabší FPS. Mohlo by se předpokládat, že alespoň výkon fyzikálního frameworku by měl být lepší, protože ten na rozlišení nezávisí. Jenže metoda pro simulaci fyziky je volána z renderovací smyčky - z toho vyplívá, že fyzikální FPS nikdy nemohou překonat rendrované FPS. Z tohoto měření lze vyvodit jednu radu - pokud aplikaci spouštíte na notebooku s přepínatelnou

Procesor	RAM	Rozlišení	Grafická karta	Prohlížeč	FPS	FFPS
Core I5 - 2,56GHz	6GB	HD+	INTEL HD 3000	Chrome	35	35
Core I5 - 2,56GHz	6GB	HD+	INTEL HD 3000	Firefox	31	20
Core I5 - 2,56GHz	6GB	HD+	Radeon HD7650M	Chrome	53	50
Core I5 - 2,56GHz	6GB	HD+	Radeon HD7650M	Firefox	-	-
Core I3 - 2,26GHz	4GB	HD Ready	INTEL HD 3000	Chrome	38	30
Core I3 - 2,26GHz	4GB	HD Ready	INTEL HD 3000	Firefox	24	15
Core I3 - 2,26GHz	4GB	HD Ready	Nvidia GT620M	Chrome	59	50
Core I3 - 2,26GHz	4GB	HD Ready	Nvidia GT620M	Firefox	34	17
Dual-Core T4200 - 2GHz	4GB	HD Ready	INTEL GM45	Chrome	30	19
Dual-Core T4200 - 2GHz	4GB	HD Ready	INTEL GM45	Firefox	19	12

Tabulka 5.1: Porovnání zobrazovaných FPS na různých počítačích

grafikou, povolte webovému prohlížeči využívání dedikované grafické karty. Tím se zároveň zlepší i výkon fyziky.

5.2 Hodnocení uživatelů

Po zkušnostech z jiných projektů, jsem dospěl k názoru, že dotazníkové hodnocení je téměř k ničemu. Podle mých zkušeností uživatel aplikaci pouze zlehka projde. Na otázku, kde se očekává textová odpověď neodpoví nebo v lepším případě odpoví 3mi slovy. A otázky s odpovědmi A B C D, nebo s hodnocením zase neposkytují dostatečnou informační hodnotu. Proto jsem se rozhodl pro beta testování s dohledem. Testování tedy probíhalo tak, že jsem stál za testerem, sledoval jeho reakce a hned poslouchal jeho zpětnou vazbu. Díky tomuto způsobu jsem v průběhu vývoje změnil nebo upravil mnoho drobností, které mě přišli samozřejmé, ale uživatelům činili téměř nepřekonatelné obtíže. Také jsem oběvil velkou spoustu bugů, jelikož každý uživatel postupoval při ovládání hry trochu jinak, takže narazil na chyby, na které já ne. Zde je malý seznam vybraných vylepšení, které jsem provedl na základě připomínek.

- Ovladač odpalování kuličky - původně se zobrazoval kousek napravo od místa, kde uživatel klikl na kuličku. Díky tomu bylo vidět na odpalovanou kuličku. Pro nastavení síly odpočtu je nutné kliknout přímo na tento HTML element, ale uživatelé opakovaně klikali stále na kuličku. Proto se nyní tento prvek zobrazí přímo pod kurzorem myši.
- Opakovaná možnost odpálení kuličky - původně šla kulička odpálit pouze jednou. Pak se zablokovala a do spuštění simulace nešlo změnit sílu odpalu. Uživatelům ale přišlo otravné, že když netrefí čas kliknutí musí restartovat celý level. Proto jsem tuto blokaci odstranil a kuličce jde měnit sílu odpalu.
- Zoomování - původně se kamera mohla pohybovat pouze po povrchu myšleného válce okolo středu scény. Jenže uživatelům se občas špatně cílili přidávané dílky a proto jsem přidal možnost zoomování - mění se tím poloměr tohoto válce.
- Automatické zobrazení tipu - původně byl tip v levém dolním rohu schovaný a když uživatel nevěděl jak na to tak si ho měl kliknutím zobrazit. Ovšem v praxi to bylo jinak - když uživatel nevěl co dělat tak na sobě dával znát určitou flustraci a nechuť ke hře. Po permamentní zobrazení tipu ho všichni testovaní uživatelé vždy přečetli a ve hře postupovali notně rychleji.

Shrnutí uživatelských názorů na aplikaci bude složitější. Vzhledem k tomu, že mi ho říkali přímo do očí, tak šlo především o pozitivní kritiku. Když shrnu opakující se názory, tak uživatelé pozitivně hodnotili navigaci v 3D prostoru, grafické zpracování (především hlavního menu) a ducha při přidávání dílků. Naopak jim vadil fakt, že pokud při kliknutí při přidávání dílku pohnou myší, dílek se neumístí a pootočí se kamera. K tomu mohu říct, že nějaké kompromisní přidávání by na druhou stranu znamenalo nekomfortní otačení kamerou, alespoň prvních pár pixelů. Druhá velká stížnost mířila na prvek odpalující kuličky. Při vytížení totiž jQuery někdy nepředá event kliknutí objektu a nic se nestane. To je nepříjemné ale neporařilo se mi to opravit.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a vytvořit 3D webovou hru s fyzikou využívající technologii WebGL. Výsledkem práce je hra PuzzleBall, ve které hráč umisťuje do hry dílky, tak aby se kulička po odpálení dostala pomocí fyzikálních principů do jamky. Hra nepotřebuje instalaci, stačí zadat její webou adresu a hra se stustí v okně webového prohlížeče. Hra byla testována v Google Chrome a Mozzila Firefox, doporučuje se však použití prvního z jmenovaných.

Hlavní brzdou aplikace je fyzikální framework, který neposkytuje takové možnosti, jako pokročilé frameworky pro OpenGL. Pokud by došlo k jeho vylepšení, samotné aplikaci by se otevřeli nové možnosti.

V průběhu práce jsem se velmi dobře seznámil s technologií WebGL a jejimi možnostmi. Dále jsem se naučil pracovat s frameworky THREE.js a Physi.js, které jsem používal pro zjednodušení práce. Bylo potřeba vyřešit velké množství problémů od navigace po 3D scéně přes označování objektů ve scéně a přidávání dílů po odpalování kuliček.

Při další práci na projektu bych vytvořil uživatelský editor pro tvorbu vlastních úrovní. Dále bych do hry přidával další díly a modely a z nich vytvářel nové úrovně. Hra by se takto mohla velmi rychle rozvinout. Dále bych přidal sociální funkce pro sdílení výsledků. Posledním vylepšením by bylo přidání zvukových efektů a hudby.

Literatura

[1] Vyjádření Microsoftu k podpoře WebGL. http://blogs.technet.com/b/srd/archive/2011/06/16/webgl-considered-harmful.aspx, 2011-6-16.

[2] Rozhovor s vedoucím projektu MONO Miguelem de Icaza. http://www.infoq.com/news/2012/05/Miguel-Moonlight, 2012.

[3] Uživatelé internetu ve světě. http://www.internetworldstats.com/stats.htm, 2012.

[4] Videohry v Evropě: Uživatelský průzkum. http://www.isfe.eu/sites/isfe.eu/files/attachments/euro_summary_-_isfe_consumer_study 2012.

[5] Využití appletů na webu. http://trends.builtwith.com/docinfo/Applet, 2012.

[6] Cantor, D.; Jones, B.: *WebGL Beginner's Guide*. Packt publishing, 2012, iSBN 978-1-84969-172-7.

[7] Parisi, T.: *WebGL Up and Running*. O'REILLY, 2012, iSBN 978-1-449-32357-8.

[8] WWW stránky: Statistiky používaných rozlišení z prosinece 2012. http://gs.statcounter.com/#resolution-ww-monthly-201212-201212-bar.

Dodatek A

Obsah DVD

Na přiloženém DVD se nacházejí následující soubory:

- source/ - složka se zdrojovými kódy
- source/css/ - obsahuje kaskádové styly
- source/img/ - obsahuje obrázky a textury použité ve hře
- source/js/ - obsahuje soubor s JavaScriptovým jádrem aplikace
- source/libs/ - obsahuje externí knihovny
- source/models/ - obsahuje JSON modely do hry
- source/index.html - spouštěcí soubor aplikce
- plakat.jpg - plakát ve formátu jpg
- projekt.pdf - tato práce ve formátu pdf

Dodatek B

Manuál

Pro zprovoznění aplikace je nutné nahrát veškeré zdrojové soubory na webový server a velmi důležité je zachovat adresářovou strukturu. Pro spuštění aplikace pak zobrazíme stránku source/index.html.

Pro spuštění aplikace bez instalace je možné navštívit stránku :
http://www.stud.fit.vutbr.cz/~xmaixn01/IBP/

Webovou aplikaci doporučuji spouštět ve webovém prohlížeči Google Chrome. Funguje i v Mozzila Firefoxu, ale ten má dle testů skoro 2x pomalejší interpret JavaScriptu a hra je mnohem pomalejší.

Co se týče ovládání - to je velmi intuitivní. V případě nejasností doporučují sledovat Tip v levém dolním rohu.

Dodatek C

Plakát