Framework PlayCanvas a jeho praktické využití

Jiří Čepela

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jiří Čepela**
Osobní číslo:	**A19012**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Framework PlayCanvas a jeho praktické využití**
Téma práce anglicky:	**PlayCanvas Framework and its Use in Practice**

Zásady pro vypracování

1. Seznamte se s webovými technologiemi HTML a CSS. V práci popište vlastnosti a funkce v nejnovějších verzích.
2. Prostudujte programovací jazyk JavaScript a framework PlayCanvas, který se používá především pro vývoj her ve webových prohlížečích.
3. Vypracujte základní výukové tutoriály pro framework PlayCanvas, které usnadní jeho studium začínajícím programátorům.
4. Pomocí frameworku PlayCanvas navrhněte a realizujte komplexnější aplikaci, která bude demonstrovat jeho možnosti.
5. Vytvořenou aplikaci odlaďte a zdrojový kód doplňte o vhodné komentáře.

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

Seznam doporučené literatury:

1. PlayCanvas Manual. Playcanvas [online]. London: PlayCanvas, 2014 [cit. 2021-10-13]. Dostupné z: https://developer.playcanvas.com/en/
2. MDN Web Docs. Resources for developers, by developers. [online]. San Francisco: MDN contributors, 2021 [cit. 2021-10-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Building_up_a_basic_demo_with_PlayCanvas/editor
3. W3Schools. JavaScript Tutorial [online]. Norway: Refsnes Data, 2018 [cit. 2021-10-13]. Dostupné z: https://www.w3schools.com/js/default.asp
4. FLANAGAN, David. JavaScript: the definitive guide : master the world's most-used programming language. Seventh edition. Beijing: O'Reilly, 2020 [cit. 2021-10-13]. ISBN 978-1491952023.
5. GRANT, Keith J. CSS in Depth. New York: Manning, 2018 [cit. 2021-10-13]. ISBN 978-1617293450 .

Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 20. 5. 2022

Jiří Čepela, v. r.
podpis studenta

ABSTRAKT

Cílem této práce je seznámení se s webovými technologiemi a s open-source frameworkem PlayCanvas. Následně popsání jednotlivých funkcí a možností frameworku. Dále také vytvoření podrobných výukových tutoriálů pro usnadnění studia pro začínající uživatele frameworku. Dalším cílem práce je vytvoření komplexnější aplikace, která bude sloužit k demonstraci možností frameworku.

Klíčová slova: framework, PlayCanvas, JavaScript, HTML, CSS, entita, skript

ABSTRACT

The aim of this thesis is to introduce web technologies and the opensource framework PlayCanvas. Then, the individual functions and possibilities of the framework are described. Furthermore, the creation of detailed tutorials to facilitate learning for new users of the framework. Another goal is to create complex application that will be used to demonstrate the capabilities of the framework.

Keywords: framework, PlayCanvas, JavaScript, HTML, CSS, entity, script

Poděkování:

Touto bakalářskou prací chci vyjádřit poděkování vedoucímu mé bakalářské práce Ing. Pavlu Pokornému, Ph.D., za odborné vedení, rady a připomínky, které mi pomohly k vypracování této práce. Dále chci poděkovat své rodině, přítelkyni a přátelům za jejich neustálou podporu během celého studia.

Prohlášení:

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **9**

I TEORETICKÁ ČÁST **10**

1 WEBOVÉ TECHNOLOGIE **11**

1.1 HTML 11

1.1.1 Historie HTML 11

1.1.2 HTML5 12

1.1.3 HTML syntaxe 13

1.2 CSS 14

1.2.1 CSS syntaxe 14

1.2.2 Propojení CSS s HTML 14

1.2.3 Historie CSS 15

1.2.4 CSS 3 15

1.3 JAVASCRIPT 16

1.3.1 JavaScript syntaxe 16

1.3.2 Historie jazyka JavaScript 16

1.3.3 ES6 17

1.3.4 WebGL 17

1.3.4.1 Výhody WebGL 17

2 FRAMEWORK **18**

2.1 VYUŽITÍ FRAMEWORKŮ 18

2.2 VLASTNOSTI DOBRÉHO FRAMEWORKU 18

2.3 WEBOVÉ FRAMEWORKY 19

2.4 FRAMEWORKY VS KNIHOVNY 19

3 PLAYCANVAS **20**

3.1 DOKUMENTACE 20

3.2 KLÍČOVÉ VLASTNOSTI 20

3.2.1 Editor 21

3.2.2 Editor kódu 21

3.2.3 Assety 21

3.2.4 Cloud 21

3.2.5 Publikace 21

3.3 POSTUP VYTVÁŘENÍ 3D APLIKACE 21

3.3.1 Vytváření a nahrávání assetů 22

3.3.2 Sestavení scény 22

3.3.3 Přidání interaktivity 22

3.3.4 Publikace aplikace 22

3.4 UŽIVATELSKÉ PROSTŘEDÍ 22

3.4.1 Menu a panel nástrojů 23

3.4.2 Hierarchie 24

3.4.3 Inspector 25

3.4.4 Assety 25

3.4.5 Viewport 27

3.5 SIMULACE FYZIKY 29
3.5.1 Typy simulace 29
II PRAKTICKÁ ČÁST 30
4 VYUKOVÉ TUTORIÁLY 31
4.1 ZÁKLADY PRO PRÁCI S FRAMEWORKEM PLAYCANVAS 31
4.2 PŘIDÁNÍ A NASTAVENÍ KOMPONENTŮ ENTITY 32
4.3 SVĚTLO A SKYBOX 32
4.4 POHYB ENTITY 33
4.5 ČÁSTICOVÉ EFEKTY A ZVUK 34
4.6 UŽIVATELSKÉ ROZHRANÍ 34
5 TVORBA HRY 35
5.1 PŘÍPRAVA ASSETŮ 35
5.2 VYTVOŘENÍ PROJEKTU A SCÉNY 36
5.2.1 Vložení assetů do prostředí 37
5.3 PŘIDÁNÍ KOMPONENTŮ 37
5.3.1 Animace a zvuk 39
5.3.2 Částicové efekty 40
5.3.3 Skripty 41
5.4 PROGRAMOVÁNÍ 41
5.4.1 Pohyb hráče a kamery 41
5.4.2 Pohyb zvířete 44
5.4.3 Měření času 44
5.4.4 Ukončení hry 45
5.5 VYTVOŘENÍ LEVELU 46
5.5.1 Hrací plocha 46
5.5.2 Přidání překážek 46
5.5.3 Přidání zvířat 47
5.5.4 Prostředí 47
5.5.5 Skybox a světlo 48
5.6 UŽIVATELSKÉ ROZHRANÍ 49
5.6.1 Rozhraní pro start a hru 49
5.6.2 Rozhraní pro konec hry 50
5.6.3 Propojení rozhraní s hrou 51
5.7 OPTIMALIZACE 53
5.8 PROBLÉMY PŘI VÝVOJI 54
5.9 PUBLIKACE HRY 54
ZÁVĚR 55
SEZNAM POUŽITÉ LITERATURY 56
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 58
SEZNAM OBRÁZKŮ 59
SEZNAM TABULEK 61
SEZNAM PŘÍLOH 62

ÚVOD

Vývoj nových aplikací se stává s příchodem moderních technologií stále mnohem jednodušší. Jednou z těchto technologií je framework PlayCanvas, který se využívá pro tvorbu trojrozměrných interaktivních aplikací nebo her. Dále framework PlayCanvas podporuje simulování fyziky, 3D animace a prostorový zvuk.

Framework má vlastní platformu, která umožňuje vytvářet aplikace prostřednictvím webového rozhraní v prohlížeči. Práce s frameworkem PlayCanvas je dostupná na každém prohlížeči, který podporuje aplikační programové rozhraní WebGL. PlayCanvas podporuje psaní skriptů v jazyce JavaScript, který slouží k funkčnosti celé aplikace. Kód lze upravovat přímo v editoru, který framework poskytuje v rámci svého rozhraní. Samotné vytváření aplikací je tak přívětivější pro uživatele. Veškerá práce se následně ukládá na cloud, který umožňuje úpravy z více počítačů přes frameworkem podporovaný prohlížeč. Samotný framework také nabízí hosting služeb zdarma pro vytvořené aplikace.

I když má framework rozsáhlou dokumentaci a základní tutoriály, nemá velkou komunitu, která by vytvářela naučné materiály pro nové uživatele. Potenciální uživatelé jsou tak většinou odrazeni malým množstvím pomoci ze strany komunity.

Hlavním cílem této práce bylo seznámení se s frameworkem PlayCanvas a jeho následný podrobný popis frameworku PlayCanvas. Kvůli tomuto důvodu bylo potřeba se podrobně seznámit s frameworkem PlayCanvas. V práci byly vytvořeny výukové tutoriály, které slouží začínajícím uživatelům při práci s frameworkem. Výukové tutoriály popisují širokou škálu možností frameworku a poskytují ukázky využití jednotlivých funkcí.

Dalším cílem práce bylo vytvoření komplexnější aplikace za využití frameworku PlayCanvas. Aplikace měla prezentovat, co nejširší spektrum funkcí a možností, které lze ve frameworku využívat. Finální podoba aplikace byla publikována tak, aby byla volně dostupná na webových stránkách frameworku.

I. TEORETICKÁ ČÁST

1 WEBOVÉ TECHNOLOGIE

Jedná se o technologie, při kterých lze za jejich pomoci vytvořit webové stránky. Pro vytvoření webových stránek je potřeba využít ze tří základních nástrojů pro tvorbu webu.

Prvním nástrojem pro tvorbu je značkovací jazyk HTML, který poskytuje strukturu nebo způsob jakým se text a digitální obsah zobrazí na webu. Druhým základním nástrojem jsou kaskádové styly (CSS), které se využívají k nastavení vizuálních vlastností, jako jsou barvy, formát a rozvržení jednotlivých prvků z HTML. Posledním nástrojem je programovací jazyk JavaScript. JavaScript lze použít k vytvoření funkcí pro prvky v závislosti na akci uživatele. Například provedení výpočtu nebo pro zobrazení či schování elementu při zmáčknutí tlačítka.[6]

1.1 HTML

Jazyk HTML (hypertext markup language) je nejrozšířenějším hypertextovým značkovacím jazykem. Hypertext v názvu HTML definuje propojení mezi webovými stránkami. Jedná se tedy o základní technologii při tvorbě webových stránek. Používá se také pro tvorbu webových stránek za pomocí značkovacího jazyka.

Tento jazyk se využívá k anotaci textu tak, aby mu stroj mohl porozumět a mohl s textem odpovídajícím způsobem pracovat. Většina značkovacích jazyků je čitelná i pro člověka. HTML je jazyk, který prohlížeč využívá k manipulaci s textem, obrázky a dalším digitálním obsahem, a aby jej zobrazil v požadovaném formátu. [6]

1.1.1 Historie HTML

HTML značkovací jazyk vytvořil v roce 1991 Tim Berners-Lee. První verzí jazyka byla verze HTML 1.0. Tato verze však byla využívána pouze malým množství developerů a tím pádem i malou komunitou.

Další verze vyšla v roce 1995 s příchodem HTML 2.0. Verze 2.0 obsahovala veškeré funkce předchozí verze a také přidala další nové funkce, které se zavedly jako standardní značkovací jazyk pro návrh a vytvoření webových stránek.

V roce 1997 vyšla verze HTML 3.0, která obsahovala vylepšené funkce. Tyto funkce vývojářům poskytovaly větší množství možností při vytváření webových stránek.

Poté přišla v roce 1999 verze HTML 4.01, která začala být široce využívaná a úspěšná až do doby, než vyšla nejnovější verze HTML5, která vyšla v roce 2014. [7]

1.1.2 HTML5

HTML5 je pátou v pořadí a momentálně nejaktuálnější verzí značkovacího jazyka HTML. HTML5 výrazně zlepšilo interaktivitu uživatelů a snížilo zátěž na zařízení při prohlížení webu. Dále HTML5 přineslo také spoustu novinek, některé z nich jsou popsány v tabulce níže (Tabulka 1.) [7]

Tabulka 1. Rozdíly mezi HTML a HTML5 [7]

HTML	HTML5
Nepodporuje audio a video bez použití flash přehrávače.	Podporuje audio a video za použití speciálních značek.
Používá cookies pro ukládání dočasných dat.	Používá SQL databáze a mezipaměť aplikace pro ukládání off-line dat.
Nepodporuje běh JavaScriptu v prohlížeči.	Podporuje běh JavaScriptu v pozadí.
Podporován starými prohlížeči.	Podporován všemi novými prohlížeči.
Starší verze jsou méně přívětivé pro mobilní zařízení.	Více přívětivé pro mobilní zařízení.
Malé množství značek pro webovou strukturu.	Přidané značky pro tvorbu webové struktury.
Vektorová grafika možná pouze za použití různých technologií.	Vektorové grafika je součástí HTML5.
Nepodporuje drag and drop efekt.	Podporuje drag and drop efekt

1.1.3 HTML syntaxe

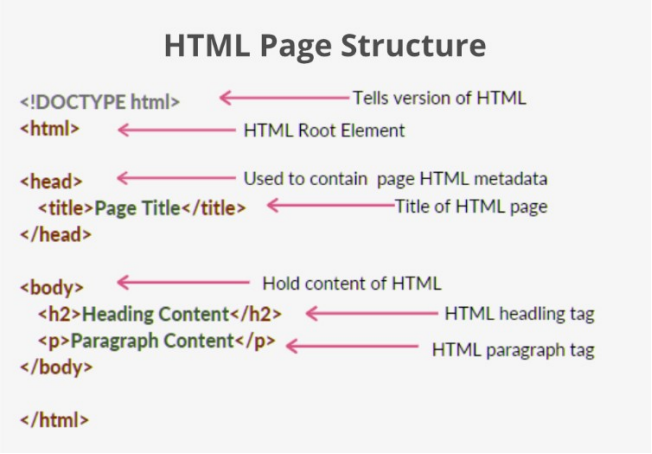
Značkovací jazyk se používá k definování textového dokumentu za pomoci značek, které vytvářejí strukturu webových stránek. Značky definují, jaká manipulace má být s textem provedena. Značkovací jazyk HTML se skládá z několika elementů, které se využívají k obalení různých částí obsahu tak, aby vypadal nebo se choval určitým způsobem.

Každý element obsahuje začínající značku, která udává, kde element začíná a jaký bude mít na obsah vliv. Dále se v elementu objevuje jeho obsah, obsahem elementu může být prostý text nebo také další element. Na konci každého elementu se nachází ukončení značky, které udává, kde element končí.

Struktura webové stránky je složena ze základních elementů. Na začátku každé stránky se nachází deklarace typu značkovacího jazyka souboru. Tato deklarace sděluje, v jakém značkovacím jazyce je aktuální stránka napsána. Pod deklarací se nachází první element se značkou <*html*>. Ten se využívá k definici kořenového elementu dokumentu HTML. Značka sděluje prohlížeči, že se jedná o HTML dokument. Tento element dále v sobě obsahuje všechny další elementy.

V elementu se značkou <*html*> se nachází další element se značkou <*head*>. Značka se používá k definování hlavičky dokumentu HTML. Hlavička obsahuje informace týkající se dokumentu. Prvky vepsané uvnitř elementu, nejsou vidět v popředí webové stránky.

Pod elementem se značkou <*head*> se nachází element se značkou <*body*>. Tato značka se využívá k zobrazení veškerého viditelného obsahu webové stránky. Vše, co se do elementu se značkou <*body*> vypíše, se zobrazí na webové stránce. [6]



Obrázek 1. Struktura webové stránky [6]

1.2 CSS

CSS neboli kaskádové styly jsou jednoduchým jazykem, které mají za účel zjednodušit proces tvorby webových stránek. Slovo „Kaskádové“ označuje pravidla, která určují, jak jsou selektory upřednostňovány při změně vzhledu stránky. Jedná se o velice důležitou funkci, protože složité webové stránky mohou obsahovat také i tisíce pravidel CSS.

Dále CSS umožňuje aplikaci stylů na webové stránky. Lze jej použít pro základní stylování textu dokumentu, k tvorbě rozložení obsahu a také například k animaci. Styly byly navrženy v roce 1994 a poprvé implementovány v prohlížeči Internet Explorer 3 v roce 1996. [5]

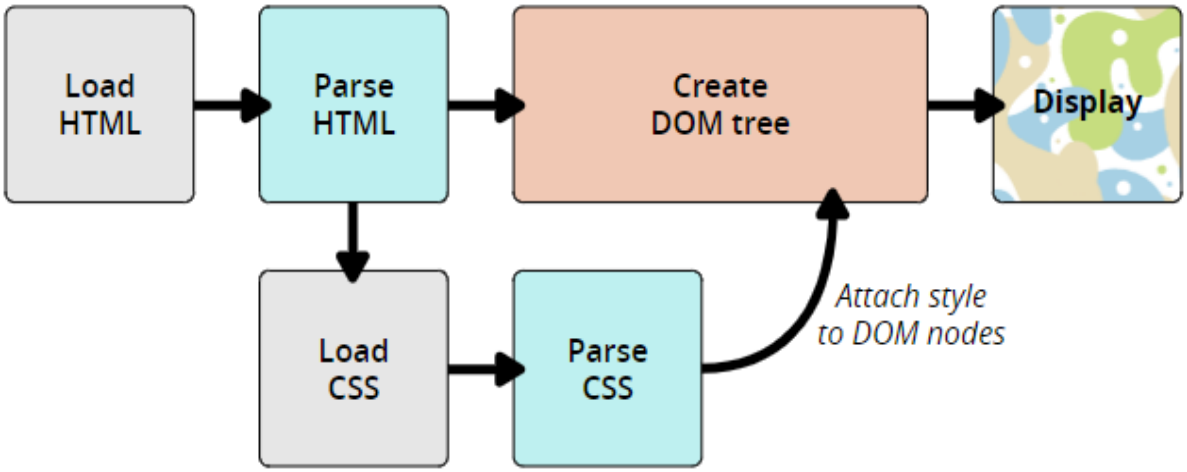
1.2.1 CSS syntaxe

Jazyk CSS je založen na pravidlech. Samotný tvůrce si nadefinuje pravidla určující skupiny stylů, které se mají použít na určité prvky nebo skupiny prvků na webové stránce. Tyto pravidla se otevírají za pomoci selektoru, který vyhledá prvek v příslušném HTML. Na daný prvek jsou následně naneseny styly podle pravidel.

Do složených závorek se následně zapisují deklarace, které mají podobu dvojice (vlastnost, hodnota). Každá dvojice má vlastnost prvku, který lze nadefinovat a hodnotu, které se přiřazují vlastnosti. Některé vlastnosti mohou mít také i více hodnot. Tyto dvojice jsou rozděleny dvojtečkou a zakončeny středníkem. Soubor stylů CSS obsahuje mnoho po sobě jdoucích zapsaných pravidel. [5]

1.2.2 Propojení CSS s HTML

Při zobrazení HTML dokumentu v prohlížeči, se musí také kombinovat jeho obsah v podobě kódu s informacemi o stylu. Dále se dokument zpracovává v několika fázích. Jako první se načítá HTML. Načtený obsah HTML se převede do DOM (Document Object Model). DOM představuje dokument v počítačové paměti. Prohlížeč následně načte většinu zdrojů, na které dokument HTML odkazuje, jako jsou například CSS a digitální obsah (obrázky a videa). Dále prohlížeč analyzuje načtené CSS a roztřídí pravidla podle typů selektoru. Na základě nalezených selektorů zjistí, která pravidla se mají použít na jednotlivé uzly v DOM, a podle potřeby k nim připojí styl. Tento krok se nazývá vykreslovací strom. Strom je rozložen do struktury, v jaké by se měl zobrazit po aplikaci pravidel. Poté se spustí fáze vykreslení, kdy se zobrazí vizuální zobrazení stránky na obrazovce. [15]



Obrázek 2. Načtení CSS [15]

1.2.3 Historie CSS

CSS vznikly v roce 1994. Poprvé je navrhl Håkon Wium Lie 10. října 1994 při práci s autorem HTML Timem Bernersem-Lee. Od jejich vzniku se postupně vyvíjeli a vylepšovali.

V roce 1996 byla vydána verze CSS 1 na kterou následně navázala v roce 1998 verze CSS 2, která navazovala na předešlou verzi. Po vydání CSS 2.1 bylo samotné CSS modulárně upraveno tak, že každý modul je možné vyvíjet nezávisle na ostatních. To umožňuje nejen samostatné vrstvení existujících modulů, ale i vytváření nových modulů, které buď definují zcela nové sady funkcí, nebo rozšiřují stávající funkce. Každá verze CSS navazuje na předešlou, přidává větší množství funkcí a opravuje problémy předešlých verzí. Vše, co bylo zveřejněno po verzi CSS 2.1 se označuje jako verze CSS 3. [5]

1.2.4 CSS 3

CSS 3 je nejnovější a v současnosti nejpoužívanější verzí CSS s podporou XHTML (kombinace HTML a XML). Zaměřuje se především na modularizaci. Různé moduly procházejí různými fázemi procesu doporučení.

CSS 3 má podporu pro téměř všechny nejnovější webové prohlížeče. CSS 3 má také několik nových vlastností CSS. Dále obsahuje nové selektory spolu s novými kombinátory a novými pseudoelementy. Podporuje animace, které nebyly součástí dřívějších rozšíření. Byly přidány také různé vlastnosti, jako jsou transformace, gradienty a přechody pro animační efekty na webových stránkách. Mezi nejnovější doplňky patří například *border-radius, box-shadow, flex-box* a CSS grid. [8]

1.3 JavaScript

JavaScript je textový programovací jazyk, používaný na straně klienta i serveru, který umožňuje vytvářet interaktivní webové stránky. Zatímco HTML a CSS jsou jazyky, které webovým stránkám dodávají strukturu a styl, JavaScript dodává webovým stránkám interaktivní prvky. Dále umožňuje implementovat složité funkce, vytvářet dynamicky se aktualizující obsah, ovládat multimédia, animovat obrázky a spoustu dalších věcí spojených s vývojem webových stránek. [4]

1.3.1 JavaScript syntaxe

JavaScript lze implementovat pomocí příkazů, které jsou umístěny uvnitř značek <*script*> v HTML. Značky, které obsahují JavaScript, je možno umístit kamkoli v rámci webové stránky, ale obvykle se doporučuje, aby byly umístěny v rámci hlavičky HTML souboru. Značka <*script*> upozorňuje program prohlížeče, aby začal veškerý text mezi těmito značkami interpretovat jako skript. [3]

Jazyk JavaScript rozlišuje malá a velká písmena. To znamená, že klíčová slova jazyka jako jsou proměnné, názvy funkcí a jakékoli další identifikátory musí být vždy psány jednotně s dodržením velikosti písmen.

JavaScript ignoruje mezery, odsazení a nové řádky, které se objevují v programu. Mezery, odsazení a nové řádky je možné v programu libovolně používat. Programy tak lze formátovat a odsazovat úhledným a konzistentním způsobem, který usnadňuje čtení a pochopení kódu pro vývojáře. [14]

1.3.2 Historie jazyka JavaScript

Webové stránky před existencí jazyka JavaScript byly velmi statické. Tabulky, data a odkazy byly pevně zakódovány do jazyka HTML a jakýkoliv druh dynamické funkce byl zakódován v hlavičce dokumentu HTML jako záhlaví, které nebylo na webové stránce vidět.

JavaScript vymyslel Brendan Eich a v roce 1997 se stal standardem ECMA. Oficiální název jazyka je ECMAScript. První iterace jazyka JavaScript se nejmenovala JavaScript. Jmenovala se Mocha. Tento jazyk byl vytvořen jako jazyk vyšší úrovně pro návrháře i začínající programátory. Když byl programovací jazyk Mocha dodáván s prohlížečem Netscape Navigator 2.0, stal se jeho produkční název LiveScript a v pozdějších verzích JavaScript. [9]

1.3.3 ES6

JavaScript ES6, známý také jako ECMAScript 2015 nebo ECMAScript 6, je nejnovější verzí jazyka JavaScript, která byla představena v roce 2015. Verze umožňuje psát kód chytrým způsobem, díky němuž je kód modernější a čitelnější.

ES6 zavedlo několik klíčových funkcí, jako jsou *const*, *let*, výchozí parametry a mnoho dalšího. Od vydání verze z roku 2015, verze dostává každý rok update s novými funkcemi a standardizací. [9]

1.3.4 WebGL

WebGL je grafické aplikační programové rozhraní (API) vytvořené pro použití ve webových aplikacích. Je založeno na vestavěném standardu OpenGL.

WebGL se využívá k vytváření interaktivních grafických aplikací na webu nezávisle na platformě. Používá se nejen k vykreslení grafiky 2D a 3D her, ale také k urychlení funkcí webových editorů obrázků a jejich efektů a také fyzikálních simulací.

WebGL původně vznikl z experimentu s elementem canvas. Experiment začal v roce 2007 americko-srbský softwarový inženýr Vladimir Vukicevic pod společností Mozilla. V roce 2011 vznikl WebGL, který je v současné době navržen a udržován neziskovou skupinou Khronos Group jako dvoudílný kód, který se dělí na řídicí kód a kód shaderu. Řídicí kód je napsán v jazyce JavaScript a zpracovává se na procesoru. Přičemž kód shaderu, který je napsán v OpenGL ES, je zpracováván na GPU. [10]

1.3.4.1 Výhody WebGL

Práce s WebGL API přináší velké množství výhod, mezi kterými jsou:

- **Rozšíření podpory mobilních prohlížečů** – WebGL podporuje mobilní prohlížeče jako je iOS safari, Android Browser a Chrome pro Android.
- **Otevřený zdrojový kód** – každý má přístup do zdrojové knihovny kódu a je snazší tak pochopit, jak WebGL funguje.
- **Snadné vytvoření** – WebGL je integrováno v rámci HTML 5, není potřeba vytvářet nic dodatečného. K napsání aplikace tak stačí pouze textový editor a prohlížeč.
- **Není potřeba kompilace** – pro spuštění skriptu není potřeba soubor kompilovat. Aplikace vytvořené za pomoci WebGL také není potřeba kompilovat. Pouze stačí otevřít soubor v prohlížeči. [21]

2 FRAMEWORK

Framework neboli softwarový framework je platforma, která poskytuje základ pro vývoj softwarových aplikací. Představit si jej lze jako šablonu fungujícího programu, kterou je možné selektivně upravovat přidáváním kódu. Využívá sdílené zdroje – například knihovny, obrazové soubory a referenční dokumenty, které následně spojuje do jednoho balíčku. Tento balíček lze upravit podle konkrétních potřeb projektu. Pomocí frameworku může vývojář přidávat nebo nahrazovat vlastnosti, aby aplikaci poskytl nové funkce. [11]

2.1 Využití frameworků

Účelem frameworku je pomoc při vývoji, a poskytnout tak standardní a nízko úrovňové funkcionality, aby se vývojáři mohli soustředit na prvky, které dělají projekt jedinečným.

Použití vysoce kvalitních a předem prověřených funkcí, zvyšuje spolehlivost softwaru, urychluje programování a zjednodušuje testování. Díky aktivní základně uživatelů a neustálému vylepšování kódu pomáhají frameworky zvyšovat bezpečnost a nabízejí základnu podpory. Frameworky se primárně používají k úspoře času a peněz při vývoji. [11]

2.2 Vlastnosti dobrého frameworku

Existuje mnoho druhů frameworků, přičemž některé jsou oblíbenější než jiné. Vývojáři si často vybírají frameworky, které nejlépe znají, ale tento framework nemusí být pro danou práci ten pravý. Při rozhodování o výběru vhodného frameworku pro daný projekt je dobré raději zvážit následující vlastnosti:

- **Funkčnost** – možnost vybrání si frameworku, který poskytuje potřebné funkce pro daný projekt.
- **Konzistence** – framework může pomoci v konzistenci pro velké nebo distribuované týmy.
- **Dokumentace** – většina frameworku má rozsáhlou dokumentaci, která poskytuje proškolení pro práci s daným frameworkem.
- **Aktivní komunita** – frameworky jsou tak silné, jak silná je uživatelská základna podpory. Framework s aktivní uživatelskou základnou je vždy lepší při řešení problémů. [11]

2.3 Webové Frameworky

Webové aplikační frameworky (WAF) nebo webové frameworky (WF) podporují vývoj webových aplikací s webovými službami, webovými zdroji a webovými rozhraními API. Existují různé webové frameworky jak pro front-end (jak webová aplikace vypadá), tak pro back-end (jak aplikace funguje). [11]

Obrázek 3. Princip webového frameworku [11]

2.4 Frameworky vs knihovny

Frameworky mohou zahrnovat softwarové aplikace, integrace, čísla knihoven, nástroje a aplikační programovací rozhraní které integrují různé komponenty pro vývoj projektu nebo programu.

Knihovna je sada nízko úrovňových komponentů, které lze vyvolat za účelem dosažení určitého výsledku. Frameworky jsou velmi podobné knihovnám a jsou v dnešní době populární, ale mají vzhled, který je odlišuje od tradičních knihoven. [11]

3 PLAYCANVAS

PlayCanvas je open-source framework pro vývoj interaktivního webového obsahu. Framework je založený na API WebGL. PlayCanvas byl založen roku 2011 v Londýně ve Velké Británii a zveřejněn roku 2014 jako startup. Jeho zakladatelé jsou Dave Evans a Will Eastcott. V roce 2018 odkoupila startup společnost Snap Inc., která se chystala vstoupit do oblasti herního vývoje.

Vytvořené aplikace a nástroje jsou poháněny technologií HTML5. Platforma je hostována na webu, takže není potřeba nic instalovat a ke své práci je možno přistoupit z jakéhokoliv zařízení, na kterém běží podporovaný webový prohlížeč.

S použitím JavaScriptového programovacího jazyku lze tento framework použít pro široké spektrum aplikací s kvalitními grafickými výstupy – od jednoduchých 2D her až po 3D grafické simulace. Framework také podporuje vytváření aplikací pro virtuální realitu.

PlayCanvas je zdarma pro každého, kdo by si ho chtěl vyzkoušet. Po registraci obdržíte 1 GB volného místa na projekty a hosting projektu a publikované hry zdarma. V případě větších projektů lze verzi PlayCanvasu vylepšit na personální čí firemní verzi. Obě verze přidávají více úložného místa na cloudu a přidávají nové možnosti. [1]

3.1 Dokumentace

PlayCanvas obsahuje rozsáhlou dokumentaci ke všem svým funkcím. Ať už se jedná o samotné ovládaní prostředí nebo jednotlivé příkazy kódu. Dokumentace obsahuje také velké množství tutoriálů s postupem a obrázky, které jsou určené pro naučení se základních i pokročilých funkcí a možností frameworku. [1]

3.2 Klíčové vlastnosti

Framework PlayCanvas má určité vlastnosti, které ho oddělují od ostatních frameworků. Nejvýhodnější je, že není zapotřebí nic stahovat a dá se používat kdekoliv, kde je připojení k internetu. Tato vlastnost je zároveň také i mínusem, protože je vždy zapotřebí mít internet k dispozici, tak aby byl framework použitelný. Uživatel musí být tedy pro práci s PlayCanvasem neustále online.

3.2.1 Editor

Editor PlayCanvas je vizuální editační nástroj, který umožňuje vytvářet scény, aplikace a hry v krátkém čase. Editor se využívá ke správě assetů projektu, k přidávání interaktivity a ke komunikaci a práci s týmem. Editor umožňuje spolupráci v reálném čase, což znamená, že okamžitě lze vidět provedené změny, na které je možné okamžitě reagovat například vytvářením a testováním aplikací na všech zařízeních.[1]

3.2.2 Editor kódu

Editor kódu je online editor pro spolupráci v reálném čase, který umožňuje upravovat skriptové prostředky a také všechny ostatní textové prostředky, jako je JSON, HTML, CSS. [1]

3.2.3 Assety

PlayCanvas se využívá k vytváření a správě všech assetů, které je potřeba pro interaktivní webovou aplikaci. PlayCanvas přijímá všechny hlavní formáty 3D souborů, a navíc umožňuje nahrávání obrázků, zvuku a dalších typů aktiv. Framework má také svůj obchod, ve kterém se dají stáhnout zdarma assety či skripty. [1]

3.2.4 Cloud

Další vlastností frameworku je poskytnutý cloud, do kterého se ukládá projekt se všemi změnami, skripty a assety. Ke Cloudu lze přistoupit odkudkoliv. Jediné, co je potřeba, je prohlížeč a přístup k internetu. V základní verzi je uživateli poskytnuto 1 GB volného místa. [1]

3.2.5 Publikace

PlayCanvas poskytuje okamžitý a bezproblémový hosting pro vaši WebGL aplikaci jediným kliknutím. Dále podporuje okamžité stažení kompletního projektu připraveného k hostování na vlastním webovém serveru. [1]

3.3 Postup vytváření 3D aplikace

Vytváření 3D webových aplikací pomocí PlayCanvas je snadné. K využití celého potenciálu v PlayCanvas je zapotřebí napsat funkční kód v JavaScriptu. Sada nástrojů je navržena tak, aby umožnila vizuální úpravy projektu a jeho publikování s neuvěřitelnou jednoduchostí. [2]

3.3.1 Vytváření a nahrávání assetů

PlayCanvas podporuje širokou škálu základních assetů. Umožňuje také nahrávat vlastní assety například obrázky, 3D modely, zvukové soubory nebo vlastní formáty textových či binárních souborů. [2]

3.3.2 Sestavení scény

Editor PlayCanvas je vizuální stavební nástroj, který se používá ke konstrukci scén. Umožňuje sestavit hierarchii entit pomocí vestavěných komponentů, jako jsou 3D modely, kolize, částicové efekty a další. [2]

3.3.3 Přidání interaktivity

Za pomoci webového standartu JavaScript lze k jednotlivým entitám přiřadit jedinečné chování. Interaktivita lze také přidat v libovolném rozsahu od jednoduchého kliknutí až po plnohodnotnou online hru. [1]

3.3.4 Publikace aplikace

PlayCanvas umožňuje jednoduché zveřejnění vytvořené aplikace na internet za pomoci poskytnutého hostingu od PlayCanvas. Tento proces je zcela zdarma. Vytvořená aplikace lze také stáhnout pro vlastní hostování na webovém serveru. [1]

3.4 Uživatelské prostředí

Veškeré vytváření aplikace se odehrává v editoru, který funguje v prohlížeči. Obrázek editoru frameworku PlayCanvas je uveden v příloze P II této práce. Za pomoci PlayCanvas enginu editor vykresluje scény v reálném čase a každá změna lze vidět také ve spuštěné aplikaci. Hra se jednoduše spouští za pomoci tlačítka „*Launch*“. Uživatelské prostředí se dělí na pět základních částí.

První část se skládá z menu a panelu nástrojů. Menu obsahuje všechny dostupné příkazy editoru, panel nástrojů nabízí nejčastěji používané příkazy pro rychlejší přístup.

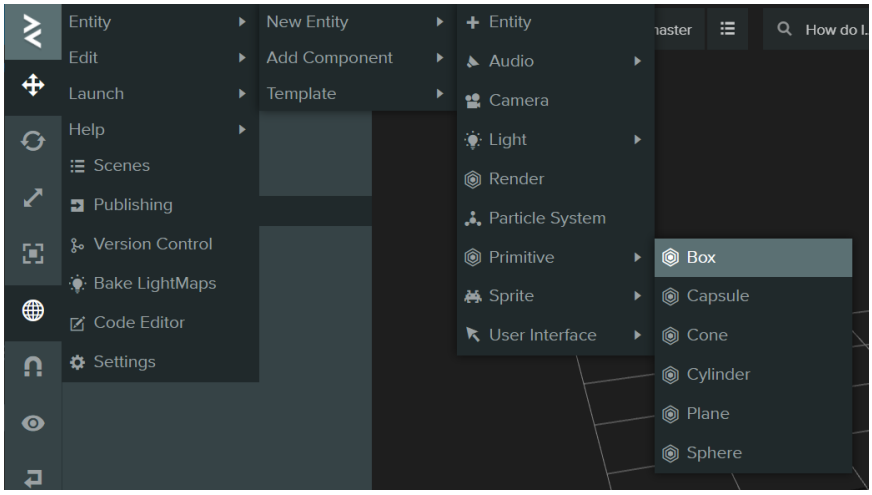
Druhou částí je hierarchie, která nabízí hierarchické zobrazení entit, které vytvářejí scénu. V hierarchii lze vybírat, duplikovat, odstraňovat a přesouvat entity. Dalším prvkem editoru je *Inspector*, který zobrazuje podrobné vlastnosti vybrané entity. *Inspector* se využívá také k nastavení entit.

Ve spodní části editoru se nachází zobrazení všech assetů v aktuálním projektu. Lze tak jednoduše vybírat a vyhledávat mezi assety. Se využívá také k jednoduchému vytvoření a nahrání nových assetů do projektu skrz import.

Nejdůležitější částí editoru je jeho *Viewport*, který zobrazuje trojrozměrnou scénu v reálném čase. *Viewport* umožňuje výběr, umístění a nastavení orientace jednotlivých entit. Obrázek *Viewportu.* [12]

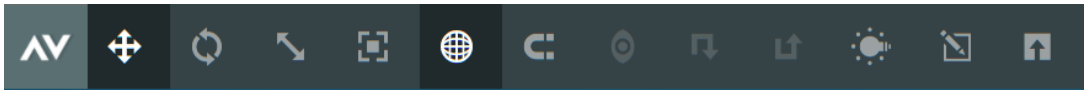
3.4.1 Menu a panel nástrojů

Menu je dostupné po kliknutí na ikonu PlayCanvas a obsahuje kompletní seznam všech příkazů, které můžete na scéně provádět. Při zapomenutí klávesové zkratky lze příkaz dohledat v nabídce menu.



Obrázek 4. Menu Editoru [12]

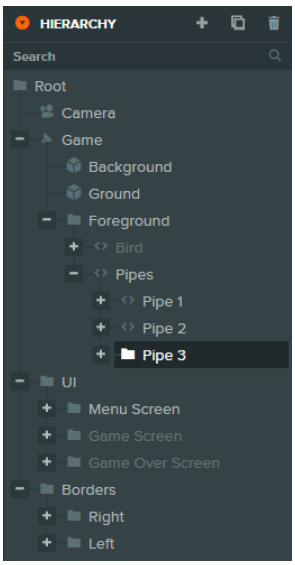
Panel nástrojů obsahuje běžné příkazy pro snadný přístup, z nichž nejužitečnější je tlačítko „*Launch*“. Tlačítko „*Launch*“ spustí instanci hry na samostatné kartě prohlížeče a načte vaši scénu. Můžete tak okamžitě začít testovat hru. [12]



Obrázek 5. Panel nástrojů [12]

3.4.2 Hierarchie

Panel Hierarchie zobrazuje stromové zobrazení celé scény, která je tvořena hierarchií entit. Scéna bude vždy obsahovat kořenovou entitu na vrcholu stromu. Všechny ostatní entity jsou pak přidávány do větví stromu.



Obrázek 6. Panel Hierarchie [12]

Hierarchie se využívá k nalezení entit. Všechny entity ve scéně jsou neustále viditelné. Kliknutím na entitu v hierarchii se entita vybere. Jakmile je entita vybrána, lze vytvořit novou podřízenou entitu nebo výběr odstranit za pomoci tlačítek v pravém horním rohu panelu hierarchie nebo otevřením kontextové nabídky pomocí kliknutí pravým tlačítkem myši. Pořadí, v jakém jsou entity uvedeny ve stromovém zobrazení je velice důležité, protože se od něj odvíjí celá scéna. Entity mohou být také potomky jiné entity. Přesunutím nebo otočením nadřazené entity, se také přesunou nebo otočí všechny děti pod nadřazenou entitou.

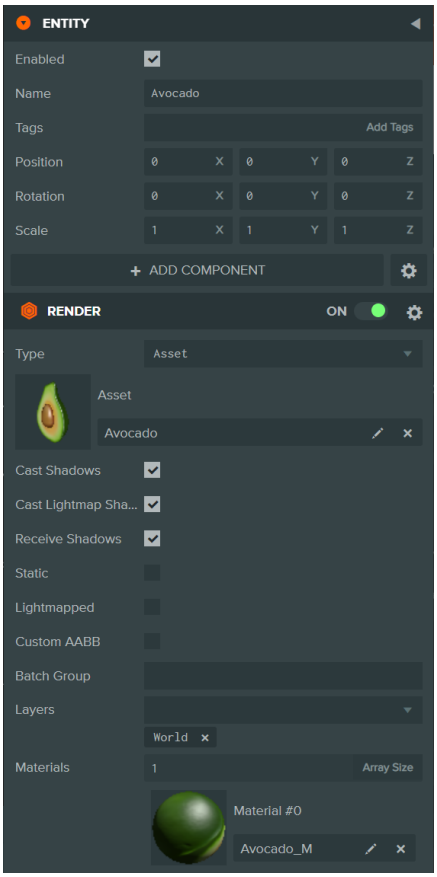
Strukturu hierarchie lze upravit vybráním a přetáhnutím libovolné entity v hierarchii. Tímto způsobem se rychle změní pořadí nebo rodič entity. Při přeřazení entity v editoru se zachová její transformace, tudíž po změně jejího rodiče se entita nepohne ani neotočí.

V horní části panelu hierarchie se nachází vyhledávání, které lze použít k dynamickému filtrování obsahu stromu entit. Entity lze také duplikovat pomocí klávesových zkratek nebo pravým kliknutím a následným vybráním možnosti duplikovat. Kopie každé entity se vygeneruje hned vedle jejího originálu. Kopírování či vkládaní entit funguje na stejném principu výběru možnosti pravým kliknutím jako u duplikace entit. Kopírovat a vkládat entity lze také mezi různými scénami nebo dokonce různými projekty. Je potřeba postupovat stejným

způsobem jako při kopírování entit a poté přejít na požadovanou scénu, vybrat požadovaného rodiče a vložit entity. Editor se pokusí přiřadit všechny odkazy na assety k vloženým entitám podle cesty v novém projektu.[12]

3.4.3 Inspector

Na panelu *Inspector* se zobrazují hodnoty komponenty a jejich atributy aktuálně vybraného objektu. V závislosti na vybrané entitě se zobrazí různé panely s nabídkou.

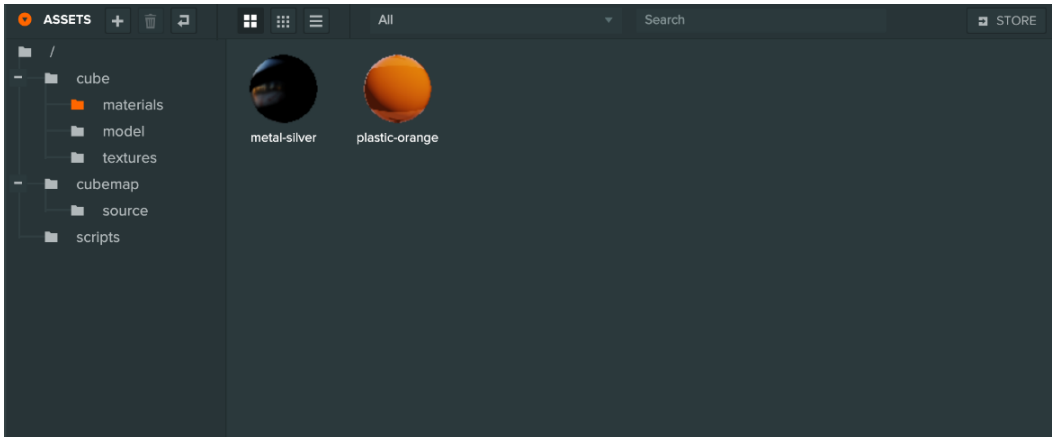


Obrázek 7. Panel *Inspector* [12]

Úpravou těchto hodnot se určuje, jak se bude vybraná entita chovat. Lze široce nastavovat a upravovat atributy entity. Některé atributy jsou prostý text nebo číselné hodnoty. Složitější atributy vyžadují rozsáhlejší způsob zadávání hodnot. Při současném běhu hry i editoru se změny atributů přenesou do entit ve spuštěné aplikaci v reálném čase. [12]

3.4.4 Assety

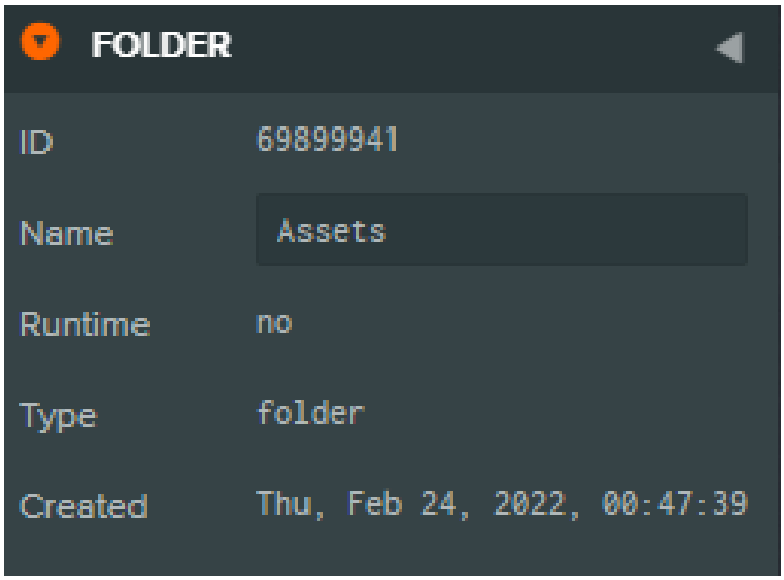
Panel assetů spravuje všechny assety, která jsou v projektu k dispozici. Umožňuje vytvářet, nahrávat, odstraňovat, kontrolovat a upravovat libovolné assety.



Obrázek 8. Panel assetů [12]

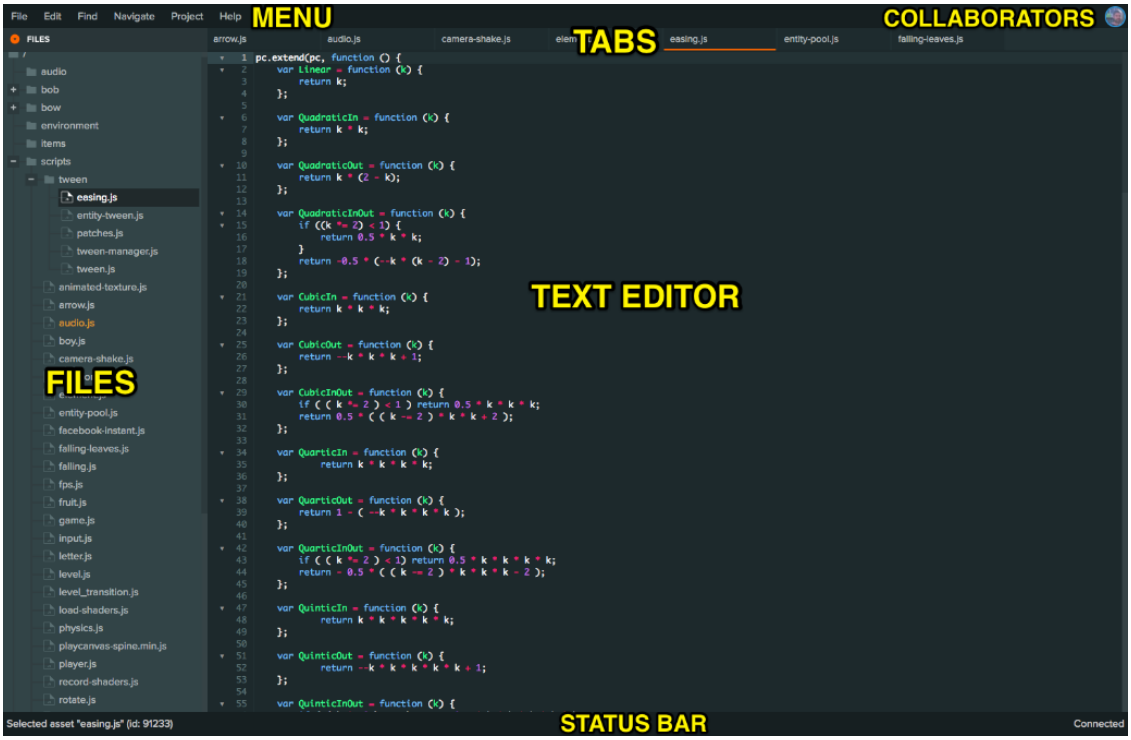
Nové položky lze vytvořit za využití tlačítka s ikonou „+“ nebo jednoduchým přetažením souboru, ze souborového systému počítače, do panelu assetů. Editor tak naimportuje soubor automaticky. Assety lze odstranit kliknutím na tlačítko s ikonou koše.

Panel obsahuje také panel složek, který umožňuje uspořádat assety do hierarchie složek. Pro přejmenování složky stačí na ni dvakrát kliknout myší a ve zobrazeném panelu v *Inspectoru* upravit název v poli „*Name*“. Složky jdou jednoduše přeorganizovat jednoduchým přetažením v hierarchii složek.



Obrázek 9. Přejmenování složky [12]

Dvojklikem na miniaturu assetu se otevře editor kódu, ve kterém lze otevřít a upravit některé textové prostředky: textové soubory, json, shader, html, css a skriptové assety. [12]



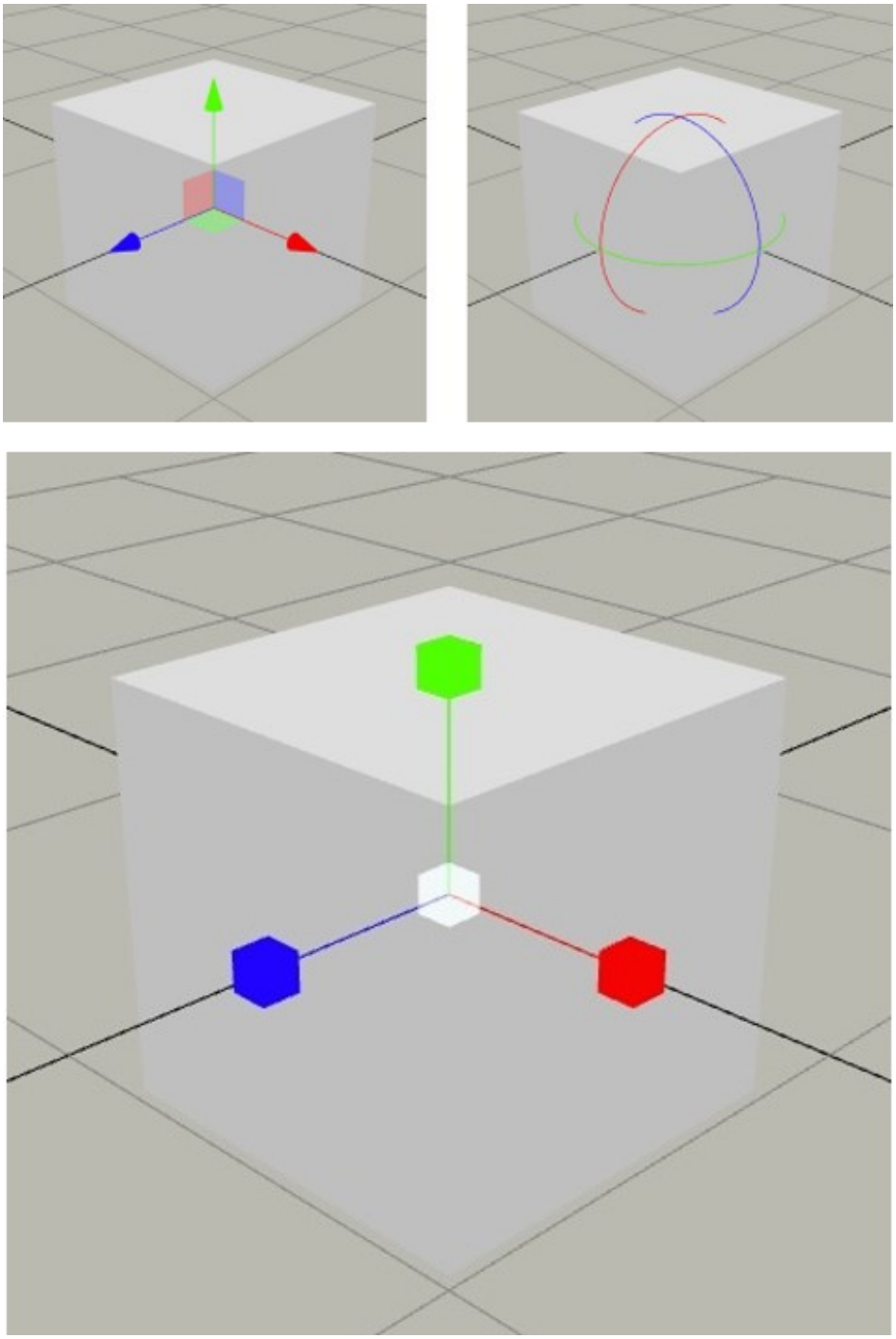
Obrázek 10. Editor kódu [12]

3.4.5 Viewport

Viewport zobrazuje aktuálně vykreslenou scénu. Ve scéně se dá volně pohybovat za pomoci náhledové kamery editoru. Editor je zpočátku nastaven na použití perspektivní kamery. Tato kamera je jako filmová kamera, se kterou se dá volně pohybovat po scéně. Za pomoci rozbalovací nabídky kamery lze scénu zobrazit v různých přednastavených kamerách. Ortografické kamery: Horní, Dolní, Přední, Zadní, Levá, Pravá se používá k zobrazení pohledů scény bez perspektivy.

Obrázek 11. Možnosti kamery [12]

Tříbarevná osa, která se nachází ve *Viewportu,* se nazývá *Gizmo.* Využívá se k manipulaci a transformaci vybrané entity po trojrozměrných osách. Dělí se na tři typy. Prvním typem je *Translate,* který se využívá k přesouvání entity v prostoru. Dalším typem je *Rotate.* Za jeho pomoci lze entitu otáčet v prostoru. Posledním typem je *Scale*, který lze využít ke zvětšování nebo zmenšování entity v prostoru. [12]



Obrázek 12. Gizmo [12]

3.5 Simulace fyziky

Framework PlayCanvas má vestavěný fyzikální engine, který usnadňuje implementaci mnoha fyzikálních simulací a urychluje proces návrhu hry. Je také nejkritičtější částí kódové základny z hlediska výkonu. Je implementován jako tenká, ručně psaná vrstva, která obaluje Ammo.js, port open source fyzikálního enginu Bullet generovaného v jazyce Emscripten.

K popisu fyzikálních objektů se používají dvě hlavní součásti enginu – *Rigid Body* (simulující působení sil na těleso) a kolize (oblast kolem tělesa). *Rigid Body* umožňuje objektu účastnit se fyzikálních interakcí na scéně (gravitace, kolize mezi tělesy a působení ostatních sil). Naopak kolize popisuje, v jaký moment bude objekt reagovat při kontaktu s jiným objektem, který má komponent *Rigid Body*.

Například u skákajícího míče se pomocí *Rigid Body* nastaví fyzikální vlastnosti (hmotnost, pružnost, tření atd.), následně pomocí kolizního faktoru se definují parametry pro kolizi. Samotná kolizní oblast může mít mnoho pravidelných tvarů, které automaticky poskytuje framework PlayCanvas. Pokud existují skutečně unikátní kolizní tvary, které je třeba speciálně zohlednit, existuje také možnost importovat *mesh* z 3D modelovacího programu. Po vytvoření bude PlayCanvas testovat průsečíky mezi kolizními parametry v malých časových intervalech. Pokud k průsečíku dojde, vstoupí do hry faktor simulace fyziky tělesa. V aplikaci PlayCanvas lze sledovat mnoho kolizí najednou. [1]

3.5.1 Typy simulace

Statická tělesa se při kolizi nepohybují a běžně se používají ke znázornění povrchů, po kterých se odrážejí nebo kutálejí jiné objekty. Mají vlastnosti tření a restituce, z nichž první modeluje tření objektu a druhá modeluje, kolik energie se ztratí při kolizi neboli jak moc bude těleso odraženo.

Dynamická tělesa mají určenou hmotnost a při srážce se budou pohybovat. Mají také počáteční lineární a úhlový faktor (vektory popisující směrovou, resp. rotační rychlost).

Kinematická tělesa jsou podobná statickým tělesům v tom, že mají nekonečnou hmotnost a při srážce se nepohnou. Budou se však pohybovat, když vývojář napíše skript, který bude objektem specificky pohybovat. [1]

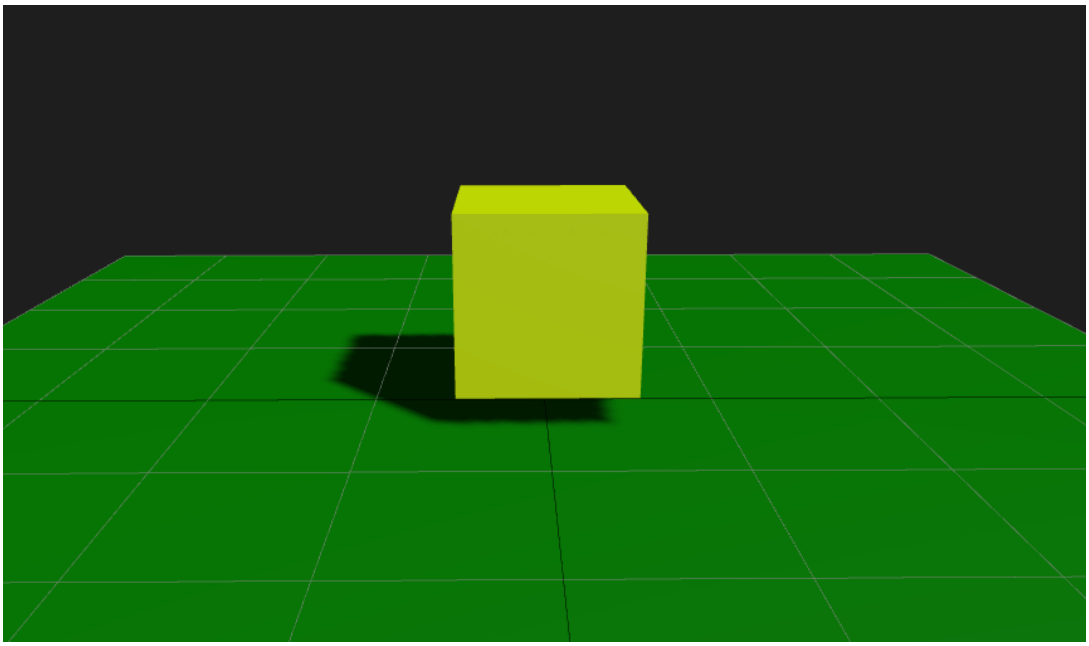
II. PRAKTICKÁ ČÁST

4 VYUKOVÉ TUTORIÁLY

Pro začátečníky, kteří by chtěli začít pracovat s frameworkem PlayCanvas byly vypracovány jednoduché tutoriály, jejichž účelem je provést nového uživatele prostředím a naučit ho základy práce s frameworkem. Bylo vytvořeno celkem šest tutoriálů, které lze najít v příloze P I. Tutoriály postupně pokrývají základní a pokročilé funkce frameworku. Veškeré tutoriály byly vytvořeny tak, aby na sebe navazovaly a výsledkem tak byla funkční aplikace.

4.1 Základy pro práci s frameworkem PlayCanvas

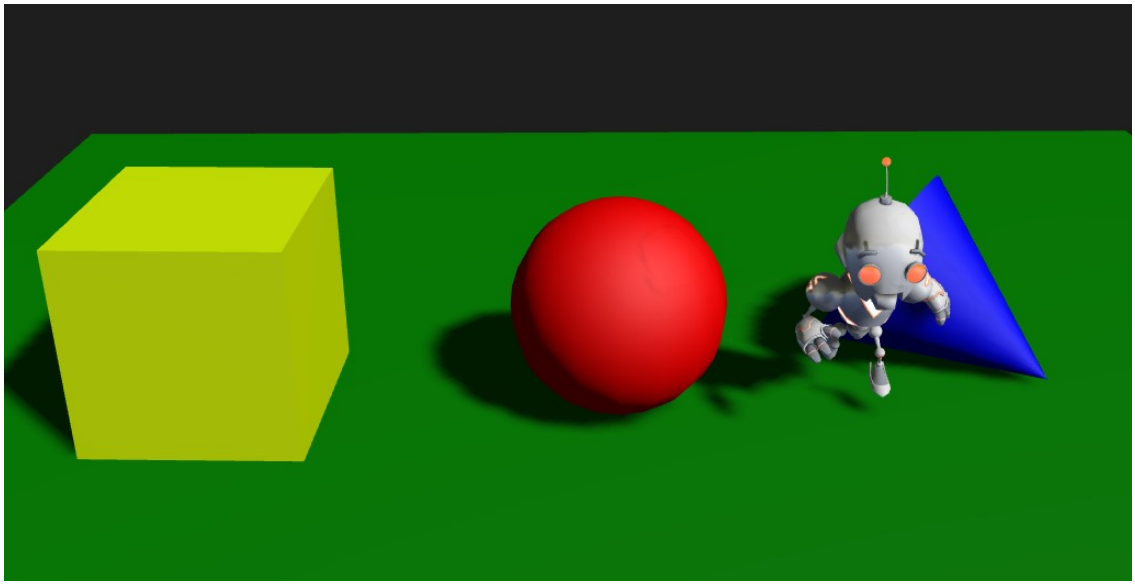
První tutoriál má převážně sloužit k seznámení uživatele se základy a pohybem v prostoru editoru frameworku. Tutoriál popisuje registraci nového profilu a vytvoření nového projektu ve frameworku PlayCanvas. Dále se tutoriál zaměřuje na naučení největších základů pro práci s editorem. Podrobně popisuje ovládání, pohyb a možnosti pohledu kamery. Tutoriál se zaměřuje také na manipulaci a transformaci entit, jako jsou pohyb, rotace a změna velikosti. Další částí tutoriálu je vytvoření nových entit jejich duplikace, kopírování nebo jak jednotlivé entity odstranit. Pro vysvětlení práce s materiálem byly do tutoriálu přidány základy pro vytvoření a nanesení materiálu na entitu.



Obrázek 13. Základy práce s frameworkem PlayCanvas

4.2 Přidání a nastavení komponentů entity

Druhý tutoriál byl vytvořen tak, aby se zaměřoval na přidávání jednotlivých komponentů entitám a nastaveních atributu daných komponentů. Během tutoriálu se uživateli vysvětluje, jak vytvořit a následně i aplikovat materiál na jednotlivé objekty. Dále se tutoriál zaměřuje na přidání a popis komponentů, které simulují fyziku daného objektu. Tutoriál obsahuje podrobný popis dvou komponentů *Rigid Body* a *Collison* a následně jejich aplikaci v názorné ukázce. Dále se tutoriály zabývají importem nových assetů a aplikováním animace na vybraný objekt. Poprvé je představeno spuštění aplikace.



Obrázek 14. Přidání a nastavení komponentů entity

4.3 Světlo a skybox

Třetí tutoriál vznikl za účelem vysvětlit uživateli práci se světlem, stíny a *skyboxem*. Tutoriál popisuje vytvoření jednotlivých typů světel v editoru, jejich možnosti nastavení a využití ve scéně. Vytvoření a nastavení simulace stínů vykreslované v reálném čase. Čtenář se také dozví o práci s *Cubemapou* a jejím aplikováním pro vytvoření *skyboxu*. Všechny tyto znalosti uživatel využije při tvoření nasvícení scény.

Obrázek 15. Světlo a skybox

4.4 Pohyb entity

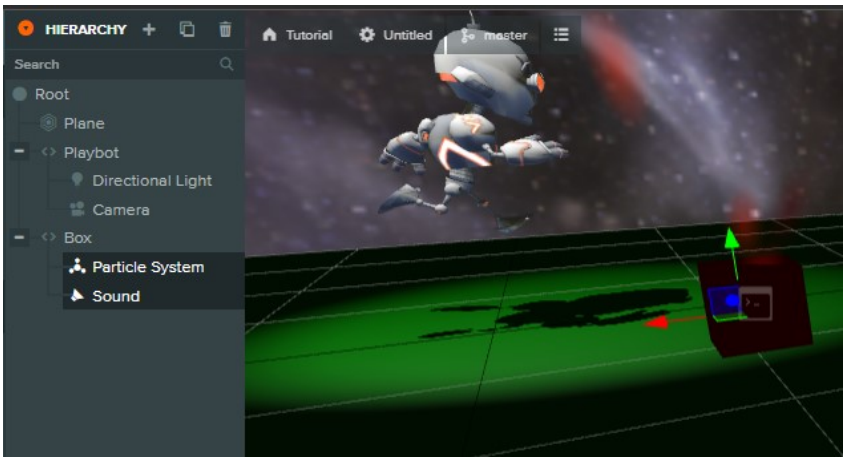
Čtvrtý tutoriál byl zaměřen na vytvoření a následnou práci se skriptem. Tutoriál vysvětluje vytvoření nového skriptu, jak vypadá samotný skript a popisuje základní funkce skriptu a jejich využití. Čtenář si vyzkouší přidat pohyb entitě a následně i vytvoření ovládání tohoto pohybu přes klávesové inputy a funkce, které na entitu aplikují sílu. V závěru tutoriálu byl vytvořen návod na úpravu scény tak, aby byla optimalizována pro pohyb entity.

```
10
11  // update code called every frame
12  Movement.prototype.update = function(dt) {
13
14      if(app.keyboard.isPressed(pc.KEY_W))
15      {
16          entity.rigidbody.applyForce(0,0,5);
17      }
18      if(app.keyboard.isPressed(pc.KEY_S))
19      {
20          entity.rigidbody.applyForce(0,0,-5);
21      }
22      if(app.keyboard.isPressed(pc.KEY_A))
23      {
24          entity.rigidbody.applyForce(5,0,0);
25      }
26      if(app.keyboard.isPressed(pc.KEY_D))
27      {
28          entity.rigidbody.applyForce(-5,0,0);
29      }
30
31
32  };
```

Obrázek 16. Pohyb entity

4.5 Částicové efekty a zvuk

Pátý tutoriál byl vytvořen tak, aby se zaměřil na přidání částicových efektů a zvuku do scény. V základu tutoriálu je vysvětleno vytvoření samotného *Particle Systemu* a následný popis možností nastavení, jednotlivých simulovaných částic. Tutoriál se dále zaměřuje na import souboru se zvukem do projektu a následně jeho aplikováním s popisem možností nastavení.



Obrázek 17. Částicové efekty a zvuk

4.6 Uživatelské rozhraní

Posledním tutoriál byl vytvořen pro naučení základů při tvorbě uživatelského rozhraní ve frameworku. Tutoriál popisuje vytvoření nového uživatelské rozhraní a přidávání elementů do tohoto rozhraní. Dále se zaměřuje na možnosti nastavení vzhledu a formátování jednotlivých elementů a import nového fontu do projektu. Obsahuje také návod na vytvoření funkce, která se zavolá po stisknutí tlačítka.

Obrázek 18. Uživatelské rozhraní

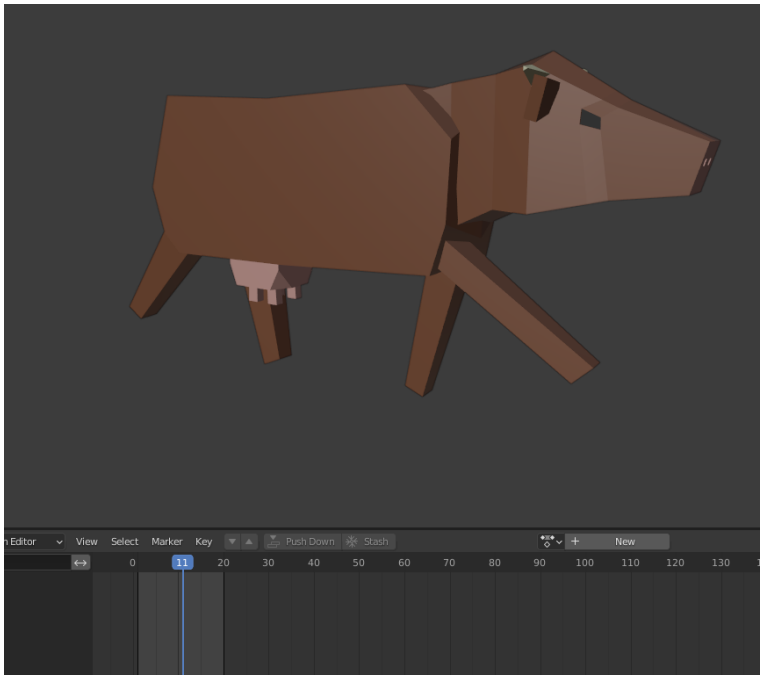
5 TVORBA HRY

Pro prezentování možností a práce ve frameworku PlayCanvas, byla vytvořena nová hra za použití co nejvíce nástrojů frameworku. Jedná se o 3D *first person* hru s *low poly* grafikou. Princip hry spočívá v tom, že se hráč musí v nejrychlejším čase dostat do cíle. Hráč se objeví ve vytvořeném levelu a za pomoci přeskakování přes pohybující se zvířata se snaží doskákat do cíle. Hráč se nesmí dotknout země nebo jakéhokoliv jiného assetů prostředí (stromu nebo kamene) než zvířete. Hra končí výhrou, jestliže hráč doskáče do cíle nebo také prohrou, kdy se hráč dotkne země či objektu. Po celou dobu hraní se hráči měří čas, který se pak ukáže na závěrečné obrazovce, pokud hráč vyhraje.

5.1 Příprava assetů

Jako první bylo potřeba najít a připravit assety, které se následně využívaly při vytváření hry. Pro vyhledání volně dostupných assetů byla využita webová stránka s volně dostupnými assety *www.cgtrader.com* [13]. Všechny použité assety byly bezplatné a s volnou licencí.

V případě zvířat [16], bylo potřeba assety trochu poupravit. Za využití volně dostupného softwaru Blender z assetů byl vybrán model zvířete, který se při tvorbě hry následně využíval. Model měl již implementovanou kostru, která posloužila k vytvoření nové animace běhu zvířete. Model se následně vyexportoval jako soubor s příponou .fbx.

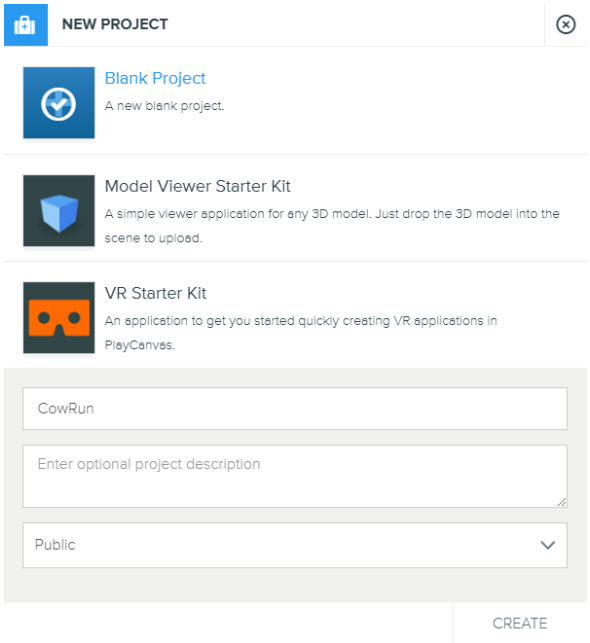


Obrázek 19. Model zvířete

Pro prostředí byl vybrán *low poly environment* balíček assetů [17]. Tento balíček byl vybrán, protože obsahuje široké množství assetů prostředí určených k vytváření levelu. Dále se použil model dřevěného plotu [18], který se následně ve hře využil jako označení viditelného cíle.

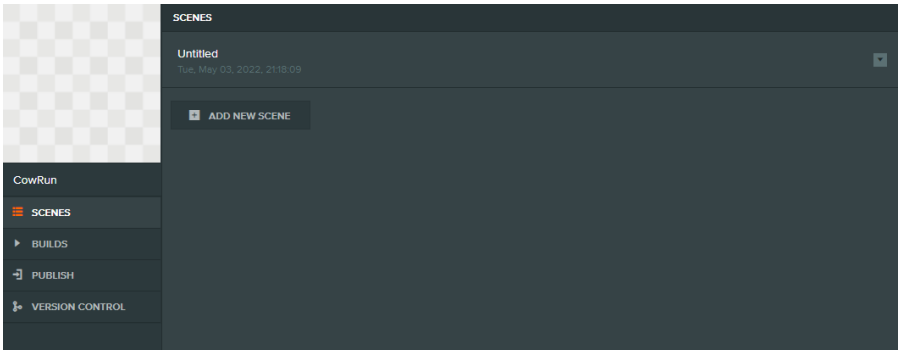
5.2 Vytvoření projektu a scény

Ve frameworku PlayCanvas přes nabídku v profilu byl vytvořen nový prázdný projekt s názvem „CowRun“ (testovací název hry).



Obrázek 20. Vytvoření Projektu

Po vytvoření nového projektu se přes jeho nabídku otevřel editor nového projektu. Po načtení editoru vyskočilo okno s možností výběru scény. Otevřela se předem vytvořená scéna „*untitled*“. Jakmile se scéna načetla vymazaly se nepotřebné předem vygenerované entity. Zůstala tak prázdná scéna, do které se následně importovaly předem připravené assety.



Obrázek 21. Výběr scény

5.2.1 Vložení assetů do prostředí

Před importem assetů se nejprve musel v nastavení scény vypnout import hierarchie modelu. Kdyby byla tato možnost zapnutá, editor by neumožňoval vytvoření entity z modelu jednoduchým přetažením z panelu assetů. Každá entita by se tak musela vytvářet ručně a následně přes komponenty přidávat vyrenderovaný model.

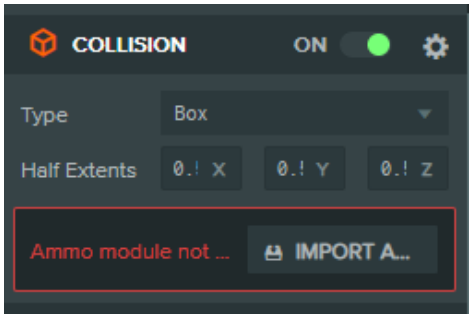
Veškeré připravené assety se přes panel assetů vložily do editoru, aby se s nimi mohlo dále pracovat. Přes hierarchii složek se assety rozdělily do dvou podsložek. První složka byla určena pro model zvířete, jeho materiálu a animace. Druhá poté sloužila pro veškeré modely prostředí. Jednoduchým přetažením modelu z panelu do *Viewportu* byla vytvořena první entita, která se během vývoje aplikace využívala.

Obrázek 22. Vložení assetů

5.3 Přidání komponentů

Vzhledem k tomu, že bylo potřeba docílit, aby assety mezi sebou kolidovaly nebo v případě hráče a zvířete se pohybovaly, muselo se přidat jednotlivým entitám komponenty. Nejdůležitějšími komponenty tak byly *rigidbody,* a *collision.* První zmíněný komponent se využíval k simulaci realistické fyziky pro objekt. Bylo možné tak nastavit, zda bude objekt statický nebo na něj budou působit kinematické či dynamické síly. Naopak komponent *collison* se využíval k vytvoření oblasti kolem objektu, která při kolizi s dalším objektem s komponentem *collison* spustila trigger event na základě typu jejich *rigidbody* komponentu.

Oba tyto komponenty na sebe navazovaly a jeden bez druhého by neměl význam. Pro funkčnost těchto komponentů bylo potřeba naimportovat „*Ammo*“ modul. Pokud editor tento modul neobsahoval, tak jej při přidání komponentů doporučil editor naimportovat.



Obrázek 23. Import modulu

Pro nejlepší nastavení kolizí se využíval typ *mesh*, který vytvořil oblast podle vloženého modelu. Při testování entit s přidanými atributy nastal problém, kdy dvě entity s kolizním typem *mesh* mezi sebou vzájemně nekolidovaly a nespouštěly tak triggery. Po zjištění, že se nejednalo o chybu při vývoji hry, ale o to, že framework nepodporoval kolize dvou objektů s kolizním typem *mesh* (ostatní typy se sebou navzájem kolidovaly), bylo využito jiné řešení. Pro všechny objekty, se kterými zvíře či hráč kolidovalo, se nastavilo typ *mesh*. Pro hráče a zvíře se využil kolizní typ „kapsle“, který umožňoval hladký pohyb po všech osách. Důvodem tohoto nastavení bylo, aby hráč na zvířeti stál rovně a neklouzal. Dále bylo potřeba ke zvířeti přidat podřazenou entitu, ke které se přidal kolizní typ boxu. Rozměry boxu se poté upravily tak, aby vytvořily tenkou rovnou plochu nad zvířetem. Docílilo se tak objektu, který se dokázal pohybovat bez většího omezení. Hráč tudíž mohl na něj přistát a odrážet se od něj.

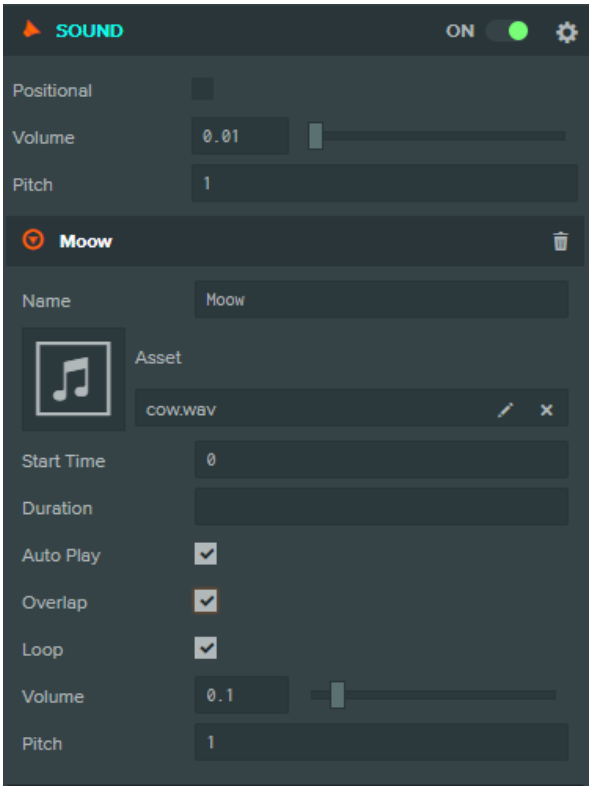
Obrázek 24. Oblasti kolize

5.3.1 Animace a zvuk

Jako další byl zvířeti přidán komponent *Animation.* Tento komponent umožňoval přiřazení animace k objektu a nastavení rychlosti, zda se měl cyklus opakovat nebo zda měla být animace již aktivní. Jelikož byl model z programu Blender exportován ve formátu .fbx, tak se po vložení modelu do panelu assetů vytvořil i soubor s animací modelu. Tento soubor se následně přiřadil přetažením do komponentu animace.

Pro vytvoření zvuku zvířete se využil komponent *sound,* který se využívá k přehrávaní zvukového souboru. K simulaci zvuku zvířete byl stažen soubor s volně dostupnou a bezplatnou nahrávkou zvířete [19]. Soubor se přidal do prostředí editoru a následným přetažením přiřadil do komponentu *sound*.

Soubor se zvukem byl v základu velmi hlasitý. Jelikož se jednalo se o zvuk, který měl hrát v pozadí aplikace. V nastavení komponentu tak bylo potřeba nastavit hlasitost na minimum. Dále se nastavilo spuštění zvuku při zapnutí aplikace, překrývání se zvuku s ostatními zvuky a smyčka, která se použila k tomu, aby se zvuk opakoval. Stejným postupem byla do hry také přidána volně dostupná melodie [20].



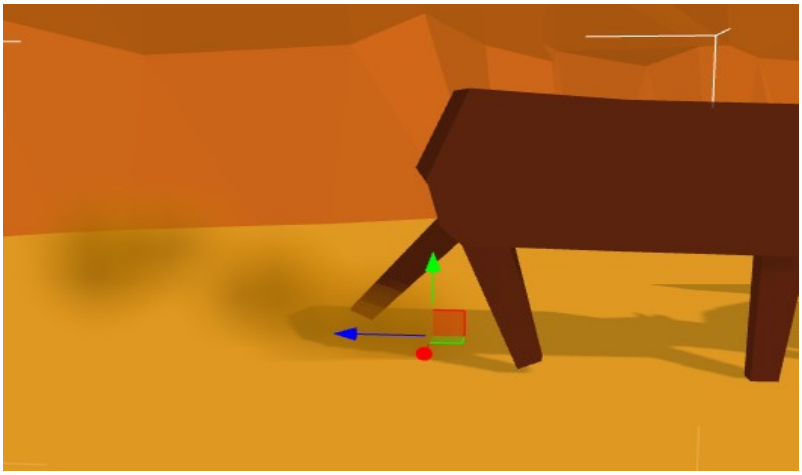
Obrázek 25. Nastavení zvuku

5.3.2 Částicové efekty

Zvířata se měla pohybovat po prašné ploše, proto do entity zvířete byl přidán také částicový efekt vyzdviženého prachu ze země. Tohoto efektu se docílilo tak, že se pod objektem vytvořila nová entita s komponentem *particle system*.

Tento komponent vygeneruje a nasimuluje pohyb částic. Komponent obsahuje velkou možnost nastavení a vytvoření efektu. V nastavení komponentu bylo nastaveno generování tři sta částic s životností přes sekundu. Barva jednotlivých částic byla nastavena na barvu povrchu hrací plochy. Částicím byl také nastaven směr tak, aby se pohybovaly směrem od samotného objektu zvířete. Efekt se následně nastavil na spuštění při zapnutí aplikace a vytváření částic ve smyčce tak, aby efekt nikdy neskončil. Za pomoci tohoto nastavení byl simulován efekt zvedajícího se prachu, který byl následně umístěn pod nohy modelu zvířete.

Obrázek 26. Nastavení částic



Obrázek 27. Částicový efekt

5.3.3 Skripty

Dále se objektům přidal komponent *script*, který se používá k rozšíření funkcí entity přiřazením JavaScriptového souboru. JavaScript se vykonával s možností přístupu k jednotlivým entitám za pomoci komponentu *Script*. Do jednotlivých atributů komponentů se poté přidaly vytvořené skripty.

5.4 Programování

Za pomoci tlačítka „+“ v panelu assetů se vytvořily skripty. Po vytvoření skriptového souboru se přes dvojklik na vytvořený soubor dalo dostat do editoru kódu, ve kterém bylo možné napsat kód pro jednotlivé skripty. Všechny kódy byly napsány v jazyce JavaScript.

5.4.1 Pohyb hráče a kamery

Nejdůležitějším skriptem byl skript pro pohyb hráče a pohled kamery z první osoby. Jako první se vytvořil atribut, ke kterému se v editoru přidávaly jednotlivé entity a hodnoty. Atribut *camera* se využíval k propojení skriptu s kamerou. Atributy *power* a *powerJump* se využívaly k rychlosti pohybu a síle skoku. Posledním přidaným atributem byl *lookspeed*, který se využíval k nastavení senzitivity kamery.

first-person-movement.j

```
1   var FirstPersonMovement = pc.createScript('firstPersonMovement');
2
3   FirstPersonMovement.attributes.add('camera', {
4       type: 'entity',
5       description: 'assign a camera entity'
6   });
7
8   FirstPersonMovement.attributes.add('power', {
9       type: 'number',
10      default: 2500,
11      description: 'adjust the speed of player movement'
12  });
13
14  FirstPersonMovement.attributes.add('powerJump', {
15      type: 'number',
16      default: 2500,
17      description: 'adjust the height of jump'
18  });
19
20
21  FirstPersonMovement.attributes.add('lookSpeed', {
22      type: 'number',
23      default: 0.25,
24      description: 'adjust the sensitivity of looking'
25  });
26
```

Obrázek 28. Vytvoření atributů

Další částí skriptu byla jeho inicializace. Ta se volala pouze při načtení entity. V této funkci se deklarovaly proměnné a trigger eventy. Přidala se také funkce *onCollisionStart*, která kontrolovala, zda se hráč něčeho dotknul. Tato funkce zamezila tomu, aby hráč nemohl skákat do nekonečna ve vzduchu a mohl tak vyskočit pouze, když se dotkne jiného objektu.

```
// initialize code called once per entity
FirstPersonMovement.prototype.initialize = function() {
    this.force = new pc.Vec3();
    this.eulers = new pc.Vec3();
    this.app.keyboard.on(pc.EVENT_KEYDOWN, this.onKeyDown, this);
    this.entity.collision.on("collisionstart",this.onCollisionStart,this);
    this.canJump = 1;
    var app = this.app;

    // Listen for mouse move events
    app.mouse.on("mousemove", this._onMouseMove, this);

};

//entity is on ground
FirstPersonMovement.prototype.onCollisionStart = function (result){
    if(result.other.rigidbody)
    {
        this.canJump = 1;
    }
};
```

Obrázek 29. Inicializace a funkce pro kolizi

Pro pohyb v prostoru se využila funkce *update*, která se každým novým *framem* zavolá znovu. Tato funkce byla ideální pro vytvoření pohybu postavy, jelikož bylo potřeba měnit směry co nejrychleji. Na začátku funkce *update* se kontroluje, zda je hra stále v běhu nebo zda byla ukončena. Pokud je hra spuštěna, kód pokračuje a deklaruje nové proměny, které se při dalším obnovení znovu přepíší. Pod deklarací se nachází kontrola stisknutí kláves určených pro pohyb postavy. Po stisku některých z kláves se odečte nebo přičte pohyb ve dvou směrech. Jestliže byl zaznamenán nějaký pohyb, vypočte se a aplikuje síla na objekt. Hráč se tak pohybuje ve směru, v jakém na něj byla aplikována síla. Nakonec se ve funkci aktualizuje natočení kamery.

Posledními vytvořenými funkcemi ve skriptu byly *onKeyDown* a *onMouseMove*. První zmíněná funkce se využívá k ověření, zda byla stisknuta klávesa „mezerník“ a zda hráč může skákat. Pokud se splnily oba tyto požadavky, tak se na entitu aplikuje síla zespodu a zamezí se dalšímu skoku. Hráč tak vyskočí do vzduchu a nemůže vyskočit znovu, dokud se opět nedotkne další entity s kolizí. Poslední funkce kontroluje pohyb počítačové myši. Pokud se myš pohne, tak se kamera přesune na pozici kurzoru v závislosti na rychlosti senzitivity.

```
49  // update code called every frame
50  FirstPersonMovement.prototype.update = function(dt) {
51      this.playing = playing;
52      if(this.playing)
53      {
54          var force = this.force;
55          var app = this.app;
56
57          // Get camera directions to determine movement directions
58          var forward = this.camera.forward;
59          var right = this.camera.right;
60
61          // movement
62          var x = 0;
63          var z = 0;
64
65          // Use W-A-S-D keys to move player
66          // Check for key presses
67          if (app.keyboard.isPressed(pc.KEY_A)) {
68              x -= right.x;
69              z -= right.z;
70          }
71          if (app.keyboard.isPressed(pc.KEY_D)) {
72              x += right.x;
73              z += right.z;
74          }
75          if (app.keyboard.isPressed(pc.KEY_W)) {
76              x += forward.x;
77              z += forward.z;
78          }
79          if (app.keyboard.isPressed(pc.KEY_S)) {
80              x -= forward.x;
81              z -= forward.z;
82          }
83          // use direction from keypresses to apply a force to the character
84          if (x !== 0 && z !== 0) {
85              force.set(x, 0, z).normalize().scale(this.power);
86              this.entity.rigidbody.applyForce(force);
87          }
88          // update camera angle from mouse events
89          this.camera.setLocalEulerAngles(this.eulers.y, this.eulers.x, 0);
90      }
91
92  };
```

Obrázek 30. Funkce *update*

```
94  //jump on key down
95  FirstPersonMovement.prototype.onKeyDown = function (){
96      this.powerjump = powerjump;
97
98      if (this.app.keyboard.wasPressed(pc.KEY_SPACE) && this.canJump!=0) {
99          this.entity.rigidbody.applyImpulse(0, powerjump, 0);
100         this.canJump = 0;
101     }
102 };
103 //update camera on move
104 FirstPersonMovement.prototype.onMouseMove = function (e) {
105     if (pc.Mouse.isPointerLocked() || e.buttons[0]) {
106         this.eulers.x -= this.lookSpeed * e.dx;
107         this.eulers.y -= this.lookSpeed * e.dy;
108     }
109 };
110
```

Obrázek 31. Funkce *onKeyDown* a *onMouseDown*

5.4.2 Pohyb zvířete

Pohyb zvířete oproti pohybu hráče byl vyřešen aplikováním síly z jednoho směru na objekt. Vytvořil se skript, kde každý *frame* aplikuje sílu na entitu, ke které byl skript přiřazen. Zvíře se tak pohybovalo konstantní rychlostí jedním směrem.

```
var CowRun = pc.createScript('cowRun');

// update code called every frame
CowRun.prototype.update = function(dt) {

//pokud je hra spustena aplikuje se sila na entitu
if(playing == true)
{
    this.entity.rigidbody.applyForce(0,0,-3000);
}


};
```

Obrázek 32. Skript pro pohyb zvířete

5.4.3 Měření času

Pro měření času byly vytvořeny stopky, které připočítávaly a vracely aktuální čas od spuštění hry. Nejprve se v inicializaci deklarovaly jednotlivé proměnné pro minuty, sekundy a milisekundy. Ve funkci *update* se následně ověřilo, zda byla hra spuštěna. Pokud byla hra spuštěna, začal se přičítat čas od milisekund. Po uplynutí 60 milisekund se začaly přičítat sekundy a vynulovaly se milisekundy. Poté po uplynutí 60 sekund se přičítaly minuty. Na konci funkce bylo vytvořeno vypsání kompletního času do elementu text.

```
// update code called every frame
// postupne pricitani casu, cas se na konci kodu vypise do elementu text
Time.prototype.update = function(dt) {

    this.playing = playing;
    if(this.playing)
    {
    this.milsec += 1;

    if (this.milsec == 60) {
      this.sec += 1;
      this.milsec = 0;
    }
    if (this.sec == 60) {
      this.min += 1;
      this.sec = 0;
      this.milsec = 0;
    }
    this.textElement.element.text = this.min +":" + this.sec + ":" + this.milsec;
    time = this.min +":" + this.sec + ":" + this.milsec;
    }

};
```

Obrázek 33. Skript pro stopky

5.4.4 Ukončení hry

Ve skriptu *gameover* se vytvořila funkce pro kontrolu, zda se hráč dotkl objektu. Kontrola se prováděla přes tagy entity. Každé entitě lze v editoru přiřadit tag, který se využívá k rozpoznání nebo rozdělení jednotlivých entit. Pokud se jedna entita s nastavenou kolizí dotknula jiné entity s kolizí, zavolala se *trigger* funkce, která zjistila tagy jednotlivých entit. Pokud jedna entita měla tag *player* a druhá tag *ground,* hra se ukončila. Pokud měla druhá entita tag *end* znamenalo to, že hráč doskákal do konce a hru dohrál.

```
var GameOver = pc.createScript('gameover');


var end = false;

GameOver.prototype.initialize = function() {
        this.entity.collision.on('triggerenter', this.onTriggerEnter, this);

};

GameOver.prototype.onTriggerEnter = function (otherEntity) {

    if (otherEntity.tags.has('Player'))
    {
        if(this.entity.tags.has('Ground'))
        {
        playing = false;
        }
        if(this.entity.tags.has('End'))
        {
        end = true;
        }
    }


};
```

Obrázek 34. Ukončení hry

5.5 Vytvoření levelu

Po přípravě všech entit a souborů, které ve hře byly využívány, se začalo pracovat na vytváření levelu, ve kterém se hra odehrávala. Hra byla zasazena do westernového prostředí, proto se použily převážně odstíny oranžové, žluté a hnědé barvy.

5.5.1 Hrací plocha

Jako první se vytvořila hrací plocha, po které se zvířata pohybovala a obsahovala veškeré překážky. Pro plynulý pohyb zvířat, bylo potřeba, aby plocha byla vodorovná a rovná. Plocha byla vytvořena za pomoci vytvoření nové entity s komponentem *render*. V komponentu se zvolil typ renderování na *box* a nastavila se velikost entity se souřadnicemi x = 100, y = 5, z =1000, kde *x* značilo šířku, *y* výšku a *z* délku objektu. Na plochu byl poté přetažením z panelu assetů nanesen materiál *Brown.15.* Kolem hrací plochy byla vytvořena neviditelná bariéra, aby hráč či zvíře nemohli hrací plochu opustit.

Na konec hrací plochy se umístila entita s vygenerovanou novou plochou. Tato plocha značila cíl hry. Na plochu se proto nanesl materiál *Brown.8,* aby novou plochu bylo možné rozeznat od hrací plochy. Kolem cíle byl poté rozmístěn model dřevěného plotu tak, aby napodoboval ohradu.

5.5.2 Přidání překážek

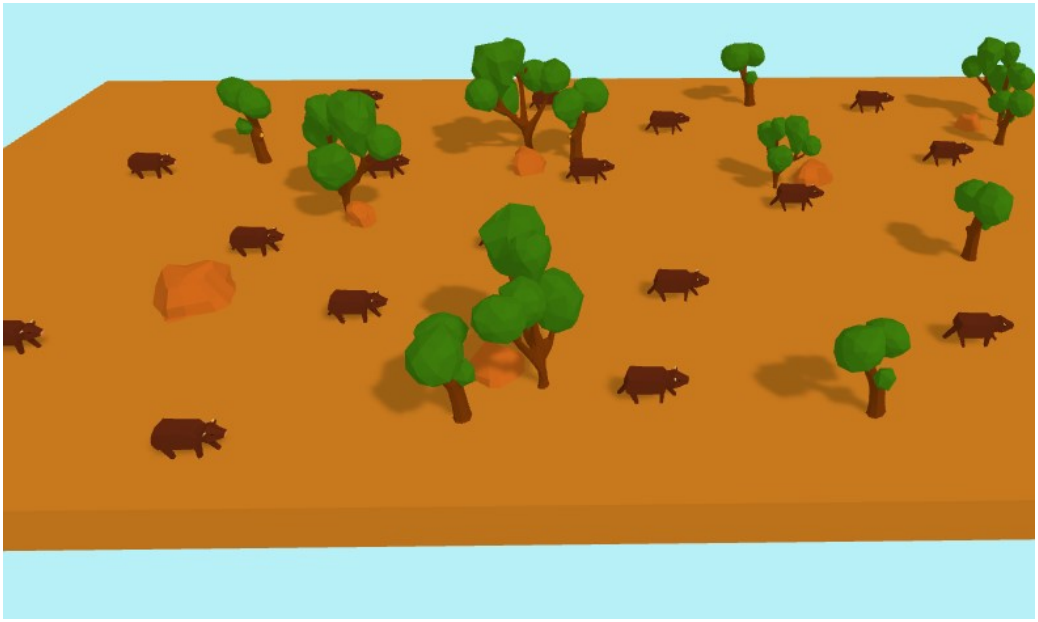
Po vytvoření plochy se začaly rozmisťovat různé objekty, které měly fungovat jako překážky pro pohybující se zvířata a hráče. Ke každé entitě bylo potřeba přidat komponenty, aby mohly interagovat s entitou hráče a zvířete.

Každá překážka obsahovala komponenty *rigidbody* se statickým typem a *collision* s *meshem* modelu. Tyto komponenty se využívaly pouze k omezení pohybu zvířete. Pro kolizi hráče s objektem se muselo pod entitu vytvořit novou entitu s tagem *Ground*, komponentem *collison* s typem mesh a skriptem *gameover*. Tato entita se poté zvětšila o jednu setinu ve všech směrech. Tento složitý krok byl nezbytný, jelikož hráč by se kvůli *rigidbody* komponentu nadřazené entity nedostal do kolizní oblasti a nespustil tak trigger na spuštění skriptu. Velikost oblasti *collison* se ve frameworku PlayCanvas odvíjí od velikosti entity a nelze tak upravovat její velikost bez závislosti na entitě. Ke každému assetů byl přiřazen tag Ground.

Po přidaní všech potřebných atributů se tímto způsobem vytvořily různé překážky (stromy a kameny), které se následně už jen duplikovaly a rozmisťovaly po hrací ploše tak, aby dodaly hře větší obtížnost a vzhled.

5.5.3 Přidání zvířat

Další věcí, která byla potřeba přidat do levelu, byla zvířata. Po kterých se hráč mohl pohybovat a přeskakovat z jednoho zvířete na druhé. Jelikož entita zvířete byla již dříve přichystána a přidaly se k ní veškeré komponenty, mohla se použít k rozmístění za pomoci duplikace. Zvířata byla rozmístěna tak, aby mezi nimi nebyla příliš velká mezera a hráč mohl mezi zvířaty přeskakovat. Rozmisťování entit proběhlo po celé šířce a zhruba do poloviny délky hrací plochy tak, aby hráč měl dostatek prostoru pro doskákání do cíle.



Obrázek 35. Hrací plocha

5.5.4 Prostředí

Následně se kolem hrací plochy začalo vytvářet prostředí tak, aby hráč neviděl pod mapu. Cílem bylo vytvořit westernové prostředí hor. Z připravených assetů se postupným přetahováním začala vytvářet okolní scenérie. Objekty byly různě zvětšovány a otáčeny tak, aby nešlo na první pohled poznat, že se jednalo o pár assetů spojených dohromady. Na různá místa na hrací ploše se poté rozmístily další objekty jako byly například stromy a kameny. Dále se také přidaly různé keře a rostliny, aby se docílilo reálnějšího efektu prostředí. Rostliny se nanesly také na hrací plochu. Tyto rostliny nijak neovlivňovaly pohyb entit po ploše.

5.5.5 Skybox a světlo

Pro nastavení oblohy se využila funkce *skybox*. Tato funkce se nachází v nastavení scény v sekci *rendering*. Pro využití funkce se nejdříve musela vytvořit *cubemapu*. Ta se vytvořila za pomoci tlačítka „+“ v panelu assetů. Do *cubemapy* bylo potřeba vložit jednotlivé stěny *skyboxu*, které byly vytvořeny ve volně dostupném softwaru *Gimp* za pomoci barevného převodu mezi bílou a modrou. Stěny se poté přetažením umístil do *cubemapy*, která se následně vložila do *skybox* atributu. Výsledkem bylo napodobení oblohy.

Pro přidání světla do scény bylo potřeba přidat novou entitu *Directional Light*. Tato entita se používá k napodobení světla ze slunce. Barvu světla se nechala téměř bílá a přidalo se pouze trochu žluté barvy. Čímž se napodobila teplota světla. V nastavení světla se také zapnuly stíny. Pro vykreslování reálných stínu se použila *realtime* funkce vykreslení a hodnoty se nastavily tak, aby stíny vypadaly co nejlépe.

Obrázek 36. Finální podoba prostředí

5.6 Uživatelské rozhraní

Dalším krokem bylo vytvoření uživatelského rozhraní. V hierarchii byla vytvořena nová prázdná entita, která se jmenovala UI. Do entity se následně přes nabídku vytvoření nové entity a rozkliknutí možnosti *user interface* vložila entita *2D screen*. Vygenerovala se tak nová 2D plocha, do které se následně mohly přidávat HTML elementy.

5.6.1 Rozhraní pro start a hru

Jako první bylo potřeba vytvořit rozhraní pro spouštění hry za pomoci tlačítka start, které se nacházelo uprostřed obrazovky. Přes nabídku vytvoření nové *user face* entity se vytvořil nový *button* element. Vygenerované tlačítko bylo zarovnáno na střed přes nabídku ukotvení v inspektoru. Dále se upravil vzhled tlačítka za pomoci široké škály nastavení. Do textového pole tlačítka byl vložen text „Start“ a změnil se typ fontu pro text.

Následně bylo vytvořeno rozhraní, které se zobrazovalo po celou dobu hry. Vytvořila se proto nová skupinu elementů. Do skupiny se následně přidaly dva textové elementy. První element byl určen pro jednoduchý text „Time:“, který označoval čas. Druhý textový element obsahoval aktuálně uběhlý čas od spuštění hry. Oba elementy se vedle sebe ukotvily do levého horního rohu. Do skupiny elementů se následně přidal komponent skript, který volal skript *Time*. Přes atribut skriptu se propojil druhý textový element se skriptem.

Obrázek 37. Uživatelské rozhraní před spuštění hry

5.6.2 Rozhraní pro konec hry

Pro konec hry byla potřeba vytvořit dvě různá rozhraní. Jedno rozhraní určovalo neúspěšné dokončení hry, kdy se hráč dotknul nějakého povrchu, kterého by se dotknout neměl. Druhé rozhraní určovalo úspěšné dokončení, kdy se hráč dostal úspěšně do cíle.

Vytvořily se tedy dvě nové skupiny elementů. První skupina elementů byla určena k neúspěšnému dokončení. Do skupiny se přidaly dva textové elementy. Do prvního byl přidán text „Failed“. Druhý textový element byl zarovnán pod první a také se přidal text, který oznamoval, že pro restart hry stačí zmáčknout mezerník. Oba elementy se následně zarovnaly na střed. Podobným postupem se upravila i druhá skupina, určená pro úspěšné dokončení hry. Pouze text „Failed“ se změnil na text „Completed“ a mezi dva texty byl přidán nový textový element, který ukazoval aktuální naměřený čas, při dokončení hry.

Obě rozhraní měli v pozadí také lehce průhlednou barvu, která se používá pro lepší rozpoznání ukončení hry. Tato barva se musela vytvořit jako nové 2D plochy s image elementem, které se nacházely za hlavní plochou určenou pro rozhraní. Tento krok byl potřeba z toho důvodu, aby obrázky nepřekrývaly text a ten byl čistě viditelný.

Obrázek 38. Rozhraní dokončené hry

5.6.3 Propojení rozhraní s hrou

Když bylo rozhraní vytvořené, bylo potřeba ho propojit s hrou a přidat mu funkčnost. Vytvořil se tedy nový skript *Game.js*, který se využíval k inicializaci globálních proměn. Do těchto proměn se dosazují převzaté entity z atributů skriptu.

```
1   var Game = pc.createScript('game');
2
3   Game.attributes.add('uiStart', {type: 'entity'});
4   Game.attributes.add('uiInGame', {type: 'entity'});
5   Game.attributes.add('uiOver', {type: 'entity'});
6   Game.attributes.add('uiEnd', {type: 'entity'});
7
8
9   //user interface
10  var uiMenu;
11  var uiInGame;
12  var uiGameOver;
13  var uiEnd;
14
15  //game mechanics
16  var playing;
17
18  // initialize code called once per entity
19  Game.prototype.initialize = function() {
20
21      uiMenu = this.uiStart;
22      uiInGame = this.uiInGame;
23      uiGameOver = this.uiOver;
24      uiEnd = this.uiEnd;
25  };
26
```

Obrázek 39. Skript pro inicializaci rozhraní

Dále byly vytvořeny skripty pro jednotlivá rozhraní. Skript *Ui-menu.js* se využívá ke spuštění hry za pomocí tlačítka, které je převzato z atributu skriptu. Funkce *update* kontroluje, zda je hra spuštěna nebo ne. Pokud hra není spuštěna kontroluje se, zda bylo tlačítko zmáčknuto. Pokud se tlačítko zmáčknulo, nastavila se proměnná *playing* na *true* a uzamkl se kurzor. Pokud již hra byla spuštěna, schová se rozhraní obsahující tlačítko.

```
var UiMenu = pc.createScript('uiMenu');
UiMenu.attributes.add("button", {type: "entity"});

// initialize code called once per entity
UiMenu.prototype.initialize = function() {
    this.uiMenu = uiMenu;
};
UiMenu.prototype.update = function(dt) {
   if(!playing)
   {
    this.button.button.on('click', function(event) {

        playing = true;
        console.log('ok');
        this.app.mouse.enablePointerLock();

    }, this);
   }else
   {
       uiMenu.enabled = false;
   }
};
```

Obrázek 40. Skript *Ui-menu.js*

Skript *Ui-gameover.js* se používá k zobrazení jednotlivých elementů do rozhraní. Pokud mají jednotlivé proměnné atribut *enabled* nastaven na *false*, znamená to, že jsou skryté. V případě, že je atribut *enabled* nastaven na *true*, jsou elementy vidět v rozhraní. Skript tak kontroluje, zda byla hra ukončena. Pokud ano, tak se zobrazí rozhraní *uiGameOver* s přiřazenou entitou *overlay* a schová se rozhraní *uiInGame*. Skript poté po krátké pauze zastaví hráče tak, aby se nehýbal po skončení hry. Následně skript poslouchá, zda se zmáčknula klávesa mezerník. Po zmáčknutí klávesy se znovu načte okno s hrou a hra se tak vyresetuje na začátek.

Na stejném principu funguje také skript *Ui-end.js*. Skript místo proměnné *playing* kontroluje proměnnou *end*. Ve skriptu je navíc také atribut *time*, který se využívá pro přiřazení změřeného času do textového elementu.

Všechny vytvořené scripty se následně zavolaly v entitě UI, která obsahuje všechny rozhraní. Do atributů skriptu se poté přiřadily veškeré entity a elementy.

```
var UiEnd = pc.createScript('uiEnd');
UiEnd.attributes.add('overlay', {type: 'entity'});
UiEnd.attributes.add('time', {type: 'entity'});
UiEnd.attributes.add('player', {type: 'entity'});

// initialize code called once per entity
UiEnd.prototype.initialize = function() {
};

UiEnd.prototype.update = function() {
this.end = end;
if(this.end == true)
{
    uiEnd.enabled = true;
    this.time.element.text = time;
    this.overlay.enabled = true;
    uiInGame.enabled = false;
    setTimeout(() => {
        this.player.enabled = false;
    }, 100);
    if (this.app.keyboard.wasPressed(pc.KEY_SPACE)) {
        location.reload();
    }
}
};
```

Obrázek 41. Script *Ui-end.js*

5.7 Optimalizace

Dále bylo potřeba provést optimalizaci hry. Pohyb hráče nebyl plynulý a bylo obtížené se trefovat na zvířata. Proto se poupravily hodnoty síly pohybu a skoku hráče. Tímto byla hra poté jednodušší a plynulejší.

Původním plánem bylo vytvořit obtížnější hru tak, aby pro její dokončení musel hráč nejprve nějakou dobu hru hrát. Hra však byla příliš obtížná. Proto bylo přidáno více zvířat do scény a všechna zvířata zvětšena tak, aby se na ně jednodušeji trefovalo.

Další problém nastal s malým počtem FPS, kvůli kterému se hra sekala a nebyla plynulá. Tento problém se vyřešil nastavením menšího rozlišení hry, zmenšením kvality stínu a vypnutím mlhy. Následně se také upravilo zorné pole kamery z 45° na 60°. A do rozhraní před spuštěním hry byly také přidány informace o ovládání a principu hry.

Do samotných kódů skriptu byly vloženy komentáře, které se využívaly k lepšímu rozpoznání funkčnosti a čitelnosti kódu. Jednotlivé komentáře v kódu lze vidět v přílohách P V a P VI.

5.8 Problémy při vývoji

Během vytváření hry nastalo velké množství problémů, nebo omezení ze strany frameworku. Například dříve zmíněná nepodpora kolize dvou entit s kolizním typem *mesh*.

Velkým problémem bylo načítaní všech atributů a skriptů entit při otevření uloženého projektu. Často se při vývoji stávalo, že bylo potřeba atributy ke komponentům entit přiřazovat od začátku, jelikož framework nenačetl dříve přiřazené atributy. Komponenty sice ukazovaly, že atributy přiřazené jsou, ale ve skutečnosti přiřazené nebyly. To znamenalo, že po každém spuštění se hra musela testovat, zda vše funguje a pokud něco nefungovalo, tak se musel hledat problém. Několikrát se také stalo, že po načtení projektu zmizely veškeré materiály objektů a materiál se musel tak znovu přiřazovat.

5.9 Publikace hry

Hra při publikování měla jeden vytvořený level, ve kterém se odehrávala. Plánovalo se přidání více levelů, ale kvůli problémům s možnostmi frameworku nebyly vytvořeny. Framework sice podporuje změnu scén, ale při změnách často dochází k chybám s rozpoznáním jednotlivých skriptů. Hratelnost hry nemusí vyhovovat každému, ale bylo ve snaze vytvořit co nejlepší plynulý pohyb hráče.

Hra byla publikována na webu *playcanvas.com* [1], který hru hostuje. Jednoduchým otevřením přes vytvořený odkaz https://playcanv.as/p/2iekHIiz/ lze hru spustit. Pro spuštění hry není potřeba mít zaregistrovaný účet.

Hra se také nachází v příloze P I, pro spuštění této hry je však potřeba vytvoření vlastního serveru, podle přiloženého návodu SelfHosting.txt, který se nachází ve stejné příloze.

Finální podoba hry lze vidět na obrázcích v příloze P III a P IV.

ZÁVĚR

Tato bakalářská práce měla za cíl popsání frameworku PlayCanvas a následné vytvoření výukových tutoriálů pro začínající uživatele, kteří s frameworkem chtějí pracovat. Dalším cílem bylo vytvoření komplexní aplikace, která měla za cíl využít co největšího množství funkcí frameworku.

V teoretické části této práce se popisují vlastnosti a funkce nejnovějších verzí webových technologií HTML a CSS. Dále se tato část zaměřuje na popis programovacího jazyka JavaScript a API WebGL na které je založen framework PlayCanvas. Také se zde nachází popis a vysvětlení pojmu frameworku. Velká část se pak následně zaměřuje na samotný framework PlayCanvas a popis jeho funkcí a možností.

Praktická část obsahuje popisy vytvořených tutoriálů. Tyto tutoriály seznamují nového uživatele frameworku se základními i pokročilejšími funkcemi. Tutoriály jsou vytvořeny tak, aby na sebe navzájem navazovaly a výsledkem tutoriálů je tak funkční aplikace.

Další kapitolou praktické části bylo vytvoření komplexnější aplikace. Tato aplikace byla vytvořena s cílem využití co nejvíce možností a funkcí frameworku. Finální aplikace má podobu 3D hry. Část popisuje podrobný popis tvorby hry a využívání jednotlivých funkcí frameworku.

V závěru praktické části se nachází optimalizace samotné hry a okomentování kódu pro lepší přehled a znovu čitelnost. Práce zde popisuje, také problémy s frameworkem PlayCanvas, které nastaly během vývoje.

Během práce bylo zjištěno, že práce s frameworkem je na naučení složitější, kvůli malému množství výukových materiálů. Dalším zásadním nedostatkem frameworku bylo velké množství bugů, které se při vývoji objevovaly (například nefunkčnost komponent po načtení uloženého projektu). Vývojáři však postupem času framework upravují a stále rozšiřují. Proto je framework vhodný pro menší projekty, u kterých lze jednodušeji dohledat a odstranit případný problém.

SEZNAM POUŽITÉ LITERATURY

[1] PlayCanvas Manual. Playcanvas [online]. Santa Monica: PlayCanvas, 2014 [cit. 2021-10-13]. Dostupné z: https://developer.playcanvas.com/en/

[2] MDN Web Docs. Resources for developers, by developers. [online]. San Francisco: MDN contributors, 2021 [cit. 2021-10-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Building_up_a_basic_demo_with_PlayCanvas/editor

[3] W3Schools. JavaScript Tutorial [online]. Norway: Refsnes Data, 2018 [cit. 2021-10-13]. Dostupné z: https://www.w3schools.com/js/default.asp

[4] FLANAGAN, David. JavaScript: the definitive guide: master the world's most-used programming language. Seventh edition. Beijing: O'Reilly, 2020 [cit. 2021-10-13]. ISBN 978-1491952023.

[5] GRANT, Keith J. CSS in Depth. New York: Manning, 2018 [cit. 2021-10-13]. ISBN 978-1617293450

[6] HTML. *GeeksforGeeks* [online]. Noida: geeksforgeeks, 2022 [cit. 2022-03-10]. Dostupné z: https://www.geeksforgeeks.org/html/?ref=lbp

[7] HTML5. *GeeksforGeeks* [online]. Sector-136, Noida, Uttar Pradesh: geeksforgeeks, 2022 [cit. 2022-03-10]. Dostupné z: https://www.geeksforgeeks.org/difference-between-html-and-html5/

[8] Difference between CSS. CodingNinjas [online]. Sunrise Mentors Pvt. Ltd.: Sunrise Mentors Pvt., 2021 [cit. 2022-03-10]. Dostupné z: https://www.codingninjas.com/blog/2021/01/07/difference-between-css-css2-css3/

[9] JavaScript Versions. Educative [online]. Seattle: Educative, 2020 [cit. 2022-03-10]. Dostupné z: https://www.educative.io/blog/javascript-versions-history

[10] WebGL. WhatIs [online]. Auburndale: TechTarget, 2019 [cit. 2022-03-10]. Dostupné z: https://whatis.techtarget.com/definition/WebGL

[11] What is a Framework in Programming. Net solutions [online]. London: Net Solutions, 2021 [cit. 2022-03-10]. Dostupné z: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/

[12] PlayCanvas Manual Editor. Playcanvas [online]. Santa Monica: PlayCanvas, 2014 [cit. 2022-04-10]. Dostupné z: https://developer.playcanvas.com/en/user-manual/designer/

[13] CGTrader [online]. Vilnius: CGTrader, 2011 [cit. 2022-04-24]. Dostupné z: https://www.cgtrader.com/

[14] JavaScript Syntax. Tutorialspoint [online]. Tutorialspoint [cit. 2022-04-28]. Dostupné z: https://www.tutorialspoint.com/javascript/javascript_syntax.htm

[15] How CSS works. In: MDN Web Docs [online]. USA: Mozilla Foundation, 2022 [cit. 2022-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_works

[16] Rigged farm animals Free low-poly 3D model. Cgtrader [online]. cgtrader, 2016 [cit. 2022-04-28]. Dostupné z: https://www.cgtrader.com/free-3d-models/animals/other/farm-rigged-animals-lowpoly

[17] LowPoly Environment Pack Free low-poly 3D model. Cgtrader [online]. cgtrader, 2019 [cit. 2022-04-28]. Dostupné z: https://www.cgtrader.com/free-3d-models/exterior/landscape/lowpoly-environment-pack

[18] Low Poly Fence Free low-poly 3D model. Cgtrader [online]. cgtrader, 2018 [cit. 2022-04-28]. Dostupné z: https://www.cgtrader.com/free-3d-models/exterior/exterior-public/low-poly-fence-abc210ab-78cf-4faa-b64f-61b1648faf10

[19] Cow. Freesound [online]. Barcelona: MTG [cit. 2022-04-28]. Dostupné z: https://freesound.org/people/Benboncan/sounds/58277/

[20] Cowbell Phonk Melody-Speed Up. SampleFocus [online]. Sample Focus [cit. 2022-04-28]. Dostupné z: https://samplefocus.com/samples/cowbell-phonk-melody-speed-up

[21] WebGL Introduction. Tutorialspoint [online]. Tutorialspoint [cit. 2022-04-29]. Dostupné z: https://www.tutorialspoint.com/webgl/webgl_introduction.htm

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

2D	Dvourozměrný prostor
3D	Trojrozměrný prostor.
API	Application Programming Interface (aplikační programovací rozhraní)
CSS	Cascading Style Sheets (tabulky kaskádových stylů)
DOM	Document Object Model (objektový model dokumentu)
FPS	Frames per Second (počet snímků za sekundu)
GPU	Graphic Proces Unit (grafický procesor)
HTML	Hypertext Markup Language (textový značkovací jazyk)
UI	User Interface (uživatelské rozhraní)
WAF	Web Application Framework (webový aplikační framework)
WF	Web Framework (webový framework)
XML	eXtensible Markup Language (rozšiřitelný značkovací jazyk)

SEZNAM OBRÁZKŮ

Obrázek 1. Struktura webové stránky [6] 13
Obrázek 2. Načtení CSS [15] 15
Obrázek 3. Princip webového frameworku [11] 19
Obrázek 4. Menu Editoru [12] 23
Obrázek 5. Panel nástrojů [12] 23
Obrázek 6. Panel Hierarchie [12] 24
Obrázek 7. Panel *Inspector* [12] 25
Obrázek 8. Panel assetů [12] 26
Obrázek 9. Přejmenování složky [12] 26
Obrázek 10. Editor kódu [12] 27
Obrázek 11. Možnosti kamery [12] 27
Obrázek 12. Gizmo [12] 28
Obrázek 13. Základy práce s frameworkem PlayCanvas 31
Obrázek 14. Přidání a nastavení komponentů entity 32
Obrázek 15. Světlo a skybox 33
Obrázek 16. Pohyb entity 33
Obrázek 17. Částicové efekty a zvuk 34
Obrázek 18. Uživatelské rozhraní 34
Obrázek 19. Model zvířete 35
Obrázek 20. Vytvoření Projektu 36
Obrázek 21. Výběr scény 36
Obrázek 22. Vložení assetů 37
Obrázek 23. Import modulu 38
Obrázek 24. Oblasti kolize 38
Obrázek 25. Nastavení zvuku 39
Obrázek 26. Nastavení částic 40
Obrázek 27. Částicový efekt 40
Obrázek 28. Vytvoření atributů 41
Obrázek 29. Inicializace a funkce pro kolizi 42
Obrázek 30. Funkce *update* 43
Obrázek 31. Funkce *onKeyDown* a *onMouseDown* 43
Obrázek 32. Skript pro pohyb zvířete 44

Obrázek 33. Skript pro stopky 44
Obrázek 34. Ukončení hry 45
Obrázek 35. Hrací plocha 47
Obrázek 36. Finální podoba prostředí.....48
Obrázek 37. Uživatelské rozhraní před spuštění hry 49
Obrázek 38. Rozhraní dokončené hry.....50
Obrázek 39. Skript pro inicializaci rozhraní 51
Obrázek 40. Skript *Ui-menu.js* 52
Obrázek 41. Script *Ui-end.js*.....53

SEZNAM TABULEK

Tabulka 1. Rozdíly mezi HTML a HTML5 [7].....12

SEZNAM PŘÍLOH

Příloha P I – obsah CD

Příloha P II – Uživatelské prostředí editoru [12]

Příloha P III – Podoba aplikace

Příloha P IV – Podoba aplikace

Příloha P V – Okomentovaný kód

Příloha P VI – Okomentovaný kód

PŘÍLOHA P I: OBSAH CD

- fulltext.pdf
- \prilohy
    - \hra
        - \assety
            - \animal
            - \enviroment
            - \font
            - \skybox
        - \Cow Run (hra)
        - SelfHosting.txt
    - \tutorialy
        - \materialy
        - Pohyb_entity_framework_PlayCanvas.docx
        - Pohyb_entity_framework_PlayCanvas.pdf
        - Pridani_komponentu_framework_PlayCanvas.docx
        - Pridani_komponentu_framework_PlayCanvas.pdf
        - Svetlo_skybox_framework_PlayCanvas.docx
        - Svetlo_skybox_framework_PlayCanvas.pdf
        - Uzivatelske_rozhrani_framework_PlayCanvas.docx
        - Uzivatelske_rozhrani_framework_PlayCanvas.pdf
        - Zaklady_prace_framework_PlayCanvas.docx
        - Zaklady_prace_framework_PlayCanvas.pdf
        - Zvuk_casticove_efekty_framework_PlayCanvas.docx
        - Zvuk_casticove_efekty_framework_PlayCanvas.pdf

PŘÍLOHA P II: UŽIVATELSKÉ PROSTŘEDÍ EDITORU [12]

PŘÍLOHA P III: PODOBA APLIKACE

PŘÍLOHA P IV: PODOBA APLIKACE

PŘÍLOHA P V: OKOMENTOVANÝ KÓD

```
1   //vytvoreni scriptu
2   var Time = pc.createScript('time');
3
4   //deklarace globalni promenne
5   var time;
6
7   //pridani atributu pro entitu s textem
8   Time.attributes.add('textElement', {type: 'entity'});
9
10  // initialize code called once per entity
11  //deklarace promen
12  Time.prototype.initialize = function() {
13      this.time = 0;
14      this.milsec = 0;
15      this.sec = 0;
16      this.min = 0;
17      this.playing = playing;
18
19  };
20
21  // update code called every frame
22  // postupne pricitani casu, cas se na konci kodu vypise do elementu text
23  Time.prototype.update = function(dt) {
24
25      this.playing = playing;
26
27      // pokud hra probiha pricitaji se milisekundy
28      if(this.playing)
29      {
30      this.milsec += 1;
31
32      //pokud se napocitalo 60 milisekund pricte se sekunda
33      if (this.milsec == 60) {
34        this.sec += 1;
35        this.milsec = 0;
36      }
37      //pokud se napocitalo 60 sekund pricte se minuta
38      if (this.sec == 60) {
39        this.min += 1;
40        this.sec = 0;
41        this.milsec = 0;
42      }
43      //do entity prevzane z atributu se vypise vysledny cas
44      this.textElement.element.text = this.min +":" + this.sec + ":" + this.milsec;
45
46      //ulozeni casu do promenne time
47      time = this.min +":" + this.sec + ":" + this.milsec;
48      }
49
50  };
```

PŘÍLOHA P VI: OKOMENTOVANÝ KÓD

```
//vytvoreni scriptu
var GameOver = pc.createScript('gameover');

//globalni promena pro uspesne dokonceni hry
var end = false;

//pri kolizi se zavola funkce onTriggerEnter
GameOver.prototype.initialize = function() {
        this.entity.collision.on('triggerenter', this.onTriggerEnter, this);

};

//funkce trigger, ktera prebira entitu, s kterou se kolidovalo
GameOver.prototype.onTriggerEnter = function (otherEntity) {

    //kontroluje se zda ma entita tag player
    if (otherEntity.tags.has('Player'))
    {
        ///kontroluje se zda ma druha entita tag player
        //pokud ano hra skoncila neuspechem
        if(this.entity.tags.has('Ground'))
        {
        playing = false;
        }
        //kontroluje se zda ma druha entita tag end
        //pokud ano hra byla dohrana
        if(this.entity.tags.has('End'))
        {
        end = true;
        }
    }
};

```

```
//vytvoreni scriptu
var CowRun = pc.createScript('cowRun');

// update code called every frame
CowRun.prototype.update = function(dt) {
    //pokud je hra spustena aplikuje se sila na entitu
    if(playing == true)
    {
        //aplikovani sily na entitu v pozadovanem smeru
        this.entity.rigidbody.applyForce(0,0,-3000);
    }
};
```