Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v ČB
Pedagogická fakulta
Katedra informatiky

Webové 3D aplikace s využitím open-source frameworku Babylon.js

The use of open-source framework Babylon.js for creating 3D web application.

Bakalářská práce

Vypracoval: Filip Chytra

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip CHYTRA**
Osobní číslo:	**P190049**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Téma práce:	**Webové 3D aplikace s využitím open-source frameworku Babylon.js**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování

Cílem bakalářské práce bude představení tvorby 3D modelů/scén a jejich vkládání do webových aplikací za pomocí open-source frameworku Babylon.js. V dnešní době se stále častěji setkáváme s využíváním trojrozměrných technologii, virtuálních a rozšířených realit (VR a AR) napříč mnoha medii a odvětvími, jako jsou například kinematografie, herní průmysl, výcvikové trenažéry aj. Podobný trend implementace těchto technologií je předpokládán v následujících pár letech i u webových technologií a je podmíněn vývojem nových webových standardů a JavaScriptových API, jako jsou OpenGL, WebGl a v neposlední řadě nejnovější WebGPU. Ty umožňují využití fyziky a zpracování obrazu a efektů zrychlených na GPU jako součást plátna webové stránky. Autor se ve své práci zaměří na popis funkcí samotného 3D enginu Babylon.js a porovná jej s dalšími technologiemi pro tvorbu 3D počítačové grafiky a její implementování do kódu webu. Základem práce bude na dílčích ukázkách demonstrovat možnosti základních nástrojů a funkcí pro založení scény, vytvoření základních objektů (světlo, kamera, polygonové sítě) a práci s materiály, ale také představení pokročilejších technik, jako jsou průnik a sloučení objektů, animace, práci s uživatelskými vstupy, fyzikální engine, kostry, partikly a shadery. Cílem praktické části pak bude vytvoření vlastní 3D webové prezentace, na které bude demonstrováno vkládání jednotlivých praktických ukázek do kódu webových stránek v HTML5 a popsání dalších možných uplatnění frameworku Babylon.js na webu.

Rozsah pracovní zprávy:	**40**
Rozsah grafických prací:	**CD ROM**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Home | Babylon.js Documentation [online]. Dostupné z: https://doc.babylonjs.com/
2. MOREAU-MATHIS, Julien. Babylon.js Essentials. Packt, 2016. ISBN 9781785884795.
3. MOREAU-MATHIS, Julien. LEARNING BABYLON.JS [online]. [cit. 04.04.2021]. Dostupné z: http://learningbabylonjs.com/
4. WebGL: 2D and 3D graphics for the web – Web APIs | MDN. [online]. Copyright © 2005 [cit. 04.04.2021]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
5. WebGPU: GitHub Pages [online]. Copyright © 2021 the Contributors to the WebGPU Specification, [cit. 04.04.2021]. Dostupné z: https://gpuweb.github.io/gpuweb/

Vedoucí bakalářské práce:	**PaedDr. Petr Pexa, Ph.D.**
	Katedra informatiky

Datum zadání bakalářské práce: **7. dubna 2021**
Termín odevzdání bakalářské práce: **30. dubna 2022**

L.S.

doc. RNDr. Helena Koldová, Ph.D.
děkanka

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 7. dubna 2021

Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 3. července 2023.

Filip Chytra

Podpis

Abstrakt/Anotace

Cílem této bakalářské práce bude představení tvorby 3D modelů/scén a jejich vkládání do webových aplikací za pomocí open-source frameworku Babylon.js. V dnešní době se stále častěji setkáváme s využíváním trojrozměrných technologii, virtuálních a rozšířených realit (VR a AR) napříč mnoha medii a odvětvími jako jsou například kinematografie, herní průmysl, výcvikové trenažéry aj. Podobný trend implementace těchto technologií je předpokládán v následujících pár letech i u webových technologií a je podmíněn vývojem nových webových standardů a JavaScriptových API, jako jsou OpenGL, WebGl a v neposlední řadě nejnovější WebGPU, které umožňují využití fyziky a zpracování obrazu a efektů zrychlených na GPU jako součást plátna webové stránky.

Autor se ve své práci zaměří na popis funkce samotného 3D enginu Babylon.js a porovnání s dalšími softwary pro tvorbu 3D počítačové grafiky a na její implementování do kódu webu. Základem práce bude na jednoduchých ukázkách důkladně demonstrovat možnosti základních nástrojů a funkcí pro založení scény, vytvoření základních objektů (světlo, kamera, polygonové sítě) a práci s materiály, ale také ukázání pokročilejších technik jako jsou průnik a sloučení objektů, animace, práci s uživatelskými vstupy, fyzikální engine, kostry, partikly a shadery.

Cílem praktické části pak bude vytvoření vlastní 3D webové prezentace, na které bude demonstrováno vkládání jednotlivých praktických ukázek do kódu stránek v rámci HTML5 a popsáním možných uplatnění frameworku Babylon.js na webu.

Klíčové slova

BABYLON.JS, 3D render, JavaScript, framework, HTML5, kanvas, API, WebGL

Abstract

The goal of the thesis is to introduce the creation of 3D models/scenes and their implementation into applications using the open-source framework Babylon.js. In recent years, we are increasingly encountering the use of three-dimensional technologies, and virtual and augmented realities (VR and AR) across many media and industries such as cinema, gaming, training simulators, etc. A similar trend in the implementation of these technologies is expected in the next few years web technologies and is conditioned by the development of new web standards and JavaScript APIs, such as OpenGL, WebGL and last but not least the latest WebGPU, which allows the use of physics and image processing and GPU-accelerated effects as part of a web page canvas.

The author will focus on the description of the function of the 3D engine Babylon.js itself and comparison with other software for creating 3D computer graphics and its implementation in the web code. The work will be based on simple demonstrations to thoroughly demonstrate the possibilities of basic tools and functions for creating a scene, creating basic objects (light, camera, meshes), and working with materials, but also demonstrating more advanced techniques such as collisions and intersections of objects, animation, working with user inputs, physics engine, skeletons, particles, and shaders.

The aim of the practical part will be to create your own 3D web presentation, which will demonstrate the implementation of individual practical examples into the code of pages within HTML5 and a description of possible utilization of the Babylon.js framework on the web.

Keywords

BABYLON.JS, 3D render, JavaScript, framework, HTML5, canvas, API, WebGL

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce panu PaedDr. Petru Pexovi, Ph.D. za nápomocné rady, ochotu a vedení při zpracování mé závěrečné práce.

Obsah

1 Úvod

Bakalářská práce se bude zabývat open-source frameworkem Babylon.js, který umožňuje na moderních internetových prohlížečích vytvářet a vkládat scény s trojrozměrnými modely.

1.1 Východiska práce

Možnost vkládání různých formátů, rastrové či vektorové grafiky, videí a animací napomohla k podobě webů, jak je známe teď. Ovšem v poslední době se nabízí otázka, jakým směrem se vývoj webů vydá dále. Jedna z možností, která se nabízí převážně v posledních letech, a jejíž úspěch byl možný být pozorován i v dalších odvětvích, je rozšíření webu o trojrozměrné technologie. Díky příchodu nových technologií jako WebGL do moderních prohlížečů, výpočetní výkonnosti dnešních zařízení a rychlého gigabitového internetu, tak jediné co zatím stálo v cestě nástupu 3D grafiky na scénu moderních stránek, byl jednoduchý, intuitivní a uživatelsky přívětivý způsob implementace pro vývojáře. A právě tyto poslední zásadní faktory zastupuje JavaScriptovy open-source framework Babylon.js. [1] [2]

1.2 Cíle práce

Cílem této bakalářské práce bude představení tvorby 3D modelů/scén a jejich vkládání do webových aplikací za pomocí open-source frameworku Babylon.js. V dnešní době se stále častěji setkáváme s využíváním trojrozměrných technologii, virtuálních a rozšířených realit (VR a AR) napříč mnoha médii a odvětvími jako jsou například kinematografie, herní průmysl, výcvikové trenažéry aj. Podobný trend implementace těchto technologií je předpokládán v následujících pár letech i u webových technologií a je podmíněn vývojem nových webových standardů a JavaScriptových API, jako jsou OpenGL, WebGl a v neposlední řadě nejnovější WebGPU, které umožňují využití fyziky a zpracování obrazu a

efektů zrychlených na GPU jako součást plátna webové stránky.

Autor se ve své práci zaměří na popis funkce samotného 3D enginu Babylon.js a porovnání s dalšími softwary pro tvorbu 3D počítačové grafiky a na její implementování do kódu webu. Základem práce bude na jednoduchých ukázkách důkladně demonstrovat možnosti základních nástrojů a funkcí pro založení scény, vytvoření základních objektů (světlo, kamera, polygonové sítě) a práci s materiály, ale také ukázání pokročilejších technik jako jsou průnik a sloučení objektů, animace, práci s uživatelskými vstupy, fyzikální engine, kostry, partikly a shadery.

Cílem praktické části pak bude vytvoření vlastní 3D webové prezentace, na které bude demonstrováno vkládání jednotlivých praktických ukázek do kódu stránek v rámci HTML5 a popsáním možných uplatnění frameworku Babylon.js na webu.

1.3 Metody práce

Úvodem bakalářské práce popíši jednotlivé potřebné technologie, které stojí za umožněním vykreslování trojrozměrných scén na webových stránkách. Dále se přímo zaměřím na popsání úloh, v které v tomto procesu zastává framework Babylon.js a na popsání jeho funkcí. V další části rozeberu způsoby tvorby 3D scén a objektů v prostředí Babylonu. Uvedu, jaké jsou jeho výhody a omezení oproti jeho alternativám a popíši, jak následně scénu implementovat do HTML kódu. Základem praktické části již bude důkladná demonstrace jednotlivých funkcí a nástrojů, a také pokročilejších funkcí a technik, které framework poskytuje. Veškeré funkce pak budou představeny na jednotlivých ukázkách na vlastní webové prezentaci. Následně vytvořím ukázku možné reálné 3D webové aplikace s implementovaním vlastní trojrozměrné grafiky vytvořené kombinací jednotlivých technik a funkcí.

2 Trojrozměrná počítačová grafika

Trojrozměrnou počítačovou grafikou se rozumí vytváření, manipulace a zobrazování trojrozměrných objektů pomocí specializovaného počítačového softwaru a hardwaru. 3D grafika zahrnuje reprezentaci virtuálního objektu nebo scény v počítači jako soubor geometrických tvarů a povrchů, které lze vykreslit a vytvořit tak realisticky vypadající obraz. [3][4]

2.1 Vývoj

Historie 3D počítačové grafiky sahá až do 60. let 20. století, kdy vědci začali experimentovat s wireframe ,neboli drátovými modely, pro vytváření 3D objektů. V průběhu let byly vyvinuty různé metodiky včetně polygonového modelování, dělení povrchů a modelování skulptur pro vyjmenování alespoň několika z nich.[5]

V současné době je 3D počítačová grafika vysoce specializovanou oblastí informačních technologií, která vyžaduje odborné znalosti algoritmů počítačové grafiky, geometrie, lineární algebry a počítačového hardwaru. Vývoj nových technik a algoritmů pro vytváření a vykreslování 3D grafiky je aktivní oblastí výzkumu a inovací.[5]

K nejnovějšímu vývoji v této oblasti patří trasování světelných paprsků v reálném čase, volumetrické vykreslování a přístupy založené na strojovém učení.[5]

2.2 Základní procesy tvorby

Proces vytváření 3D počítačové grafiky zahrnuje několik kroků, včetně 3D modelování, texturování, nasvícení a vykreslování (renderování). Trojrozměrné modelování zahrnuje vytvoření virtuálních objektů definováním jejich tvaru, velikosti a struktury. Proces texturování zahrnuje přidání povrchových charakteristik, jako je barva a textura, do 3D objektů. Osvětlení zahrnuje přidání

zdrojů světla do virtuální scény a jejich umístění tak, aby vytvořily požadovaný efekt. A konečně vykreslování zahrnuje použití matematických algoritmů k převodu 3D objektů na 2D obrázky, které lze zobrazit na obrazovce.[3]

2.3 Uplatnění technologie

3D počítačová grafika se používá v široké škále oblastí, včetně videoher, filmů, architektonické vizualizace, produktového designu, vědecké vizualizace a virtuální reality. Používání 3D grafiky způsobilo revoluci v mnoha odvětvích tím, že umožnilo vytvářet realistická a pohlcující virtuální prostředí, která nebyla s tradiční 2D grafikou možná.[5]

3 3D webové aplikace

Vzestup internetu výrazně ovlivnil způsob, jakým zacházíme s technologiemi. S příchodem webu 3.0 se internet změnil z pouhého zdroje informací na platformu pro imerzní zážitky. Jedním z nejvýznamnějších trendů posledních let v této oblasti je vznik 3D webových aplikací.[5]

3D webové aplikace jsou webové aplikace, které využívají 3D grafiku a umožňují uživatelům interakci s 3D prostředím v reálném čase. Tyto aplikace mají potenciál revolučně změnit způsob, jakým pracujeme, učíme se a hrajeme, jelikož poskytují interaktivní, pohlcující zážitek, který je dostupný odkudkoli na světě.[5]

Samotný render ve webovém prostředí se týká procesu vytváření a zobrazování trojrozměrného obrazu nebo animace pomocí webových technologií a standardů.[5]

3.1 Počátky 3D technologií pro web

Mezi některé z významných předchůdců moderních technologií, které vydláždily cestu moderním 3D webovým aplikacím, patří brzký standard VRML (Virtual Reality Modeling Language). Dále Java Applet, který umožnil vývojářům vkládat interaktivní 3D grafiku a animace přímo do webových stránek pomocí programovacího jazyka Java, jenž je následně spouštěn v prostředí Virtual Java Machine. A v neposlední řadě za zmínku také stojí multimediální softwarové platformy Adobe Shockwave a Flash Player, které vývojářům umožňovaly vytvářet interaktivní animace, hry a další multimediální obsah, který bylo možné vkládat přímo do webových stránek.[5][6]

3.1.1 Přínos

Tyto technologie přinesly na web nové možnosti a umožnily vývojářům vytvářet interaktivní a dynamický obsah, který dříve nebyl možný pouze pomocí HTML a CSS. To podpořilo experimentování a inovace při vývoji webových stránek. Umožnily vytvářet interaktivní aplikace a hry na webu, což přineslo poutavější uživatelský zážitek[5]

3.1.2 Nevýhody

Jednalo se o dodatečný proprietární software a technologie, jejichž vývoj byl řízen vždy jedinou společností nebo organizací. To omezovalo jejich propojitelnost a ztěžovalo vytváření konzistentních aplikací v různých prohlížečích a zařízeních. Často byly kritizovány za problémy s výkonem. Dalším problémem bylo mnoho možných bezpečnostních chyb, které mohly ohrozit data a soukromí uživatelů. Z tohoto důvodu mnoho webových prohlížečů jejich podporu ve výchozím nastavení zakázalo nebo zcela odstranilo.[5]

3.2 Současné standardy pro tvorbu 3D webových aplikací

Moderní přístupy k vytváření 3D webových aplikací jsou proto často postaveny na několika webových standardech a technologiích jejichž otevřenost a vzájemná kompatibilita umožňuje vývojářům webových aplikací vytvářet konzistentní webové aplikace, které fungují na široké škále zařízení a platforem. Mezi hlavní webové standardy a technologie pro tvorbu 3D webových aplikací patří následující.

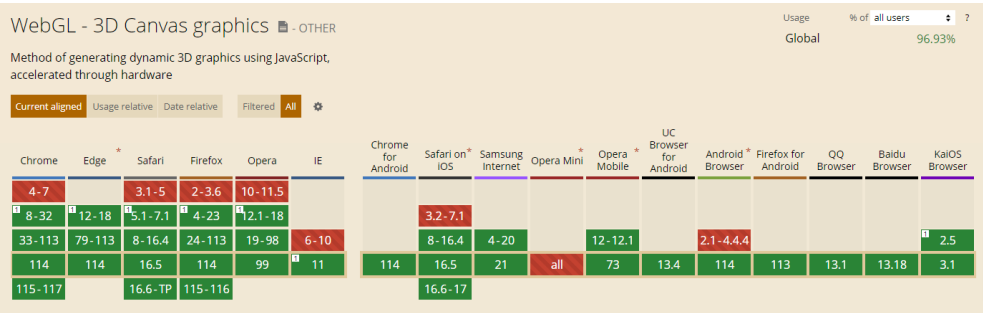
3.2.1 HTML5 a canvas element

HTML5 je nejnovější verze hypertextového značkovacího jazyka, který se používá pro strukturování webových stránek a začlenění 3D grafiky na webovou stránku. V této verzi byl představen <canvas> element, který poskytuje nízkoúrovňový rastrový kreslicí povrch, který lze použít k vytváření složité 2D a 3D

grafiky, včetně vykreslování 3D scén prostřednictvím WebGL.[7]

3.2.2 WebGL

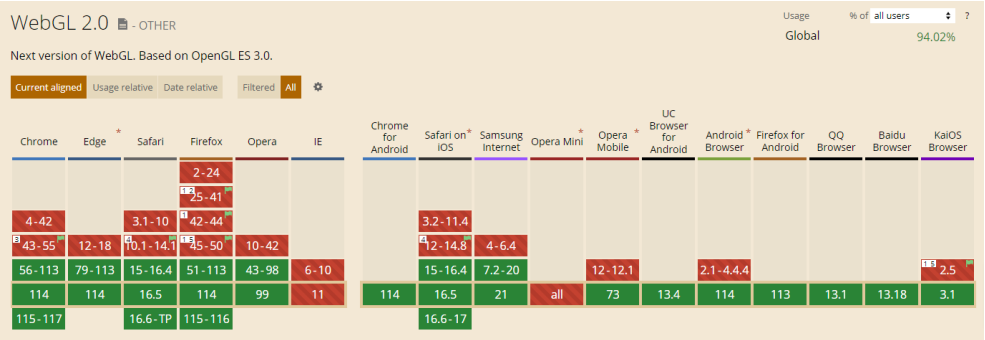
WebGL (Web Graphics Library) je JavaScriptové API pro renderování interaktivních 2D a 3D grafik v kompatibilních webových prohlížečích bez nutnosti použití jakéhokoli speciálního softwaru nebo pluginů. Poskytuje nízkoúrovňový přístup ke grafickému procesoru a umožňuje rychlé vykreslování složitých 3D scén. WebGL byl vyvinut jako standard webové technologie skupinou Khronos a jeho první verze byla vydána v roce 2011. I přes širší podporu starší verze standardu WebGL 1.0 napříč verzemi webových prohlížečů je v současné době pro své vylepšení výkonnosti, bezpečnosti a pro poskytování dodatečných nástrojů a funkcí pro vytváření pokročilých 3D grafických aplikací novější verze WebGL 2.0 používanější.[7][8][9]



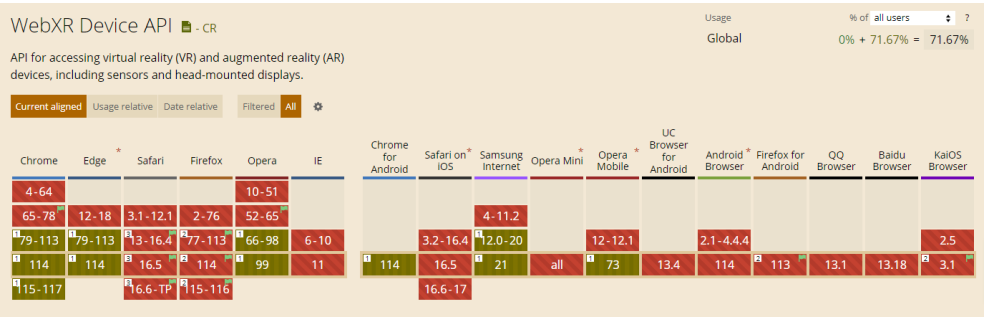
Obrázek 1: Podpora standardu webGl[10]

3.2.3 WebXR

WebXR je API, které umožňuje vytváření virtuální (VR) a rozšířené reality (AR) přímo na webu. Tato technologie umožňuje vytvářet bohaté a imerzivní zážitky, přičemž podporuje širokou škálu zařízení od mobilních telefonů po stolní počítače a specializované VR headsety. Standard WebXR je stále ve vývoji a je spravován World Wide Web Consortium (W3C), který je hlavní mezinárodní standardizační organizací pro internet.[7]



Obrázek 2: Podpora standardu webGl 2[10]



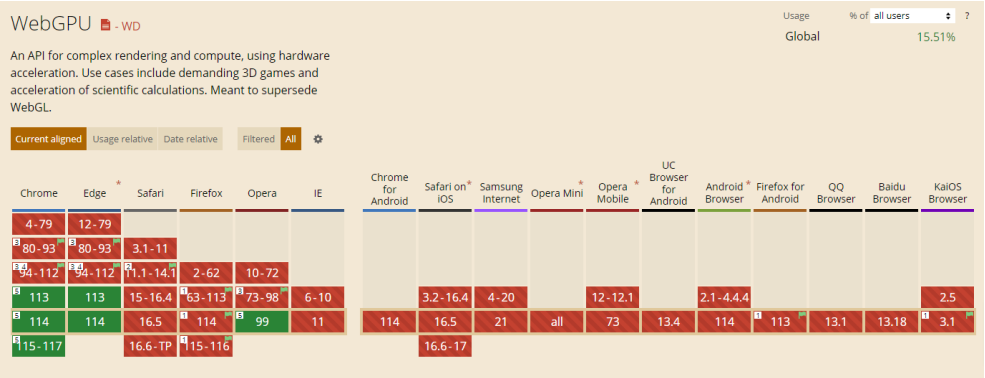
Obrázek 3: Podpora standardu webXR[10]

3.2.4 WebGPU

WebGPU je novější standard, který se snaží poskytnout moderní 3D grafiku a lepší využití výpočetních schopností hardwaru. Nabízí přímější kontrolu nad GPU ve srovnání s WebGL a je navržen tak, aby dobře fungoval na různých systémech. Stejně jako WebXR je i WebGPU spravován World Wide Web Consortium.[7]

3.2.5 High level frameworky

Vysokoúrovňové frameworky pro 3D webový vývoj, jako jsou Babylon.js a Three.js nebo například A-Frame, hrají zásadní roli ve vývoji 3D webových aplikací. Poskytují vyšší úroveň abstrakce než nízkoúrovňové API standardy, a také mnoho užitečných funkcí. Navíc zajišťují kompatibilitu napříč různými



Obrázek 4: Podpora standardu webGPU[10]

platformami a prohlížeči a umožňují import a použití 3D modelů vytvořených v populárních 3D modelovacích programech. Frameworky značně zjednodušují vývoj 3D webových aplikací, činí jej přístupnějším a efektivnějším.[5][7][12]

4 BabylonJS

BabylonJS je JavaScriptový framework, který se zařadil mezi oblíbené nástroje pro tvorbu 3D webové grafiky. BabylonJS je postaven na rozhraních API HTML5 Canvas a WebGL, která poskytují základ pro vytváření 3D webové grafiky.[1]

Babylon.js je projekt s otevřeným zdrojovým kódem, který je hostován na serveru GitHub a je napsán převážně v jazyce TypeScript, což je nadstavba jazyka JavaScript, která přidává volitelné statické typování a další funkce jazyka.To umožňuje lepší organizaci kódu, zvýšení produktivity vývojářů a zlepšení kvality kódu.[1]

Jednou z klíčových předností vykreslovacího jádra Babylon.js je jeho podpora široké škály platforem a zařízení. Je navržen tak, aby byl kompatibilní s různými webovými prohlížeči, včetně desktopových a mobilních prohlížečů. Podporuje také řadu vstupních zařízení, včetně klávesnic, myší, dotykových obrazovek a herních ovladačů.[1]

4.1 Historie

BabylonJS byl původně vytvořen Davidem Catuhem v roce 2012, softwarovým inženýrem společnosti Microsoft. V té době Catuhe pracoval na projektu, který vyžadoval použití WebGL, nízkoúrovňového 3D grafického API, které umožňuje hardwarově akcelerované vykreslování 3D grafiky ve webových prohlížečích. Catuhe však zjistil, že práce přímo s WebGL je složitá a časově náročná, a vyžaduje značné znalosti nízkoúrovňového programování.[12]

Aby tento problém vyřešil, vytvořil Catuhe BabylonJS jako vysokoúrovňový framework, který by zjednodušil proces vytváření 3D webové grafiky pomocí WebGL. Jako jazyk pro tento framework zvolil JavaScript, protože byl již široce používán při vývoji webových aplikací a měl velkou komunitu vývojářů.[12]

BabylonJS byl poprvé uvolněn jako open-source projekt na GitHubu v roce

2013 a díky snadnému použití, výkonné sadě nástrojů a obsáhlé dokumentaci si rychle získal oblibu mezi vývojáři.[1]

V průběhu let se BabylonJS dále vyvíjel a zdokonaloval. Pravidelně přibývaly nové funkce a vylepšení. V roce 2016 byl BabylonJS oceněn jako nejlepší open-source herní engine v soutěži Develop Awards, což ještě více upevnilo jeho pozici předního frameworku pro 3D webovou grafiku.[12]

4.2 Alternativy

4.2.1 ThreeJS

Tato JavaScriptová knihovna je významnou volbou pro vývojáře webových 3D aplikací. Je oblíbená pro svou flexibilitu a rozsáhlou sadu funkcí, které umožňují tvorbu detailních a komplexních 3D scén s širokou kontrolou nad materiály, světly a stíny. S Three.js je možné vytvářet efektní vizualizace a interaktivní zážitky s vysokým stupněm přizpůsobení. Stejně jako v případě Babylonu jej lze použít k vytváření 3D webové grafiky pro širokou škálu platforem, včetně stolních počítačů a mobilních zařízení, ale také pro virtuální a rozšířenou realitu. Nicméně, Three.js může být náročnější pro nováčky v oblasti 3D webového vývoje, neboť vyžaduje hlubší porozumění 3D programování. Three.js má však v porovnání s Babylonem velkou a aktivní komunitu vývojářů, kteří vytvořili širokou škálu zdrojů, včetně výukových programů, příkladů a zásuvných modulů.[13][14]

4.2.2 A-Frame

Tento framework, který je založen na entity-component systému, a používá HTML-podobnou deklarativní syntaxi, je unikátní ve své snaze o přístupnost a snadné použití. A-Frame je speciálně navržen tak, aby usnadnil vytváření zážitků virtuální a rozšířené reality pro web. Má řadu funkcí pro práci s VR a AR. A-Frame umožňuje vývojářům snadno vytvářet VR zážitky přímo v prohlížeči. Jeho největší výhodou je, že nevyžaduje hluboké znalosti JavaScriptu

a 3D grafiky, což umožňuje rychlý vývoj a testování bez nutnosti psát velké množství kódů. Nicméně, pokud se projekt stává složitějším a vyžaduje více specifické funkce, A-Frame může být omezený v porovnání s robustnějšími knihovnami jako Babylon.js nebo Three.js. To znamená, že zatímco A-Frame je ideální pro menší projekty nebo prototypy, může se stát omezením pro větší a složitější projekty. Je postaven nad frameworkem Three.js, díky čemuž zdědil mnoho jeho vlastností a silných stránek.[13]

4.2.3 Unity WebGL

Unity je jedním z nejpopulárnějších nástrojů pro vývoj her na světě a jeho WebGL rozšíření umožňuje týmům, které jsou již seznámeny s tímto nástrojem, rychle a efektivně přizpůsobit své nativní hry a aplikace webovému prostředí. Díky tomu je Unity WebGL často využíván v herním průmyslu pro tvorbu webových verzí stávajících her. Představuje jakýsi most mezi světem nativních aplikací a webovým prostředím. Unity nabízí řadu funkcí pro tvorbu 3D her a simulací, včetně fyzikálních enginů, animačních nástrojů a podpory skriptování v jazyce C#. S podporou pro import aktiv z široké škály formátů a kompatibilitou s mnoha různými typy hardware, Unity WebGL je také vhodný pro náročné projekty, které vyžadují širokou škálu funkcí a optimalizace.[15] Přestože je Unity WebGL silný nástroj, má i svá omezení. Například časy načítaní mohou být delší než u ostatních webových 3D knihoven a mohou existovat problémy s kompatibilitou se staršími prohlížeči a systémy. Navíc, Unity WebGL vyžaduje licenci pro komerční využití, což může být překážkou pro menší týmy a nezávislé vývojáře.[15]

5 Architektura frameworku

Babylon.js renderovací engine je postaven na WebGL, nízkoúrovňové grafické API, které mu umožňuje přímý přístup ke grafickému procesoru v počítači uživatele. To umožňuje vysoce výkonné 3D vykreslování ve webovém prohlížeči bez nutnosti používat proprietárních zásuvných modulů nebo softwaru.[11]

Celý framework je pak strukturován jako modulární knihovna, která se skládá z jednotlivých funkčních modulů. Tato modulární konstrukce systému Babylon.js umožňuje vývojářům vybrat si, které komponenty chtějí do svých projektů zahrnout. To pomáhá udržet projekt lehký a zaměřený na konkrétní požadavky, a přitom v případě potřeby poskytnout přístup k výkonným nástrojům.[16]

5.1 Moduly

1. babylonjs (jádro Babylonu)

   Hlavní knihovna frameworku Babylon.js, která zahrnuje základní funkčnosti potřebné pro vytvoření 3D scén. Poskytuje vysoce úrovňové API pro vytváření, manipulaci a vykreslování 3D scén ve webovém prohlížeči pomocí WebGL. Funkce zahrnují scénický graf, světla, kamery, materiály, textury, modely sítí, animace, detekci kolizí, podporu fyzikálního enginu a mnoho dalšího.[16]

2. babylonjs-materials (kolekce pokročilých materiálů)

   Tento modul poskytuje sadu pokročilých materiálů, které lze použít pro konkrétní vzhled nebo pocit vašich 3D objektů. Zahrnuje materiály jako kožešina, gradient, normály, jednoduché, obloha a terénní materiály. Každý materiál má vlastní sadu vlastností, které lze přizpůsobit pro dosažení požadovaného efektu.[16]

3. babylonjs-loaders (oficiální loadery)

   Tento modul poskytuje třídy načítačů pro import 3D modelů z různých formátů do scény Babylon.js. V současné době podporuje formáty jako

.babylon, .gltf, .glb (binární glTF), .obj, .stl a další. Každý načítač je schopen analyzovat formát souboru a převést ho do formátu, kterému Babylon.js může porozumět a vykreslit.[16]

4. babylonjs-post-process (postprocesy)
   Tento modul umožňuje aplikovat postprocesingové efekty na render scény. Postprocesingové efekty jsou techniky používané k vylepšení kvality vykreslování nebo vytvoření speciálních efektů. Zahrnuje efekty jako černobílé zobrazení, rozmazání, chromatická aberace, konvoluce, hloubka ostrosti, zrno, zvýraznění a další.[16]

5. babylonjs-procedural-textures (podporované procedurální textury)
   Modul poskytuje sadu procedurálních textur. Procedurální textury jsou textury, které jsou definovány matematickou funkcí, namísto toho, aby byly načteny ze souboru obrázku. To může poskytnout unikátní vizuální efekty a také ušetřit paměť, protože textura nemusí být uložena jako obrázek.[16]

6. babylonjs-serializers (serializéry scény / sítě)
   Modul poskytující funkčnost pro serializaci (převod do formátu, který lze uložit nebo přenášet) a deserializaci (převod zpět do původní formy) scén a sítí. To je užitečné pro ukládání stavu scény nebo sítě pro pozdější použití, či pro přenos po síti.[16]

7. babylonjs-gui (grafické uživatelské prostředí)
   Tento modul poskytuje nástroje a funkce pro vytváření interaktivních grafických uživatelských rozhraní (GUI) v dané 3D scéně. Podporuje různé typy ovládacích prvků jako jsou tlačítka, posuvníky, vstupní text, výběr barvy, obrázek a další. S prvky GUI lze plně interagovat pomocí myši nebo dotykových událostí. Umožňuje debugování a profilování aplikací Babylon.js a zahrnuje funkce jako je nástroj Inspector a nástroje pro logování.[16]

8. babylonjs-inspector (inspektor)

   Tento modul je nástrojem pro ladění, který poskytuje detailní pohled na strukturu a data 3D scény. Umožňuje vývojářům prozkoumat a manipulovat s grafem scény, materiály, texturami, světly a dalšími aspekty scény v reálném čase. Inspektor Babylon.js lze otevřít přímo z běžícího programu Babylon.js a umožňuje tak rychlé a pohodlné ladění a optimalizaci aplikace.[16]

9. babylonjs-viewer (samostatný prohlížeč Babylon.js)

   Tento modul poskytuje jednoduchý a snadný způsob, jak vložit 3D scénu do webové stránky bez nutnosti psaní jakéhokoli JavaScriptového kódu. Prohlížeč lze konfigurovat pomocí HTML atributů nebo konfiguračního souboru JSON. Podporuje načítání souborů .babylon, .gltf a .glb, a také zahrnuje podporu pro HDR prostředí a různé další funkce.[16]

Samotné jádro Babylon.js je páteří celého frameworku a poskytuje základní infrastrukturu, která umožňuje bezproblémovou spolupráci ostatních modulů. Ostatní základní moduly poskytují další doplňující funkce a jejich použití je čistě vázané na potřebu v rámci funkcionalit projektu.[16]

6 Online nástroje

Babylon.js nabízí ve svém jádru výkonný vykreslovací engine, ale jeho skutečný potenciál spočívá také, mimo jiné, v komplexní sadě online nástrojů, které poskytuje nad tímto rámcem. Ty se starají o různé aspekty vývoje 3D webových stránek a svými funkcemi zjednodušují proces vývoje, a to vše v rámci jednoho uceleného a propojeného ekosystému. Nabídkou těchto nástrojů nad základním frameworkem se Babylon.js zařadil mezi přední volby pro vývojáře, kteří chtějí vytvářet špičkové 3D aplikace pro web.[17]

6.1 Playground

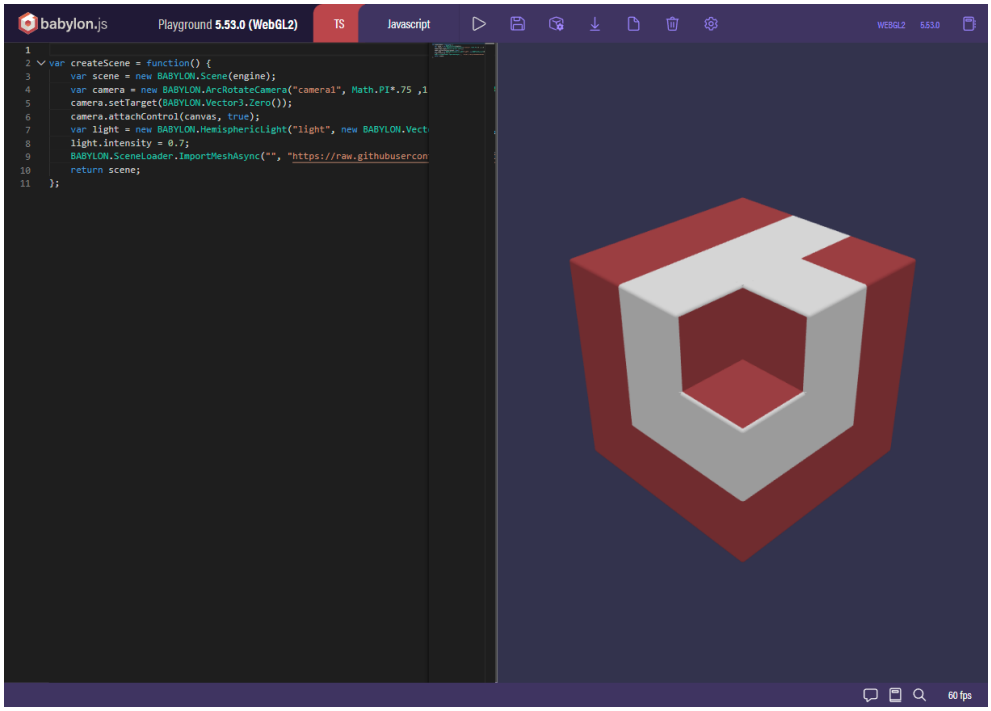
Babylon.js Playground dostupný na webové adrese https://playground.babylonjs.com/ je funkčně bohatý a výkonný online nástroj, který zjednodušuje proces práce s 3D scénami v rámci Babylon.js. Díky integrovanému editoru kódu, náhledu v reálném čase, možnostem sdílení a spolupráce, integraci nástroje Inspector, a možnostem exportu a importu, je Playground nepostradatelným nástrojem pro vývojáře, kteří se chtějí učit, vytvářet a zdokonalovat své 3D zážitky na webu. Umožňuje rychlé vytváření prototypů a experimentování s Babylon.js přímo z prostředí webového prohlížeče. [17]

6.1.1 Editor kódu

Playground obsahuje vestavěný editor kódu, který vývojářům umožňuje psát a upravovat kód v jazyce JavaScript nebo TypeScript přímo v nástroji. Editor kódu je vybaven funkcemi zvýrazňování syntaxe, automatickým doplňováním a kontrolou chyb, které usnadňují bezproblémové psaní kódu.[17]

Editor musí vždy obsahovat definici funkce s názvem createScene, která vrací objekt scény. Na základě definice tohoto objektu je následně vykreslená požadovaná 3D scéna. Při prvním otevření nástroje již editor obsahuje výchozí kód funkce s vytvořením základní scény. Tato scéna kterou funkce navrací se skládá z kamery, zdroje světla a jednoduché 3D geometrie koule a plochy.

Předpřipravený kód slouží jako výchozí bod, se kterým mohou vývojáři experimentovat, upravovat a rozšiřovat jej podle svých potřeb.[16][17]



Obrázek 5: Prostředí nástroje Babylon.js Playground

6.1.2 Náhled 3D scény

Playground nabízí náhled 3D scény v reálném čase, takže vývojáři mohou okamžitě vidět výsledky změn svého kódu. Tato funkce eliminuje nutnost přepínat mezi samostatnými okny kódu a náhledu, čímž zefektivňuje proces vývoje.[17]

Pro zobrazení výsledků změn provedených v kódu je potřeba kliknout na tlačítko "Run"v liště nástrojů nebo stisknout klávesy Alt + Enter. Tím se kód spustí a náhled 3D scény se aktualizuje v reálném čase.[16]

6.1.3 Ukládání změn

Po provedení požadovaných změn mohou uživatelé svou práci uložit kliknutím na tlačítko "Save"nebo klávesovou zkratku Ctrl + S. Tím se vytvoří nová

verze Playgroundu a vygeneruje se adresa URL (příklad: https://playground.babylonjs.com/#420KZS) s jedinečným identifikátorem projektu (v tomto případě: #420KZS). Tuto adresu URL pak mohou uživatelé sdílet s ostatními nebo ji přidat do záložek pro budoucí použití.[16]

S každým dalším uložením nedochází k přepsání předchozího uloženého stavu aplikace, jelikož si playground udržuje historii verzí uložených projektů, což uživatelům umožňuje přístup k předchozím iteracím jejich práce. Každá tato verze je pak přístupná přidáním znaménka # a indexu verze za jedinečnou adresu projektu (příklad url adresy s verzí projektu: https://playground.babylonjs.com/#420KZS#1) [16][17]

6.1.4 Export a import

Vývojáři mohou do svých projektů importovat externí prostředky (například 3D modely, textury nebo skripty) nebo exportovat své výtvory v různých formátech, včetně .gltf, .glb a .babylon.[16][17]

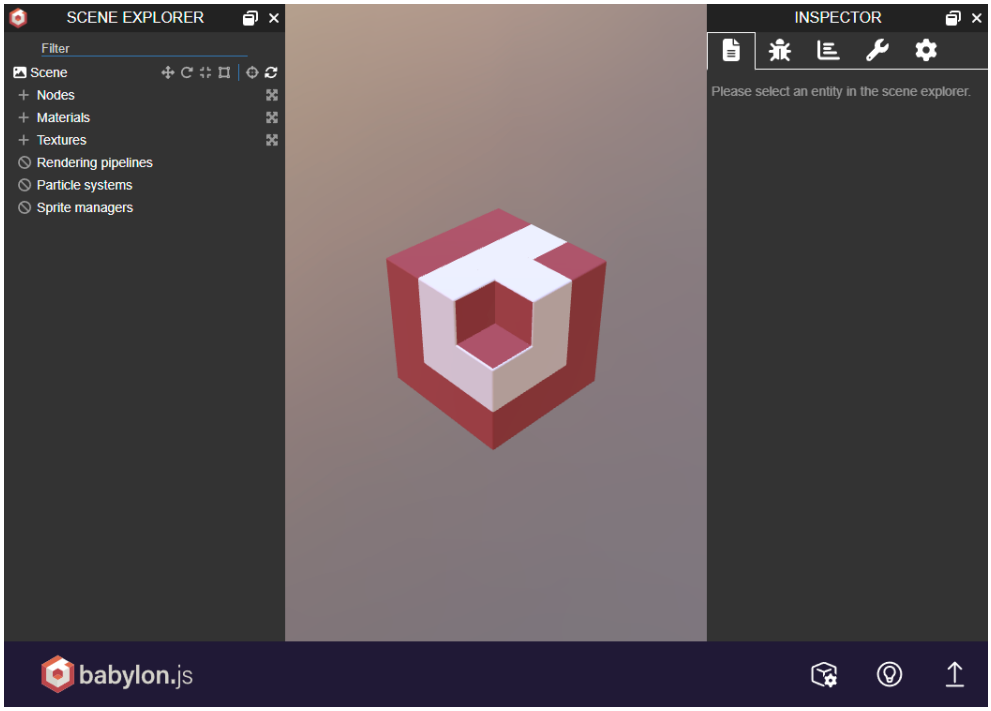
Použití externích prostředků sice může zlepšit proces vývoje, je však nutné zajistit, aby si vytvořený projekt zachoval dostupnost a funkčnost pro budoucí použití. Zjednodušení pomocí základních meshů, existujících textur a modelů může vést k rychlejšímu vývoji, porozumění problému a jejich řešení v prostředí playgroundu. Pokud je však použití externích zdrojů nezbytné, je zapotřebí zajistit přístup k souborům v rámci internetu skrze URL adresu, například jejich hostováním skrze služby Githubu, Gitlabu, Dropboxu, Imguru a dalších. Při výběru je třeba mít na paměti, že všechny stránky hostujících zdrojů musí být kompatibilní s CORS (Cross-Origin Resource Sharing) a používat protokol HTTPS.[16]

6.2 Sandbox

Babylon.js Sandbox je důležitý nástroj pro vývojáře pracující s externími modely v rámci frameworku. Sandbox je určen jako intuitivní a uživatelsky pří-

větivé online prostředí pro rychlé testování, iteraci a ladění 3D modelů a scén, díky integraci nástroje inspektoru a všech jeho funkcionalit, a to vše izolovaně a nezávisle na zbytku projektu. Nástroj je dostupný na webové adrese https://sandbox.babylonjs.com/.[16][17]

Import 3D modelů do nástroje Sandbox je snadný. Lze provést buď pomocí tlačítka pro import v pravém dolním rohu, nebo přetáhnutím souboru 3D modelu (například .gltf, .glb nebo .babylon) do okna prohlížeče. Model se poté automaticky načte a zobrazí ve viewportu. Tím odpadá nutnost psát kód jen pro náhled modelu, což usnadňuje kontrolu a úpravu 3D zdrojů.[16]



Obrázek 6: Importovaný model v prostředí Sandboxu

Jakmile je 3D model načten do Sandboxu, mohou s ním vývojáři pracovat a upravovat v reálném čase, a jakmile jsou s úpravami spokojeni, mohou za pomoci funkce inspektoru opětovně exportovat projekt do různých formátů, včetně .gltf, .glb a .babylon.[16]

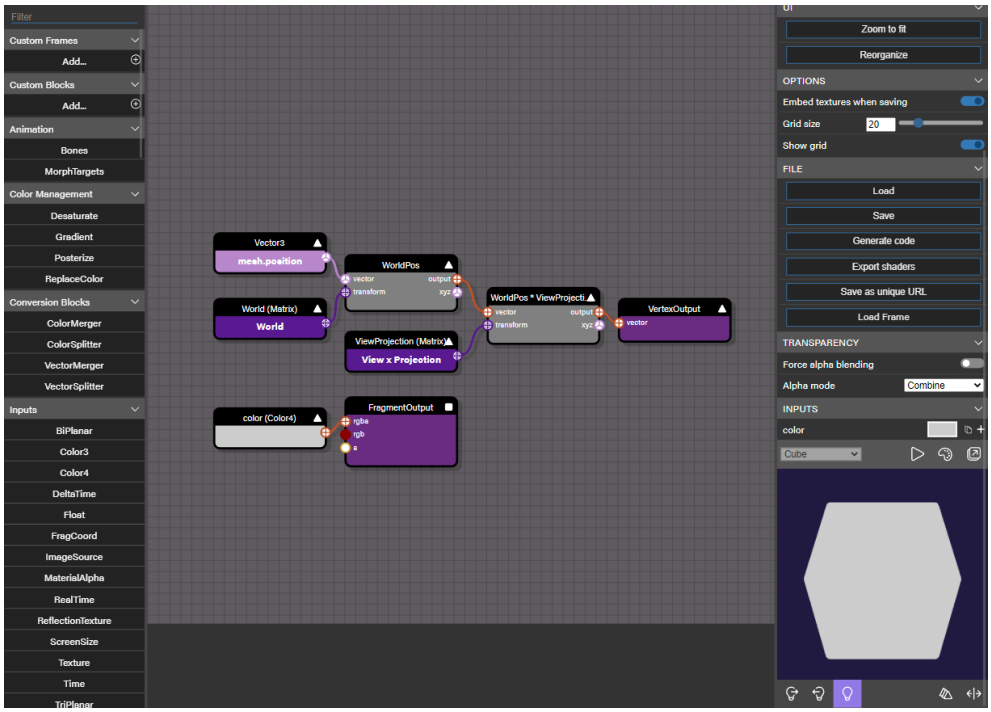
6.3 Node Material Editor

Node Material Editor je nástroj určený k usnadnění tvorby a přizpůsobení shaderů pro 3D scény v Babylon.js projektech. Je navržen tak, aby vyhovoval specifickým potřebám a požadavkům projektu. Tento editor eliminuje nutnost ručního psaní kódu shaderů a umožňuje vizuálně vytvářet a upravovat složité materiály pomocí vizuálního programovacího rozhraní. Tento nástroj tak umožňuje vytvářet pokročilejší a jedinečné vizuální efekty, které mohou vylepšit celkový vzhled 3D scény.[16][17]

Toto rozhraní se skládá z plátna, na které lze přidávat, propojovat a uspořádávat logické bloky a definovat tak požadovaný materiál. Jednotlivé bloky představují různé operace, vstupy, výstupy a další prvky, které se podílejí na výsledném shaderu. Editor poskytuje rozsáhlou kolekci zabudovaných logických prvků jako jsou základní matematické operace, vzorkování textur a vstupní uzly reprezentující různá data o scéně, například pozici ve virtuálním světě nebo UV souřadnice.[16][17]

Při vytváření materiálu lze jednoduše přetahovat uzly z knihovny na plátno a propojovat je do struktury podobné grafu. Spojení mezi bloky určují tok dat a pořadí operací, a to v konečném důsledku určuje vzhled materiálu. Editor také umožňuje snadnou změnu uspořádání, přidávání nebo odstraňování bloků, což umožňuje flexibilní proces návrhu. Mimo to poskytuje Node Material Editor náhled shaderu v reálném čase a zobrazuje vzhled materiálu na 3D modelu. Tato funkcionalita umožňuje uživatelům rychle pozorovat výsledky svých změn a provést odpovídající úpravy.[16][17]

Jakmile je materiál vytvořen a doladěn, mohou jej vývojáři exportovat jako soubor JSON nebo snippet Babylon.js, který lze následně importovat do projektu Babylon.js. Editor také podporuje import existujících materiálů pro další úpravy nebo kontrolu.[16]



Obrázek 7: Node Material Editor

7 Implementace

Před samotnou implementací Babylon.js v rámci projektu je třeba zvážit několik faktorů, které je třeba vzít v úvahu při rozhodování se mezi způsoby použitím Babylon.js. Existuje totiž více možností jak framework pro účel vizualizace 3D aktiv na stránkách použít a pro každý způsob pak existují ruzné způsoby implementace.[11][16]

7.1 Babylon.js Viewer

Prohlížeč Babylon.js Viewer je nenáročná a snadno použitelná komponenta, která umožňuje vkládat a zobrazovat 3D scény a modely na webových stránkách bez nutnosti detailní znalosti frameworku Babylon.js nebo konceptů 3D programování. Pomocí Vieweru lze zjednodušit proces přidávání 3D obsahu do svých projektů a zároveň využívat výkonné vykreslovací schopnosti frameworku Babylon.js.[11][16]

Viewer slouží jako souhrnný element pro engine Babylon.js a dodržuje stejný systém verzování jako engine samotný. Na základě toho je distribuován ve dvou verzích. První je stabilní verze využívající aktuální stabilní verzi enginu Babylon.js a následně preview verze, která využívá preview verzi enginu.[16]

Viewer výrazně snižuje složitost integrace 3D obsahu do webových stránek, což z něj činí ideální řešení pro vývojáře, kteří vyžadují jednoduchý a rychlý způsob přidávání 3D scén nebo modelů. Viewer také automaticky zpracovává změnu velikosti a responzivitu, čímž zajišťuje, že 3D obsah vypadá a funguje dobře na různých zařízeních a při různých velikostech obrazovky. [16]

Babylon.js Viewer je jednoduchý za cenu omezené kontroly nad 3D scénou. Ačkoliv viewer do jisté míry úpravy zobrazení umožňuje, pro pokročilejší úpravy a přesnější ovládání chování scény může být zapotřebí použití přímo frameworku Babylon.js. Přestože je Viewer optimalizován z hlediska výkonnosti, nemůže nabídnout takové možnosti optimalizace pro konkrétní případy použití jako vývoj v rámci samotného frameworku.[16]

7.1.1 Přidání Vieweru do projektu

Pro přidání Vieweru Babylon.js do projektu je nejdříve zapotřebí přidání script tagu v hlavičce HTML souboru, v rámci něhož se získá reference na hostovanou verzi knihovny Vieweru sdílenou na CDN serverech.[16]

```
1 <script src="https://cdn.babylonjs.com/viewer/babylon.
    viewer.js"></script>
```

Příklad 1: Script tag s referencí stabilní verze Vieweru[16]

Pak již stačí přidat do těla souboru HTML element babylon s požadovanými atributy, například atributem model s hodnotou adresy URL k souboru 3D modelu.[16]

```
1 <babylon model="model.gltf"></babylon>
```

Příklad 2: Script tag s referencí stabilní verze Vieweru[16]

7.2 Export HTML souboru z nástroje Playground

Další způsobem přidání 3D prvků do svého projektu je možnost vyžití nástroje Playgroundu pro export plně funkčního HTML souboru, obsahující kanvas již s plně implementovaným enginem Babylon.js a veškerou logikou daného vyvíjeného projektu Playgroundu.[11][16][17] Tento přístup umožňuje rychlou tvorbu prototypů 3D scén, což uživatelům umožňuje experimentovat s různými nastaveními a prohlížet si výsledky v reálném čase před jejich začleněním do svých projektů. Navíc díky exportu souboru HTML odpadá proces nastavení rámců Babylon.js v rámci projektu, tím se šetří čas a úsilí, zejména u menších projektů nebo projektů s omezeným 3D obsahem.[11][16]

Další výhodou používání exportovaných souborů HTML je snadné sdílení s ostatními. Tato pohodlná metoda usnadňuje spolupráci na 3D scénách nebo modelech, aniž by příjemce musel konfigurovat rámec Babylon.js. Exportovaný soubor HTML zahrnuje všechny potřebné kódy a prostředky Babylon.js. To umožňuje samostatné spuštění 3D scény nebo modelu bez nutnosti připojení k externím zdrojům.[16][17]

Exportování souboru HTML z Babylon.js Playground má také některé nevýhody, například omezenou flexibilitu, problémy s údržbou a výkonnostní problémy. Při použití exportovaného souboru HTML nemusí být k dispozici stejná úroveň kontroly a flexibility jako při přímém použití frameworku Babylon.js. To může představovat problém pro pokročilé případy použití nebo v případech, kdy je zapotřebí přesná kontrola nad scénou.[16][17]

Navíc s rozšiřováním projektu může být správa více exportovaných souborů HTML stále složitější a obtížnější, a to ztěžuje sledování změn a údržbu zdrojového kódu. To může vést k potenciálním problémům při aktualizaci nebo optimalizaci 3D obsahu. Z hlediska výkonu nemusí být exportovaný soubor HTML tak optimalizovaný jako projekt vytvořený přímo pomocí rámce Babylon.js. U složitých scén nebo při cílení na specifické hardwarové konfigurace se může ukázat jako vhodnější volba přímé použití frameworku.[16]

7.2.1 Export z nástroje Playground

Nejprve je třeba vytvořit požadovanou 3D scénu nebo model v online nástroji Babylon.js Playground a provést všechny potřebné úpravy, aby bylo dosaženo požadovaného výsledku. Jakmile je scéna vyhovující, stačí kliknout na tlačítko "Download"na panelu nástrojů či použít klávesovou zkratku CTRL+SHIFT+S a uložit do lokálního úložiště komprimovaný balíček (.zip) souborů, obsahující soubory textur, modelů a jiných prvků spolu se zcela funkčním index.html souborem.[11][16]

Obrázek 8: Zvýrazněné tlačítko z lišty nástrojů pro stažení souboru obsahující HTML soubor se scénou.

7.2.2 Přidání do projektu

Integrace exportovaného souboru HTML z Babylon.js Playgroundu do projektu může být provedena několika způsoby v závislosti na struktuře a požadavcích projektu. Každý přístup má své výhody a nevýhody, které je třeba zvážit, aby byla zajištěna bezproblémová integrace se stávajícím projektem.[11][16]

Jednou z možných metod je přímé vložení obsahu exportovaného souboru HTML do stávající struktury HTML. Kód Babylon.js, včetně elementu canvas a všech potřebných skriptů, lze zkopírovat a vložit do souboru HTML svého projektu na požadované místo, kde 3D scénu nebo model chceme zobrazit. Tato metoda umožňuje snadnou integraci 3D obsahu do stávajícího rozvržení a stylování, což z ní činí vhodnou volbu pro mnoho projektů.[11][16]

Alternativně lze k vložení exportovaného souboru HTML do projektu použít iframe. Tento přístup udržuje kód Babylon.js nezávislý na hlavním souboru HTML, což efektivně odděluje 3D obsah od ostatních prvků stránky. Za tímto účelem se v souboru HTML projektu vytvoří prvek iframe a jeho atribut

"src"se nastaví na cestu k exportovanému souboru HTML. Tato metoda může být obzvláště užitečná pro správu a údržbu 3D obsahu odděleně od zbytku projektu.[16]

Další možnost zahrnuje vytvoření odkazu nebo tlačítka v projektu, které otevře exportovaný soubor HTML v novém okně nebo kartě prohlížeče. Tato metoda udržuje 3D obsah zcela oddělený od hlavního projektu a poskytuje uživatelům vyhrazené okno pro interakci s 3D scénou nebo modelem. Pro implementaci této možnosti se do souboru HTML projektu přidá prvek hypertextového odkazu nebo tlačítka a atribut "href"(u hypertextových odkazů) nebo událost onclick (u tlačítek) se nastaví tak, aby se otevřela adresa URL exportovaného souboru HTML.[16]

7.3 Přímá implementace frameworku

Třetím přístupem k integraci 3D obsahu na webové stránky pomocí Babylon.js je přímé použití rámce Babylon.js. Tato metoda poskytuje největší flexibilitu a kontrolu nad 3D obsahem a je vhodná pro projekty, které vyžadují pokročilé funkce, přizpůsobení a optimalizaci výkonu. Přímé použití frameworku zahrnuje integraci knihovny Babylon.js do projektu a následné napsání vlastního kódu pro vytváření, manipulaci a vykreslování 3D scén.[11][16]

7.3.1 CDN

Oblíbenou a pohodlnou metodou začlenění rámce Babylon.js do projektu je použití sítě pro doručování obsahu (CDN). Sítě CDN hostují knihovnu Babylon.js a doručují ji uživatelům s nízkou latencí, čímž poskytují rychlý a efektivní způsob přístupu ke knihovně. Pomocí sítě CDN mohou vývojáři snadno začlenit rámec Babylon.js do svého projektu, aniž by museli sami stahovat nebo spravovat soubory knihovny.[7][16]

Využití sítě pro doručování obsahu nabízí při začlenění rámce Babylon.js do projektu několik výhod. Jednou z těchto výhod je jeho jednoduchost. Začlenění

knihovny vyžaduje pouze script tag v souboru HTML, což je snadná a nenáročná metoda pro vývojáře, kteří chtějí s rámcem rychle začít. Další výhodou jsou automatické aktualizace. Verze rámce hostovaného v síti CDN je obvykle automaticky aktualizována při vydání nové verze. Tím je zajištěno, že projekt vždy používá nejnovější a nejstabilnější verzi knihovny, aniž by vývojář musel provádět ruční aktualizace. Kromě toho sítě CDN ukládají soubory knihovny do mezipaměti na více serverech po celém světě, což může zlepšit rychlost načítání 3D obsahu, zejména pro uživatele, kteří se nacházejí daleko od serveru hostujícího projekt. A konečně, přenesením hostování knihovny Babylon.js na síť CDN se sníží zatížení vašeho vlastního serveru, což může pomoci zlepšit celkový výkon projektu.[7][16]

Na druhou stranu má toto použití i určitá omezení. Jedním z omezení je, že není možné mít plnou kontrolu nad verzí knihovny, což může být problém, pokud projekt závisí na konkrétní verzi nebo pokud potřebujete použít vlastní úpravy knihovny. Další potenciální nevýhodou je riziko výpadku nebo přerušení provozu, což by mohlo ovlivnit dostupnost knihovny Babylon.js. Ačkoli jsou sítě CDN obecně spolehlivé, vždy existuje možnost, že k těmto problémům dojde.[7][16]

Každý modul je poskytován ve formě vlastní adresy URL hostované v CDN, což vývojářům umožňuje zahrnout pouze specifické funkce potřebné pro jejich projekty, čímž se snižuje celková velikost a složitost aplikace. Vždy je zapotřebí zahrnout minimálně babylon.js core modul. Toho lze docílit přidáním značky skriptu do hlavičky souboru HTML.[16]

```
1 <script src="https://cdn.babylonjs.com/babylon.js"></
    script>
```

Příklad 3: Script tag užívající CDN pro začlenění core modulu frameworku[16]

7.3.2 Použití Babylon.js s UMD/NPM

Univerzální definice modulů (UMD) umožňuje JavaScriptovým modulům pracovat v mnoha různých prostředích jako je prohlížeč, Node.js nebo AMD (Asynchronous Module Definition). Pokud je Babylon.js používán s UMD/-NPM, umožňuje využít sílu npm ke správě dependencí projektu.[7]

Je zapotřebí nejprve nainstalovat Node.js a npm. Po jejich instalaci lze přidat Babylon.js do projektu tak, že v terminálu se přejde do adresáře projektu a spustí příslušným terminálovým příkazem.[7][16]

```
1 npm install --save babylonjs
```

Příklad 4: Terminálový příkaz pro přidání frameworku za pomocí metody UMD/NPM[16]

Tím se nainstaluje Babylon.js a přidá se do adresáře node_modules projektu. Poté lze již importovat Babylon.js v rámci projektu.[16]

```
1 const BABYLON = require('babylonjs');
```

Příklad 5: UMD/NPM import Babylonu a jeho částí v souborech JavaScript[16]

Při použití UMD/NPM metody je zapotřebí využívat moderní nástroje a systémy pro sestavení jazyka JavaScript, jako jsou kupříkladu Webpack nebo Parcel, které kód a závislosti spojí do jediného souboru JavaScriptu.[7]

7.3.3 Použití Babylon.js s ES6/NPM:

ES6 (známý také jako ECMAScript 2015) zavedl do JavaScriptu systém modulů, který usnadňuje strukturování rozsáhlých kódových bází a správu závislostí mezi různými částmi kódu.[7]

Při použití Babylon.js s ES6/NPM, budete postupovat podobně jako při nastavení UMD/NPM. Nejdříve je zapotřebí nainstalovat Node.js a npm a poté přidat Babylon.js do projektu opět za pomocí terminálového příkazu.[16]

```
1 npm install --save @babylonjs/core
```

Příklad 6: Terminálový příkaz pro přidání frameworku za pomocí metody ES6/NPM[16]

Rozdíl při postupu přidání frameworku je tak v použití odlišného názvu balíčku. Balíček '@babylonjs/core' poskytuje právě verzi ES6 Babylon.js. Jednotlivé komponenty lze následně podle potřeby importovat z Babylon.js. Při užívání například tříd Engine a Scene import provedeme následovným způsobem.[16]

```
1 Import { Engine, Scene } from '@babylonjs/core';
```

Příklad 7: ES6/NPM import částí frameworku v souborech JavaScript[16]

To, že lze importovat pouze ty části Babylon.js, které jsou skutečně třeba, je hlavní výhodou používání Babylon.js s ES6/NPM. Výsledkem pak může být menší velikost souboru a rychlejší načítání aplikace.[7][16]

V obou případech se bude moci použít npm pro správu dalších závislostí projektu. Použití modulů ES6 s Babylon.js však může vést k efektivnějšímu kódu, protože umožňuje stromové protřepávání, což je proces, který eliminuje nepoužívané exporty ve konečném balíčku.[7]

8 Inicializace a vytvoření první 3D scény

8.1 Canvas

Element <canvas> byl zaveden jako součást specifikace HTML5, která byla dokončena a zveřejněna konsorciem World Wide Web Consortium (W3C) v říjnu 2014. Účelem elementu je poskytnout vyhrazenou kreslicí plochu pro dynamické vykreslování za pomoci jazyka JavaScript. Umožňuje tak provádět programové práce s grafikou včetně 2D a 3D grafiky, animací a vizuálních efektů přímo ve webovém prohlížeči.[7]

Od svého začlenění do specifikace HTML5 se stal široce podporovaným moderními webovými prohlížeči a stal se standardní součástí vývoje webových aplikací pro vytváření dynamické a interaktivní grafiky.[7]

V kontextu Babylon.js tak hraje HTML element <canvas> klíčovou roli při vytváření 3D scén, jelikož slouží jako kontejner pro vykreslovací jádro Babylon.js. Poskytuje vyhrazenou oblast na webové stránce, kde se zobrazuje 3D scéna. Zadáním rozměrů prvku canvas můžete řídit velikost vykreslované scény.[16]

Prvek canvas funguje jako hostitel kontextu WebGL, který umožňuje rozhraní Babylon.js komunikovat s příslušným grafickým hardwarem. Prostřednictvím kanvasu získává Babylon.js přístup k nízkoúrovňovým funkcím akcelerace GPU pro efektivní vykreslování 3D scény.[5][7][16]

Dále umožňuje interakci uživatele s 3D scénou. Babylon.js poskytuje různé vstupní metody, jako jsou události myši, dotyku a klávesnice, které umožňují manipulaci s objekty a navigaci ve scéně. Připojením posluchačů událostí k prvku canvas lze zachytit vstup uživatele a spustit odpovídající akce v aplikaci Babylon.js.[16]

8.1.1 Integrace v rámci HTML a CSS

Kanvas se bez problémů integruje s ostatními prvky HTML a styly CSS. Plátno je možné umístit v rámci rozvržení webové stránky, použít styly a překrýt prvky HTML nad 3D scénou. Tato flexibilita umožňuje kombinovat výkon Babylon.js s všestranností HTML a CSS.[11][16]

```
1 <canvas id="renderCanvasId"></canvas>
```

Příklad 8: Kód pro přidání kanvasu v rámci těla HTML[16]

Při vytváření plátna je potřeba přiřadit mu jedinečný identifikátor pomocí atributu elementu 'id', pomocí jehož bude v pozdější fázi možné v rámci JavaScriptu vytvořit instance elementu pro pozdější práci s ním.[11][16]

Pokud element <canvas> do kódu HTML pouze vložíte, aniž by se výslovně nastavily jeho rozměry, bude se ve výchozím nastavení vykreslovat s těmito počátečními rozměry 300x150 pixelů. Poměr stran plátna je také nastaven na 2:1 (šířka:výška).[7]

Chceme-li však zajistit konzistentní a požadované rozměry plátna, obecně se doporučuje explicitně nastavit atributy šířky a výšky nebo použít pravidla CSS pro prvek canvas. To umožní definovat velikost plátna podle vašich konkrétních požadavků a hledisek návrhu.[7]

```
1 <canvas id="renderCanvas" width="800" height="600"></
    canvas>
```

Příklad 9: Explicitní nastavení rozměrů skrze atributy elementu

Tento přístup nastavuje velikost plátna skrze atributy 'width' a 'height explicitně pomocí hodnot pixelů na 800 a 600. Případně lze použít i jiných měrných jednotek podporovaných atributy HTML.[7]

```
1 #renderCanvasId { width: 100%; height: 100%; }
```

Příklad 10: Explicitní nastavení rozměrů skrze styly elementu

Pomocí CSS lze nastavit šířku a výšku prvku <canvas> a ovládat jeho velikost pomocí pravidel CSS. Tento přístup poskytuje flexibilitu, protože se používají relativní jednotky, jako jsou procenta, em nebo rem, což může pomoci při responzivním návrhu. Je možné také použít další vlastnosti CSS, jako je max-width, max-height a margin, a dále tak řídit umístění a rozložení plátna v jeho nadřazeném kontejneru.[7]

8.1.2 Získání instance kanvasu v rámci JavaScriptu

Pro získání instance plátna v jazyce JavaScript je možné použít různé metody v závislosti na konkrétním případu použití a struktuře kódu. K získání prvku canvas na základě jeho ID lze použít metodu document.getElementById(). Tato metoda umožňuje přístup k libovolnému prvku v DOM podle jeho jedinečného atributu id. [11]

```
1 var canvas = document.getElementById('renderCanvasId');
```

Příklad 11: Uložení instance HTML <canvas> elementu do proměnné canvas pomocí metody document.getElementById()

Lze také použít metodu document.querySelector(), která poskytuje flexibilnější způsob výběru prvků a to pomocí selektorů CSS. Na prvek canvas se můžete odkázat pomocí jeho id, třídy nebo jiných atributů.[11]

```
1 var canvas = document.querySelector('#renderCanvasId');
```

Příklad 12: Uložení instance HTML <canvas> elementu do proměnné canvas pomocí metody document.querySelector()

V tomto případě metoda querySelector() vybere kanvas s ID "renderCanvasId"pomocí id selektoru CSS "#renderCanvasId".

Z důvodu optimalizace lze celé získávání kontextu plátna obalit do listeneru události DOMContentLoaded(), aby bylo zajištěno, že se kód Babylon.js spustí až po úplném načtení dokumentu HTML.[16]

8.2 Babylon.js engine, scéna a renderovací smyčka

Objekt Babylon.js Engine je klíčovou součástí frameworku. Slouží jako hlavní vykreslovací jádro odpovědné za správu nízkoúrovňových vykreslovacích operací a poskytuje infrastrukturu pro vykreslování 3D scén v reálném čase. Funguje jako most mezi kódem aplikace a základním grafickým hardwarem, pro jehož obsluhu využívá implicitně WebGL pro hardwarově akcelerované vykreslování.[16][18]

Spravuje celou vykreslovací pipeline, která zahrnuje zpracování a vykreslení scény na cílové plátno. Řeší úlohy, jako je vymazání plátna, odesílání volání kreslení, použití transformací a vykreslování materiálů a shaderů.[16]

Engine je přímo spojen s objektem Scene. Vykresluje scénu a aktualizuje její obsah na základě aktuálního stavu scény.[16]

Účelem objektu Babylon.js Scene je definovat virtuální 3D svět, ve kterém koexistují a vzájemně se ovlivňují objekty, světla, kamery a další prvky. Funguje jako kontejner pro všechny entity ve scéně a poskytuje strukturu pro definování jejich vztahů, vlastností a chování.[11][16][18]

Objekt Scene shromažďuje potřebné informace o kameře, světlech, sítích a materiálech ve scéně a komunikuje s přidruženým enginem Babylon.js, který provádí vlastní vykreslování na plátno. Řídí pořadí vykreslování, zpracovává viditelnost a vyřazování entit na základě jejich polohy a zorného pole kamery, aktualizuje všechny entity, které se v ní nacházejí, pro každý snímek procesu vykreslovací smyčky. To zahrnuje aktualizaci animací, kontrolu kolizí, zpracování uživatelských vstupů a další činnosti.[11][16][18]

Vykreslovací smyčka, ve vývoji her známá také jako herní smyčka, je v systému Babylon.js klíčovým konceptem, protože poskytuje mechanismus pro průběžnou aktualizaci a vykreslování 3D scény, který vytváří iluzi pohybu a interaktivity. Vykreslovací smyčka je v podstatě funkce, která je opakovaně volána, obvykle synchronizovaně s obnovovací frekvencí obrazovky (obvykle 60krát za sekundu), aby se zachovaly plynulé animace a interakce.[11][16][18]

8.2.1 Vytvoření objektů Engine a Scene a iniciace vykreslovací smyčky

Konstruktor objektu Engine přijímá dva parametry. Prvním z nich je instance kanvasu, na který má engine provádět render. Druhým je boolean hodnota, která určuje použití antialiasingu. Antialiasing je technika, která se používá k vyhlazení grafiky a snížení výskytu pixelů vyhlazováním hran. Obvykle se nastavuje na hodnotu true pro lepší vizuální kvalitu.[16]

Konstruktor Scene přijímá jeden parametr, a to instanci objektu Engine, ke kterému má tato scéna být přiřazena.[16]

V Babylon.js se vykreslovací smyčka vytváří prostřednictvím metody runRenderLoop() objektu Engine. Tato metoda přijímá jako argument funkci, která je volána při každém snímku. Uvnitř této funkce obvykle chceme zavolat metodu render() objektu Scene, která bude vykreslovat scénu po dobu jednoho snímku.[16]

```
1 var engine = new BABYLON.Engine(canvas, true);
2 var scena = new BABYLON.Scene(engine);
3
4 engine.runRenderLoop(function () {
5     scene.render();
6 });
```

Příklad 13: Ukázka kódu vytvoření objektů Engine a Scene a iniciace vykreslovací smyčky[16]

Tento kód však na obrazovce nic nezobrazí, jelikož se jedná pouze o základní nastavení kontextu vykreslování a scény.

Pro zobrazení čehokoli na obrazovce musíme do scény zahrnout a definovat další objekty. Mezi ně patří kamera (pro určení perspektivy, ze které se na scénu díváme), zdroje světla (pro osvětlení objektů ve scéně) a nějaký druh geometrie nebo objektu (aby bylo možné něco vykreslit).[11][18]

9 Kamera

V rámci Babylon.js se 3D scéna vytváří umístěním a manipulací s objekty v trojrozměrném prostoru. Protože je však obrazovka, na které se scéna zobrazuje, dvourozměrná, je zapotřebí mechanismus pro zachycení a prezentaci zploštělého zobrazení scény. Zde hrají zásadní roli kamery.[11][15]

Objekt kamery slouží jako oko, jehož prostřednictvím je scéna vizualizována a umožňuje tak uživatelům vnímat 3D prostředí z různých perspektiv.[15]

Po vytvoření je kamera automaticky zařazena do seznamu všech kamer scény. Scéna je vždy zobrazena z pohledu jediné aktivní kamery. Poloha a natočení kamery spolu s dalšími doplňujícími vlastnosti kamery se používají k výpočtu matice pohledu použité při vykreslování. Matice pohledu spolu s projekční maticí transformuje 3D scénu na 2D obraz, který lze zobrazit na obrazovce.[11][16][18]

Aby bylo možné kameru ovládat a umožnit uživatelům interagovat se scénou, je po vytvoření kamery třeba zavolat její funkci attachControl() s dvěma parametry, z nichž první je instance dané scény a druhá boolean hodnota udávající povolení ovládání kamery uživatelem. Při nastavení na hodnotu true, lze následně například kameru natáčet pomocí tažením myši nebo přejetím prstu na dotykovém zařízení, či přibližovat a oddalovat pomocí rolovacího kolečka myši, nebo v případě mobilních zařízení gesta štípnutí.[16][18]

```
1 camera.attachControl(canvas, true);
```

Příklad 14: Ukázka kódu povolení ovládání kamery[16]

Babylon.js nabízí různé varianty kamer, přičemž každá poskytuje unikátní způsob manipulace s polohou a orientací v 3D prostoru. Každý má také své specifické vlastnosti a metody, které umožňují další přizpůsobení. Pochopením těchto vlastností a manipulací s nimi lze vytvořit širokou škálu různorodých kamer, které budou vyhovovat specifickým potřebám projektu.[11][16]

9.1 FreeCamera

"FreeCamera představuje nejjednodušší typ kamery, poskytující svobodný pohyb a rotaci v kontextu 3D scény. Jejím účelem je poskytnout uživatelům možnost volně se pohybovat a zkoumat scénu.[11][16] Pohyb kamery v různých směrech se ovládá stisknutím šipek klávesnice. Pohyby myši jsou mezitím zodpovědné za otáčení pohledu kamery. Veškerý pohyb vyvolaný šipkami je relativizovaný v závislosti na aktuálním směrovém natočení kamery.[16]

9.2 UniversalCamera

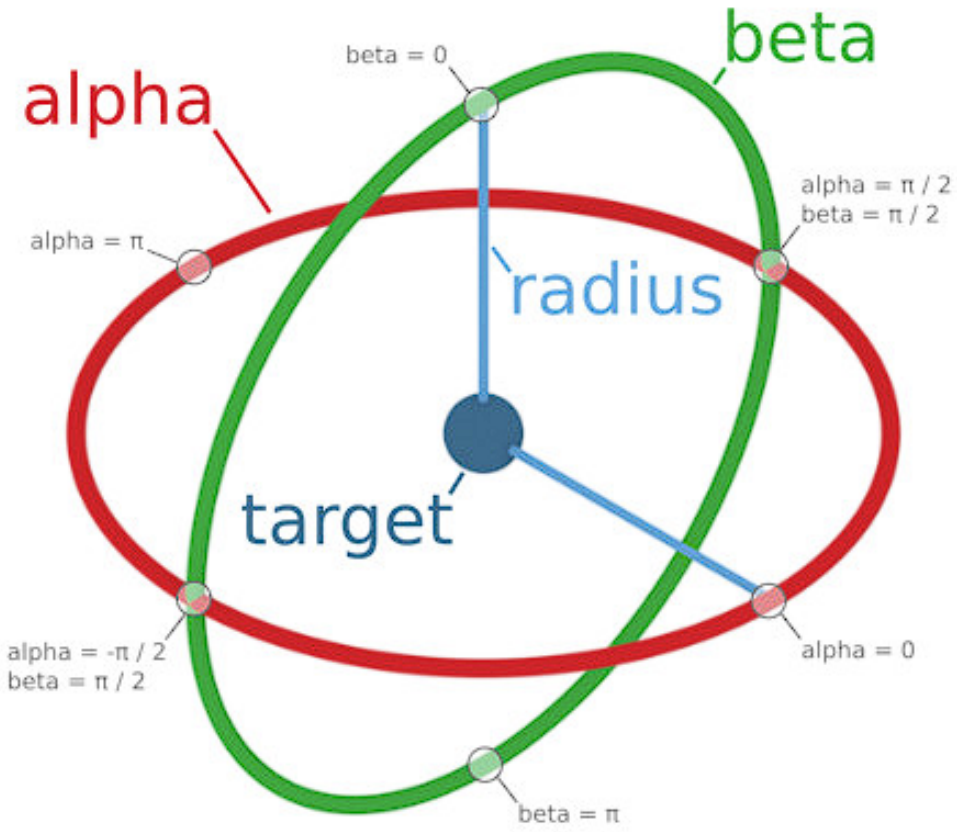
UniversalCamera představuje standardní typ kamery, který je v rámci frameworku implicitně používán. Kombinuje funkce kamer typu FreeCamera, TouchCamera a GamepadCamera a nabízí všestranné možnosti manipulace. Kamera UniversalCamera podporuje vstupy z klávesnice, myši, dotyku a gamepadu bez nutnosti explicitní specifikace ovladače. Její funkce se podobá mechanice ovládání ve hrách typu FPS (first-person shooter), což ji činí ideální volbou pro scény vyžadující tento druh navigace napříč zařízeními.[11][16]

9.3 ArcRotateCamera

ArcRotateCamera, jako efektivní varianta kamery, nabízí interaktivní manipulaci s polohou a natočením, a to pomocí horizontálních a vertikálních úhlů okolo definovaného bodu ve 3D scéně. Nabízí uživatelsky přívětivý způsob pohybu po scéně a jejího zkoumání díky možnosti plynulého otáčení a přibližování.[11][16]

Přináší několik základních vlastností, které sehrávají zásadní roli při ovládání chování kamery a jejího pohledu. Mezi tyto vlastnosti patří úhly alpha a beta, jejíž hodnoty typu number udávají hodnotu úhlů v radiánech horizontální a vertikální natočení kolem cílového bodu. Další vlastností je radius, který udává vzdálenost mezi kamerou a cílovým bodem. Úpravou poloměru můžete zvětšovat nebo zmenšovat zorné pole kamery a měnit úroveň detailů. Také se zde mění funkce vlastnosti target, který se při ovládání kamery nemění a defi-

nuje pevnou polohu cílového bodu ve scéně. Kolem tohoto bodu se kamera při rotaci otáčí a udržuje jej jako ohniskový bod.[11][16]



Obrázek 9: Diagram znázorňující vliv jednotlivých vlastností kamery na její orientaci v prostoru.[16]

Mimo uvedené základní parametry, ArcRotateCamera nabízí další nástroje pro sofistikovanější manipulaci. Například můžete nastavit limity pro vertikální rotaci a poloměr a omezit tak pohyb kamery v určitých mezích, což umožňuje přesnou kontrolu nad úhlem pohledu a úrovní přiblížení.[16]

9.4 FollowCamera

V Babylon.js je FollowCamera designována tak, aby sledovala a držela v zorném poli určitý cílový objekt během procházení 3D scénou. Jejím účelem je automa-

ticky upravovat svou polohu a orientaci tak, aby cílový objekt zůstal v pohledu. Běžně se používá v aplikacích nebo hrách s pohledem třetí osoby.[11][16]

9.5 Specializované kamery

Babylon.js dále nabízí specializované varianty kamer, mezi něž patří AnaglyphUniversalCamera pro vytváření anaglyfických efektů (3D brýle), StereoscopicUniversalCamera určenou pro stereoskopické zobrazení, nebo například VRDeviceOrientationFreeCamera vhodnou pro aplikace virtuální reality.[16]

10 Světla

Osvětlení hraje zásadní roli při vytváření realistických a vizuálně přitažlivých scén. Je to klíčový prvek, který dodává 3D objektům a prostředím hloubku a rozměr, čímž zvyšuje celkovou estetickou úroveň a věrohodnost scény.[11][15]

Zdroje světla v 3D renderech mají podobný základní účel jako ve skutečném světě - osvětlují scénu a objekty v ní. Bez světla by byla scéna zcela tmavá a objekty by vypadaly ploše, bez hloubky a rozměru. Navíc vytvářejí, při interakci s různými materiály, rozmanité efekty, jež odrážejí skutečný svět.[11]

Relativní poloha 3D objektů ve scéně nemá vliv na jejich implicitní osvětlení. To znamená, že pokud výslovně nenastavíte tvorbu stínů, bude světlo "procházet"objekty, jako by byly průhledné. Jinými slovy, objekt nebude bránit světlu v přístupu k jiným objektům.[16]

V Babylon.js je výchozí limit čtyř světel ovlivňujících jeden objekt. Důvodem je optimalizace výkonu, jelikož proces generování světelných efektů může být náročný na zdroje, zejména u složitých scén s mnoha světly a objekty. Tento limit je možné zvýšit, pokud scéna vyžaduje, aby na jednotlivý objekt působilo více světel. Toho je možné dosáhnout zvýšením hodnoty vlastnosti maxSimultaneousLights u jednotlivých materiálů objektů.[16]

10.1 Typy světelných objektů

3D enginy obvykle poskytují několik typů světelných objektů, které napodobují chování skutečných světelných zdrojů. Každý typ světla má své jedinečné vlastnosti a chování, díky čemuž je vhodný pro specifické použití.[11]

10.1.1 HemisphericLight

Hemisférické světlo typicky napodobuje plošné zdroje světla, jako je obloha, odkud světlo vyzařuje ze všech směrů shora. Má dvě hlavní vlastnosti. Direction (směrový vektor), který vede z počátku scény (0,0,0) do polohy světla a udává směr plošného světla a groundColor (podkladovou barvu) rozptýleného

světla, která udává barvu přímo neosvětlené části objektů a scény. Toto světlo je poměrně nenáročné a nepočítá se stíny ani s tvarem objektů. Nejlépe se používá v případech, kdy chcete mít jednoduché nastavení světla a nepotřebujete vytvářet složité světelné interakce.[11][15][16]

10.1.2 DirectionalLight

Směrové světlo slouží k simulaci vzdálených zdrojů světla, jako je například Slunce. Světelné paprsky vytvářené tímto zdrojem světla jsou rovnoběžné a mají stejnou intenzitu bez ohledu na vzdálenost od zdroje světla. Je to proto, že se předpokládá, že světlo je nekonečně daleko, a proto se ve frameworku DirectionalLight světlo definuje směrem, nikoli jeho polohou. Světelné paprsky se nerozptylují tak jako je tomu u hemisferického světla, a proto se neosvícená část objektu jeví ztmaveně. Rozdílem je také to, že už umí pracovat a řídit, zda a jak mají tímto světlem osvětlené objekty vrhat stíny.[11][15][16]

10.1.3 PointLight

Bodové světlo je zdroj světla, který vyzařuje rovnoměrně do všech směrů. Je definováno svou polohou ve 3D prostoru. Intenzita světla se zmenšuje se vzdáleností podle inverzního kvadratického vztahu. Tento typ světla lze použít k simulaci světelných zdrojů, jako jsou žárovky nebo svíčky, které vyzařují světlo rovnoměrně ve všech směrech. V aplikaci Babylon.js můžete ovládat intenzitu a dosah bodového světla a také to, zda vrhá stíny.[11][15][16]

10.1.4 SpotLight

Spotlight je zdroj světla, který vyzařuje paprsky v určitém směru ve tvaru kužele. Intenzita světla se zmenšuje se vzdáleností od zdroje světla a také od středové osy kužele. Tento typ světla lze použít k simulaci například svítilen, světlometů automobilů nebo reflektorů. Světlo je definováno svou polohou, směrovým vektorem určujícím směr, kterým zdroj světla míří, úhlem kužele v

radiánech a rychlostí poklesu intenzity. [11][15][16]

10.2 Vlastnosti světel

V rámci Babylon.js má každý typ světla řadu vlastností, kterými lze manipulovat a vytvářet tak požadované světelné efekty.

10.2.1 Intensity

Tato vlastnost řídí jas světla. Čím vyšší je hodnota, tím je světlo jasnější. Tuto vlastnost lze použít k tomu, aby byl zdroj světla tlumenější nebo jasnější podle požadavků scény.[15][16]

10.2.2 Range

U vlastností PointLight a SpotLight určuje vlastnost range maximální vzdálenost, do které světlo dosáhne. Intenzita světla se lineárně snižuje od pozice zdroje světla, dokud nedosáhne definovaného rozsahu, kde je intenzita světla rovna 0.[15][16]

10.2.3 Diffuse

Tato vlastnost definuje, jak je základní barva objektu ovlivněna zdrojem světla. Když světlo dopadá na objekt, rozptyluje se do mnoha směrů. Vlastnost diffuse tento efekt simuluje, a to způsobuje, že objekt vypadá osvětlený ze směru zdroje světla.[15][16]

10.2.4 Specular

Nastavuje barvu spekulárního odlesku, který vzniká při dopadu světla na lesklé předměty. Spekulární odlesky jsou malé světlé skvrny, které se objevují na lesklých površích, když odrážejí světlo. Díky nim může objekt vypadat leskle nebo kovově. Změnou barvy zrcadlového zvýraznění lze vytvořit různé vizuální efekty. Například zlatý objekt může mít žluté spekulární odlesky.[15][16]

10.2.5 Position a direction

Vlastnost pozice u světel PointLight a SpotLight určuje umístění světla v 3D prostoru scény. Vlastnost direction pro DirectionalLight a SpotLight určuje vektor směru, kterým jsou vyzařovány světelné paprsky. Poloha a směr světla mohou výrazně ovlivnit stíny a světla ve scéně a vytvořit tak hloubku a rozměr.[15][16]

10.2.6 Angle a exponent

Tyto vlastnosti jsou specifické pro světlo SpotLight. Velikost světelného kužele, který bodové světlo vytváří, je definována vlastností angle. Tato vlastnost přímo ovlivňuje soustředěnost či šířku světelného paprsku. Konkrétně menší hodnota úhlu vede k vytvoření užšího a více soustředěného paprsku, zatímco s rostoucí hodnotou úhlu se paprsek stává širším. Vlastnost exponent následně určuje míru, jakou se intenzita světla snižuje od středu reflektoru směrem k jeho okrajům. Vyšší exponent má za následek zaostřenější světlo s ostřejším poklesem intenzity, zatímco nižší exponent má za následek měkčí světlo s pozvolnějším poklesem intenzity.[15][16]

10.3 Stíny

Stíny mohou scéně dodat značnou úroveň realističnosti. Stíny jsou implementovány pomocí třídy ShadowGenerator, ke kterému je přiřazen daný objekt světla a je zodpovědný za generování stínů vrhaných tímto zdrojem. Třída poskytuje metody pro správu různých aspektů stínů, jako je povolení nebo zakázání stínů, nastavení velikosti mapy stínů, definování tmavosti stínů a určení objektů, které vrhají nebo přijímají stíny.[11][16]

11 Polygonové sítě neboli meshes

Meshové struktury a s nimi spojená geometrie představují základní stavební kameny v oblasti 3D vykreslování a modelování. Slouží jako stavební kameny pro vytváření 3D objektů a scén.[5][15]

Polygonové sítě jsou soubor vrcholů, hran a ploch, které společně reprezentují tvar 3D objektu. Každý vrchol je bod v 3D prostoru a vrcholy jsou spojeny hranami. Když se spojí tři nebo více hran, vytvoří plochu. Například v případě krychle představuje každý roh vrchol, které spojují hrany krychle a její strany jsou plochami.[5][15]

Sítě jsou významné při 3D modelování a vykreslování, protože se používají k vytváření 3D objektů, které vyplňují scénu. Každý prvek ve 3D scéně, od postav přes různé modely objektů až po celé prostředí, je tvořen pomocí meshových struktur. Složitost sítě, často označovaná jako úroveň podrobnosti, je obvykle úměrná počtu jejích vrcholů a stěn. Síť s vyšším rozlišením obsahuje více vrcholů a ploch a může reprezentovat složitější tvary a obecně vede k podrobnějšímu zobrazení, ale také vyžaduje více výpočetních prostředků k vykreslení.[5][15]

Geometrie v kontextu 3D modelování označuje matematický popis tvaru objektu. Poskytují základní strukturu pro síť. Zatímco síť se zabývá reprezentací tvaru ve 3D prostoru, geometrie poskytuje parametry, které tento tvar definují. Například geometrie koule zahrnuje její poloměr, geometrie krychle zahrnuje její délku, šířku a výšku a geometrie válce zahrnuje jeho poloměr a výšku. Vzájemné působení mezi sítěmi a geometriemi je při 3D modelování klíčové. Společně umožňují vytvářet složité a rozmanité 3D modely.[5][15]

Babylon.js nabízí různé typy sítí, které lze rozdělit do dvou hlavních kategorií. Parametrické tvary jsou jednoduché 3D tvary, jako jsou koule, kvádry, válce atd., vytvořené pomocí vestavěných metod v objektu BABYLON.MeshBuilder. Naproti tomu modelové sítě jsou složitější tvary importované z externího 3D modelovacího softwaru.[11][16]

11.1 Tvorba primitivních struktur

V kontextu systému Babylon.js je k dispozici řada základních typů sítí, například Box, Sphere, Cylinder a Plane. Pro tvorbu těchto primitivních struktur, jako jsou koule, kvádry, válce a jiné, jsou v rámci Babylonu vestavěné metody objektu BABYLON.MeshBuilder. Syntaxe těchto metod lze zobecnit tímto zápisem:[16]

```
1 BABYLON.MeshBuilder.Create<Objekt>(nazev: string,
    parametry: {} , scena: BABYLON.Scene);
```

Příklad 15: Zobecněný zápis tvorby primitivních objektů[16]

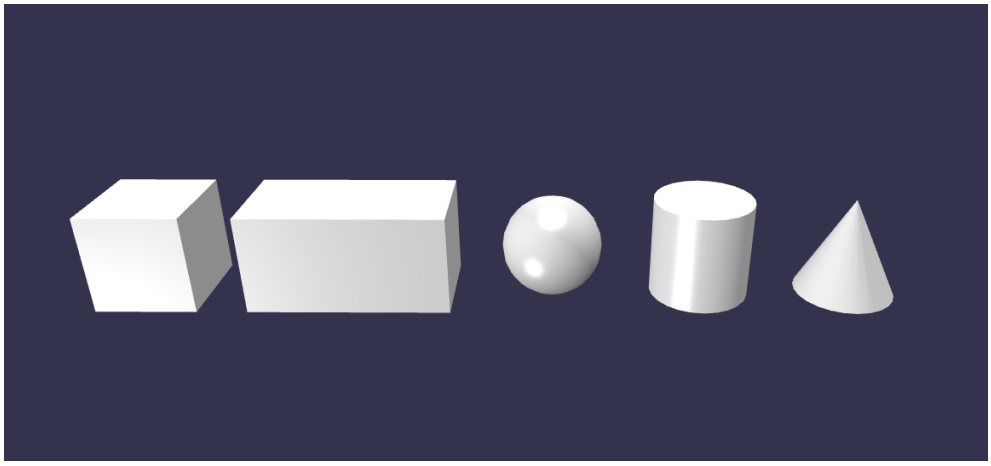
V tomto zápise je zapotřebí nahradit <Objekt> požadovaným typem primitivní struktury, a za parametry námi zvoleným název, přiřazenou scénu, a v neposlední řadě objekt s parametry, který má každý základních typů sítí jiný. Tyto sítě lze upravovat a kombinovat a vytvářet tak složitější tvary a objekty.

```
1 var krychle = BABYLON.MeshBuilder.CreateBox('krychle', {
    size: 1}, scene);
2 var kvadr = BABYLON.MeshBuilder.CreateBox('kvadr', {height
    : 1, width: 2, depth:1}, scene);
3 var koule = BABYLON.MeshBuilder.CreateSphere('koule', {
    diameter:1, segments: 10}, scene);
4 var valec = BABYLON.MeshBuilder.CreateCylinder('valec', {
    height: 1, diameter: 1,}, scene);
5 var kuzel = BABYLON.MeshBuilder.CreateCylinder("kuzel", {
    height: 1 , diameterBottom: 1, diameterTop:0}, scene);
```

Příklad 16: Vytvoření krychle, kvádru, koule, válce a kuželu

Například v rámci parametrů sítě Box lze definovat velikost (size), čímž vznikne box o stejné velikosti stran, tedy krychle. Dále můžeme v rámci objektu s parametry definovat různé hodnoty pro šířku (width), výšku (height) a hloubku (depth) zvlášť, a tím snadno vytvářet různé kvádrovité tvary. I u sítě

Cylinder můžeme definovat obecně jeho průměr pro obě jeho základny pomocí parametru diameter, nebo lze každé základně přiřadit jinou hodnotu pomocí diameterBottom a diameterTop, a tím lze vytvořit model kužele, pokud jeden z nich nastavíme na hodnotu nula. Díky těmto parametrům, úpravám a kombinací lze vytvářet i složitější tvary a objekty.[16]



Obrázek 10: Ukázka objektů z příkladu 15

Mimo tyto trojrozměrná tělesa můžeme stejným způsobem vytvořit i samostatné plochy (Plane) a podkladové plochy (Ground). Ty lze obě definovat svojí výškou a šířkou, ale liší se od sebe jejich základní orientací v prostoru. Objekt Plane je paralelní s osami X a Z, zatímco Ground je paralelně s osami X a Z. Díky tomu hodnota výšky u podkladové plochy určuje velikost na ose Z, i když zdánlivě nastavuje hloubku.[16]

11.2 Další možnosti tvorby sítí

Mimo vytváření primitivních struktur umožňuje Babylon také tvorbu komplexnějších parametrických sítí. Ty jsou definovány polem parametrů typu Vector3. To dovoluje, že jejich tvar může být nepravidelný. Tyto typy sítí umožňují vytvářet lomené nebo nelomené čáry napříč 3D prostorem a také 2D polygony, které lze extrudovat do 3D. Mohou také vytlačovat 2D obrys podél 3D cesty

s proměnným natočením a měřítkem podél dané cesty nebo obrys obtáčet a vytvářet tak tvar s rotační symetrií.[11][16]

Důležité je také poznamenat, že Babylon.js má funkci pro vytváření vlastních sítí, pokud vestavěné primitivy nevyhovují potřebám modelu. Síť lze vytvořit zadáním jejích vrcholů a indexů. Vrcholy definují tvar a polohu sítě a indexy určují, jak se vrcholy spojují, aby tvořily stěny.[16]

Kromě toho Babylon.js podporuje operace se sítěmi, jako je slučování a rozdělení. Sloučení geometrií více objektů do jedné může optimalizovat scénu snížením počtu volání kreslení. Naproti tomu rozdělení umožňuje rozdělit síť na dvě části.[16]

11.3 Import externích modelů

Framework slouží jakožto 3D vykreslovací engine Neklade si za úkol obstarat veškeré funkce plnohodnotného 3D modelovacího softwaru jako jsou kupříkladu programy Blender, Maja, Cinema4D či CAD programy pro počítačovou podporu konstrukce jako jsou AutoCAD nebo SolidWorks. Proto se lze domnívat, že při vývoji projektu může dojít k potřebě v rámci scény použít modely vzniklé z těchto externích nástrojů. Naštěstí import externích zdrojů do Babylon.js je díky robustním možnostem enginu pro import poměrně jednoduchý proces.[11]

Model lze importovat pomocí třídy SceneLoader a funkce Append(), která přebírá řetězec adresy URL souboru, řetězec názvu souboru a funkci zpětného volání, která se provede po úspěšném načtení modelu.[16]

```
1 BABYLON.SceneLoader.Append("url/cesta/souboru/", "
     nazevSouboru.babylon", scene, function (
     importovanaScena) {
2     //logika callback funkce
3 });
```

Příklad 17: Ukázka metody append pro import externího modelu[16]

Framework Babylon.js podporuje širokou škálu formátů 3D souborů, včetně, ale nejen, formátů .babylon, .gltf, .glb (binární forma GLTF), .obj a .stl. Výběr formátu souboru závisí do značné míry na potřebách projektu a nástrojích používaných k vytváření 3D modelů.[16]

11.3.1 Výhody

Možnost importovat externí sítě nebo modely přináší několik klíčových výhod. Předně umožňuje vývojářům vytvářet složité 3D scény, aniž by museli ručně kódovat každý objekt ve scéně. To může výrazně zkrátit dobu vývoje a umožňuje vytvořit komplexnější a vizuálně přitažlivější konečný produkt.[11]

Dále díky možnosti importu modelů z různých 3D modelovacích softwarů využívá Babylon.js rozsáhlé knihovny 3D modelů, které jsou k dispozici online, ať už zdarma, nebo k zakoupení. To znamená, že vývojáři mohou využít existující prostředky, místo aby vše vytvářeli od začátku.[11]

A konečně podpora různých formátů souborů znamená, že Babylon.js lze používat ve spojení s širokou škálou nástrojů pro 3D modelování, což z něj činí flexibilní volbu pro mnoho projektů.[1][11]

11.3.2 Problémy a řešení

Jedním z nejčastějších problémů je zvládnutí rozdílů v měřítku, rotaci nebo souřadnicových systémech, které jsou používány v různých softwares pro 3D modelování. To může způsobit, že se importovaný model ve scéně Babylon.js zobrazí nesprávně. Většinu těchto problémů lze vyřešit úpravou transformací modelu po importu nebo úpravou nastavení exportu v softwaru pro 3D modelování.[11]

Dalším potenciálním problémem je práce se složitými materiály nebo shadery použitými v 3D modelu. Ne všechny funkce všech formátů jsou v systému Babylon.js podporovány a některé materiály mohou při importu vypadat jinak. V takovém případě je nutné, aby vývojáři ručně upravili materiály v aplikaci Babylon.js nebo zjednodušili materiály před exportem modelu.[11]

12 Transformace objektů

Každý objekt má řadu vlastností a transformací, se kterými lze manipulovat. Lze jimi například přesouvat, otáčet a měnit jejich měřítko pomocí vlastností position (poloha), rotation (otáčení) a scaling (měřítko).[16]

12.1 Posuv

Posuv či translace je operace, při níž se objekt posune podél os X, Y a Z. Poloha objektu (position) je reprezentována vektorem, který označuje jeho souřadnice v 3D prostoru. Tato vlastnost určuje umístění objektu v prostoru a změnou této vlastnosti se síť posune. Alternativně lze použít k posunutí sítě podél určité osy o určitou hodnotu metodu translate() s parametrem směrového vektoru udávající směr posuvu a hodnotou vzdálenosti.[15][16]

```
1 mesh.position = new Babylon.Vector(10, 0, 0);
2 mesh.translate(new BABYLON.Vector3(1, 0, 0), 10);
```

Příklad 18: Příklad posuvu objektu oběma způsoby o vzdálenost 10 po ose X

12.2 Rotace

Rotací dochází k otočení objektů kolem os X, Y a Z. Vlastnost rotace (rotation) je definována vektorem představujícím natočení v radiánech. Změnou této vlastnosti se síť otočí. Alternativně lze metodu rotate() použít k otočení sítě kolem určité osy o určitý úhel, podobným způsobem jako v metodě translate().[15][16]

```
1 mesh.rotation = new BABYLON.Vector3(0, Math.PI / 2, 0);
2 mesh.rotate(new BABYLON.Vector3(1, 0, 0), Math.PI / 2);
```

Příklad 19: Příklad rotace objektu oběma způsoby o hodnotu $1/2\pi$ (90°) po ose Y

12.3 Škálování

Objekt lze škálovat pomocí vlastnosti scaling, což je objekt Vector3, který představuje faktor škálování podél os X, Y a Z. Změnou této vlastnosti se objekt prodlužuje v daném směru.[15][16]

```
1 mesh.scaling = new BABYLON.Vector3(1, 2, 1);
```

Příklad 20: Příklad škálování objektu na dvojnásobnou velikost podél osy Z

12.4 Hierarchické vztahy objektů

Objekty lze uspořádat do dědičné hierarchie rodič-dítě přiřazením rodičovského objektu do vlastnosti parent potomka. Nadřazený objekt může mít přiřazeny jeden nebo více objektů. Tento vzájemný vztah má zásadní vliv na způsob, jakým se aplikují transformace.[16]

Hierarchické pořadí rodičovských a podřízených objektů silně ovlivňuje transformace, protože transformace podřízeného objektu je vždy relativní vůči jeho rodiči. Pokud tedy dochází k otáčení, škálování nebo přesunutí rodiče, transformují se i všechny jeho potomci.[11][16]

12.4.1 Lokální vs. globální prostor

Každý objekt (neboli "uzel") v Babylon.js má svůj vlastní lokální souřadnicový prostor, který je definován vzhledem k jeho rodiči. Pokud objekt nemá rodiče, je jeho lokální prostor stejný jako globální prostor.[5][15]

Když na objekt aplikujeme transformaci, provede se v lokálním prostoru objektu. To znamená, že pokud objekt posuneme podél lokální osy Y, bude se pohybovat ve směru svého rodiče, což nemusí být stejné jako při posunu po globální ose Y, pokud byl rodič otočen.[15]

12.5 Klonování a instancování

Klonování i instancování v Babylon.js slouží k vytvoření kopie existující sítě. V některých aspektech se však liší, například v tom, jak využívají paměť a jak se chovají při transformaci.[5]

Klonování a instancování jsou důležité funkce pro optimalizaci výkonu 3D scény. Obě vytvářejí novou síť, která sdílí geometrická data původní sítě, ale s materiálovými daty pracují odlišně. Klonování vytváří novou kopii materiálu, což umožňuje měnit vzhled klonu, aniž by to ovlivnilo původní síť. Instance však sdílí materiál s původní sítí, což je rychlejší a spotřebovává méně paměti, ale neumožňuje měnit vzhled instance bez ovlivnění původní sítě.[11][16]

12.5.1 Klon sítě

Klonování sítě vytvoří zcela novou kopii této sítě se všemi jejími vlastnostmi a se stejnou geometrií, materiálem a transformacemi (poloha, rotace a měřítko) jako původní síť. Jakékoli změny provedené v geometrii, materiálu nebo transformacích klonu budou provedeny nezávisle na původní síti a tu nijak neovlivní.[5][15][16]

```
1 var original = BABYLON.MeshBuilder.CreateBox('original',
    {}, scene);
2 var clone = original.clone('klonOriginalu');
```

Příklad 21: Příklad vytvoření klonu meshe[16]

12.5.2 Instance sítě

Při vytváření instance sítě vytváříme nový objekt, který sdílí geometrii a materiály původní sítě. To znamená, že všechny instance sítě používají stejnou sadu vrcholů, indexů, normálů a její vzhled nelze měnit. Každá instance má však vlastní sadu transformačních parametrů (poloha, rotace a měřítko). Proto, i když mají společnou geometrii, mohou se ve scéně vyskytovat na různých

místech nebo v různých orientacích. V důsledku toho je instancování méně náročné na paměť než klonování, zejména u velkého počtu objektů, jelikož GPU může vykreslit instance v jediném volání renderu pro všechny takové objekty najednou.[11][15][16]

```
1 var original = BABYLON.MeshBuilder.CreateBox('original',
     {}, scene);
2 var instance = riginalMesh.createInstance("
     instanceOriginalu");
```

Příklad 22: Příklad vytvoření instance meshe[16]

13 Materiály a textury

Materiály a textury hrají klíčovou roli při zvyšování vizuální přitažlivosti a realističnosti 3D scény v Babylon.js. Oživují geometrické tvary tím, že jim dodávají barvu, vzor a reakci na světlo, což přispívá k věrohodnosti a pohlcující kvalitě virtuálního prostředí.

13.1 Materiály

Materiály v Babylon.js definují, jak povrch objektu reaguje na světlo, a tím ur- čují jeho vzhled. Každý materiál je charakterizován vlastnostmi, jako je barva, průhlednost, lesklost, mapy textur a další.[11]

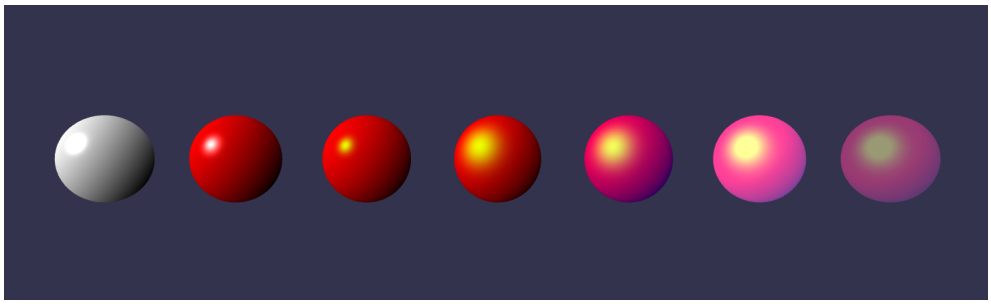
13.1.1 StandardMaterial

Tento základní a nejpoužívanější materiál v Babylon.js nabízí širokou škálu vlastností. Podporuje difúzní barvu určující základní barvu objektu, spekulární barvu odlesků a emisní barvu, kterou objekt vyzařuje jako světlo. Kromě toho podporuje ambientní barvu z okolního světla, spekulární sílu určující, jak moc lesklý objekt je, a hodnotu alfa určující průhlednost materiálu. Dokáže pracovat s různými typy textur a nejlépe se hodí pro základní 3D modely, kde není vyžadován extrémní realismus a prioritou je výkon.[11][16]

```
1 var koule = BABYLON.MeshBuilder.CreateSphere("koule", {},
     scene);
2
3 var material = new BABYLON.StandardMaterial('material',
     scene);
4
5 material.diffuseColor = new BABYLON.Color3(1, 0, 0);
6 material.specularColor = new BABYLON.Color3(0, 1, 0);
7 material.specularPower = 10;
8 material.ambientColor = new BABYLON.Color3(0,0,0.5);
```

```
9  material.emissiveColor = new BABYLON.Color3
       (0.25,0.25,0.25);
10 material.alpha = 0.5;
11
12 koule.material = material;
```

Příklad 23: Ukázka kódu pro vytvoření, nastavení jednotlivých vlastností a přiřazení materiálu objektu koule



Obrázek 11: Ukázka efektu postupně se nabalujících vlastností standardního materiálu z příkladu 23

13.1.2 PBRMaterial

Materiály založené na fyzikálním vykreslování (Physically based rendering material) nabízí realističtější vykreslování ve srovnání s materiály StandardMaterials a jsou navrženy tak, aby napodobovaly fyzikální interakci mezi světlem a materiály v reálném světě. Používá model osvětlení, který přesněji a předvídatelněji reprezentuje interakci světla s povrchy tak, jako je tomu ve skutečném světě.[11][15]

Podporují pokročilé vlastnosti, jako je například kovovost (metallic), jež s hodnotami v rozmezí od nuly do jedné určuje, zda se objekt jeví jako vyrobený z plastu nebo kovu. Další důležitou vlastností je drsnost (roughness), která stejným způsobem ovlivňuje matnost či lesklost materiálu. Krom toho také obsahují i složitější vstupy textur, jako například mapy detailů (detailMap),

pomocí které lze přidat dodatečné detaily povrchu, aniž by bylo zapotřebí sítě s vysokým počtem polygonů. Materiál PBRMaterial je náročnější na výkon, ale jeho výsledkem je vyšší vizuální věrohodnost, takže je ideální pro aplikace, kde je klíčový vizuální realismus.[15]

```
1 var koule = BABYLON.MeshBuilder.CreateSphere("koule",{
     segments: 128,diameter: 1,}, scene);
2
3 var pbrMaterial = new BABYLON.PBRMaterial("pbrMaterial",
     scene);
4
5 pbr0.metallic = 1;
6 pbr0.roughness = 1;
7
8 koule.material = pbrMaterial;
```

Příklad 24: Ukázka kódu pro vytvoření, nastavení vlastností kovovosti a hrubosti a přiřazení PBRMateriálu objektu koule

Obrázek 12: Ukázka efektů kovovosti:hrubosti s hodnotami 1:1, 1:0, 0:0 a 0:1

13.1.3 ShaderMaterial

ShaderMaterial je typ materiálu, který se používá pro případy, kdy je potřeba dosáhnout maximální kontroly nad vykreslováním sítě. Tento materiál umožňuje definovat vlastní vertexové a fragmentové shadery, což dává plnou kontrolu nad zpracováním vrcholů sítě a vykreslováním pixelů.[15][16][20]

ShaderMaterial je schopen vytvářet širokou škálu unikátních vizuálních efektů, které by s použitím ostatních typů materiálů nebyly dosažitelné. Tato pokročilá funkce vyžaduje dobré znalosti shaderovacího jazyka GLSL a základních principů 3D vykreslování.[15][20]

Přestože ShaderMaterial poskytuje maximální možnou kontrolu, jeho použití přináší složitost a vyžaduje pokročilé znalosti programování shaderů.[15]

13.1.4 MultiMaterial

MultiMaterial je speciální typ materiálu v Babylon.js, který umožňuje použití různých materiálů na různých částech jedné sítě. Tento materiál není samostatným materiálem, ale spíše kolekcí jiných typů materiálů.[16]

Tento materiál funguje jako pole materiálů, které jsou přiřazeny k specifickým ID tváří sítě. To umožňuje komplexní a variabilní vykreslení jednotlivých částí sítě s použitím různých materiálů.[16]

13.1.5 BackgroundMaterial

Tento materiál je speciálně koncipován pro použití na pozadí, což umožňuje efektivní a rychlé vytváření dynamických pozadí scén, jako jsou skyboxy. Tento materiál je odvozen od PBRMaterial a zachovává mnoho z jeho realistických vlastností, zároveň je však optimalizován pro zobrazení vzdálených objektů a prostředí. Navzdory své specializaci na pozadí, BackgroundMaterial udržuje kompatibilitu s řadou textur a efektů, což zvyšuje jeho flexibilitu a uplatnění v různých scénářích. Jeho účinné řešení pro vykreslování pozadí také přispívá k celkové optimalizaci výkonu scény v Babylon.js.[11][16]

13.1.6 Speciální materiály

Babylon.js disponuje řadou specializovaných materiálů určených pro vytvoření specifických efektů nebo pro zjednodušení určitých úloh. Každý z těchto specializovaných materiálů má jedinečné vlastnosti a je navržen pro konkrétní apli-

kace. FireMaterial například umožňuje simulovat živý oheň, zatímco FurMaterial vytváří efekt srsti. TerrainMaterial je zase optimalizován pro vykreslování terénu. Tyto specializované materiály přispívají k různorodosti a vizuálnímu bohatství scén vytvořených v Babylon.js, a umožňují vývojářům přizpůsobit a oživit své 3D modely s větší přesností a detaily.[16]

13.2 Textury

Textura v Babylon.js představuje obrázek, který je aplikován na povrch 3D modelu nebo sítě, čímž mu dodává barvu a detaily. Textury mají zásadní význam pro dosažení vizuální realističnosti, neboť poskytují detailní reprezentaci povrchu materiálu. V kombinaci s materiály umožňují vytvářet rozmanité vizuální efekty.[11][19]

V Babylon.js se využívají bitmapové textury, které lze aplikovat na materiál v různých formách, přičemž každý typ textury ovlivňuje jinou část vzhledu materiálu. Existuje celá řada typů textur, které odpovídají různým vlastnostem materiálu.[11]

Každý typ textury má specifický účel a případ použití. Některé textury nemusí být pro určitý materiál zapotřebí, zatímco jiné mohou být kombinovány s dalšími typy textur k dosažení konkrétního vizuálního efektu. Efektivní použití textur je klíčové pro vytváření vizuálně atraktivních a realistických 3D scén v Babylon.js.[11][15]

Textury jsou aplikovány na povrchy sítě pomocí procesu známého jako mapování textur. Tento proces je řízen pomocí UV souřadnic, které efektivně mapují každý vrchol sítě na určitý bod na 2D textuře. Babylon.js nabízí pokročilé možnosti transformace UV, které umožňují detailnější a sofistikovanější manipulaci s texturami na povrchu sítě.[15][21]

Mají schopnost simulovat komplexní vlastnosti povrchu, jako je drsnost, lesk, hrbolatost, a dokonce i světelné vyzařování. Výsledkem je široké spektrum vizuálních efektů a stylů, od fotorealistické až po stylizovanou grafiku.[15]

13.2.1 DiffuseTexture

Difuzní textury či mapy představují základní barvu nebo vzor materiálu. V kontextu 3D grafiky definují barvu a vzor, který bude materiál na modelu zobrazovat. Tyto textury se kombinují s difuzní barvou materiálu, přičemž barevné informace z mapy se obvykle mísí s barvou modelu. Difuzní mapa je rovnoměrně nanesena na celý objekt a přímo ovlivňuje jeho základní barevné podání. Výsledkem je pocit celistvosti a vizuální soudržnosti na povrchu objektu.[11][15]

13.2.2 SpecularTexture

Spekulární textury či mapy slouží k ovládání lesku a barvy zvýraznění materiálu. Určují, které části modelu jsou lesklejší než jiné. Ve většinou černobílé spekulární mapě světlejší hodnoty značí oblasti s větším leskem a odrazivostí, zatímco tmavší oblasti jsou méně odrazivé. To umožňuje jemné ovládání interakce světla na povrchu objektu, vedoucí k realističtějšímu zobrazení materiálů, jako je kov nebo plast.[11][15]

13.2.3 BumpTexture

Bump textury nebo také normálové mapy se používají k vytvoření iluze hloubky na jinak hladkých površích. Změnou normál povrchu objektu a využitím výpočtu osvětlení vytvářejí efekt hrbolatosti, jakoby měl povrch nerovnosti nebo vrásy. Bump mapy mohou zvýšit vizuální komplexitu a realističnost objektu bez nutnosti zvyšování jeho geometrické složitosti.[11][15]

13.2.4 EmissionTexture

Emisní textura je typ textury určující, které části modelu se budou zdát, že vyzařují světlo. Tato textura je užitečná pro realistické texturování objektů, které emitují světlo, jako jsou obrazovky, světla nebo bioluminiscenční tvorové.[11][15]

13.2.5 AmbientTexture

Okolní textura ovlivňuje barvu a osvětlení 3D objektu v oblastech, které nejsou přímo osvětleny zdrojem světla. Tato textura pomáhá vytvářet dojem hloubky a realističnosti ve scéně tím, že přidává přirozeně vypadající stíny a odstíny tam, kde je omezené nebo žádné přímé světlo. Je to důležitý prvek pro vytvoření uvěřitelného osvětlení ve 3D scéně.[11][15]

13.2.6 OpacityTexture

Tato textura slouží k ovládání průhlednosti různých částí sítě modelu. V textuře průhlednosti černé oblasti indikují úplnou průhlednost (tedy oblasti, které jsou průhledné), zatímco bílé oblasti jsou zcela neprůhledné. Tato textura umožňuje vytváření složitých vizuálních efektů, jako jsou poloprůhledné povrchy, sklo nebo průzračné materiály.[11][15]

13.2.7 Ostatní textury

Mimo základní textury se můžeme setkat i s texturami méně používanými. Mezi ně lze zařadit textury odrážející okolní prostředí, například reflectionTexture, která simuluje vzhled kovových nebo zrcadlových povrchů. Další pak ambientTexture, která obvykle obsahuje explicitní panoramatické nebo kubické mapy okolí, které se mají odrážet. Za zmínku také patří například refractionTexture, která je používána k simulaci efektu refrakce, kdy světlo prochází průsvitným objektem a je přitom ohýbáno. Toto ohýbaní světla způsobuje zkreslení objektů za průhledným materiálem, vytvářející tak realistický efekt například u skla nebo vody.[11][15][16]

```
1 const material = new BABYLON.StandardMaterial("material",
     scene);
2
3 material.diffuseTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
4 material.specularTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
5 material.bumpTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
6 material.emissiveTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
7 material.ambientTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
8 material.opacityTexture = new BABYLON.Texture("cesta/k/
     souboru/textury", scene);
```

Příklad 25: Ukázka kódu pro přiřazení obrázku textury k jednotlivým typům textur

14 Pokročilé funkce frameworku

14.1 Animace

Součástí 3D enginu Babylonu je také vytváření, správa a řízení animací. Poskytuje robustní podporu pro animace s klíčovými snímky (keyframes), které definují stav objektu v konkrétních časových bodech. Animační systém následně interpoluje mezi těmito klíčovými snímky, čímž vytváří plynulé přechody a oživuje objekty.[11][16]

Téměř každý atribut objektu, jako je pozice, rotace, měřítko, barva, ale i další, je možné animovat. Třída Animation v Babylon.js drží data klíčových snímků a také definuje chování interpolace jako je lineární, bezierova nebo hermitova spline interpolace.[16]

Další neoddělitelnou součástí animačního systému v Babylon.js jsou takzvané easing funkce. Tyto matematické funkce lze použít k vytvoření složitějších a realističtějších animací definováním vzorců akcelerace. Babylon.js podporuje řadu standardních funkcí easingu jako lineární, kruhová, elastická, odskočná a další, stejně jako schopnost definovat vlastní.[15][16]

Podstatnou funkcí je také možnost skeletální (kosterní) animace. Kostra v Babylon.js je vytvořena jako hierarchická struktura kostí, kde každá kost představuje část modelu, která se může pohybovat nezávisle. To je zásadní pro animace postav, kde je požadován přirozený, organický pohyb.[16]

Framework také podporuje export a import animací spolu s 3D modely. To usnadňuje vytváření animací v dedikovaném 3D modelovacím softwaru a následný import do Babylon.js pro vykreslení.[16]

14.2 Uživatelské vstupy

Babylon.js poskytuje rozsáhlé API pro zpracování uživatelských vstupů, což umožňuje širokou škálu interaktivity v 3D aplikacích. To zahrnuje podporu pro tradiční vstupy myši a klávesnice, dotykové události pro mobilní zařízení,

a dokonce specializovaný vstup pro ovladače virtuální reality, včetně sledování pozice a rotace ovladače, detekci stisknutí tlačítek a manipulaci s hand trackingem.[11][16]

Tento engine zjednodušuje proces práce s uživatelskými vstupy prostřednictvím event-driven metod. Jedná se v podstatě o funkce, které se provedou vždy, když dojde k určité vstupní události. Lze tak definovat konkrétní akci, která by měla být provedena pro každý typ vstupní události.[16]

Na uživatelských vstupech často závisí například ovládání kamery. Babylon.js poskytuje různé typy kamer, každou z nich lze ovládat pomocí vstupů myši nebo dotyku. Vývojáři mohou tyto ovládací prvky přizpůsobit podle svých potřeb a dokonce vytvářet vlastní ovládací prvky kamery.[19]

Události myši a dotyku lze také použít k interakci s jednotlivými sítěmi ve scéně. Babylon.js poskytuje systém výběru, který usnadňuje určení, s jakým objektem uživatel interaguje. To je zásadní pro aplikace jako 3D hry, kde uživatelé mohou potřebovat kliknout na konkrétní postavy nebo objekty.[16]

14.3 Fyzikální engine

Fyzikální engine je robustní systém, který dokáže simulovat fyziku reálného světa a přidávat tak další vrstvu realismu do 3D scén. Dokáže zvládnout různé fyzikální jevy, jako je gravitace, kolize a síly, které jsou všechny kritické složky při vytváření poutavého a interaktivního 3D prostředí.[15]

Klíčové funkce poskytované fyzikálními enginy zahrnují detekci kolizí, která identifikuje, když se dva nebo více objektů na scéně dotkly. Dále například dynamiku tuhých těles pro simulaci objektů s pevným tvarem, jako je odskakující míč nebo padající bedna a dynamiku měkkých těles, pro simulaci objektů, které se mohou deformovat, jako je tkanina nebo želé.[5][15][16]

Babylon.js využívá pro dosažení svých fyzikálních simulací knihoven pluginů třetích stran. Zahrnuje podporu pro několik fyzikálních enginů, a to Cannon.js, Ammo.js a Oimo.js. Každý má své vlastní silné a slabé stránky,

takže vývojáři si mohou vybrat nejvhodnější na základě potřeb konkrétního projektu.[16][19]

Cannon.js je lehký, snadno použitelný engine, který poskytuje dobrý rozsah funkcí pro simulaci tuhých těles. Ammo.js je na druhé straně přímým portem výkonného fyzikálního enginu Bullet do JavaScriptu. Nabízí širší rozsah funkcí, včetně dynamiky měkkých těles, ale může být složitější na použití. Oimo.js je lehký a orientovaný na výkon, nejlépe se hodí pro scénáře, kde interaguje mnoho objektů.[16]

14.4 Systémy částic

Systémy částic jsou funkcí pro vytváření vizuálních efektů, jako je kouř, oheň, déšť, sníh a další atmosférické jevy, které by bylo nepraktické reprezentovat jednotlivými sítěmi s fyzickou geometrií, ale také solid částic, které polygonovou strukturu mají.[5][15]

Babylon.js podporuje robustní systém částic, který umožňuje vytváření, ovládání a vykreslování velkého množství malých částic pro generování těchto efektů. Tento systém poskytuje řadu vlastností k řízení, včetně životnosti, barvy, textur, velikosti a rychlosti částic.[15][16]

Systém částic se nejprve nastaví definováním zdroje částic (emmiter), který může být jediný bod, linka, box, nebo dokonce vlastní tvar. Tento zdroj funguje jako počátek částic, které budou generovat částice na základě přiřazených parametrů.[16]

Chování částic v průběhu času lze definovat a ovládat pomocí různých vlastností. Například můžete určit, jak rychle jsou částice vypouštěny, jejich životnost, jejich počáteční velikost, barvu a jak se tyto vlastnosti mění v průběhu času. Částicím lze také přiřadit počáteční směr a rychlost.[11][16]

14.5 Inspektor

Inspektor představuje komplexní a efektivní nástroj, který byl navržen za účelem maximalizace produktivity během vývoje v rámci frameworku Babylon.js. Tento nástroj je vybaven širokou škálou funkcí pro zkoumání scény, modifikaci atributů, profilaci výkonu a manipulaci s vykreslováním v reálném čase, stejně jako disponuje nástroji pro ladění a úpravu prostředí. To vše umožňuje vývojářům efektivně konstruovat, optimalizovat a zdokonalovat jejich 3D projekty.[11][16]

Je přístupný prostřednictvím online nástrojů, jako jsou Playground a Sandbox. Ty jsou hlavními platformy, které tento inspektor integrují pro zjednodušení přístupu a zefektivnění procesu vývoje. Nicméně Babylon.js Inspector je možné použít také jako samostatný nástroj ve vlastních projektech Babylon.js. K tomuto účelu je nutné do vlastního projektu zahrnout příslušný balíček a následně zavolat metodu show() na vlastnosti debugLayer objektu scene.[11][16]

```
1 scene.debugLayer.show();
```

Příklad 26: Volání metody show() pro zobrazení inspektoru[16]

15 Praktická část

V rámci praktické části mé bakalářské práce jsem se ponořil do detailního prozkoumání a praktické aplikace různých funkcí, nástrojů a pokročilých technik, které poskytuje framework Babylon.js. Nejdříve jsem se zaměřil na důkladné porozumění těmto aspektům a následně jsem se snažil demonstrativně ukázat, jak lze tyto nástroje a techniky efektivně využívat v praxi. K tomuto účelu jsem vytvořil dvě webové aplikace. Stránka je dostupná na adrese https://filipchytra.github.io/BabylonJS/.

První aplikace, webová stránka, slouží jako doplňkový zdroj k informacím o práci s Babylon.js, které jsem podrobněji rozebral v teoretické části mé práce. Stránka poskytuje ukázky kódu a renderovaných 3D scén, které cíleně demonstrují různé techniky a metody používané v Babylon.js. Tyto zahrnují, ale nejsou omezeny na vytváření a manipulaci s 3D objekty (meshes), jejich texturování, práci s osvětlením a stíny, animací, importování externích 3D modelů a práci s kamery a pohledem uživatele. Tato stránka slouží jako interaktivní vzdělávací nástroj pro ty, kteří se chtějí naučit více o Babylon.js a jeho možnostech.

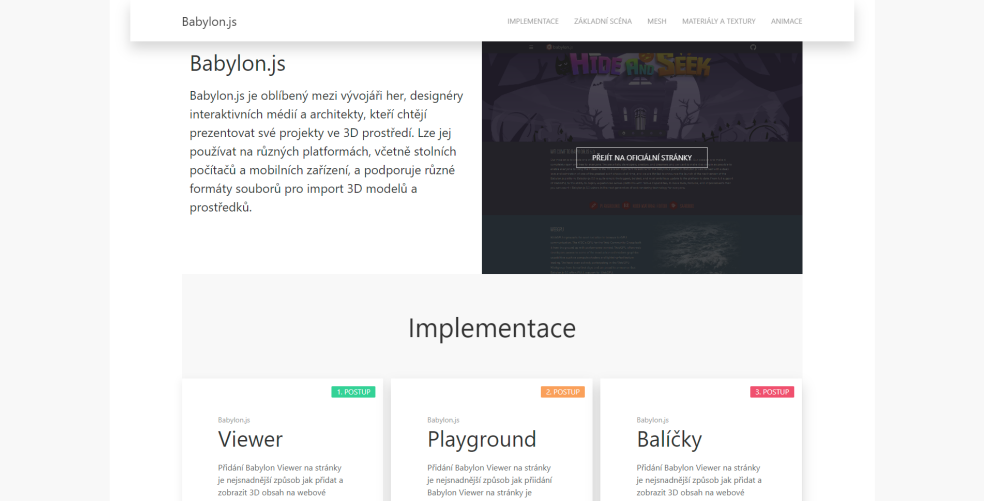
Druhou aplikací je detailní 3D model sluneční soustavy. Tato aplikace byla navržena a implementována tak, aby ukázala potenciál Babylon.js pro vytváření složitějších a interaktivních 3D scén a aplikací. Model poskytuje uživatelům možnost prozkoumávat sluneční soustavu, přibližovat se k jednotlivým planetám a pozorovat jejich orbity. Tato aplikace je ukázkou, jak lze Babylon.js využít pro vytváření náročnějších a uživatelsky příjemných 3D webových aplikací.

Během vývoje těchto aplikací jsem si osvojil řadu technických dovedností a získal hlubší porozumění pro práci s 3D grafikou na webu. Začal jsem se detailněji zabývat problematikou 3D technologií a grafiky, učil jsem se, jak správně využívat jednotlivé nástroje a techniky, které Babylon.js nabízí. Zkoumal jsem, jak je možné tyto nástroje a techniky zapojit do standardního vývoje webových

aplikací a technologií. Mým cílem bylo nejen rozšířit své technické dovednosti, ale také prozkoumat možnosti, které přináší práce s Babylon.js a ukázat, jak lze tento framework využít pro vytváření kvalitních a uživatelsky příjemných 3D webových aplikací.

15.1 První aplikace - Informační webová stránka

V rámci praktické části mé bakalářské práce jsem vytvořil informační webovou prezentaci. Platforma je navržena tak, aby poskytovala doplňkový a interaktivní zdroj informací k teoretické části práce. Návštěvníci zde mohou nalézt ukázky implementace různých funkcí a technik Babylon.js, které jsou podloženy konkrétními příklady z praxe.



Obrázek 13: Úvodní náhled informačního webu

15.1.1 Použité technologie a nástroje

V rámci vývoje jsem vycházel z osvědčených webových technologií, kterými jsou HTML, CSS a JavaScript. Využití těchto základních technologií jsem zvolil s ohledem na univerzálnost a přístupnost pro širokou škálu uživatelů. Pro generování a vizualizaci 3D scén jsem využil nástroj Babylon.js Playground,

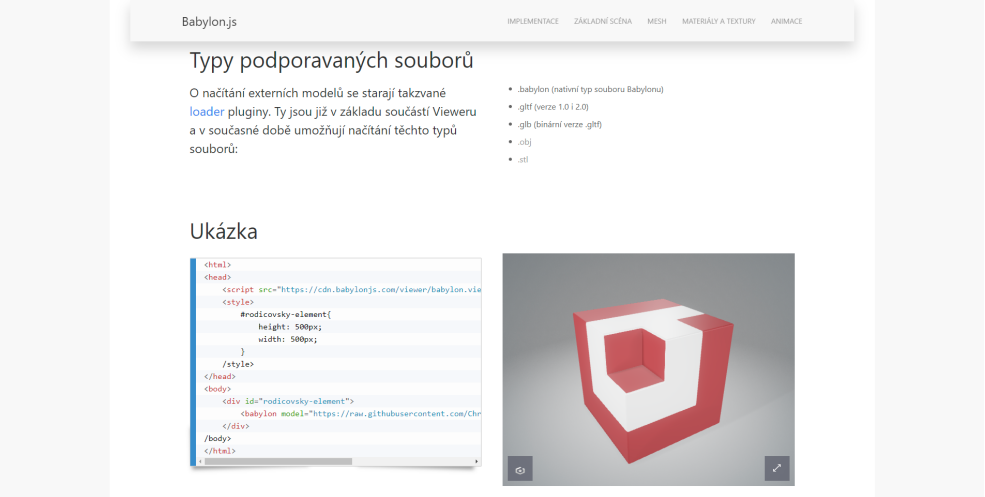
který mi poskytl flexibilní a intuitivní prostředí pro tvorbu a export 3D modelů. Následně jsem tyto modely integroval přímo do kódu aplikace.

Vzhled aplikace jsem sjednotil s využitím stylovacího nástroje UIKit, který mi umožnil rychlé a efektivní vytváření konzistentního uživatelského rozhraní.

Dalším důležitým nástrojem, který jsem při vývoji využil, byl Prism. Prism je JavaScriptový balíček, který se zaměřuje na formátování ukázek kódu na webových stránkách. Díky Prismu jsem byl schopen prezentovat ukázky kódu v přehledné a pro uživatele srozumitelné formě, blízké klasickým vývojovým prostředím.

```
1 <pre class="language-markup"><code><!--<script src="https
    ://cdn.babylonjs.com/viewer/babylon.viewer.js"></script
    >--></code></pre>
```

Příklad 27: Ukázka použití knihovny Prism pro zobrazení kódu v rámci stránky.



Obrázek 14: Příklad zobrazení kódu v rámci stránky s příslušným ukázkovým renderem

15.1.2 Struktura a vývoj aplikace

Vývoj této aplikace byl založen na prvotním rozboru a návrhu, v rámci kterého jsem si stanovil požadavky a strukturu aplikace. Výsledný produkt je

vícestránková aplikace, jejíž úvodní stránka slouží jako rozcestník do dalších tématicky zaměřených částí.

Tato struktura mi umožnila postupně a logicky prezentovat jednotlivé aspekty Babylon.js a poskytnout uživatelům důkladného průvodce tímto frameworkem.

Samotný vývoj aplikace jsem zahájil tvorbou základní struktury a stylizací webové stránky. Poté jsem začal přidávat konkrétní informace a ukázky kódu, které jsem připravil v nástroji Babylon.js Playground. Tyto ukázky jsem následně začlenil do webové prezentace, kde doplňují teoretické informace o Babylon.js.

15.1.3 Přehled a hodnocení práce na první aplikaci

Volba použití Babylon.js Playground pro integraci Babylon.js byla založena na jeho schopnostech pro rychlé prototypování a okamžitou vizualizaci výsledků, což se ukázalo jako vhodné pro specifika tohoto projektu.

V rámci tohoto procesu bylo možné realizovat 3D scény bez nutnosti složitých konfigurací nebo obtížného ladění, což výrazně usnadnilo celý proces vývoje. Babylon.js Playground jako nástroj pro vytváření a testování 3D scén přímo v prohlížeči se prokázal jako efektivní řešení pro splnění cílů tohoto projektu.

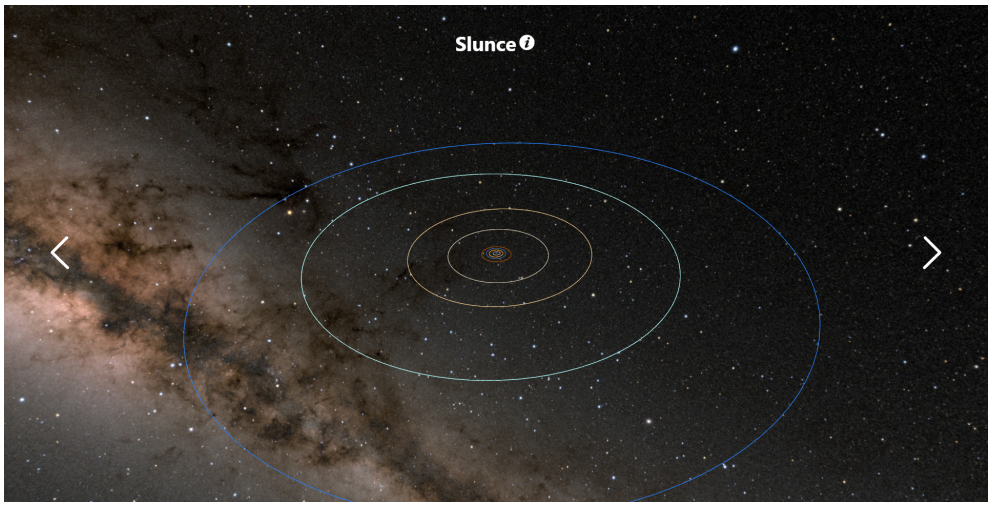
Během vývoje jsem se nesetkal s žádnými významnými překážkami nebo problémy. Věřím, že to je výsledek jasné dokumentace Babylon.js a intuitivního rozhraní Playground. Kombinace těchto nástrojů mi umožnila snadno pochopit a využít širokou škálu funkcí a nástrojů, které Babylon.js nabízí.

Na základě mých zkušeností z tohoto projektu mohu konstatovat, že práce s Babylon.js byla intuitivní a bez větších komplikací. Framework nabízí uživatelsky příjemné prostředí a poskytuje užitečné nástroje pro rychlý a efektivní vývoj 3D webových aplikací. Tato zkušenost mě posílila v přesvědčení, že Babylon.js je silný a flexibilní nástroj pro vývoj 3D webových aplikací.

15.2 Druhá aplikace - Model Sluneční soustavy

Druhá aplikace, kterou jsem vytvořil, slouží jako interaktivní model sluneční soustavy, vycházející z reálných hodnot pro zobrazení oběžných drah a modelů planet. Struktura aplikace je koncipována jako Single Page Application (SPA), což znamená, že veškerý obsah je dostupný na jedné webové stránce bez nutnosti jejího obnovování. Model naleznete na adrese https://filipchytra.github.io/SlunecniSoustava/.

Hlavní komponentou je 3D scéna, která zobrazuje model sluneční soustavy. Abych umožnil uživatelům detailnější prozkoumání, přidal jsem interakci s modelem umožňující uživatelům přiblížit se k vybraným vesmírným tělesům a to buďto pomocí kolečka myši, nebo příslušných gest na dotykových obrazovkách. Každý model planet je animací rotován kolem své osy Y, aby bylo možné si prohlédnout textury planet i ze strany, ze které nejsou osvětlovány Sluncem.



Obrázek 15: Úvodní náhled webové aplikace Sluneční soustavy

Aplikace poskytuje také doplňující informace o každé z planet. Tyto informace jsou zobrazeny na informativních kartách, které se objeví po dvojitém poklepání na model rotující planety nebo po kliknutí na ikonu 'i'. Pro přepínání se mezi jednotlivými tělesy sluneční soustavy mohou uživatelé využít ikon šipek, které jsou umístěny po stranách stránky.

Obrázek 16: Náhled na přiblížený model planety Země s postranní kartou s informacemi.

Pro tématiku modelu jsem se rozhodl na základě toho, aby mi umožňoval maximálně využít možnosti Babylonu jako renderovacího 3D enginu a zároveň pro jeho tvorbu využít předpřipravených funkcí pro tvorbu jednoduchých těles jako je čára a koule. To mi umožnilo vyhnout se náročnému modelování složitých polygonových sítí v externích nástrojích.

15.2.1 Použité technologie a nástroje

V procesu vývoje mé aplikace jsem se rozhodl využít soubor nástrojů, které mi pomohly optimalizovat pracovní postupy a maximalizovat efektivitu vývoje.

Klíčovým rozhodnutím bylo přímé začlenění frameworku Babylon.js do mého projektu, místo využití Playground nástroje, který jsem používal při tvorbě první aplikace. Pro tuto cestu jsem se rozhodl díky tomu, že tento přístup umožňuje větší kontrolu nad funkcemi a strukturou aplikace.

Dále jsem se rozhodl využít jazyk Typescript pro vývoj samotného projektu. Toto rozhodnutí bylo motivováno skutečností, že samotný framework je postaven a vyvíjen v tomto jazyce. Další motivací pro výběr Typescriptu byla jeho silná typová kontrola a sada funkcí, které se ukázaly jako cenné při řešení složitosti tohoto projektu.

Jelikož však webové prohlížeče primárně podporují Javascript, bylo nutné zajistit kompilaci kódu z Typescriptu do Javascriptu. Tento proces jsem realizoval pomocí Vite.js, což je velmi rychlý a efektivní nástroj pro bundlování JavaScriptu. Tato volba mi usnadnila proces konfigurace a ladění aplikace, což bylo klíčové pro efektivní vývoj.

15.2.2 Struktura a vývoj aplikace

Vývoj této aplikace byl velmi odlišný od mé předchozí zkušenosti s vývojem aplikací, především z důvodu, že jsem se rozhodl pro přímé využití Babylon.js Frameworku, a také díky volbě vývoji v jazyce TypeScript. Pro ulehčení celého procesu jsem využil komunitou vytvořeného a vývojáři frameworku doporučeného Vite boilerplate, který je předpřipraven pro rychlé spuštění projektu.

Na začátku vývojového procesu bylo nezbytné provést několik počátečních kroků. Nejprve bylo třeba naklonovat Vite boilerplate z GitHubu do místního úložiště pomocí nástroje Git. Tento repositář lze nalézt na URL adrese https://github.com/paganaye/babylonjs-vite-boilerplate. Poté, za použití Node.js a terminálového příkazu npm install, jsem nainstaloval potřebné balíčky zahrnující i core modul Babylonu, které byly již definovány v příslušném souboru naklonovaného projektu package.json. Samotný Boilerplate již obsahoval index.html soubor s implementovaným kanvasem a potřebnými TypeScript soubory inicializující základní scénu.

Zde je vhodné poznamenat, že jak verzovací nástroj Git, tak i Node.js jsou základními nástroji pro vývoj JavaScriptových aplikací a jejich používání je v dnešní době standardem. Já jsem již tyto nástroje před vývojem aplikace používal, a proto jsem mohl jejich instalaci přeskočit.

Po úspěšném nastavení vývojového prostředí jsem začal pracovat na úpravách poskytnuté základní scény. Změnil jsem konfiguraci kamery z FreeCamera na ArcRotateCamera, která se lépe hodila pro potřeby mého projektu. Poskytuje vhodnější možnost pohybu kamery kolem určitého bodu v prostoru. Kameře jsem definoval vlastnost automatického chování na samostatné otá-

čení pro zlepšení dynamičnosti zobrazení. Tato funkce byla ideální pro moji aplikaci, kde hlavním záměrem bylo prohlížení planet ze všech úhlů.

```
1 const camera: BABYLON.ArcRotateCamera = new BABYLON.
     ArcRotateCamera("Kamera", camAlpha, camBeta, camDist,
     camTarget, scene);
2 this.camera = camera;
3 camera.useAutoRotationBehavior = true;
4 camera.autoRotationBehavior!.idleRotationSpeed = -0.05;
5 camera.autoRotationBehavior!.idleRotationSpinupTime =
     5000;
6 camera.autoRotationBehavior!.idleRotationWaitTime = 2000;
7 camera.attachControl(true);
```

Příklad 28: Inicializace ArcRotate kamery a nastavení volnoběžného otáčení.

Následujícím krokem bylo nastavení osvětlení scény. Původní scéna implementovala světlo typu HemisphericLight, která osvětluje scénu rovnoměrně ze všech směrů. Na základě této jeho vlastnosti byla nahrazena za světlo PointLight, které lépe odpovídalo dynamice osvětlení napodobující Sluneční soustavu. Tento typ světla vychází z jednoho bodu a šíří se do všech směrů, což napodobuje chování Slunce jako zdroje světla ve Sluneční soustavě. Toto světlo jsem umístil do středu scény.

```
1 const light: BABYLON.PointLight = new BABYLON.PointLight("
     starLight", BABYLON.Vector3.Zero(), scene);
2 light.intensity = .15;
3 light.diffuse = new BABYLON.Color3(8, 8, 8);
4 light.specular = new BABYLON.Color3(1, 1, 1);
```

Příklad 29: Inicializace a nastavení bodového světla.

Jeden z mých hlavních cílů bylo vytvořit co nejrealističtější model sluneční soustavy. K tomu jsem využil externí zdroje dat pro získání reálných fyzikálních vlastností planet. Z těchto dat jsem mohl vytvořit a definovat struktury

typů a interfaců v TypeScriptu pro informace o planetách. Následně jsem vytvořil třídy, které obsahují metody provádějící potřebné výpočty poloh planet v průběhu jejich oběhu kolem Slunce.

Každá planeta byla reprezentována instancí třídy Planeta, která byla inicializována s parametry odpovídajícími fyzikálním vlastnostem dané planety. Třída Planeta obsahuje metody odpovídající za tvorbu modelů planet a jejich oběžných drah na základě poskytnutého jména planety jakožto parametry metod. Všechny z planet v modelu disponují minimálně diffuse texturou ve vysokém rozlišení.

Tyto textury jsem získal z externího webového zdroje. Byly vytvořeny na základě dat o výšce a snímků získaných od NASA. Barvy a odstíny textur byly upraveny podle fotografií v přirozených barvách pořízených různými vesmírnými sondami, jako jsou Messenger, Viking a Cassini, a samozřejmě Hubbleův vesmírný dalekohled. Navíc pro planetu Zemi bylo možné z tohoto zdroje získat normálovou mapu a texturu zachycující světelné zdroje nočního povrchu planety, a proto jako jediná disponuje jak normálovou, tak emisní texturou materiálu, které jí propůjčují ještě více reálný vzhled. Avšak v modelu lze ještě naleznout jednu výjimku vzhledu a to u planety Saturn, která má navíc přiřazenou texturu jeho prstence.[22]

Pro dodání většího realismu jsem ve scéně vytvořil za pomoci equirectangular textury 360° panoramatické fotografie hvězdné oblohy. Ta mapuje každý bod sféry na unikátní bod na rovině, čímž usnadňuje manipulaci a zpracování dat. Tuto texturu lze přidat pomocí objektu Babylonu PhotoDome, která vytvoří texturu pozadí celé scény.

```
1 const starMap: BABYLON.PhotoDome = new BABYLON.PhotoDome("
    hveznaObloha", StarMap, {size: 10000}, scene);
```

Příklad 30: Vytvoření objektu PhotoDome nastavující texturu hvězdného nebe.

Na závěr jsem přidal ovládací prvky šipek pro iteraci mezi jednotlivými vesmírnými objekty a prvek karty s informacemi o planetách, které se dyna-

micky mění v závislosti na zvolené planetě.

15.2.3 Přehled a hodnocení práce na druhé aplikaci

Během vývoje této aplikace jsem čelil hned několika výzvám. Jednou z nejnáročnějších byla implementace algoritmického převodu oběžných drah planet do trojrozměrného prostoru. Původně jsem předpokládal, že to bude jednoduchý úkol, ale brzy jsem zjistil, že tento proces je velmi složitý a vyžaduje značné množství fyzikálních výpočtů. Tyto výpočty jsem vyčlenil do samostatné třídy Orbits.

Vycházel jsem z práce pana Paula Schlytera dostupné na webu, který tento výpočet podrobně popsal v jedné z jeho prací. Ta mi pomohla lépe pochopit příslušné výpočty pro modelování oběžných drah planet ve 3D prostoru.[23]

Další výzvou bylo zvládnout zachování měřítka a velikosti planet v kontextu celé sluneční soustavy. Existuje značný rozdíl mezi velikostí sluneční soustavy a velikostí jednotlivých planet, což představovalo významnou výzvu pro vykreslování ve 3D prostoru.

Když jsem se snažil vytvořit pohled na celou sluneční soustavu, narazil jsem na limitace enginu, který má omezenou schopnost zobrazovat velké vzdálenosti. Na druhé straně, při nastavení vhodného měřítka pro vykreslení celého modelu, po snaze zobrazit jednotlivé planety, jejich velikosti byly příliš malé pro to, aby jejich geometrie a síť byly patrné.

Tento problém jsem vyřešil tím, že jsem vymyslel dynamický algoritmus pro výpočet vzdálenosti kamery od počátku. Tento algoritmus dynamicky navyšuje násobek zvětšení za každý krok přiblížení, což mi umožnilo zobrazit jak celou sluneční soustavu, tak detaily jednotlivých planet.

```
1 private scaleOnZoom(): void {
2     const solarSystem = this.scene.getMeshById("
          SlunecniSoustava");
3     if (solarSystem) solarSystem.scaling.setAll(1 / Math.
          pow(1.05, this.zoomFactor))
```

```
4       this.meshIdInView.scaling.setAll(300000 / Math.pow
            (1.05, this.zoomFactor))
5   }
```

Příklad 31: Metoda zajišťující dynamické zvětšení objektů Sluneční soustavy na základě přiblížení.

Pokud jde o moji zkušenost s vývojem této aplikace, musím říci, že byla jak vysoce vzdělávací, tak i náročná. Použití Babylon.js se ukázalo jako užitečné pro tento konkrétní projekt, umožnilo mi využít jeho nástroje a funkce pro vytvoření požadované aplikace.

Volba pro reálný model sluneční soustavy se však ukázala jako náročnější úloha, než jsem původně očekával. Neznamená to však, že bych litoval svého rozhodnutí - naopak, tato výzva mě motivovala k dalšímu učení a zdokonalování mého přístupu k vývoji softwaru.

15.3 Porovnání přístupů

Během tvorby svých 3D webových aplikací jsem měl možnost vyzkoušet dva různé přístupy k vývoji. Díky tomu jsem měl příležitost srovnat jejich vlastnosti, výhody a nevýhody na základě skutečných zkušeností, které jsem získal během své práce. Tyto zkušenosti mi poskytly ucelený pohled na oba přístupy a umožnily mi pochopit, jak se každý z nich může nejlépe uplatnit v různých scénářích a kontextech.

Při práci s Playgroundem jsem ocenil jeho okamžitou odezvu a interaktivitu. Díky tomu jsem mohl provádět změny a ladit scénu v reálném čase a sledovat jak se mé změny projevují ve výsledné 3D scéně. Toto prostředí je ideální pro rychlý prototypový vývoj, experimentování s různými funkcemi a možnostmi frameworku a je plně dostupný ve webovém prostředí.

Přímý vývoj v rámci frameworku Babylon.js mi poskytl plnou kontrolu nad celým projektem. Měl jsem možnost strukturovat kód podle svých preferencí a potřeb a využívat výhod moderních vývojových nástrojů a technologií, jako je

Typescript a bundling pomocí Vite. Tento přístup je vhodný pro komplexnější projekty a aplikace, kde je důležitá údržba kódu a jeho škálovatelnost.

15.4 Potenciální uplatnění 3D webových aplikací

3D webové aplikace těží z výhod, které jsou společné pro všechny 3D technologie. Přinášejí možnost vizualizovat složité koncepty a objekty způsobem, který je intuitivní a snadno pochopitelný. To umožňuje představit objekty v jejich skutečné třírozměrné podobě, což usnadňuje porozumění a zkoumání.

Webové aplikace představují významnou alternativu k tradičním nativním 3D programům. Jedním z důležitých aspektů je univerzální dostupnost těchto aplikací, jelikož mohou být snadno sdíleny prostřednictvím odkazů. Jediným požadavkem pro uživatele je mít moderní webový prohlížeč. To eliminuje potřebu stahování nebo instalace specifického softwaru a usnadňuje přístup pro široké spektrum uživatelů. Věřím, že 3D webové aplikace mají obrovský potenciál v řadě odvětví.

15.4.1 Herní průmysl

Odvětví, které velmi výrazně ovlivňuje vývoj a adopci 3D na webu již dnes, je herní průmysl. Ten je obecně na vrcholu 3D technologického pokroku, jelikož hry často vyžadují nejnovější a nejvyspělejší technologie pro vytváření bohatých, strhujících a interaktivních zážitků. Herní průmysl také generuje značné příjmy a má velký počet uživatelů, což jej činí atraktivním pro investice a inovace.

Díky tomu má herní průmysl potenciál posunout 3D webové technologie vpřed, protože hry vyžadují pokročilé grafické schopnosti a rychlé výpočty. Pokud budou webové platformy schopny podporovat tyto požadavky, otevře se cesta pro širší využití 3D na webu v dalších oblastech.

Pokud se 3D webové hry stanou běžné, zvýší to celkovou povědomost a komfort uživatelů s 3D technologiemi na webu. To by mohlo vést k vyššímu

očekávání a poptávce po 3D vizualizacích a interakcích v dalších oblastech, jako je e-commerce nebo vzdělávání. Takže rozvoj 3D her na webu by mohl mít pozitivní dopad na celou řadu odvětví a aplikací.

15.4.2 Školství

Například v oblasti školství mohou tyto aplikace vizualizovat komplexní koncepty a procesy, čímž zlepšují porozumění a zapamatovatelnost učiva. Mohly by tak také nahradit obvykle velmi drahé fyzické modely, které si škola pro podporu výuky často pořizuje. Moje druhá aplikace modelu sluneční soustavy by do jisté míry mohla sloužit jako příklad tohoto typu edukačního nástroje.

15.4.3 Komerční a reklamní sféra

V oblasti e-commerce mohou 3D aplikace poskytnout zákazníkům možnost prohlížet si produkty z různých úhlů. To může vést k lepšímu rozhodování o nákupech a zvýšené spokojenosti zákazníků. Navíc 3D vizualizace by mohly být použity k vytvoření poutavých reklamních kampaní, které zvyšují angažovanost a zájem o produkty.

15.4.4 Architektura a průmyslový design

Stejně jako v architektuře tak i v oblasti průmyslového designu, mohou 3D webové aplikace usnadnit prezentaci návrhů budov a městských plánů, či návrhů výrobků klientům a veřejnosti ve formě, která umožňuje prohlížení z různých úhlů s možností interakce.

15.4.5 Zdravotnictví

V zdravotnictví mohou 3D vizualizace pomoci lékařům a pacientům lépe porozumět zdravotním stavům a lékařským procedurám. Mohou také sloužit jako nástroje pro trénink chirurgických dovedností.

15.4.6 Výzkum a věda

A nakonec se domnívám, že 3D webové aplikace mohou také najít uplatnění v oblasti výzkumu a vědy, kde mohou pomoci vizualizovat a analyzovat komplexní data a fenomény.

16 Závěr

V průběhu mé bakalářské práce jsem se zaměřil na tvorbu 3D modelů, scén a jejich integraci do webových aplikací pomocí open-source frameworku Babylon.js. Účelem této práce bylo podrobně představit Babylon.js jako výkonný a multifunkční nástroj pro vytváření 3D webových aplikací a demonstrovat jeho praktické využití.

Během mé práce jsem dospěl k závěru, že Babylon.js je skutečně užitečný a efektivní nástroj pro implementaci 3D renderů ve webovém prostředí. Jeho abstraktnější a zjednodušený přístup oproti nízkoúrovňovým standardům, na kterých je postaven, jej činí atraktivním a přístupným nástrojem pro vývojáře. 3D webové aplikace jsou sice stále na počátku svého masového využití, ale díky svému výkonu a schopnosti práce s 3D aktivy jsou již nyní plně implementace schopné, a do jisté míry konkurence schopné s nativními řešeními. Díky velkému prostoru pro další rozvoj existuje velký potenciál pro jejich zdokonalení.

Dále jsem si uvědomil, že 3D webové aplikace mají široké uplatnění napříč různými obory, včetně vzdělávání, lékařství a e-commerce. Při pohledu na růst herního průmyslu v posledních dekádách, který je hnacím motorem v oblasti 3D technologií, předpokládám, že rostoucí popularita a investice do 3D her na webu mohou značně urychlit vývoj a adopci 3D webových technologií.

Na základě mé práce se nabízí mnoho možností pro budoucí výzkum. Například, blízké spojení s technologiemi VR a AR, které jsou závislé na 3D technologiích, může představovat důležitou oblast pro další zkoumání. Téma metaverze a virtuálních světů se v posledních letech stává stále relevantnějším a diskutovaným v rámci odborné komunity, což dále podkresluje potenciál a význam 3D technologií pro budoucnost webových aplikací.

Seznam použité literatury a zdrojů

[1] Babylon.js: Powerful, Beautiful, Simple, Open - Web-Based 3D At Its Best [online]. [cit. 2023-01-08]. Dostupné z: https://www.babylonjs.com/

[2] Why 3D is the future of the internet?. Medium [online]. [cit. 2023-01-21]. Dostupné z: https://medium.com/naker/why-3d-is-the-future-of-the-internet-44d616633aba

[3] What is 3D rendering?. Treekit [online]. [cit. 2023-01-23]. Dostupné z: https://www.threekit.com/blog/what-is-3d-rendering

[4] BADLER, Norman I. a Andrew S. GLASSNER. *3D Object Modeling* [online]. 27 [cit. 2023-01-30]. Dostupné z: http://gamma.cs.unc.edu/courses/graphics-s09/LECTURES/3DModels_SurveyPaper.pdf

[5] Real-Time 3D Graphics with WebGL 2 [online]. 2nd ed. Packt, 2018 [cit. 2023-02-06]. ISBN 9781788629690. Dostupné z: https://www.packtpub.com/product/real-time-3d-graphics-with-webgl-2-second-edition/9781788629690

[6] Elementor. 10 Cutting-Edge Uses of 3D in Web Design [online]. 2021 [cit. 2023-02-15]. Dostupné z: https://elementor.com/blog/3d-web-design/

[7] MDN Web Docs: Resources for Developers, by Developers [online]. [cit. 2023-02-19]. Dostupné z: https://developer.mozilla.org/en-US/

[8] Khronos Group. WebGL 2.0 Specification. [online]. 2017 [cit. 2023-02-19]. Dostupné z: https://registry.khronos.org/webgl/specs/latest/2.0

[9] WebGL Fundamentals [online]. [cit. 2023-02-19]. Dostupné z: https://webglfundamentals.org/

[10] Can I Use [online]. [cit. 2023-03-02]. Dostupné z: https://caniuse.com/

[11] WebGPU is coming to Babylon.js. Medium [online]. 2020 [cit. 2023-03-08]. Dostupné z: https://babylonjs.medium.com/a-brief-summary-of-babylon-js-tooling-14fb6c0b5fec

[12] *Going the Distance with Babylon.js* [online]. Packt, 2022 [cit. 2023-03-17]. ISBN 9781801076586. Dostupné z: https://www.packtpub.com/product/going-the-distance-with-babylonjs/9781801076586

[13] 20 Interactive 3D Javascript Libraries & Frameworks. Bashooka [online]. 2019 [cit. 2023-04-03]. Dostupné z: https://bashooka.com/coding/3d-javascript-libraries

[14] WebGL Frameworks: Three.js vs Babylon.js. In plain english [online]. 2020 [cit. 2023-04-24]. Dostupné z: https://plainenglish.io/blog/webgl-frameworks-three-js-vs-babylon-js-36975d915694

[15] Unity Documentation [online]. [cit. 2023-04-30]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[16] Babylon.js Documentation [online]. [cit. 2023-05-04]. Dostupné z: https://doc.babylonjs.com/

[17] Medium: A brief summary of Babylon.js tooling [online]. 2020 [cit. 2023-05-10]. Dostupné z: https://babylonjs.medium.com/a-brief-summary-of-babylon-js-tooling-14fb6c0b5fec

[18] WEBER, Raanan. *Game Development - Babylon.js: Building a Basic Game for the Web.* MSDN Magazine Issues [online]. 2015, 12/01/2015, 30(13) [cit. 2023-05-15]. Dostupné z: https://learn.microsoft.com/en-us/archive/msdn-magazine/2016/january/game-development-babylon-js-advanced-features-for-enhancing-your-first-web-game

[19] WEBER, Raanan. *Babylon.js: Advanced Features for Enhancing Your First Web Game.* MSDN Magazine Issues [online]. 2016, 02/01/2019,

31(1) [cit. 2023-05-25]. Dostupné z: https://learn.microsoft.com/en-us/archive/msdn-magazine/2016/january/game-development-babylon-js-advanced-features-for-enhancing-your-first-web-game

[20] CUTAHE, David. *Building Shaders With Babylon.js.* Smashing Magazine: For Web Designers And Developers [online]. nov 1, 2016 [cit. 2023-06-01]. Dostupné z: https://www.smashingmagazine.com/2016/11/building-shaders-with-babylon-js/

[21] Understanding UVs: Love or Hate Them, They're Essential. Pluralsight [online]. Nov 1 2022 [cit. 2023-06-12]. Dostupné z: https://www.pluralsight.com/blog/film-games/understanding-uvs-love-them-or-hate-them-theyre-essential-to-know

[22] Solar Textures: Solar System Scope [online]. [cit. 2023-06-20]. Dostupné z: https://www.solarsystemscope.com/textures/

[23] SCHLYTER, Paul. *How to compute planetary positions* [online]. [cit. 2023-06-20]. Dostupné z: https://stjarnhimlen.se/comp/ppcomp.html

Seznam obrázků

Seznam příkladů

A Příloha

CD/DVD obsahuje plné znění mé bakalářské práce s názvem Zadání-BP-Chytra.pdf a soubor Příloha.pdf obsahující odkazy na repositáře se zdrojovými kódy obou aplikací.

B Příloha

Informační webová stránka Babylon.js: https://filipchytra.github.io/BabylonJS/

Model Sluneční soustavy: https://filipchytra.github.io/SlunecniSoustava/

GitHub repozitář se zdrojovými kódy aplikace informační webové stránky Babylon.js: https://github.com/FilipChytra/BabylonJS

GitHub repozitář se zdrojovými kódy aplikace Model Sluneční soustavy: https://github.com/FilipChytra/BabylonJS