VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

PREZENTACE AUDIOVIZUÁLNÍHO DÍLA VE VIRTUÁLNÍ REALITĚ
AUDIOVISUAL WORK PRESENTATION IN VIRTUAL REALITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Daniel Lazorčák**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Kamil Říha, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Daniel Lazorčák ***ID:*** 203738
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Prezentace audiovizuálního díla ve virtuální realitě

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s herními enginy, např. s Unity a Unreal, a s dalšími vývojovými prostředky pro vytváření prostředí ve virtuální realitě a porovnejte je mezi sebou z hlediska možnosti vytváření vlastních modulů pro zpracování prostorového zvuku. Pomocí vybraného nástroje realizujte alespoň dvě různá prostředí virtuální výstavy, do kterých umístíte vybraná audiovizuální a grafická díla. Tato prostředí mohou být vytvořená na základě reálných prostředí nebo zcela smyšlená. Kritériem je pouze možnost pohybu uživatele ve virtuálním prostoru. Zjistěte možnosti načítání grafických, zvukových a video souborů za běhu programu na základě vyhledávacích kritérií zadaných uživatelem a implementujte toto načítání a jednoduché rozhraní pro zadávání vyhledávacích kritérií. Prostudujte také možnost navázání vyhledávání na relační databázi a možnosti prostorové reprezentace zvuku ve vybraném nástroji. Ty využijte pro audiovizuální díla umístěná ve virtuálním prostoru.

DOPORUČENÁ LITERATURA:

[1] HOCKING, Joseph. Unity in action: multiplatform game development in C#. Shelter Island, NY: Manning Publications Co., [2015]. ISBN 978-1617292323.

[2] LINOWES, Jonathan. Unity Virtual Reality Projects. Second Edition. Birmingham: Packt Publishing, 2018. ISBN ISBN 978-1-78847-880-9.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: doc. Ing. Kamil Říha, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cieľom tejto práce je vyhotovenie rešerše o dostupných herných enginoch pre tvorbu prehliadky audiovizuálnych diel vo virtuálnej realite, naštudovanie problematiky mapovania textúr, voľba herného enignu ,ktorý spĺňa podmienky, vytvorenie scény virtuálnej prehliadky vo vybranom engine, demonštrácia premietania audiovizuálneho diela na výstavné teleso pomocou streamovania z internetového úložiska s využitím priestorového zvuku a s možnosťou ovládať prehrávanie daného diela. Vytvorenie vyhľadávania videí v databáze pomocou zadaných tagov s následnou možnosťou nájdené video premietať na teleso. Popísané sú vlastnosti a informácie o enginoch Unity, Unreal Engine, Game Maker Studio, Godot a CryEngine. Vytvorená je spustiteľná aplikácia virtuálnej prehliadky s audiovizuálnymi dielami.

KĽÚČOVÉ SLOVÁ

audiovizuálne dielo, C#, herný engine, MySQL, stream, tag, Unity, virtuálna galéria, virtuálna realita

ABSTRACT

Goal of this thesis is research of Game Engines aviable for the creation of gallery of audiovisual artworks in virtual reality, research of texture mapping, choice of suitable game engine for our work, creation of scene of the virtual gallery in selected Game Engine, demonstration of projecting the artwork onto exhibit body by streaming it from internet storage with use of ambisonic sound that engine provides and with abillity to control the playback of the piece. Creation of search engine for looking up artpieces in database by using tags with option to project the found piece to the body. Properties of Game Engines Unity, Unreal Engine, Game Maker Studio, Godot and CryEngine are given. A working application of virtual gallery with implented audiovisual artpieces is created.

KEYWORDS

audiovisual artpiece, C#, Game engine, MySQL, streamtag, Unity, virtual gallery, virtual reality

LAZORČÁK, Daniel. *Prezentace audiovizuálního díla ve virtuální realitě*. Brno, 2020, 62 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Ing. Kamil Říha, Ph.D.

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „Prezentace audiovizuálního díla ve virtuální realitě“ som vypracoval samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu práce doc. Ing. Kamilovi Říhovi, Ph.D. a garantovi práce doc. Ing. Jiřímu Schimmelovi, Ph.D. za rady, konzultácie a pomoc pri tvorbe tejto bakalárskej práce.

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

V tejto bakalárskej práci sa venujeme problematike vytvárania prehliadky audiovizuálnych diel vo virtuálnej realite. Vyhotovíme rešerš dostupných herných enginov na tvorbu prehliadky audiovizálnych diel, popíšeme ich vlastnosti, vrátane licenčných podmienok a na základe týchto vlastností sa rozhodneme pre konkrétne vývojové prostredie. Oboznámime sa s teóriou mapovania textúr na objekty.

Vytvoríme scénu virtuálnej prehliadky obsahujúcu výstavné telesá s premietanými audiovizuálnymi dielami. Tieto audiovizuálne diela budeme streamovať z internetového úložiska. Aplikujeme možnosti priestorového zvuku vybraného enginu. Budeme demonštrovať možnosť komunikácie medzi herným enginom a internetovým úložiskom pomocou protokolu FTP.

Vytvoríme menu pre vyhľadávanie v databáze audiovizuálnych diel pomocou priradených tagov s možnosťou vyhľadané video premietať na teleso. Vytvoríme ovládanie prehrávania jednotlivých audiovizuálnych diel. Po dohode sme sa rozhodli, že sa virtuálna prehliadka nebude konať vo výstavných priestoroch, ale na asteroide vo vesmíre a tak sa v práci simuluje vybrané prostredie.

1 Herné Enginy

Herné enginy sú programy, ktoré primárne slúžia na tvorbu a vývoj počítačových hier, no dajú sa použiť aj na tvorbu architektúry, filmu, rôznych simulácií, alebo v našom prípade na tvorbu prehliadky audiovizuálnych diel vo virtuálnej realite. Pri týchto programoch sa stretávame s dvoma pojmami: API (application programming interface) a SDK (software development kit). API je softvérové prostredie operujúce so systémami, knižnicami a službami, ktoré užívateľovi umožňujú používanie týchto nástrojov. SDK je súbor nástrojov, knižníc a množstva API, ktoré umožňujú programovanie a tvorbu v engine. Herné enginy teda ponúkajú API vo svojom SDK. [1] V tejto kapitole sa pozrieme na niektoré z najznámejších a najviac používaných herných enginov na trhu. Pre tvorbu nášho produktu je potrebné si zvoliť engine, ktorý bude obsahovať všetky potrebné nástroje na tvorbu virtuálnej prehliadky a licenčné obmedzenia budú minimálne.

1.1 Unity

1.1.1 Základné informácie

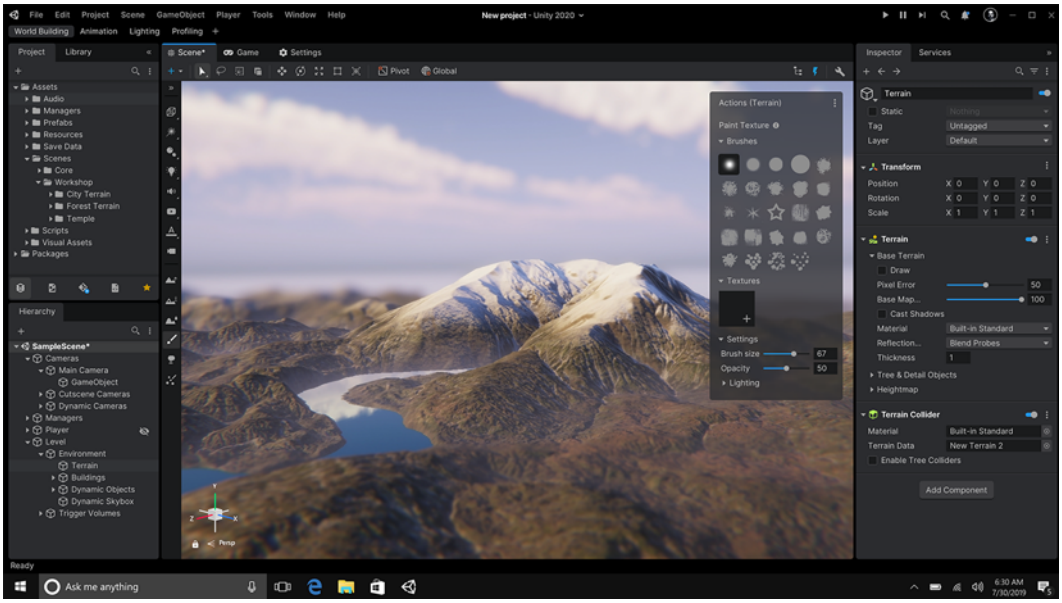
Unity, vyvíjaný spoločnosťou Unity Technology, patrí medzi najznámejšie herné enginy. Vznikol v roku 2005 s cieľom sprístupniť herné vývojárstvo širšej verejnosti bez ohľadu na technické znalosti a rozpočet. V roku 2006 tento engine vyhral cenu na celosvetovej konferencií vývojárov organizovanou spoločnosťou Apple. [2]

1.1.2 Programovacie jazyky a vlastnosti enginu

Engine primárne využíva kódovanie skriptov v programovacom jazyku C# a to pre programovanie pluginov ako aj samotných hier. Pre 3D developing ponúka engine špecifikácie kompresie textúr, minimapy, nastavenia rozlíšenia pre každú platformu, ktorú engine podporuje. Podporuje taktiež bump mapping, reflection mapping, parallax mapping, SSAO (screen space ambient occlusion), dynamické tiene, render-to-texture a full-screen post-processing efekty. [3]

1.1.3 Licencia

Licenčný model obsahuje dve ponuky - prvá je zdarma a tá je určená pre osobné použitie alebo pre menšie spoločnosti generujúce zisk menší ako 100 000 $ ročne. Druhá ponuka je platená a to formou „subscription“ – mesačné predplatné ktorého cena sa odvíja od ziskovosti hier. [4]



Obr. 1.1: Užívateľské prostredie Unity [5]

1.2 Unreal

1.2.1 Základné informácie

Unreal Engine, vyvíjaný spoločnosťou Epic Games, je ďalší z najznámejších herných enginov. Prvú verziu, ktorá vznikla už v roku 1998, vytvoril zakladateľ Epic Games Tim Sweeney pri tvorbe hry Unreal. [6] Pôvodne bol engine vyvinutý pre tvorbu hier typu FPS (First Person Shooter) no uplatnenie našiel vďaka svojej versatilite aj v mnohých iných žánroch.

1.2.2 Programovacie jazyky a vlastnosti enginu

Engine využíva programovací jazyk C++ a podporuje využívanie a tvorbu takzvaných blueprintov (uľahčenie práce so skriptami). Počas svojej existencie získal mnoho ocenení, medzi ktoré patrí aj cena za najúspešnejší herný engine roku 2014, ktorú vydala Guinessová kniha svetových rekordov. Engine obsahuje kvalitné nástroje a vďaka systému blueprintov a bohatému starter-contentu dokáže užívateľ jednoducho a efektívne pracovať s obsahom vyvíjaného produktu. [7]

1.2.3 Licencia

Unreal Engine podlieha free to use licencií, no po prekročení zárobkov v hodnote 3000 $ musí vývojár odvádzať 5 % zisku spoločnosti Epic Games. Ak sa Engine používa na architektúru, film, televíziu, podujatia, tréning a simulácie alebo iné projekty ktoré nespadajú pod kategóriu hier, tvorca nemusí odvádzať žiaden zisk a môže engine používať úplne zadarmo.[8]

Obr. 1.2: Užívateľské prostredie Unreal Engine [9]

1.3 Game Maker Studio

1.3.1 Základné informácie

Game Maker Studio (YoYo Games) je herný engine zameraný na tvorbu hier v 2D priestore. Pôvodne bol vytvorený Markom Overmarsom v roku 1999 pod názvom Animo. [10] Najnovšia verzia Game Maker Studio 2 bola vydaná v roku 2017 s prerobeným užívateľským prostredím, novými nástrojmi na editáciu, vylepšenou kompatibilitou medzi platformami a zlepšeným pracovným prostredím vďaka Drag and Drop systému. [11]

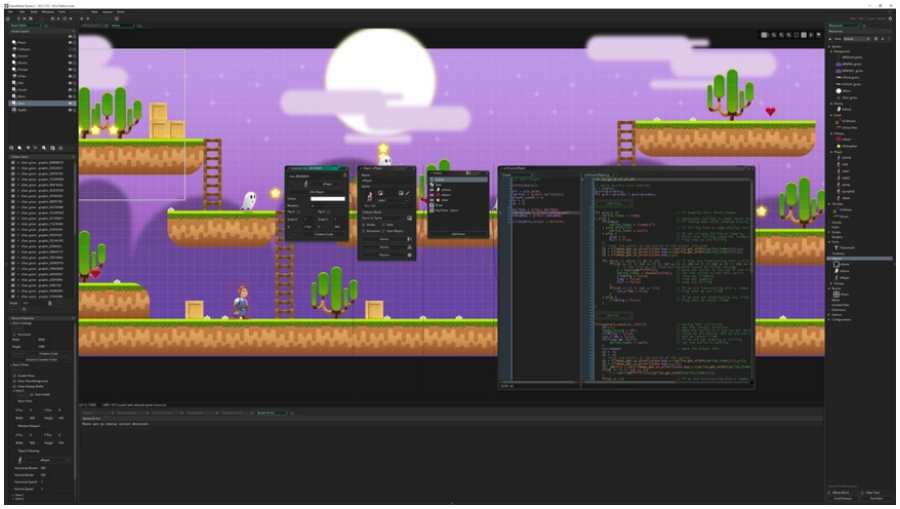
1.3.2 Programovacie jazyky a vlastnosti enginu

Engine využíva vlastný programovací jazyk GML (Game Maker Language), no podporuje aj C++ a Javascript.[12] V engine je aj systém Drag and Drop (DnD),

ktorý slúži na vykonávanie jednoduchých úloh ako vkladanie objektov, volanie funkcií alebo prácu so súbormi a dátami. [13]

1.3.3 Licencia

Pre používanie Game Maker Studio existuje viacero licencií. Prvá z nich je Trial Licence, ktorá vám dá prístup k enginu na 30 dní zdarma a slúži iba na testovanie enginu a nepovolí vám vytvoriť spustiteľný súbor. Druhá licencia je Creator Licence, ktorá je určená pre malé projekty. Pri kúpe tejto licencie dostanete prístup k enginu na jeden rok. Developer Licence je permanentná licencia určená pre väčšie projekty a vlastník získava prístup ku všetkým funkciám enginu. Console License je určená pre developerov hier určených na konzolové platformy (PS4, Xbox One, Nintendo Switch). Posledná licencia je Eudcation Licence a tá je určená pre učiteľov, ktorý chcú tvorbu v tomto engine vyučovať. [14]



Obr. 1.3: Užívateľské prostredie Game Maker Studio [15]

1.4 Godot

1.4.1 Základné informácie

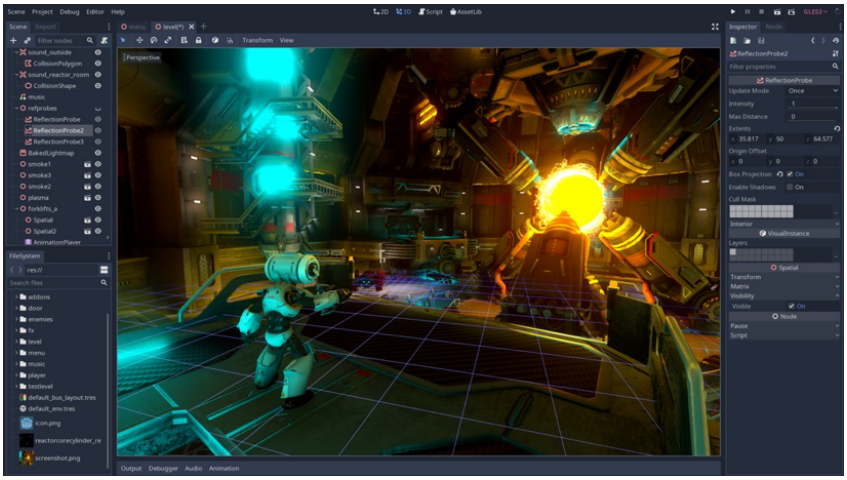
Godot je herný engine pre tvorbu 2D a 3D hier, ktorého autormi sú Juan Linietsky a Ariel Manzur. Prvú verziu tohto herného enginu vydali v roku 2014. [16] Engine je aj súčasťou vzdelávacieho programu na strednej škole West Virgina High School vďaka svojej jednoduchosti a nenáročnosti pre študentov stredných škôl. [17]

1.4.2 Programovacie jazyky a vlastnosti enginu

Engine využíva programovací jazyk GDScript, ktorý používa podobný syntax ako programovací jazyk Python. [18]

1.4.3 Licencia

Godot Engine je open-source softvér, ktorý podlieha licencií MIT. Užívateľ môže engine slobodne používať na akýkoľvek účel, môže študovať a upravovať zdrojový kód, zverejnovať upravené verzie programu a vydávať ich pod inými licenciami (zverejnená verzia nemusí byť zdarma). Jediné obmedzenie je, že vo výslednom produkte musí byť napísané, že je derivát Godot Engine-u. [19]



Obr. 1.4: Užívateľské prostredie Godot [20]

1.5 CryEngine

1.5.1 Základné informácie

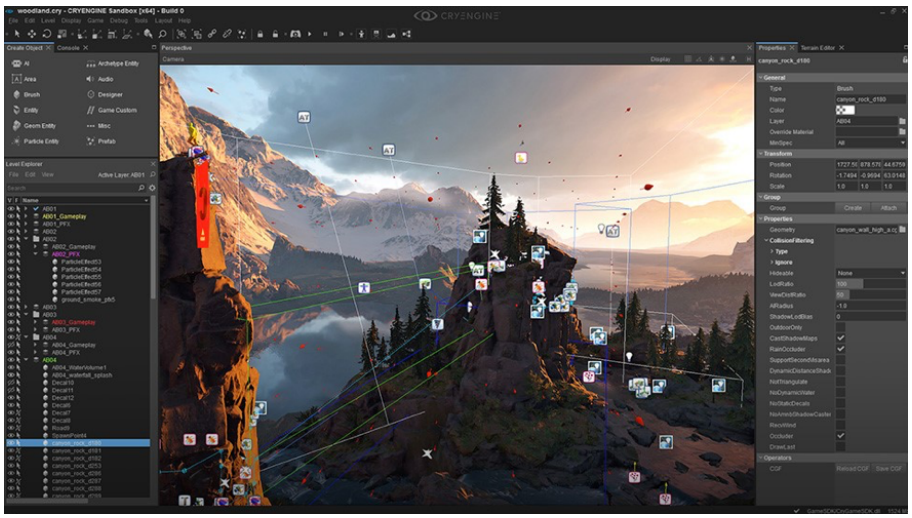
CryEngine je herný engine vyvíjaný spoločnosťou Crytek, ktorého prvá verzia bola vydaná v roku 2002. Najaktuálnejšia verzia CryEngine V je dostupná od roku 2016. Engine bol použitý na tvorbu mnohých známych a úspešných hier ako séria FarCry a Crysis (Crytek). [21]

1.5.2 Programovacie jazyky a vlastnosti enginu

CryEngine podporuje tvorbu v skriptov v programovacích jazykoch C++ a C#. Podporuje tvorbu vizuálnych efektov, sandboxov, AI [1], animácií, audia, fyziky a vizualizáciu v reálnom čase. [22] Pre tvorbu v Engine je užívateľ obmedzený na operačný systém Windows 7 a vyššie verzie tohto systému, pretože Engine nepodporuje iné operačné systémy.

1.5.3 Licencia

Použitie a inštalácia CryEnginu je zdarma, avšak po presiahnutí zisku 5000$ pri distribúcii diela, je tvorca povinný odvádzať 5% zo zisku spoločnosti Crytek bez ohľadu na účel a typ vytvoreného diela. [23]



Obr. 1.5: Užívateľské prostredie Cryengine [24]

[1] Artificial intelligence - umelá inteligencia

2 Mapovanie textúr

2.1 Textúry

Ak chceme vytvorenému objektu pridať realistickosť, využívame textúry. Tieto textúry majú za úlohu popisovať vlastnosti povrchov reálnych premetov ako:

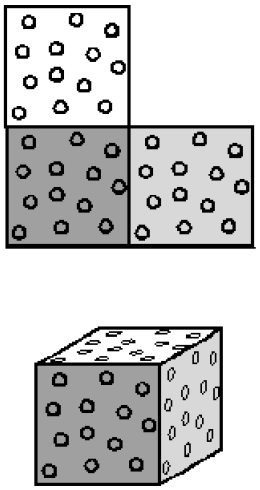
- Farba
- Lesk, zrkadlenie (množstvo odrazeného svetla)
- Priehľadnosť
- Zvrásnenie
- Hypertextúra (popisuje čo sa deje nad povrchom telesa)

Pri popisovaní povrchov môžeme skombinovať viacero textúr ako farba, lesk, zvrásnenie. Túto kombináciu textúr nazývame multitexturing. Podľa počtu dimenzií delíme textúry na:

- Jednorozmerné – typ prerušovania čiar
- Dvojrozmerné – povrch telesa, zvrásnenie
- Trojrozmerné – materiál, z ktorého sú telesá vyrobené
- Štvorrozmerné – animácia štruktúry telies (oheň, tečúca kvapalina a pod.) [25]

2.2 Mapovanie

Dvojrozmerné textúry sú najčastejšie vo forme štvorca alebo obdĺžnika. Mapovanie je aplikácia (nanesenie) textúry na povrch telesa – určovanie každému miestu dvojrozmernej textúry príslušnú súradnicu na trojrozmernom telese. Pre mapovanie textúry je potrebné vedieť definíciu textúry (textúra sa dá popísať pomocou textúrovacej funkcie *T(u, v),* ktorá priradzuje bodu na súradniciach [u, v] hodnotu modelovanej vlastnosti), tvar telesa (funkcia inverzného mapovania *M(x,y,z)* priradzuje každému bodu [x, y, z] na povrchu telesa bod [u, v] z definičného priestoru textúry **T**) a mapovanú veličinu (zložením funkcie definície textúry *T(u, v)* a funkcie inverzného mapovania *M(x, y, z)* je textúra aplikovaná na povrch). Pri plochách, ktoré sa nedajú transformovať do roviny (napríklad pri póloch gule) dochádza pri nanášaní textúry ku skresleniu. [25]



Obr. 2.1: Mapovanie 2D textúry na 3D objekt [26]

2.3 Mapovanie závislé na pohľade

Každý povrch telesa odráža svetlo do určitej miery (zrkadlové plochy sú príkladom skoro úplného odrazu). Pre určenie správnych odrazov svetla sa používa textúrovacia technika, nazvaná pohľadovo závislé mapovanie, mapovanie prostredia alebo chromové mapovanie. Odraz od premetu závisí na povrchu telesa, na okolitom prostredí a na polohe pozorovateľa. Analyzované teleso sa virtuálne umiestni do stredu gule alebo kocky, na ktorej vnútorný povrch je premietaná textúra okolia. Hľadaná hodnota sa zistí z bodu textúry, do ktorého dopadne lúč od pozorovateľa, odrazený od sledovaného bodu povrchu. [25]

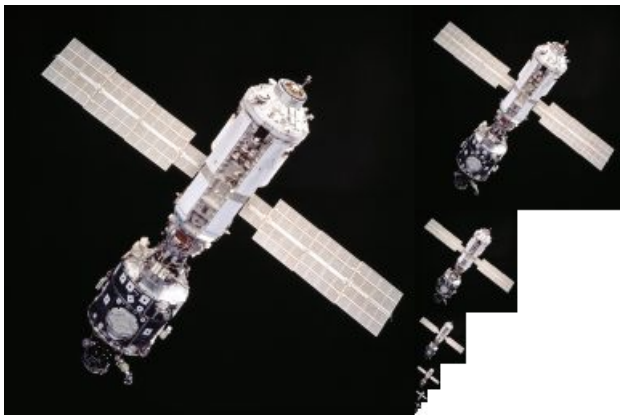
2.4 Vrásnenie povrchu

Zvrásnenie povrchu telesa môžeme dosiahnuť optickým klamom pomocou aplikovania bump texture (textúra hrboľatosti). Pomocou tohto prístupu dokážeme vytvoriť dojem zvrásneného povrchu, bez toho aby sme museli zmeniť geometriu telesa. Každému bodu telesa sa priradí hodnota textúry a od týchto hodnôt sa odvodia modifikované normálové vektory na povrchu a tým sa zmení pôvodný smer odrazu svetla. Vznikne dojem zvrásneného, resp. hrboľatého povrchu. Pri tejto metóde ale treba dávať pozor na to, že zvrásnený povrch v reáli tieni sám seba a to táto metóda neumožňuje. Preto je túto metódu vhodné používať len pre mierne zvrásnenia. Taktiež tieň vrhaný telesom nie je ovplyvňovaný nanesenou textúrou a tým pádom môže pôsobiť neprirodzene. [25]

2.5 Ukladanie textúr v pamäti počítača

2.5.1 Mipmapping

Textúru potrebujeme nanášať v rôznych mierkach. Ak by tomu tak nebolo, tak by pri každom nanášaní textúry bolo potrebné danú textúru prevzorkovávať, interpolovať a filtrovať, čo by bolo náročné na výpočet. Pre zníženie výpočtovej náročnosti je textúra na začiatku prepočítaná pre niekoľko mierok. Technika MIP (multum in pravo, veľa v malom) je spôsob uloženia textúry v rôznych mierkach súčasne (najbližšia mierka je polovica predchádzajúcej až do veľkosti jedného pixelu). Pri mapovaní sa vyberie najbližšia vyššia mierka a pomocou lineárnej interpolácie sa dosiahne potrebná veľkosť. Táto technika zvyšuje pamäťovú náročnosť na úkor zníženia výpočtovej náročnosti. Pri malých mierkach je nevýhodou tejto techniky rozmazanie textúry. [25]



Obr. 2.2: Príklad mipmappingu - hlavný obrázok a kópie s menšími mierkami [27]

2.5.2 Kompresia rastrových textúr

Textúry je pre náročnosť na pamäť často nutné komprimovať. Metódy komprimácie ako JPEG sa nedajú použiť kvôli nedostatočnej rýchlosti a preto sa používajú špeciálne kompresie ako S3TC a 3DFX Interactive. [25]

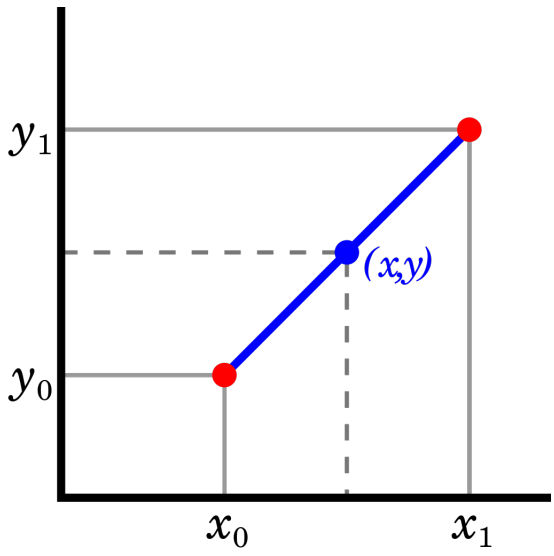
2.6 Priestorové textúry

Textúrová funkcia priestrorovej textúry *T(u, v, w)* je funkcia troch priestorových súradníc [u, v, w]. Inverzné mapovanie *M(x, y, z)* prevádza priestorové súradnice [x, y ,z] textúrovaného bodu na súradnice textúry [u, v, w]. Definícia takejto textúry

je možná pomocou trojrozmernej tabuľky, čo ale vytvára nároky na rýchlosť spracovania a prehliadania. Existuje aj možnosť generovať textúru pomocou algoritmu alebo procedúry a hodnota textúry je tak získaná výpočtom funkcie. Pri textúrach s nepravidelnými vzormi ako drevo, kameň sa využívajú fraktály alebo šumové funkcie. Jedným z použiteľných šumov je biely šum, kedže má vyrovnané kmitočtové spektrum, obsahuje detaily na rôznych úrovňach vďaka možnosti zväčšovania a zmenšovania. Hrozil by však vznik aliasingu, vyžadovaný je veľký výpočetný výkon pri vykresľovaní a realizácia by bola neopakovateľná. Využívajú sa Perlinove šumové funkcie, ktoré majú predom známi rozsah hodnôt, sú štatisticky invarianté vzhľadom k posunutiu a otáčaniu, majú obmedzené kmitočtové spektrum a sú opakovateľné.

2.7 Interpolácia textúr

Často je potrebné použiť textúru v určitom rozlíšení, ktorá v tu chvíľu nie je v pamäti ani na úložisku k dispozícií, preto sa na zistenie vzoriek používa interpolácia. Ak je daná jednorozmerná textúra tvoriaca diskrétny signál a jeho hodnoty pomznáme v celočíselných súradniciach *f[i], i* = 1,...,N , je potrebné odhadnúť hodnotu *f* v neceločíselnom bode a využije sa preloženie diskrétneho signálu spojitým signálom a odoberie sa vzorka na požadovanom mieste.[25]



Obr. 2.3: Princíp interpolácie [28]

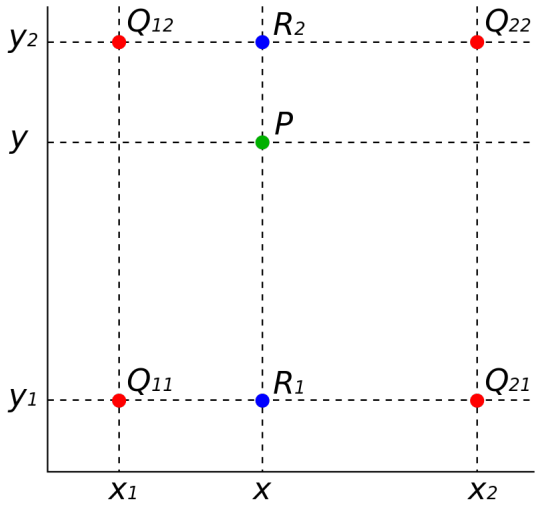
2.7.1 Lineárna interpolácia

Kompromisom medzi rýchlosťou a kvalitou v grafických aplikáciách je lineárna metóda interpolácie. Princíp spočíva v tom, že je signál prekladaný úsečkami, ktoré spájajú vždy dva najbližšie body *f[i]* a *f[i+1]*, $i = 1,...,N-1$. Ak je požadovaná poloha nového bodu medzi týmito bodmi, tak sa hodnota vypočíta ako konvexná kombinácia dvoch susedných hodnôt.

$$f(s) = (s - i) * f[i] + (i + 1 - s) * f[i + 1]$$

Obr. 2.4: Konvexná kombinácia dvoch susedných hodnôt [25]

V prípade 2D sa hovorí o bilineárnej interpolácií a v prípade 3D sa hovorí o trilineárnej interpolácií. Princíp ostáva ten istý ale pracuje sa vo dvoch (2D) a troch (3D) dimenziách. Čím vyššiu kvalitu odhadu potrebujeme, tým sú vyššie nároky na výpočtový výkon. Pri bilineárnej interpolácií sa najprv vypočíta interpolácia v jdenom smere a potom v druhom. Každý krok je lineárny, no bilineárna interpolácia ako celok je skôr kvadratického charakteru. [25]



Obr. 2.5: Princíp bilineárnej interpolácie [29]

Na obr.2.5 môžeme vidieť príklad bilineárnej interpolácie, kde body Q sú vzorky uložené v pamäti, body R sú vypočítane v prvom kroku interpolácie a následne získaný hľadaný bod P v druhom kroku interpolácie.

3 Tvorba scény v Unity

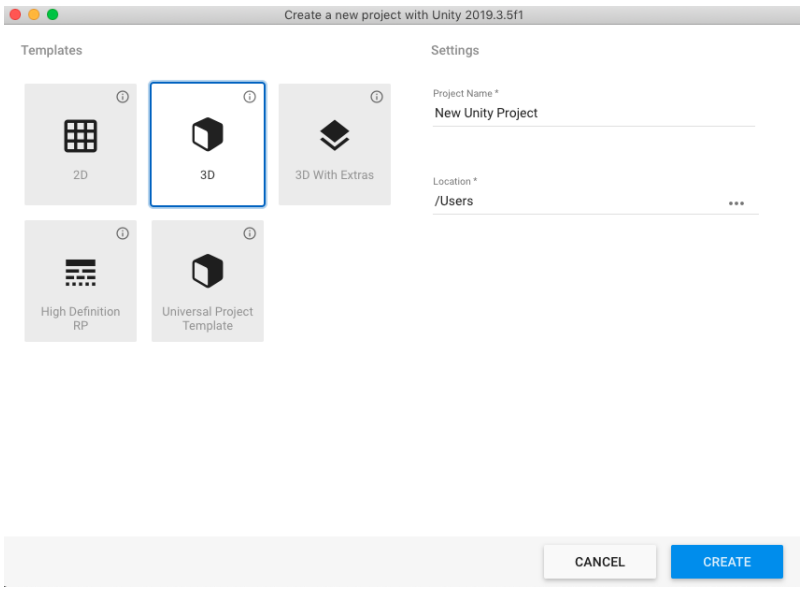
Pre tvorbu nášej práce sme vybrali z preskúmaných enginov Unity vo verzií 2019.3.14f1 pre macOS. Tento engine ponúka všetky nástroje ktoré potrebujeme, je licenčne vyhovujúci, ponúka riešenie pre streamovanie a tvorbu skriptov C#.

3.1 Založenie projektu

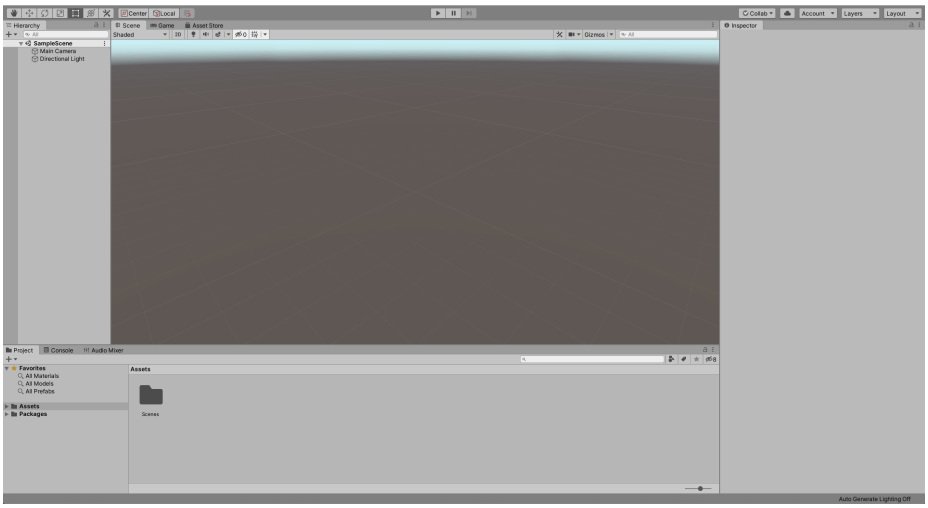
Pri vytváraní projektu nám Unity dáva na výber z piatich šablón:

- **2D** - šablóna obsahujúca prázdny projekt pre tvorbu 2D aplikácií, využívajúca vstavaný Unity rendrovač.
- **3D** - šablóna obsahujúca prázdny projekt pre tvorbu 3D aplikácií, využívajúca vstavaný Unity rendrovač.
- **3D With Extras** - šablóna obsahujúca nový post-processing so vstavaným Unity rendovačom a rôznymi presetmi a vzorovými príkladmi.
- **High Definition RP** - šablóna využívajúca High Definition Render Pipeline, zameraná na high-end grafiku a hry pre platformy podporujúce Shader Model 5.0 (DX11 a vyššie)
- **Universal Project Template** - šablóna obsahujúca Universal Render Pipeline a Shader Graph, ktoré sú určené pre tvorbu graficky optimalizovaných wide-platform aplikácií.

Z týchto možností sme vybrali možnosť 3D - prázdny 3D projekt.



Obr. 3.1: Okno s nastaveniami nového projektu



Obr. 3.2: Užívateľské prostredie Unity s prázdnym projektom

Na obrázku č.3.2 môžeme vidieť pracovné prostredie a jednotlivé panely enginu. V hornej časti sa nachádza panel s transformačnými nástrojmi ako *Move Tool, Scale Tool, Rotate Tool* a v strede obrazovky nástroje *Play, Pause* a *Step*. Panel *Hierarchy* slúži ako zoznam všetkých GameObjectov[1] v scéne, panel *Project* slúži ako prehliadač súborov projektu a panel *Inspector* slúži na zobrazenie komponentov vybraného GameObjectu.

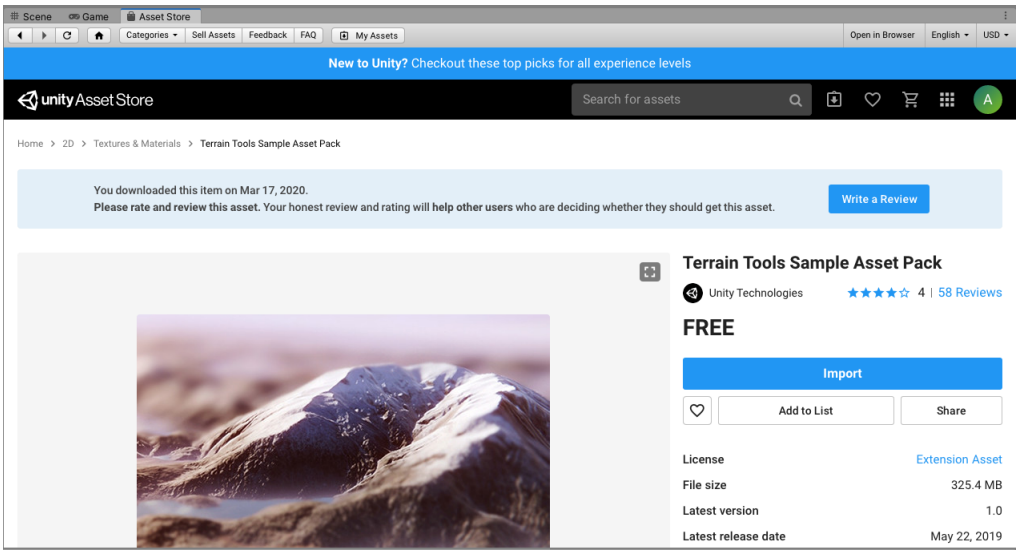
3.2 Assets

Asset Store je knižnica zdarma a platených doplnkov vytvorených priamo Unity Technologies ale aj širokou verejnosťou. Asset Store je prístupný priamo z Enginu alebo prostredníctvom webstránky, kde sa dá jednoducho stiahnuť a importovať požadovaný asset. Pre naše potreby sme stiahli a nainštalovali assety pre tvorbu prvkov terrain *(Terrain Tools Sample Pack)*, skybox *(Starfield Skybox)* a foliage (*World Space Trees)*. Po importovaní sú assety prístupné v Unity Editore.

3.3 Terrain

Na vytvorenie plochy scény slúži v Unity Engine prvok *Terrain*. Pomocou Asset Store sme stiahli a aktivovali *Terrain Tools Sample Asset Pack*, ktorý obsahuje nástroje pre prácu s terénami v našom projekte. Vytvorili sme tak nový *GameObject* typu *Terrain (GameObject/3D Object/Terrain*).

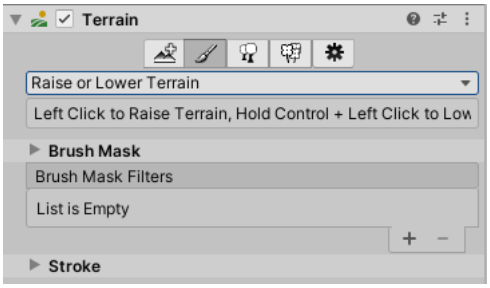
[1]Základné objekty v Unity, ktoré reprezentujú prvky scény a obsahujú komponenty, ktoré majú určitú funkcinalitu



Obr. 3.3: Okno Asset Store s Terrain Sample Pack

3.3.1 Modelovanie plochy

V záložke Inspector sa nachádza zoznam a nastavenia všetkých komponentov pripojených k vybranému GameObjectu a pre modifikáciu plochy Terrain slúži komponent Terrain, kde nájdeme nástroje Create *Neighbor Terrain, Paint Terrain, Paint Trees, Paint Details, Terrain Settings*. Pre modelovanie výšky terénu slúži nástroj *Paint Terrain* v nastavení *Raise Or Lower Terrain*. V *Brush Mask* sa nastavuje typ použitého štetca na modelovanie a v *Stroke* veľkosť a vlastnosti štetca. Kedže sa má scéna nachádzať vo vesmíre, nastavili a aplikovali sme rôzne typy štetcov a ich vlastností a vytvorili skalnatý terén, ktorý tvorí povrch pripomínajúci Asteroid.

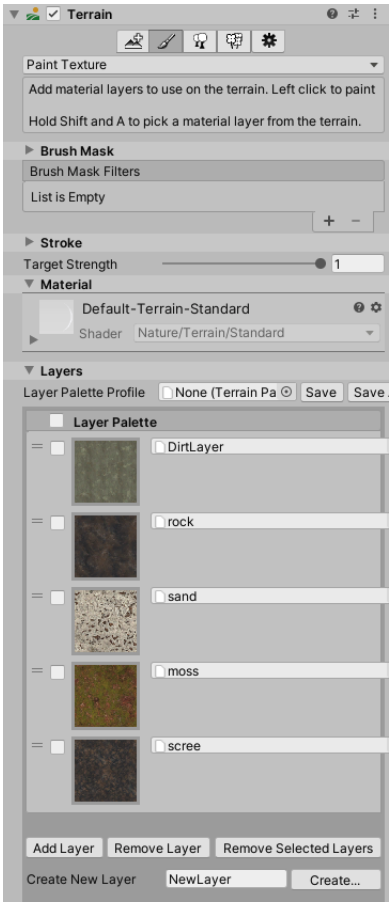


Obr. 3.4: Okno nastavení komponentu Terrain

3.3.2 Nanášanie textúr

Pre nanášanie textúr slúži v *Paint Terrain* nastavenie *Paint Texture*. Pri tomto nastavení sa do ponuky okrem *Brush Mask* a *Stroke pridá* aj *Material* a *Layer* . *Ma-*

terial sme ponechali nastavený na defaultný *Default-Terrain-Standard*, ktorý slúži k tvorbe prírodných scén. V zložke *Layer* sme vytvorili paletu textúr z importovaného assetu a to hlavne textúry *rock* a *sand*, ktoré tvoria najväčšiu časť povrchu a medzistupeň medzi nimi tovria *DirtLayer*, *moss* a *scree*.



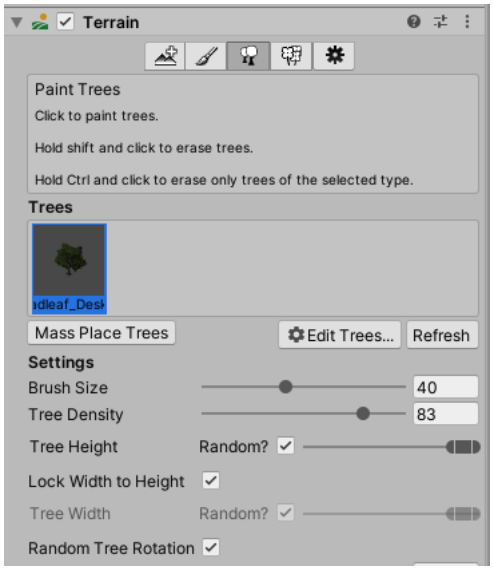
Obr. 3.5: Okno nastavení komponentu Terrain pre Paint Texture

3.3.3 Pridávanie prkov foliage

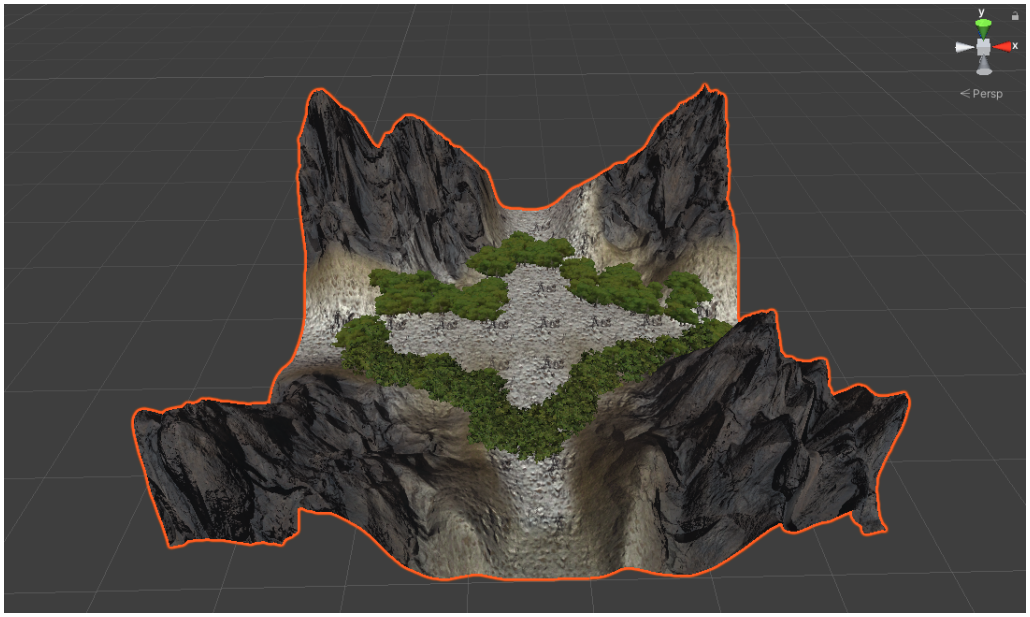
Pre pridávanie prvkov foliage slúži nástroj *Paint Trees*, resp. *Paint Details*. V tomto nástroji dokážeme pridávať prvky pomocou Edit Trees, kde môžeme pripájať Prefaby[2] rôznych prvkov. Takýmto spôsobom sme pridali Prefab *Broadleaf_Desktop*, ktorý je šablonou pre model stromu z importovaného assetu. V nastaveniach tohto prvku sme nastavili rozmedzie pre veľkosť pridávaných modelov, aby odpovedala našej scéne a náhodnosť rotácie a výšky kvôli rôznorodosti jednotlivých stromov. Tento

[2]Vzorový GameObject daného prvku, ktorý slúži ako šablona

prvok sme následne pomocou štetca naniesli na požadované miesta. Obdobným spôsobom pridáme v nástroji *Paint Details* textúru trávy *GrassFrond02AlbedoAlpha*, a pomocou štetca nanesieme prvky na terén.



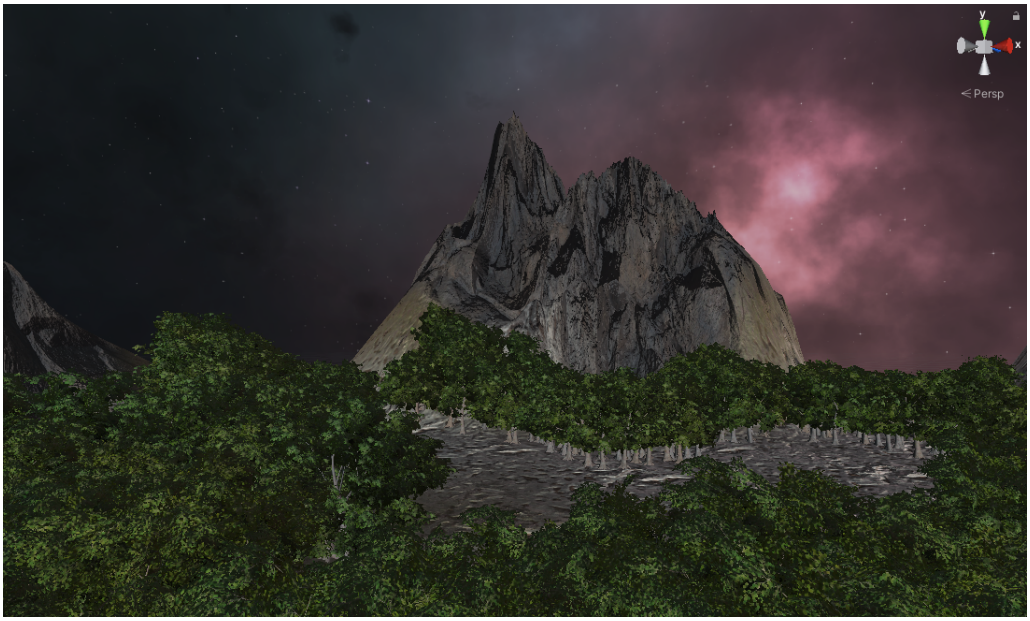
Obr. 3.6: Okno nastavení komponentu Terrain pre Paint Trees



Obr. 3.7: Terén s nanesenými textúrami a foliage

3.4 Skybox

Prvok Skybox slúži na vytvorenie pozadia scény za účelom spraviť scénu väčšou než v skutočnosti je. Textúry nanášané na tento prvok sú zvyčajne textúry oblohy s okolitým prostredím a pod. V Unity sa textúra tohto prvku nastavuje v menu *Lightning* v položke *Enviroment.*V našom prípade by sa textúra skyboxu mala približovať voľnému vesmíru, a tak pomocou naimportovaného assetu *Starfield Skybox* priradíme textúru skyboxu *Skybox* do poľa *Skybox Material*. Textúra je následne rendrovaná na pozadie scény.



Obr. 3.8: Vložený Skybox

3.5 Svetelné zdroje a Fog

3.5.1 Light

Prvky typu *Light* slúžia ako svetelné zdroje scény. Engine nám ponúka tieto *GameObjecty* typu *Light*:

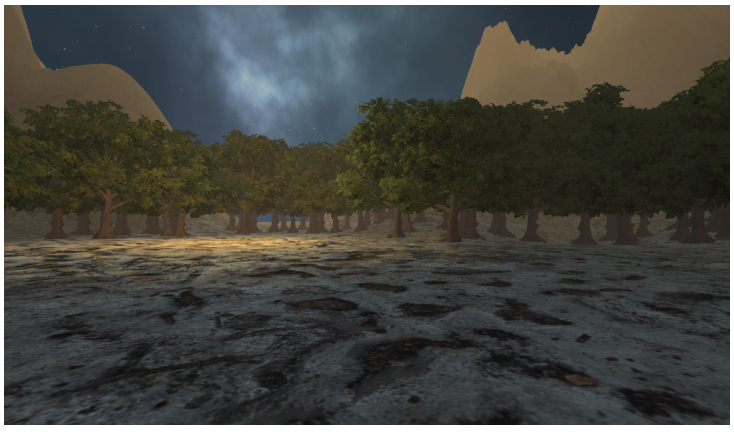
- **Directional Light** - Vyžaruje svetlo ako vzdialený zdroj svetla (rovinná vlna),vhodný na simuláciu svetelného žiarenia slnka.
- **Point Light** - Vyžaruje svetlo do všetkých strán rovnako (guľová vlna), vhodné na simuláciu lámp a iných lokálnych zdrojov svetla.
- **Spotlight** - Vyžaruje svetlo iba pod zadaným uhlom, vhodné na simuláciu svietidiel, svetiel na autách, divadelných svetiel a pod.

- **Area Light** - Je definovaný pomocou obdlžníka, pod ktorým je svetlo vyžarované uniformne iba z jednej strany, vhodné na simolovanie nasvietenia ulice, vnútorné osvetlenie domu a pod.
- **Reflection Probe** - Slúži ako sledovač prostredia, ktorý vytvorý Cubemap[3], ktorý môže byť použitý objektami s reflektivnými vlastnosťami.
- **Light Probe Group** - Slúžia na vytvorenie informácií o prechádzajúcom svetle cez prostredie, využitie na definovanie vysko kvalitného nasvietenia.

Do scény sme tak pridali *Area Light*, kedže sa scéna nachádza vo vsemíre a v blízkosti nieje žiadny zdroj pre typ *Diretional Light*. Tvar sme zmenili na *Disc*, *Range* nastavili na 10, *Radius* na 2000, intenzitu zdroja *Intensity* na 0.3, kedže chceme iba mierne podsvietené prostredie a pomocou transformačných nástrojov sme zdroj umiestnili na požadované miesto nad terénom v dostatočnej vzdialenosti pre požadovaný efekt. Ako ďalšie sme vytvorili 4 svetelné zdroje typu *Point Light*, ktoré budú slúžiť ako bodové osvetlenie jednotlivých stanovíšť výstavy. Na jednotlivých komponentoch sme nastavili *Range* na 63, *Intensity* na 10 a *Color* sme zmenili na svetlo-oranžovú (FFBE34). Pomocou transformačných nástrojov sme zdroje umiestnili na požadované miesta nad jednotlivými stanovišťami. Zmeny v nasvietení je potrebné vygenerovať (Bake) v záložke *Lightning/Generate Lightning*.

3.5.2 Fog

Pre pridanie hmly do scény slúži prvok Fog. Tento prvok nájdeme v Unity v záložke *Lightning* . Aktivovali sme tento prvok, nastavili jeho farbu *Color* na hnedú (AD9569) a *Density* znížili na 0.0025.



Obr. 3.9: Nasvietená scéna s vloženým prvkom Fog

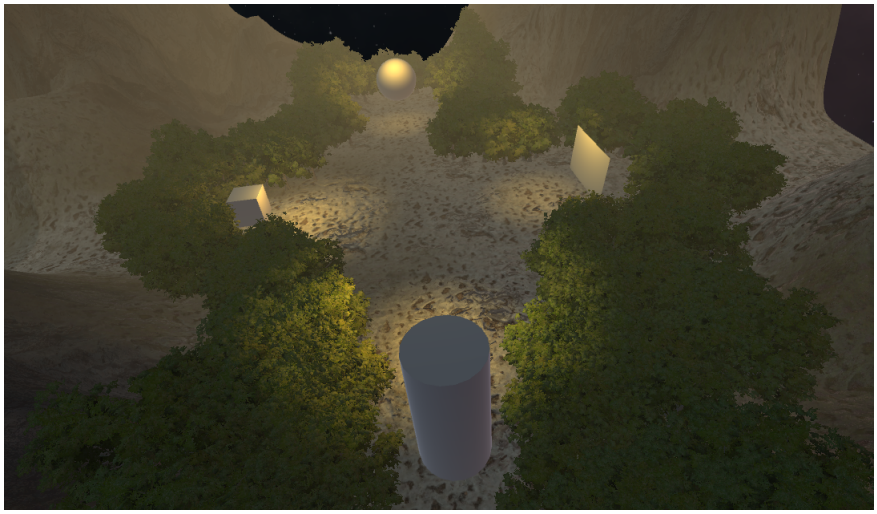
[3]6 štvorcových textúr, ktoré tvoria imaginárnu kocku okolo objektu

3.6 Výstavné telesá

Vo virtuálnej výstave sa budú nachádzať rôzne telesá, na ktoré budú premietané videá. Pre našu scénu sme vytvorili 4 stanovištia s takými telesami. Pri každom telese sa bude nachádzať text s názvom premietaného diela.

3.6.1 Vloženie telies

Pre vloženie rôznych telies môžme použiť vstavané *GameObjecty*. Unity ponúka možnosť vložiť základné 3D telesá z ktorých sme si pre našu scénu vybrali kocku, guľu, valec a plátno. Tieto telesá sme do scény vložili pomocou vytvorenia nových *GameObjectov (GameObject/3D Object/Cube, Sphere, Cylinder, Plane).*



Obr. 3.10: Scéna s vloženými výstavnými telesami

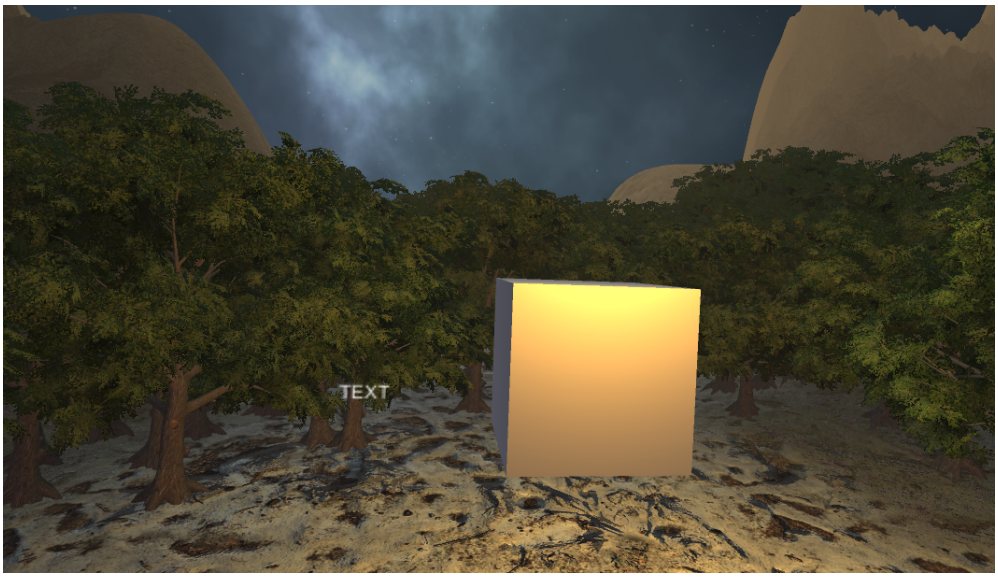
3.6.2 Text

Pre vloženie textu k vytvoreným telesám môžeme použiť:

- **3D text** - Text sa nachádza priamo v priestore scény ako 3D objekt *(GameObject/3D Object/Text - TextMeshPro).*
- **2D text** - Text je súčasť UI[4] a HUD[5] a nachádza sa tak na obrazovke ako 2D text *(GameObject/UI/Text).*

Kedže chceme text využiť ako štítok pri výstavnom telese, využijeme možnosť 3D textu v priestore a vložíme tieto objekty k jednotlivým telesám pomocou transformačných nástrojov a nastavíme veľkosť textu *Font Size* na 10.

[4]User Interface - užívateľské prostredie, komunikácia medzi hráčom a hrou
[5]Head-Up Display - 2D objekty na obrazovke (status bar, health bar, timer a pod.)



Obr. 3.11: Scéna s vloženým textom pri telesách

3.7 Audio

Pre spracovanie audio súborov poskytuje Unity 2 typy *GameObjectov*:

- **Audio Source** - vytvorí prvok scény, ktorý prehráva audio súbor.
- **Audio Reverb Zone** - Vytvorí prvok scény s nastaviteľným efektom *Reverb.*

Ako statický zvukový prvok sme vybrali zvukový klip *"cicada.mp3"*ktorý obsahuje zvuk cikád. Vytvorili sme tak objekt *Audio Source* a pomocou transformačných nástrojov sme ho umiestnili medzi stromy. V komponente *Audio Source* sme priradili zvukový klip k položke *AudioClip,*zaškrtli *Loop*[6] *Spatial Blend* nastavili na 3D, čo spôsobí, že sa zvukový klip bude chovať ako zvukový zdroj v priestore. Nastavili sme preto *Logarithmic Rolloff* a *Min Distance* na 10 a *Max Distance* na 200, čo nastaví pokles hlasitosti v závislosti so vzdialenosťou. Možnosť *Spread* sme dvihli na 100, čo spôsobí, že sa zvuk v priestore bude šíriť lepšie. Tento *GameObject* sme skopírovali a vytvorili tak 4 miesta na scéne, kde je možné počuť zvuk cikád.

[6]Klip sa po skončení začne znovu prehrávať

3.8 Vytvorenie prvku hráča

Aby sme sa v scéne mohli pohybovať, porebujeme vytvoriť prvok hráča. V našom prípade sa jedná o hráča typu FP[7], ktorý sa pohybuje po scéne pomocou klávesnice a myši. Ako prvé sme vytvorili hlavný *GameObject* typu *Empty Object*s názvom *FP*, ktorý slúži ako hlavný objekt hráča, a priradili mu komponent *CharacterController*. V tomto objekte sme vytvorili *GameObject* typu *Main Camera*, ktorý rendruje obraz pri hraní. Následne sme pridali *Cube* ktorý slúži ako *Collider*[8] s prostredím. Ako posledný sme pridali objekt *Ground Check,* ktorý zisťuje, či sa hráč nachádza na zemi, čo je dôležité pre správne nastavenie gravitácie. Pre tieto prvky sme následne vytvárali C# skripty, ktoré nastavujú základný pohyb po mape, rotáciu kamerou a vplyv gravitačnej sily.

3.8.1 Pohyb po scéne

Pre pohyb po mape sme vytvorili skript s názvom *movement.cs*. Tento skript sme priradili ako komponent pre vytvorený *GameObject FP*. Novovytvorené skripty v Unity obsahujú definíciu triedy s názvom súboru a nachádzajú sa v nej dve funkcie typu *void* (funkcia nevracajúca hodnotu):

- **Start** - Jedná sa o funkciu, ktorá je volaná vždy pri zapnutí hry.
- **Update** - Pri nastavení triedy na *MonoBehaviour* je táto funkcia volaná pri každom rámci, preto je vhodná pre sledovanie zmien scény.

Pre vytvorenie pohybu si tak vystačíme s funkciou *void Update()* v ktorej budeme zisťovať vstup z klávesnice, podľa ktorého bude vykonaný pohyb.

Výpis 3.1: Skript movement.cs - vytovrené premenné

```
1  public class movement : MonoBehaviour {
2      public CharacterController controller;
3      public float speed;
4      public float gravity = -9.81f;
5      public Transform groundCheck;
6      public float groundDistance = 0.4f;
7      public LayerMask groundMask;
8      Vector3 velocity;
9      bool isGrounded;
10 ...
```

[7]First Person - pohľad z prvej osoby

[8]prvok, ktorý zisťuje kolíziu objektu s inými objektami

Najprv sme definovali premenné potrebné pre chod triedy. Premenná *controller* je určená pre vytvorený *GameObject FP* a slúži pre vstavané funkcie pre pohyb a kolízie. Peremenná *speed* slúži na nastavenie rýchlosti pohybu hráča po scéne, *gravity* nastavuje veľkosť gravitačného zrýchlenia pri páde, groundDistace nastavuje hranicu pri ktorej nastane kolízia medzi hráčom a zemou (potrebné pre vynulovanie gravitačného zrýchlenia). *Ground Mask* slúži na priradenie vrstvy terénu (v našom prípade vrstva *Ground*), *groundCheck* je premenná pre vytvorený *GameObject GroundCheck*, *isGrounded* slúži na uloženie stavu kolízie s terénom (pre gravitáciu) a trojrozmerný vektor *velocity* určuje výsledný pohyb.

Výpis 3.2: Skript movement.cs - funkcia Update()

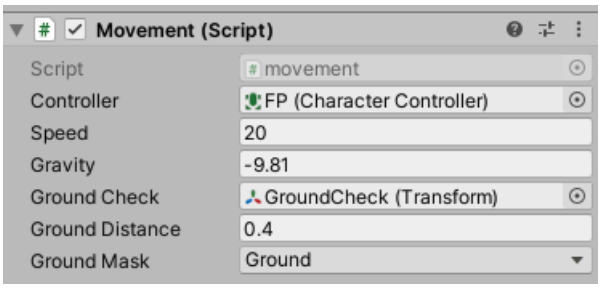
```
1  ...
2      void Update () {
3      isGrounded=Physics.CheckSphere(groundCheck.position,
4              groundDistance, groundMask);
5      if (isGrounded && velocity.y < 0) {
6          velocity.y = -2f;
7      }
8      float x=Input.GetAxis("Horizontal");
9      float z=Input.GetAxis("Vertical");
10     Vector3 move = transform.right*x+transform.forward*z;
11     controller.Move(move*speed*Time.deltaTime);
12     velocity.y +=gravity*Time.deltaTime;
13     controller.Move(velocity*Time.deltaTime);
14
15     }
16 }
```

Skript v prvom kroku zistí stav premennej *isGrounded*, a následne pomocou podmieky **if** pre stav dopadnutia na terén nastaví rýchlosť pohybu v smere osy y. Premenné x a z sú definované pomocou *Input.GetAxis*[9] a slúžia ako vstup dát o stlačení pohybových kláves. Následne pohybový trojrozmerný vektor *move* nastaví pomocou transformačnej funkcie *transform* smer pohybu podľa vstupných premnných x a z. Následne *controller* pohne s telesom za použitia vstupnej premennej *speed*, vektora *move* a časového úseku *Time.DeltaTime*[10]. Premenná *velocity* nastaví v jej vertikálnom komponente y gravitačný pohyb, ktorý v ďalšom kroku po vynásobení s

[9]Unity obsahuje vlastný input systém prístupný pomocou Edit/Project Settings/Input

[10]Čas uplynutý od posledného rámca

časovým úsekom vytvára gravitačné zrýchlen zvislej osy. Nakoniec sa *controller* pohne pomocou premennej *velocity* vynásobenej s časovým úsekom *Time.DeltaTime* , čo vytvára požadovaný pohyb v čase. Po priradení daných *GameObjectov* k vytvoreným premenným sme sa po spustení scény dokázali pohybovať pomocou kláves W,A,S,D po scéne.



Obr. 3.12: Nastavenie komponentu skritu movement.cs s priradenými objektami

3.8.2 Rotácia kamery

Pre vytvorenie rotácie kamery pomocou myši sme obdobným spôsobom vytvorili skript *mouse.cs,* ktorý sme pridali ako komponent objektu *Main Camera*.

Výpis 3.3: Skript mouse.cs - vytvorené premenné a funkcia Start()

```
1  public class mouse : MonoBehaviour {
2
3      public float mouseSensitivity = 100f;
4      public Transform playerBody;
5      float xRotation = 0f;
6
7      void Start () {
8          Cursor.lockState = CursorLockMode.Locked;
9      }
10 ...
```

Ako prvé sme definovali premennú *mouseSensitivity* ktorá určuje rýchlosť rotácie, *playerBody* je premenná typu *Transform* a slúži pre priradenie objektu hráča a manipuláciu s ním, premenná *xRotation* slúži pre rotáciu v smere osy y (rotácia okolo osy x). Tentokrát sme použili funkciu *Start()* a pomocou *Cursor.lockState* sme uzamkli kurzor v strede obrazovky, kedže vytvárame FP pohyb.

Výpis 3.4: Skript mouse.cs - funkcia Update()

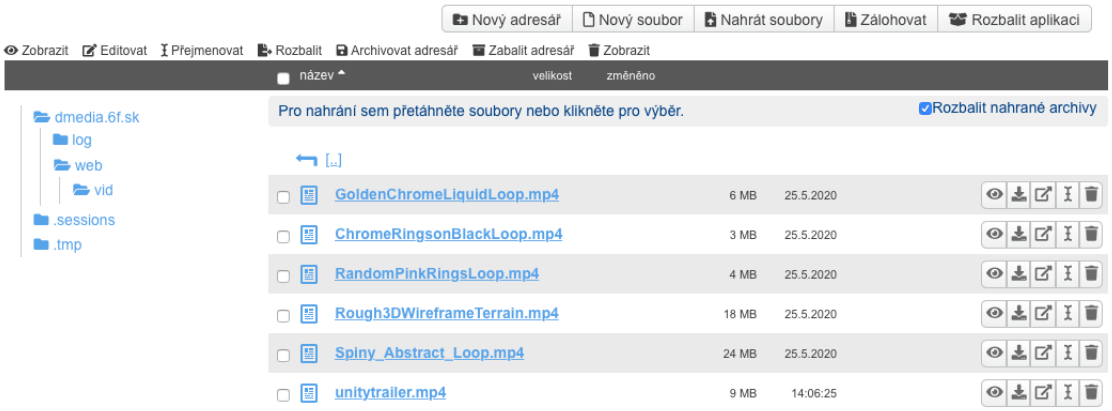
```
...
    void Update () {
        float mouseX = Input.GetAxis ("Mouse␣X") *
                       mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis ("Mouse␣Y") *
                       mouseSensitivity * Time.deltaTime;
        xRotation -= mouseY;
        xRotation = Mathf.Clamp (xRotation, -90f, 90f);
        transform.localRotation = Quaternion.Euler(xRotation,
                                  0f, 0f);
        playerBody.Rotate (Vector3.up * mouseX);

    }
}
```

Funkcia *Update()* v prvom kroku vytvorí premenné *mouseX* a *mouseY*, ktoré zisťujú vstup z myši, ktorý je následne vynásobený senzitivitou *mouseSensitivity* a časovým úsekom *Time.DeltaTime*. V ďalšom kroku *xRotation* odčíta hodnotu *mouseY* (pri pričítaní nastáva inverzia pohybu) a nastaví ohraničenie pohybu po ose y na -90 až 90 stupňov. Ďalej pomocou lokálnej transformačnej funkcie *transform.localRotation* sa nastaví rotácia pre os y. Nakoniec sa rotuje *playerBody* pomocou *mouseX* a *Vector3.up* (reprezentujúci os y,kedže pohybom v smere osy x rotujeme okolo osy y). Priradili sme objekty k nastaveným premenným a pohyb hráča tak funguje v každom smere aj s rotáciou kamery.

4 Streamovanie videa na teleso

Na jednotlivé výstavné telesá budú premietané audiovizuálne diela zo širokého výberu (jedná sa teda o veľkú databázu diel) a preto je potrebné, aby diela neboli súčasťou balíka a aby aplikácia obsahovala iba spôsob na získanie týchto videí z externého úložiska. Najvýhodnejšou metódou je preto streamovanie[1] diel zo servera priamo na výstavné telesá v Unity. Vytvorili sme preto webhosting s doménou *dmedia.6f.sk* na portály *endora.cz* a využili ho ako náš server pre ukladanie jednotlivých videí. V našom adresáry sme vytvorili zložku *vid,* a nahrali sme do nej abstraktné testovacie videá *"GoldenChromeLiquidLoop.mp4", "ChromeRingsonBlackLoop.mp4", "RandomPinkRingsLoop.mp4", "Rough3DWireframeTerrain.mp4", "Spiny_Abstract_Loop.mp4"*a upútavku na Unity Engine 5 *"unitytrailer.mp4"*obsahujúcu zvukovú stopu.



Obr. 4.1: Správca súborov webhostingu

4.1 Textúrovanie videa

Pre prácu s videom slúži v Unity Engine *Game Object* typu *Video Player*. Video je premietané na pozadie kamery alebo na nastaviteľný *RenderTexture*.Pre tento komponent exituje možnosť zadať URL link na ktorom sa požadované video nachádza a prehrávať ho tak priamo zo servera.

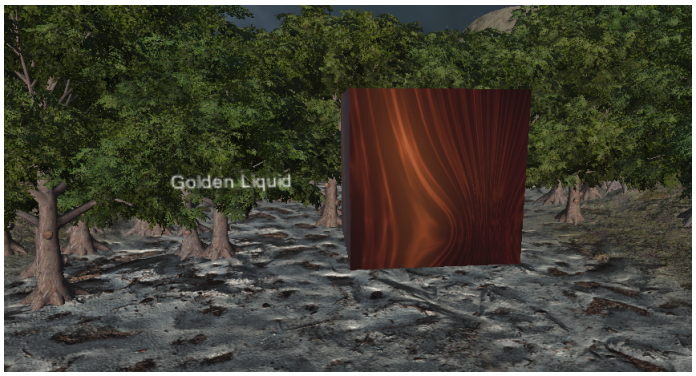
Pre pridávanie videí na výstavné telesá vytvoríme skript *addvideo.cs* ktorý bude priraďovať každému výstavnému telesu *VideoPlayer* a nastaví URL s videom.

[1]kontinuálny prenos audiovizuálneho materiálu medzi zdrojom a koncovým používateľom

Výpis 4.1: Skript addvideo.cs - vytvorenie prvku VideoPlayer

```
1  GameObject V1;
2  GameObject V2;
3  GameObject V3;
4  GameObject V4;
5  void Start () {
6      V1 = GameObject.Find ("V1");
7      var videoPlayer1 =
8          V1.AddComponent<UnityEngine.Video.VideoPlayer>();
9      videoPlayer1.url =
10         "http://dmedia.6f.sk/vid/GoldenChromeLiquidLoop.mp4";
11     videoPlayer1.isLooping = true;
12     var title = text1.GetComponent<TextMesh>();
13     title.text = "Golden␣Liquid";
14     }
15 ...
```

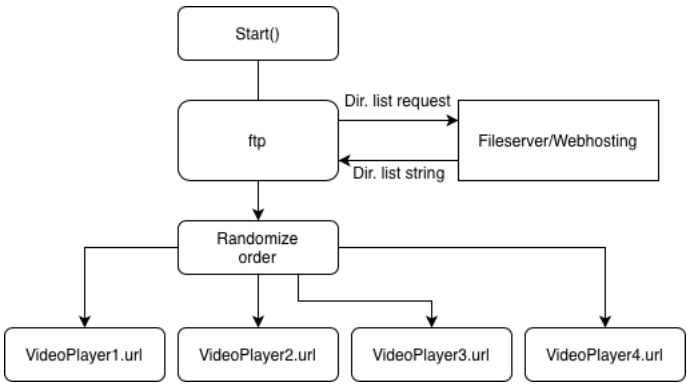
Najprv sme v skripte vytvorili 4 *GameObjecty* ku ktorým budú priradené už vytvorené telesá. Vo funkcií *Start()* sme pomocou *GameObject.Find* našli komponent V1 (prvé teleso) a priradili ho k premennej. Následne sme vytvorili lokálnu premennú *videoPlayer1* , ktorá vytvorí komponent typtu *VideoPlayer* pre *GameObject V1*. Pomocou *videoPlayer1.url* sme nastavili URL adresu požadovaného videa *GoldenChromeLiquidLoop*a *videoPlayer1.isLooping* nastaví prehrávanie na loop. Nakoniec sme vytvorili premennú *title* ktorá získa komponent *TextMesh* vytvoreného textového objektu *text1* a nastaví jeho obsah na názov prehrávaného videa. Rovnakým spôsobom sme priradili videá zvišným telesám (V2, V3, V4). Toto nastavenie je však statické a nedokážeme využiť celú databázu diel, iba 4 staticky nastavené a pri zmene na serveri nastane chyba.



Obr. 4.2: Priradený VideoPlayer k telesu

4.2 Využitie FTP

Pre dynamicky meniace sa videá potrebujeme vytvoriť spojenie so serverom, ktoré bude nastavovať videá podľa aktuále uložených videí na serveri. Jedným z možných riešení je vytvoriť spojenie so serverom pomocou FTP[2] protokolu a pomocou neho zisťovať aktuálny obsah zložky */vid/*.Pre pripojenie pomocou FTP je však potrebné vytvoriť novú triedu *ftp* a pripojiť sa pomocou nej na server.



Obr. 4.3: Bloková schéma algoritmu priraďovania videí pomocou FTP

Výpis 4.2: Skript addvideo.cs - trieda ftp

```
1  using System.Net;
2  class ftp {
3      private string host = null;
4      private string user = null;
5      private string pass = null;
6      private FtpWebRequest ftpReq = null;
7      private FtpWebResponse ftpResp = null;
8      private Stream ftpStream = null;
9      public ftp (string ip, string username, string password)
10     { host = ip; user = username; pass = password; }
11 ...
```

Vytvorili sme triedu *ftp* a definovali sme premenné *host, user* a *pass* ktoré slúžia na uloženie prihlasovacích údajov k FTP účtu na serveri. Využili sme *FtpWebRequest* z balíčka *System.Net* a vytvorili premenné pre odosielanie požiadavky na server *ftpReq* a pre získanie odpovede *ftpResp*a premennú *ftpStream* na vytvorenie odosielaných a prijímaných sekvencií. Definovali sme argumenty *ftp* ako premenné typu *string* s informáciou o IP adrese servera a prihlasovacích údajoch.

[2]File Transfer Protocol - používaný na výmenu súborov medzi klientom a serverom

Výpis 4.3: Skript addvideo.cs - trieda ftp

```
...
public string[] dirlist (string directory) {
try {
    ftpReq=
    (FtpWebRequest)FtpWebRequest.Create(host+"/"+directory);
    ftpReq.Credentials = new NetworkCredential (user, pass);
    ftpReq.Method = WebRequestMethods.Ftp.ListDirectory;
    ftpResp = (FtpWebResponse) ftpReq.GetResponse ();
    ftpStream = ftpResp.GetResponseStream ();
    StreamReader ftpRead = new StreamReader (ftpStream);
    string dirraw = null;
    try { while (ftpRead.Peek () != -1)
    { dirraw += ftpRead.ReadLine () + "|"; } }
    catch (Exception ex) {
        Debug.Log (ex.ToString ()); }
    ftpRead.Close ();
    ftpStream.Close ();
    ftpResp.Close ();
    ftpReq = null;
    try { string[] dirList =
        dirraw.Split ("|".ToCharArray ());
        return dirList; }
    catch (Exception ex) {
        Debug.Log (ex.ToString ()); }
    } catch (Exception ex) {
        Debug.Log (ex.ToString ()); }
    return new string[] { "" };
}
}
```

Pre zistenie obsahu zložky sme vytvtorili funkciu *dirlist* so vstupným prametrom typu *string* pre požadovanú cestu. Pomocou *try* skript vykonáva svoj obsah, no ak niekde nastane chyba tak sa spustí *catch* a vypíše chybu do debug logu[3]. Ako prvé skript vytvorí FTP požiadavku, následne sa pomocou *ftp.Credentials* prihlási na server pomocou zadaných údajov. *ftpReq.Method* nastaví typ požiadavky na výpis obsahu zložky *ListDirectory* a následne pomocou *ftpResp* a *ftpStream* vytvorí spätnú komunikáciu so serverom a *ftpRead* túto komunikáciu zachytí. Premená *dirraw* slúži na uloženie získaných informácií a tak pomocou ďalšieho *try* sa volá cyklus *while* pri

[3]Slúži ako konzolový výpis v Unity

ktorom sa k premennej *dirraw* pridávajú názvy súborov oddelené pomocou znaku " | ". Následne sa ukončí FTP spojenie so serverom, vytvorí sa pole *dirList* a pomocou *dirraw.Split* sa získaný *string* rozdelí na pole pomocou deliaceho znaku " | ". Funkcia na záver vracia pole *stringov dirList* kde sú uložené názvy súborov v zložke.

Túto vytvorenú triedu sme následne využili pri priraďovaní premietaných videí na telesá a preto sme modifikovali už vytvorneý skript na priraďovanie.

Výpis 4.4: Skript addvideo.cs - modifikácia funkcie Start()

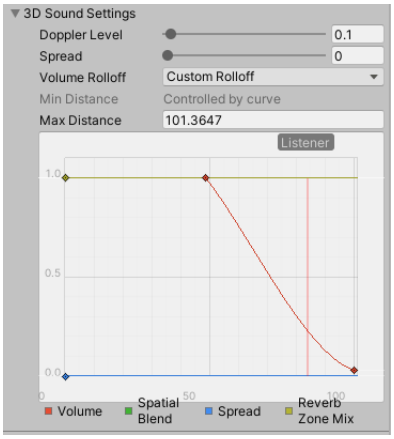
```
1  ...
2  string[] vids = new string[10];
3  int count = 0;
4  void Start () {
5      ftp ftpClient =
6          new ftp (@"ftp://endora.endora.cz",
7                       "unity", "Unity123");
8      string[] dirlisting =
9          ftpClient.dirlist ("./dmedia.6f.sk/web/vid/");
10     foreach (string urlLink in dirlsting) {
11         if (urlLink !="." && urlLink !=".." && urlLink !=""){
12             vids[count] =
13                 "http://dmedia.6f.sk/vid/" + urlLink;
14             count++;
15             }
16         }
17     System.Random rand = new System.Random ();
18     for (int i = 0; i < count - 1; i++) {
19         int j = rand.Next (i, count);
20         string temp = vids[i];
21         vids[i] = vids[j];
22         vids[j] = temp;
23     }
24     V1 = GameObject.Find ("V1");
25     var videoPlayer1 =
26         V1.AddComponent<UnityEngine.Video.VideoPlayer> ();
27     videoPlayer1.url = vids[0];
28 ...
```

Ako prvé sme pridali pole *vids* ktoré slúži na uloženie ciest k videám. Následne sme pridali *ftpClient* ktorý volá vytvorenú tiredu a vytvorí spojenie s FTP serverom pomocou zadaných prihlasovacích údajov. Do vytvoreného poľa *dirlisting* sa uloží

výpis súborov pomocou *ftpClient.dirlist* a následne sa za pomoci cyklu *foreach* zapíše do poľa *vids* požadovaný URL link na súbor. Pomocou *if* sme odstránili výpisy pre zložky ktoré zaberali v poli miesto, kedže sa jedná o súborové odkazy. Pridali sme *System.Random rand* ktorý slúži ako generátor náhodných čísel, ktoré využijeme na znáhodnenie priradených videí pri spustení a tým pádom sa pri každom novom spustení aplikácie priradia na telesá iné videá. cyklus *for* znáhodní obsah poľa *vids* náhodným premiestnením pozícií uložených *stringov* . Nakoniec sme priradili *videoPlayer1.url* k prvku poľa *vids* a rovnaký postup zopakovali pre zvyšné telesá. Po každom novom spustení aplikácie sa tak na telesá premietali iné videá. Využitie FTP je tak jednou z možností dynamického získavania URL odkazov na videá nahraté na serveri.

4.3 Nastavenie priestorového zvuku

Pre vytvorenie priestorového zvuku pri prehrávaní videa, kde sa ako zroj zvuku bude chovať samotné teleso, sme do scény pridali *GameObject* typu *Audio Source* . Tento prvok sme umiestnili k telesu a nastavili jeho parametre *Spatial Blend* na *3D* , čo spôsobí, že sa zdroj chová priestorovo - mení sa panoráma pri rotácií kamery. Pre správne nastavenie poklesu intenzity so vzdialenosťou od objektu sme položku *Volume Rolloff* nastavili na *Custom Rolloff* a krivku poklesu nastavili podľa požadovaných vzdialeností na scéne. Pre miernu simuláciu Dopplerovho javu sme využili možnosť *Doppler Level* a túto hodnotu nastavili na 0,1. Tento prvok sme rovnakým spôsobom vytvorili pre všetky telesá.



Obr. 4.4: Nastavenie 3D Sound Settings komponentu Audio Source

Pre priradenie tohto zdroja k prehrávanému videu sme modifikovali skript *addvideo.cs* .

Výpis 4.5: Skript addvideo.cs - modifikácia pre Audio Source

```
1  ...
2  public AudioSource audio1;
3  public AudioSource audio2;
4  public AudioSource audio3;
5  public AudioSource audio4;
6  void Start () {
7  ...
8      V1 = GameObject.Find ("V1");
9      var videoPlayer1 =
10         V1.AddComponent<UnityEngine.Video.VideoPlayer> ();
11     videoPlayer1.audioOutputMode =
12         VideoAudioOutputMode.AudioSource;
13     videoPlayer1.SetTargetAudioSource(0, audio1);
14     videoPlayer1.url = vids[0];
15     videoPlayer1.isLooping = true;
16 ...
```

Najprv sme vytvorili premenné typu *AudioSource* pre vytvorené zdroje. Následne sme nastavili *audioOutputMode* nášho *videoPlayeru* na *AudioSource* , čo nám umožní ďalej priradiť zdroj k videu pomocou *SetTargetAudioSource* . Toto nastavenie sme zopakovali pre všetky telesá. Po priradení zvukových zdrojov k premenným a spustení aplikácie sa zvuk šíri zo zdroja priestorovo, s rastúcou vzdialenosť klesá jeho intenzita a pri pohybe je simulovaný Dopplerov jav.

5 Tag menu

Po spustení aplikácie sa načítajú videá na telesá, no po prezretí všetkých telies by virtuálna výstava nemala čo viac ponúknuť a ak by chcel hráč načítať nové videá, musel by aplikáciu reštartovať a ani to by nezaručilo, že by sa niektoré video znovu nenačítalo. Preto je potrebné vytvoriť systém na načítanie nových videí z iniciatívy hráča. Pre tento proces sme vybrali vytvorenie vyhľadávacieho menu, kde po zadaní určitého *tagu*[1] vyskočí zoznam všetkých videí v databáze s priradeným *tagom,* a následným načítaním vybraného videa na teleso.

5.1 PhP a MySQL

Pre vytvorenie takého prvku je potrebné vytvoriť databázu obsahujúcu tieto *tagy* a odkazy na videá k ním prislúchajúce. Pomocou phpMyAdmin sme vytvorili **MySQL** databázu na našom serveri, špecifického používateľa pre Unity a tabuľku *vidlist* s nasledujúcimi stĺpcami:

- **id** - prvok typu *integer* s funkciou *auto-increment* ktorý slúži ako číslovač - každému riadku pridá unikátne identifikačné číslo.
- **name** - prvok typu *varchar* ktorý slúži pre uchovanie názvu videa, ktoré sa bude zobrazovať pri telese.
- **link** - prvok typu *varchar* ktorý slúži pre uchovanie URL cesty k danému videu.
- **tag** - prvok typu *text* do ktorého sa budú ukladať všetky priradené tagy pre dané video.

Unity však s touto databázou nedokáže komunikovať samostatne a na komunikáciu preto využíva volanie **php** skriptov. Na doméne sme preto vytvorili jednoduché užívateľské prostredie pre pridávanie videí a následné vyhľadávanie týchto videí pomocou tagov v databáze.

5.1.1 Pridávanie videí do databázy

Pre pridávanie videí sme vytvorili formulár na adrese *http://dmedia.6f.sk/add_video.php,* ktorý odosiela informácie o názve videa, tagoch a nahráva samotný súbor na server volaním skritpu *add_v_script.php.*

[1] slovo, ktoré indentifikuje daný objekt

Výpis 5.1: Skript add_v_script.php - definícia premenných

```
1  ...
2  $server ="sql.endora.cz:3306";
3  $user = "unity";
4  $pass = "Unity123";
5  $db = "unity";
6  $dir = "vid/";
7  $file = $dir . basename($_FILES["vid"]["name"]);
8  $uploadok = 1;
9  $filetype = strtolower(pathinfo($file, PATHINFO_EXTENSION));
10 $name = $_POST["vname"];
11 $tagy = $_POST["tag"];
12 ...
```

Najprv sme definovali premenné s prihlasovacími údajmi do datázy, definovali zložku pre vloženie súboru, samotný súbor *file* ktorý je pomocou HTML forumlára posielaný, a premennú *uploadok* ktorá bude zisťovať splnenie všetkých podmienok pre nahratie súboru. Premenná *filetype* zisťuje typ nahratého súboru, *name* a *tagy* sú premenné ktorým sú priradené posielane hodnoty z formulára o názve videa a o priradených tagoch.

Výpis 5.2: Skript add_v_script.php - kontrola nahrávaného súboru

```
1  ...
2  if(file_exists($file)){
3      echo "Rovnaky␣subor␣uz␣existuje<br>";
4      $uploadok = 0;
5  }
6  if($filetype != "mp4"){
7      echo "Nespravny␣format<br>";
8      $uploadok = 0;
9  }
10 if($uploadok ==0){
11     echo "nastal␣problem␣pri␣uploadovani<br>";
12 }else{
13 ...
```

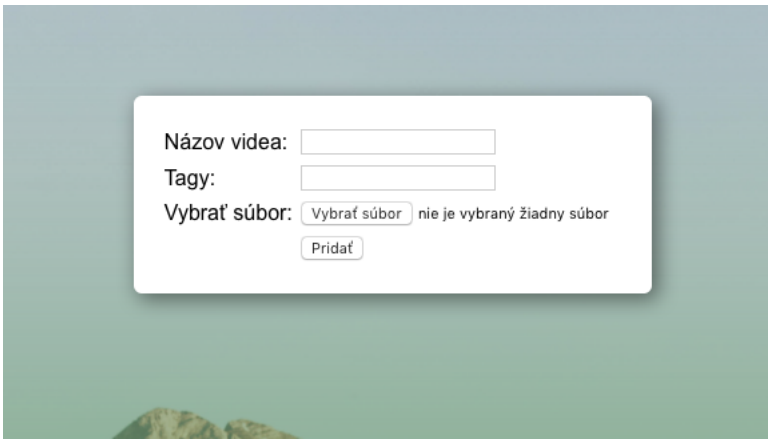
V prvom kroku pomocou podmienky **if** skript zistí, či súbor s rovnakým názvom už exituje a ak tomu tak je, nastaví hodnotu kontrolnej premennej *uploadok* na 0 a vypíše chybu o existencií rovnakého súboru. Ďalej rovnakým spôsobom zisťuje typ súburu, v tomto prípade je typ akceptovateľného súboru obmedzený na *.mp4* a ak tomu tak nieje opäť vypíše chybu o zlom type súboru a kontrolnú premennú nastaví

na 0. Ak je po týchto podmienkach kontrolná premenná rovná nule, skript vypíše chybu pri nahrávaní súboru, ukončí skript a nepostupuje k nahraniu do databázy. Ak to tak nieje tak skript pokračuje ďalej k nahrávaniu do databázy.

Výpis 5.3: Skript add_v_script.php - vloženie do databázy

```
...
if(move_uploaded_file($_FILES["vid"]["tmp_name"], $file))
{
$conn = new mysqli($server,$user,$pass,$db);
if($conn->connect_error){
    die("Problem␣MySQL");
    }else{
    $sql = "INSERT␣INTO␣‘vidlist‘␣(‘id‘,␣‘name‘,
␣␣␣␣‘link‘,␣‘tag‘)␣VALUES␣(NULL,␣’$name’,’$file’,’$tagy’)";
    if($conn->query($sql) === TRUE){
        echo"Subor␣bol␣nahraty␣do␣databazy<br>";
    }else{
        echo "SQL␣ERROR␣".$conn->error;
        echo $name.$file.$tagy;
    }
}
}else{
echo "Nastal␣problem<br>";
echo $file . $_FILES["vid"]["tmp_name"];
}
}
...
```

V prvom kroku pomocou pomienky **if** sa skript pokusí uložiť súbor a ak sa to nepodarí vypíše problém. Ak problém nieje a súbor je úspešne nahraný na server, skrpit postupuje k nahrianu dát do databázy. Premenná *conn* vytvorí MySQL pripojenie k databáze pomocou prihlasovacích údajov a následne pre prípad neúspešného pripojenia vypíše problém a ukončí skript. Premenná *sql* obsahuje sql príkaz, ktorý je odosielaný do databázy. Tento príkaz vloží hodnoty premenných do jednotlivých stĺpcov tabuľky *vidist* . Ak sa tento výkon podarí, vypíše sa správa o úspešnom nahraní súboru, ak nie, vypíše sa SQL chyba. Tento skript sme využili na pridanie videí na server, nastavení ich názvov a tagov.



Obr. 5.1: Formulár pre pridávanie videa

	id	name	link	tag
Upraviť Kopírovať Zmazať	1	Chrome Rings	vid/ChromeRingsonBlackLoop.mp4	chrome rings black loop
Upraviť Kopírovať Zmazať	2	Golden Liquid	vid/GoldenChromeLiquidLoop.mp4	golden chrome liquid loop
Upraviť Kopírovať Zmazať	3	Unity	vid/unitytrailer.mp4	unity 3d trailer
Upraviť Kopírovať Zmazať	4	Wireframe	vid/Rough3DWireframeTerrain.mp4	rought wireframe terrain loop
Upraviť Kopírovať Zmazať	5	Spiny	vid/Spiny_Abstract_Loop.mp4	spiny abstrackt loop
Upraviť Kopírovať Zmazať	6	Pink Rings	vid/RandomPinkRingsLoop.mp4	random pink rings loop

Označiť všetko *Výber:* Upraviť Kopírovať Zmazať Exportovať

Obr. 5.2: Pridané informácie v databáze MySQL na platforme phpMyAdmin

5.1.2 Hľadanie v databáze pomocou tagu

Pre hľadanie videí podľa priradených tagov sme vytvorili formulár na adrese *http://dmedia.6f.sk/search.php* . Tento formulár sme použili ako na internetovom rozhraní tak aj priamo pri hľadaní videí v Unity. Skript *search_script.php* slúži pre internetové rozranie a skript *search_ingame.php* slúži pre vyhľadávanie v aplikácií. Tieto skripty sa líšia iba v tom, že v skripte pre internetové rozhranie vykresľuje grafiku stránky a výpis je vo forme názvu videa s hyperlinkovým odkazom, zatiaľ čo skript volaný z Unity vracia názov a URL adresu videa.

Výpis 5.4: Skript search_ingame.php - získavanie dát z databázy

```
1  <?php
2  $server ="sql.endora.cz:3306";
3  $user = "unity";
4  $pass = "Unity123";
5  $db = "unity";
6  $tagy = $_REQUEST["tag"];
7  $conn = new mysqli($server,$user,$pass,$db);
8  if($conn->connect_error){
9      die("Connection␣Error");
10 }
11 $sql = "SELECT␣*␣FROM␣vidlist␣WHERE␣tag␣REGEXP
12 '([[:<:]]|^)$tagy([[:>:]]|$)'";
13 $result = $conn->query($sql);
14 if($result->num_rows > 0){
15     while($row = $result->fetch_assoc()){
16         echo $row["name"].",".$row['link'].",";
17     }
18     }else{
19         echo "␣,Nenašlo␣sa";
20 }
21 $conn->close();
22 ?>
```

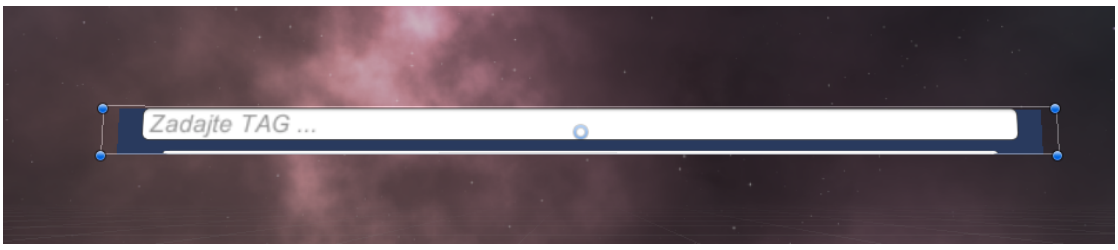
Na začiatku skriptu sme definovali premenné pre prihlasovacie údaje do databázy a premennú *tagy* ktorá získa tag z odosielaného formulára. Následne sme vytvorili pripojenie pomocou premennej *conn* a pomocou podmienky **if** sme zabezpečili výpis chyby pri zlyhaní tohto pripojenia. V premennej *sql* je uložený SQL príkaz pre vybranie informácií o všetkých videách, ktoré obsahujú zadaný tag. Pomocou príkazu **REGEXP** vieme hľadať zadaný tag v stĺpci, ktorý obsahuje všetky tagy pre dané video a nájsť presnú zhodu v slove. Ak je hľadanie úspešné, vypíše sa riadok ktorý obsahuje názov videa a URL link na nájdené video oddelené znakom **,** . V prípade, že sa žiadne také video nenašlo, sa vypíše oznam o nenájdení videa s daným tagom.

5.2 User Interface

Pre vyhľadávanie diel pomocou tagov musíme v Unity vytvoriť menu, ktoré bude hráč môcť vyvolať po priblížení k telesu a následne bude mocť vyhľadať video s priradeným tagom a po výbere tohto videa zmeniť aktuálne prehrávané video na vyhľadané.

5.2.1 Vytvorenie prvku menu

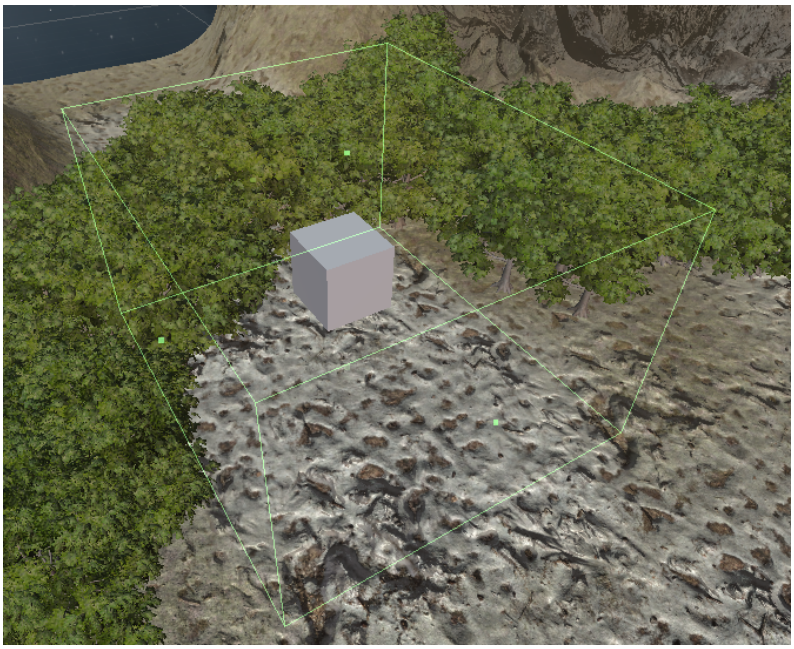
Pre vytvorenie 2D prvkov scény slúžia *GameObjecty* typu *UI* . Do tejto skupiny patria prvky ako *Text, Image, Button, Toggle, Slider, Scrollbar, Dropdown, Input Field, Canvas, Panel, Scroll View, Event System.* Pri vytváraní tohto menu sme najprv vytvorili prázdny *GameObject* ktorý sme pomenovali *Menu* a všetky novo vytvorené prvky vkladali pod neho. Prvý z týchto prvkov bol *GameObject Canvas* , ktorý vytvorí podklad pre vytváranie a zobrazovanie 2D prvkov *UI* . Pod tento prvok sme následne vytvorili *GameObject RawImage* , nastavili jeho rozmery pre potreby nášho menu a jeho farbu nastavili na modrú (293A5E). Ďalej sme pod tento prvok vytvorili *GameObject InputField* ktorý slúži ako vstup pre zadávanie textu a to v našom prípade hľadaného tagu. Komponent *Placeholder Text* tohto prvku sme nastavili na "Zadajte TAG ...äby tento text bol zobrazený pred zadaním hľadaného slova. Ďalej sme pod *RawImage* vytvorili *GameObject Button* ktorý sa bude chovať ako placeholder pre prvky *Button* ktoré sa budú dynamicky vytvárať podľa výstupu z databázy. Pod tento prvok sme vytvorili *GameObject Panel* ktorému sme pridali komponent *Vertical Layout Group* čo spôsobí vytvorenie vertikálneho zarovania. Pod tento prvok sme následne vytvorili ďalší *Button* ktorý sme umiestnili na miesto prvého vyhľadaného tlačítka. Z tohto prvku sme pretiahnutím z panelu *Hierarchy* do panelu *Project* vytvorili Prefab *Button* , ktorý využijeme ako vzor pre dynamicky vytvárané *Buttony* . Tento prvok sme následne zo scény odstránili a ponechali sme si ho iba ako vzor pre skript vytvárajúci menu.



Obr. 5.3: Vytvorený prvok menu

5.2.2 Trigger zóna

Trigger zóna je ohraničené miesto na scéne, do ktorého ak vstúpi hráč, vyvolá určitú reakicu. V našom prípade sme vytvorili trigger zónu okolo výstavného telesa. Jej zmysel spočíva v tom, že v prípade vstupu hráča do tejto zóny vyšle informáciu o tom, že sa hráč v danej zóne nachádza a to bude mať za následok ponúknutie hráčovi vyvolať menu pre zadanie tagu a zmenu videa. Pre vytvorenie tejto zóny sme pridali nový *GameObject Cube* a zmenili jeho názov na *TriggerZone.* V záložke *Inspektor* sme pre komponent *Mesh Render* odšktrli jeho viditeľnosť, čo spôsobilo, že objekt nevidíme a chová sa ako neviditeľný *collider.* Využitím možnosti *Edit Collider* komponentu *Box Collider* sme upravili veľkosť zóny okolo telesa.



Obr. 5.4: Trigger zóna okolo telesa

5.2.3 Tag Menu

Pre vytvorenie vyhľadávacieho menu sme vytvorili skript *vchange.cs* . Tento skript sa stará o zistenie kolízie s trigger zónou, následne ponúka hráčovi možnosť vyvolať menu pre vyhľadávanie v databáze videí pomocou zadaného tagu, následne tento proces vykoná a vyhľadá všetky videá s priradeným tagom a po kliknutí na vybrané tlačítko s názvom videa toto video premietne na teleso.

Výpis 5.5: Skript vchange.cs - vytvorené premenné

```
1  ...
2  public class vchange : MonoBehaviour {
3      [SerializeField] GameObject buttonPrefab;
4      [SerializeField] Transform menuPanel;
5      public GameObject nenaslo;
6      public GameObject o;
7      public GameObject menu;
8      public GameObject V;
9      public GameObject newtitle;
10     public GameObject infotext;
11     public UnityEvent onTrigger;
12     private bool colliding = false;
13     public static bool keyboard = false;
14     public InputField text;
15     public string dstring;
16 ...
```

Vytvorili sme premenné *buttonPrefab* a *menuPanel* ktoré budú slúžiť pri vytváraní nových tlačidiel menu, *nenaslo* slúži pre priradenie textu o nenájdení videa, *o* je premenná využitá pri ničení vytvorených *Buttonov* , *menu* je pre priradenie už vytvoreného objektu *Menu* , premenná *V* slúži pre priradenie objektu telesa, *newtitle* slúži pre priradenie nového názvu videa zobrazujúceho sa pri telese, *infotext* slúži na priradenie informačného textu o možnosti vyvolať menu. Následne sme vytvorili premennú *colliding* ktorá slúži na uloženie informácie z trigger zóny, premenná *keyboard* slúži na uloženie informácie o tom, či má byť klávesnica a myš použitá ako ovládanie hráča alebo ako vstupné zariadenia pre menu. Premenná *text* slúži k priradeniu vytvoreného *InputField* a nakoniec sme vytvorli premennú *dstring* do ktorej budeme vkladať odozvu zo servera.

Výpis 5.6: Skript vchange.cs - funkcie pre trigger

```
1  ...
2  void OnTriggerEnter (Collider other) {
3      infotext.SetActive (!infotext.activeSelf);
4      colliding = true;
5  }
6  void OnTriggerExit (Collider other) {
7      infotext.SetActive (!infotext.activeSelf);
8      colliding = false;
9  }
10 ...
```

Využili sme funkciu *OnTriggerEnter* kde pri vstupe do trigger zóny nastaví viditeľnosti textu *infotext* pomocou *SetActive* a nastavili premennú *colliding* na *true* . Podobne sme využili funkciu *OnTriggerExit* kde pri opustení zóny sa *infotext* zneviditeľní a *colliding* sa nastaví na *false* . Pri vstupe do zóny okolo telesa tak vyskočí informačný text o možnosti vyvolania menu a pri opustení tejto zóny tento text zmizne. Premennú *colliding* ďalej využijeme pri podmienke vyvolania menu a tak je možné toto menu vyvolať iba zvnútra zóny.

Výpis 5.7: Skript vchange.cs - funkcia Update()

```
1  ...
2  void Update () {
3      if (Input.GetKeyDown (KeyCode.Tab) && colliding) {
4          menu.SetActive (!menu.activeSelf);
5          keyboard = !keyboard;
6      }
7      if (keyboard) {
8          if (Input.GetKeyDown (KeyCode.Return)) {
9              string url =
10         "http://dmedia.6f.sk/search_ingame.php";
11             WWWForm f = new WWWForm ();
12             f.AddField ("tag", text.text);
13             WWW www = new WWW (url, f);
14             destroybuttons ();
15             StartCoroutine (wfr (www));
16         }
17     }
18 }
19 ...
```

Do funkcie *Update()* sme pridali podmienku **if** ktorá má na starosti zviditeľnenie menu pri stlačení klávesy TAB ak sa hráč nachádza v zóne. Táto pomienka zmení hodnotu premennej *keyboard* , čo spôsobí vypnutie pohybu hráča a dovolí vpisovať do textového poľa a vybrať vyhľadaný objekt. Pomocou ďalšej podmienky *if* pre prípad, že je menu zapnuté a *keyboard* má tým pádom hodnotu *true* , postlačení klávesy **Return** skript pomocou *WWWForm* pridá hodnotu v premennej *text.text* do posielaného formulára pre skrip *search_ingame.php* pre položku *tag* . Tento formulár sa odošle, pomocou funkcie *destroybuttons()* sa vymažú všetky vytvorené *Buttony* a iniciuje sa začiatok prenosu dát medzi databázou a Unity pomocou *StartCoroutine(wfr)* .

Výpis 5.8: Skript vchange.cs - Coroutine

```
1  ...
2  IEnumerator wfr (WWW www) {
3      yield return www;
4      dstring = www.text;
5      char d = ',';
6      string[] w;
7      w = dstring.Split (d);
8      int l = w.Length;
9      nenaslo.SetActive (false);
10     if (w[1] == "Nenašlo sa") {
11         nenaslo.SetActive (!nenaslo.activeSelf);
12     } else {
13         for (int i = 0; i < l - 1; i += 2) {
14         GameObject button =
15         (GameObject) Instantiate (buttonPrefab);
16         button.GetComponentInChildren<Text> ().text = w[i];
17         int index = i + 1;
18 button.GetComponentInChildren<Button>().onClick.AddListener(
19     ()=>{   ChangePlayedVideo (w[index], w[index - 1]);
20             destroybuttons ();
21             menu.SetActive (!menu.activeSelf);
22             keyboard = !keyboard; });
23         button.transform.parent = menuPanel;
24         }
25     }
26 }
27 ...
```

Pre zadefinovanie volanej *StartCoroutine(wrf)* využijeme *IEnumerator wfr(WWW www)* . Táto funkcia vracia z php skriptu získané informácie, ktoré sme vložili do stringu *dstring* . Pomocou charakteru *d* následne rozdelí prijatý string do poľa *w[]* a zistí veľkosť tohto poľa v premennej *l* . Objekt *nenaslo* sme nastavili na *false* čo spôsobí, že je objekt defaultne neviditeľný. Následne skontrolujeme pomocou podmienky **if** , či je hľadanie neuspešné a ak tomu tak je, textový objekt *nenaslo* sa zviditeľní. Ak je vyhľadávanie úspešné, pomocou cyklu **for** skript vytvorí *GameObject button* zo vzorového prefabu *buttonPrefab* a nastaví jeho text pomocou *GetComponentInChildren* na názov videa ktorý je uložený v poli *w[]* pre každý nájdený objekt. Následne je k *buttonu* pridaný *OnClick Listener* , v ktorom sa po kliknutí na daný button volá funkcia *ChangePlayedVideo* a *destroybuttons()* , vyvolané menu sa zneviditeľní a keyboard neguje svoju hodnotu.

Výpis 5.9: Skript vchange.cs - funkcie ChangePlyerVideo a destroybuttons

```
1  ...
2  void ChangePlayedVideo (string l, string n) {
3      var videoPlayer =
4          V.GetComponent<UnityEngine.Video.VideoPlayer> ();
5      videoPlayer.url = "http://dmedia.6f.sk/" + l;
6      var title = newtitle.GetComponent<TextMesh> ();
7      title.text = n;
8  }
9  void destroybuttons () {
10     o = GameObject.Find ("Panel");
11     foreach (Transform child in o.transform) {
12         GameObject.Destroy (child.gameObject);
13     }
14 }
15 ...
```

V predošlých krokoch sme v skripte volali funkcie *ChangePlayedVideo()* a *destroybuttons()* . Prvá funkcia má vstupné argumenty *l* a *n* ktoré slúžia na zaslanie URL odkazu na video a názov daného videa. Podobne ako v skripte *addvideo.cs* sme nastavili prehrávané video pomocou *videoPlayer.url* a premennej *l* a zobrazený text pomocou *title* a získanej premennej *n* . Druhá funkcia *destroybuttons()* nájde *GameObject Panel* , pod ktorý sú skriptom vytvárané *Buttony* a pomocou **foreach** a *GameObject.Destroy* všetky tieto objekty z menu odstráni.

Pre správnu funkciu potrebujeme upraviť skripty *movement.cs* a *mouse.cs* aby pri kladnej hodnote premennej *keyboard* neumožnovali pohyb po scéne. To zabezpečíme pridaním podmienky **if** a zistením hodnoty *vchange.keyboard* .

Výpis 5.10: Skript movement.cs - modifikácia

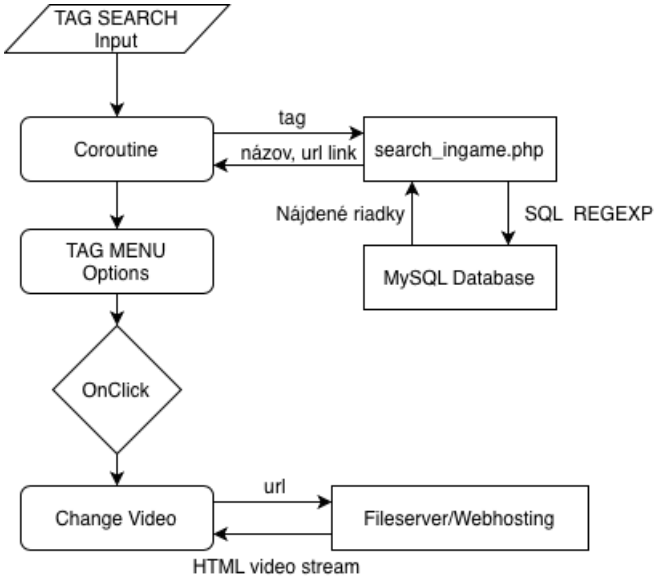
```
...
if (vchange.keyboard) { } else {
...
```

Výpis 5.11: Skript mouse.cs - modifikácia

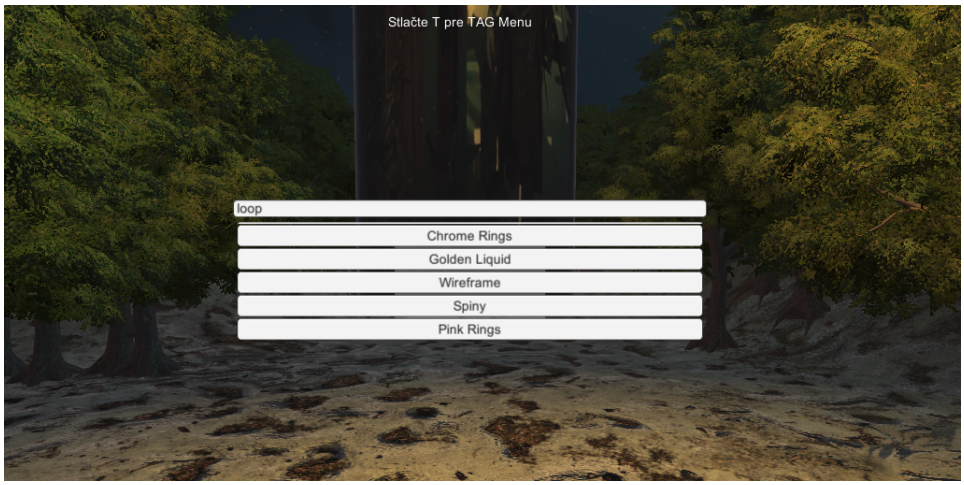
```
...
if (vchange.keyboard) {
    Cursor.lockState = CursorLockMode.None;
    } else {
...
```

Pre skript *movement.cs* sa pre kladnú hodnotu *keyboard* neudeje nič a ak tomu tak nieje, pomocou **else** pokračuje na už vytvorený kód pre pohyb. Pre *mouse.cs* potrebujeme zaistiť, že pri kladnej hodnote *keyboard* môžme pohybovať s kurzorom po obrazovke a na to sme použili príkaz *Cursor.LockMode.None* . Pri zápornej hodnote *keyboard* pokračuje pomocou **else** už vytvorený kód pre rotáciu kamery.

Vytvorený skript *vchange.cs* sme priradili ako komponent prvku *Trigger Zone* a priradili všetky požadované objekty k premenným. Po spustení aplikácie a priblížení k telesu sa nám zobrazil text o možnosti vyvolania "Tag Menu", po stlačení klávesy T sa toto menu zobrazilo, po zadaní tagu sa zobrazili vyhľadané videá vo forme tlačidiel a po kliknutí na jedno z tlačidiel sa premietane video zmenilo na nájdené video. Tento proces sme zopakovali pre všetky stanovištia za účelom vytvoriť možnosť zmeny na každom stanovišti



Obr. 5.5: Blokové schéma algoritmu pre zmenu prehrávaného videa



Obr. 5.6: Vyhľadávacie okno Tag Menu

5.3 Ovládanie prehrávania videa

Pre ovládanie prehrávania videa sme vytvorili funkcie *pausevideo(), resumevideo()* a *restartvideo().* Ovládať video bude možné po vstupe do trigger zóny a následnom stlačení numerických kláves 1 (reštart), 2 (pauza) a 3 (spustenie). Tieto funkcie sme pridali modifikáciou skriptu *vchange.cs* .

Výpis 5.12: Skript vchange.cs - modifikácia Update()

```
1  ...
2  void Update () {
3  ...
4      if (Input.GetKeyDown (KeyCode.Keypad2)&& colliding) {
5          pausevideo ();
6      }
7      if (Input.GetKeyDown (KeyCode.Keypad1)&& colliding) {
8          restarvideo ();
9      }
10     if (Input.GetKeyDown (KeyCode.Keypad3)&& colliding) {
11         resumevideo ();
12     }
13 ...
```

Najprv sme nastavili volanie jednotlivých funkcií pomocou podmienky **if** pre jednotlivé prípady stlačení kláves určených pre ovládanie a prítomnosť v trigger zóne daného telesa.

Výpis 5.13: Skript vchange.cs - definície funkcií pre ovládanie prehrávania

```
...
void pausevideo () {
    var videoPlayer =
        V.GetComponent<UnityEngine.Video.VideoPlayer> ();
    videoPlayer.Pause ();
}
void resumevideo () {
    var videoPlayer =
        V.GetComponent<UnityEngine.Video.VideoPlayer> ();
    videoPlayer.Play ();
}
void restarvideo () {
    var videoPlayer =
        V.GetComponent<UnityEngine.Video.VideoPlayer> ();
    videoPlayer.Stop ();
    videoPlayer.Play ();
}
...
```

Pre funkciu *pausevideo()* sme vytvorili premennú pre *VideoPlayer* a následne pomocou vstavnej metódy *Pause()* funkcia pozostaví prehrávané video. Funkcia *resumevideo()* funguje podobne ako *pausevideo()* no je volaná metóda *Play* . Pre reštart videa funkcia *restartvideo()* najprv volá metódu *Stop()* ktorá pozastaví prehrávanie a vráti pozíciu prehrávania na začiatok, následne pomocou *Play()* sa video spustí od začiatku.

Upravili sme informačný text vyskakujúci po vstupe do trigger zóny pridaním informácií o ovládaní prehrávania videa a po spustení aplikácie sme pomocou numerických kláves 1, 2 a 3 dokázali ovládať prehrávanie videí na jednotlivých telesách.

5.3.1 Pausemenu

Pre pozastavenie aplikácie s možnosťou ju vypnúť alebo pokračovať sme vytvorili Pause menu. Vytvorili sme *GameObject Canvas* ktorý sme premenovali na *Pausemenu* , k nemu sme priradili *GameObject Panel* a následne priradili 2 *GameObjecty* typu *Button* ktoré reprezentujú možnosti *Pokračovať* a *Ukončiť* . Vytvorili sme skript *pausescript.cs* ktorý riadi vyvolávanie tohto prvku.

Výpis 5.14: Skript pausescript.cs

```
1  public class pausescript : MonoBehaviour {
2      public static bool keyboard = false;
3      public GameObject pausemenu;
4      void Update () {
5          if (Input.GetKeyDown (KeyCode.Escape)) {
6              pausemenu.SetActive (!pausemenu.activeSelf);
7              keyboard = !keyboard;
8          }
9      }
10     public void resume () {
11         pausemenu.SetActive (!pausemenu.activeSelf);
12          keyboard = !keyboard;
13     }
14     public void exit () {
15         Application.Quit ();
16     }
17 }
```

Podobne ako pri *vchange.cs* sme vytvorili premennú *keyboard* ktorá riadi vypnutie klávesnice a myši pre pohyb po scéne. Vytvorili sme premenné *GameObject pauseme* ku ktorému priradíme vytvorený prvok *Pausemenu* . Vo funkcií Update() pomocou podmienky **if** zisťujeme stlačenie klávesy **Escape** a v prípade stlačení tejto klávesy zviditeľníme prvok *Pausemenu* a negujeme hodnotu *keyboard* . Vytvorili sme funkciu *resume()* ktorá sa stará o vypnutie zobrazeného menu a pokračovanie v hraní. Podobne ako funkcia *Update()*funkcia *resume()* zneviditeľní prvok *Pausemenu* a neguje hodnotu *keyboard* . Funkcia *exit()* spôsobí vypnutie aplikácie pomocou príkazu *Application.Quit()* . Pre tlačítko *Pokračovať* sme v komponente *Button* pridali *OnClick()* s volaním funkcie *resume()* a pre tlačítko *Ukončiť* sme v komponente *Button* pridali *OnClick()* s volaním funkcie *exit()* . Pre tento prvok sme v *movement.cs* a *mouse.cs* pridali do podmienky *pausescript.keyboard* podobne ako s *menuscript.keyboard* , aby sa pohyb pri vyvolaní menu zastavil.

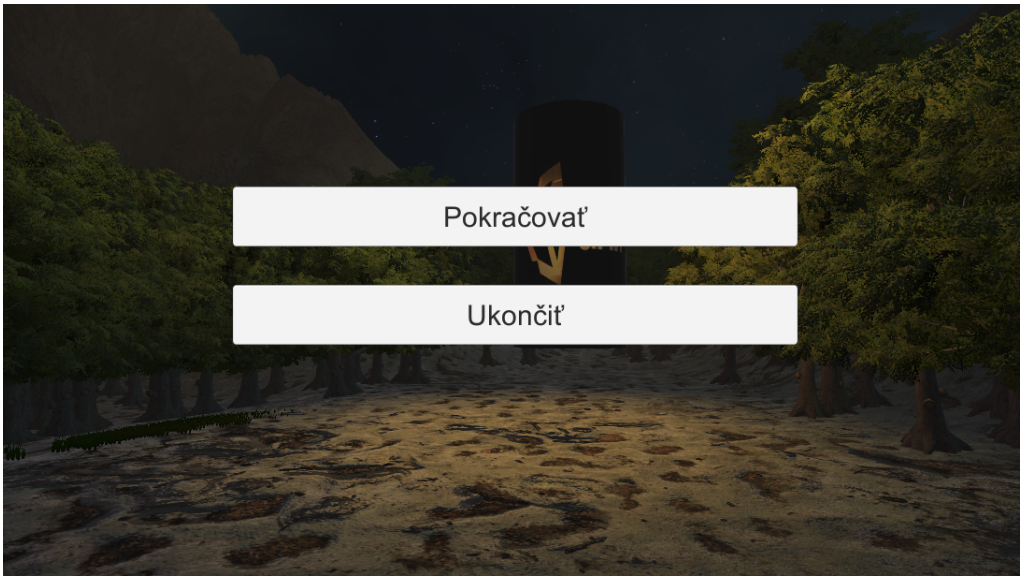
Výpis 5.15: Skript movement.cs - modifikácia

```
1 ...
2 if (vchange.keyboard || pausescript.keyboard) { } else {
3 ...
```

Výpis 5.16: Skript mouse.cs - modifikácia

```
1 ...
2 if (vchange.keyboard || pausescript.keyboard) {
3     Cursor.lockState = CursorLockMode.None;
4     } else {
5 ...
```

Po spustení aplikácie a stlačení klávesy **Escape** sa vyvolá menu s možnosťami "Pokračovať" a " Ukončiť". Po kliknutí na Pokračovať sa menu zneviditeľní a hráč sa môže voľne pohybovať a po kliknutí na Ukončiť sa aplikácia vypne.



Obr. 5.7: Vytvorené Pause Menu

Záver

V tejto bakalárskej práci sme sa venovali problematike vytvárania prezentácie audiovizuálnych diel vo virtuálnej realite. Vykonali sme rešerš dostupných herných enginov pre tvorbu tejto prehliadky, kde sme zisťovali informácie o tvorbe v jednotlivých enginoch a pozreli sme sa aj na licenčnú dostupnosť. Pomocou získaných informácií sme sa rozhodli pre prácu v hernom engine Unity. Popísali sme teoretické základy mapovania textúr na objekty.

V hernom engine sme vytvorili projekt v ktorom sme vybudovali scénu virtuálnej prehliadky pomocou vstavaných nástrojov a nástrojov z Unity Asset Store. Vytvorená scéna obsahovala natextúrovaný terén asteroidu, prvky foliage, skybox textúru, zvukové zdroje prostredia a výstavné telesá.

Vytvorili sme ovládanie postavy hráča z perspektívy prvej osoby s pohybom po scéne pomocou klávesnice a rotáciou kamery pomocou myši. Vytvorili sme základnú fyziku scény v podobe gravitačného vplyvu na postavu hráča a nastavení kolízie s prostredím.

Založili sme internetový portál na ktorom sme vytvorili jednoduché prostredie na vkladanie nových videí a MySQL databázu. Pomocou vstavaných objektov Unity a vytvorených skriptov sme dokázali streamovať video z vytvoreného internetového úložiska pomocou URL adresy na výstavné telesá s využitím priestorového spracovania zvuku enginom. Následne sme demonštrovali možnosť využia komunikácie so serverom pomocou protokolu FTP pre dynamicky meniace sa nastavenie priradených videí k jednotlivým telesám.

Kombináciou C# a PhP skriptov sme vytvorili menu pre vyhľadávanie v databáze MySQL pomocou tagov priradených k videám s možnosťou vybrať vyhľadané video pre premietanie na vybranom telese. Vytvorili sme funkcie pre ovládanie prehrávania videa na vybranom telese.

Podarilo sa nám vytvoriť scénu so štyrmi samostatne ovládateľnými a nastaviteľnými telesami pre prehrávanie audiovizuálnych diel. Ciele stanovené v tejto bakalárskej práci sa nám podarilo dosiahnuť a výsledkom je aplikácia obsahujúca základné aspekty virtuálnej prehliadky audiovizuálnych diel.

Literatúra

[1] WARD, Jeff: *What is a Game Engine* [online]. Gamecareerguide [4. 29. 2008]. Dostupné z URL: <https://www.gamecareerguide.com/features/529/what_is_a_game_.php>.

[2] AXON, Samuel: *Unity at 10: For better—or worse—game development has never been easier* [online]. Ars Technica [27. 9. 2016]. Dostupné z URL: <https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worsegame-development-has-never-been-easier/>.

[3] *Using DirectX11 in Unity 4* [online]. Unity Technologies Dostupné z URL: <https://docs.unity3d.com/520/Documentation/Manual/DirectX11.html>.

[4] MATNEY, Lucas: *With new realities to build, Unity positioned to become tech giant* [online]. TechCrunch [25. 5. 2017]. Dostupné z URL: <https://techcrunch.com/2017/05/25/with-new-realities-to-build-unitypositioned-to-become-tech-giant/>.

[5] *Unity Blog* [online]. Unity . Dostupné z URL: <https://blogs.unity3d.com/2019/08/29/evolving-the-unity-editor-ux/>.

[6] THOMSEN, Mike : *History of the Unreal Engine* [online]. IGN [14. 6. 2012]. Dostupné z URL: <https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine>.

[7] DAYAL, Preetisha: *Unreal Engine Review: Pros,Cons and Suitability* [online]. newgenapps [26. 5. 2018]. Dostupné z URL: <https://www.newgenapps.com/blog/unreal-engine-review-pros-conssuitability/>.

[8] *Frequently asked questions (FAQ)* [online]. Epic Games . Dostupné z URL: <https://www.unrealengine.com/en-US/faq>.

[9] *Unreal Engine 4.20 screenshot* [online]. Wikipedia . Dostupné z URL: <https://en.wikipedia.org/wiki/Unreal_Engine#/media/File:Unreal_Engine_4_screenshot.png>.

[10] FORD, Jerry Lee Jr.: *Getting Started with Game Maker* . Cengage Learning [1. 6. 2009]. ISBN 978-1435455214 .

[11] *GameMaker Studio 2 Released* [online]. GameFromScratch . Dostupné z URL: <https://www.gamefromscratch.com/post/2017/03/09/GameMaker-Studio-2Released.aspx>.

[12] *GDC17: GameMaker Studios 2.0 Takes On Industry Titans* [online]. Broken Joysticks . Dostupné z URL: <https://www.brokenjoysticks.net/2017/03/12/game-makerstudio-takes-on-titans/>.

[13] *Interview: James Cox of YoYo Games about GameMaker Studio 2* [online]. This Is Xbox. Dostupné z URL: <http://www.thisisxbox.com/interview-james-cox-yoyogames-gamemaker-studio-2/>.

[14] ALEXANDER, Mark: *Which Licence Is Right For me* [online]. YoYo Games [9. 2019]. Dostupné z URL: <https://help.yoyogames.com/hc/en-us/articles/115002637011Which-Licence-Is-Right-For-me->.

[15] *GameMaker Studio 2 Web* [online]. steamdb . Dostupné z URL: <https://steamdb.info/app/585600/screenshots/>.

[16] LINIETSKY, Juan: *First Public Release* [online]. Godot [14. 1. 2014]. Dostupné z URL: <https://godotengine.org/article/first-public-release>.

[17] BRASSEUR, Vicky: *Godot open source game engine helps power the future in West Virginia* [online]. Opensource.com [16. 8. 2016]. Dostupné z URL: <https://opensource.com/education/16/8/godot-open-source-game-engine>.

[18] *GDScript* [online]. Godot Docs. Dostupné z URL: <http://docs.godotengine.org/en/3.0/getting_started/scripting/gdscript/gdscript_basics.html>.

[19] *Licence* [online]. Godot . Dostupné z URL: <https://godotengine.org/license>.

[20] *Godot – open source engine pro tvorbu 2D a 3D her* [online]. gamesdev . Dostupné z URL: <https://gamesdev.cz/godot-open-source-engine-pro-tvorbu-2d-a-3d-her/>.

[21] *CryEngine Video Game Engine Review* [online]. GameDesigning. Dostupné z URL: <https://www.gamedesigning.org/engines/cryengine/>.

[22] *Features* [online]. Crytek . Dostupné z URL: <https://www.cryengine.com/features>.

[23] BATCHELOR, James: *Crytek adopts royalties model as CryEngine 5.5 arrives* [online]. Gamersindustry [20. 3. 2018]. Dostupné z URL: <https://www.gamesindustry.biz/articles/2018-03-20-crytek-adopts-royaltiesmodel-as-cryengine-5-5-arrives>.

[24] HOMOLA, Adam: *Cenu za novou verzi CryEngine si stanovíte sami* [online]. gamestiscali [16. 3. 2016]. Dostupné z URL: <https://games.tiscali.cz/oznameni/cenu-zanovou-verzi-cryengine-si-stanovite-sami-272392>.

[25] RAJMIC, Pavel a SCHIMMEL, Jiří: *Moderní počítačová grafika* Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií, 2013 .

[26] ROUSE, Margaret: *Texture mapping* [online]. techtarget [6. 2010]. Dostupné z URL: <https://games.tiscali.cz/oznameni/cenu-zanovou-verzi-cryengine-si-stanovite-sami-272392>.

[27] *Mipmapping* [online]. wikipedia . Dostupné z URL: <https://upload.wikimedia.org/wikipedia/commons/5/5c/MipMap_Example_STS101.jpg>.

[28] *Linear interpolation* [online]. wikipedia . Dostupné z URL: <https://en.wikipedia.org/wiki/Linear_interpolation#/media/File:LinearInterpolation.svg>.

[29] *Bilinear interpolation* [online]. wikipedia . Dostupné z URL: <https://en.wikipedia.org/wiki/Bilinear_interpolation#/media/File:BilinearInterpolation.svg>.

A Obsah priloženého CD

Priložené CD obsahuje tieto súbory:

- `/` ..... koreňový adresár priloženého CD
  - `C# skripty` ..... Zložka obsahujúca vytvorené C# skripty
    - `addvideo.cs`
    - `mouse.cs`
    - `movement.cs`
    - `pausescript.cs`
    - `vchange.cs`
    - `vchange2.cs`
    - `vchange3.cs`
    - `vchange4.cs`
  - `webhosting` ..... Zložka obsahujúca skripty uložené na serveri
    - `add_v_script.php`
    - `add_video.php`
    - `index.php`
    - `search_ingame.php`
    - `search_script.php`
    - `search.php`
    - `style.css`
    - `bg.jpg`
  - `Ovládanie a tagy.rtf` ..... Textový súbor popisujúci ovládanie a zoznam tagov
  - `Aplikácia.rtf` ..... Textový súbor s odkazmi na stiahnutie

V zložke *C# skripty* sa nachádzajú všetky vytvorené C# skripty používané v Unity engine v našom projekte. V zložke *webhosting* sa nachádzajú php skripty použité na internetovom portály. Textový súbor *Ovádanie a tagy.rtf* obsahuje návod k ovládaniu aplikácie a zoznam použitých tagov priradených k videám. Textový súbor *Aplikácia.rtf* obsahuje linkové odkazy na stiahnutie jednotlivých verzií aplikácie pre vybraný operačný systém. Aplikáciu je možno stiahnuť aj na adrese *http://dmedia.6f.sk* . Testovanie tejto aplikácie prebehlo na systémových verziách MacOS 10.14.6 a Windows 10 Home (verzia 1909).