Gymnázium Arabská, Praha 6, Arabská 14

Obor programování

Lrem Ipsum

Mobilní hra

Vojtěch Franc, Matěj Bittner, Jana Šrámková

Květen 2023

Prohlašujeme, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů udělujeme bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V ..... dne .....

Anotace

Předmětem této práce je vytvoření mobilní hry inspirované hrou Monumen Valley s využitím vývojového prostředí Unity. Podstatou hry je perspektiva, postava hráče se pohybuje po útvarech v prostoru a může mezi nimi přecházet podle pohledu kamery. Hra se skládá z jednotlivých úrovní a obsahuje 3D modely, animace a originální soundtrack.

Abstract

This work is centered around creating a mobile game, inspired by the game Monument Valley, using Unity development environment. The essence of the game is perspective, player's character moves along shapes in 3D space and can move between them according to the camera's view. The game consists of individual levels and features 3D models, animations and an original soundtrack.

1. Obsah

2. Zadání projektu

Hra se bude skládat z několika levelů, každý bude tvořen hlavolamem obsahujícím různé mechaniky a bude jejich prostřednictvím vyprávět příběh, který bude hráči sdělován povětšinou neverbálními prostředky. Tyto levely budou hráčem ovládané skrz pohyblivé prvky, některé levely budou navržené tak, že je bude možné otáčet celé. Jednotlivé levely budou poté rozděleny do několika kategorií podle prostředí, kterými bude například abstrakce, příroda, stavby a podobné. Tyto kategorie budou mít vliv především na vizuální stránku dané úrovně.

3. Úvod

Tato práce dokumentuje tvorbu mobilní hry, k jejímž mechanikám jsme se inspirovali u hry Monument Valley. K tvorbě hry bylo použito vývojové prostředí Unity, jakož i programy na vytváření 3D modelů a další, které v práci zmiňujeme zejména v rámci vzájemné kompatibility. Dokumentace dále uvádí i prvky hry, které byly nakonec z různých důvodů vyjmuty nebo nebyly vůbec implementovány.

Proces tvorby nebyl bez různých úskalí, které byly v podobě technických problémů, kterými se v této dokumentaci také zabýváme, nebo problémů s komunikací a efektivní dělbou práce. Přes tyto překážky můžeme předvést hru, která je, co se týče mechanik, prakticky dokončena, i když vždy je místo pro inovace. Co se nám do konce dovést nepovedlo je hlavně zamýšlený příběh a větší počet úrovní.

Největší zásluhu na této práci bezpochyby nese Vojtěch Franc, bez něj by nic z toho nebylo možné.

4. Použité technologie

Pro tvorbu této práce jsme se rozhodli využít herní engine Unity 3D. Důvodem byly možnosti využití schopností enginu práce s 3D grafikou, renderování a schopnosti budování mobilních aplikací. Alternativní možností bylo využití Unreal engine, který také podporuje zmíněné funkce a v porovnání s Unity nabízí lepší možnosti v oblasti dosahování fotorealismu, který by pro náš jednoduchý design a možnosti mobilních zařízení nepřinášel takovou přidanou hodnotu, jakou Unity nabízí v podobě podpory skriptovacích jazyků. V Unity je možné psát skripty v jazyce JavaScript nebo C#, který je podobný jazyku Java a který jseme se pro tento projekt rozhodli využívat, Unreal engine podporuje pouze jazyk C++.

Přínos užití programu Unity 3D spočíval tedy v možnosti snadné tvorby .apk souboru a interních knihoven pro práci s 3D objekty, shadery a jejich renderování. Mimo tyto funkce, externí balíčky zmíněné níže a defaultní UI objety jako je tlačítko, jsme nevyužili žádných templatů, interního systému pro fyziku ani dalších rozšíření z Asset store. Původ převzatých částí kódu je zmíněn v kapitole Zdroje využité pro tvorbu tohoto projektu.

Seznam použitých balíčků v Unity 3D

Název	Autor	Verze	Požadavky
Mobile	(Unity Technologies)	2.0.1	iOS 10 and Android 4.4 nebo vyšší
Post Processing	(Unity Technologies)	3.2.2	
Test Framework	(Unity Technologies)	1.1.31	
TextMeshPro	(Unity Technologies)	3.0.6	
Timeline	(Unity Technologies)	1.6.4	
Unity UI	(Unity Technologies)	1.0.0	
Universal RP	(Unity Technologies)	12.1.7	
Version Control	(Unity Technologies)	1.17.1	
Visual Scripting	(Unity Technologies)	1.7.8	
Visual Studio Code Editor	(Unity Technologies)	1.2.5	

Pro kooperace na tomto projektu jsme využili zabudovaný nástroj Google Plastic SCM, který sloužil také jako verzovací systém. Pro samotnou editaci kódu jsme použili program Microsoft Visual Studio Code.

Mezi programy použité pro tvorbu assetů do této hry patří:

3D modelování – Maxon Cinema 4D r25, Autodesk Maya, Pixologic ZBrush, Blender

Textury – Adobe Photoshop, Marmoset Hexels 3

Editace zvuků – Adobe Audition

Tvorba hudby – FL Studio (BBC Symphony Orchestra, Spitfire Audio Labs)

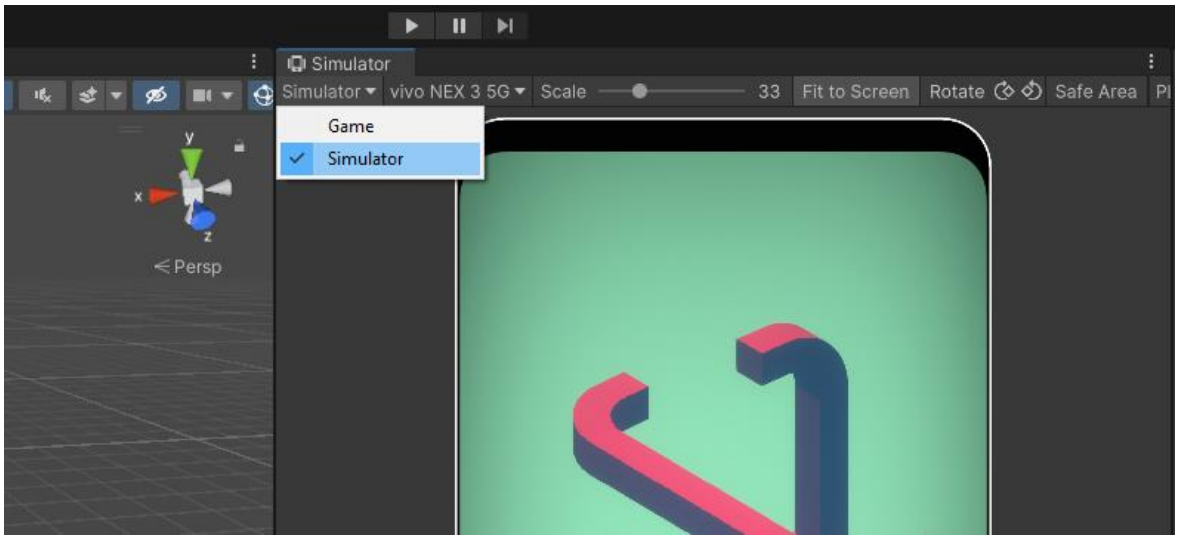
Editace hudby – PowerDirector 365

5. Použití

Pro správné spuštění projektu je zapotřebí mít nainstalovanou správnou verzi Unity 3D. My jsme pro jeho tvorbu použili Unity 3D **2021.3.4f1**. Pro další práci s tímto projektem je zapotřebí mít pro danou verzi Unity editoru nainstalovaný Android build suport modul. Projekt získáme naklonováním repositáře z Githubu, který je možné nalézt pomocí odkazu níže nebo naskenováním QR kódu.

https://github.com/gyarab/2022-3e-LremIpsum

V programu Unity 3D staženou složku otevřeme jako projekt a necháme stáhnout potřebné knihovny a balíčky. Před spuštěním projektu je nutné poznamenat, že byl zamýšlen na zařízení s dotykovou obrazovkou a standardní okno Game, ve kterém se hry spouštějí, nevnímá kliknutí jako vstupy z dotykové obrazovky, respektive funkce onTouch je zavolána, ale aplikace se nedostane k požadovaným informacím o dotyku. Řešením je hru spustit v módu Simulator, do kterého je možné se přepnout pomocí nabídky v levém horním rohu okna (viz obr. 1). V tomto režimu dotyky fungují. Alternativně je možné projekt spustit přímo na telefonu připojeném skrze USB se zapnutým laděním přes USB. Pro více informací doporučuji navštívit oficiální dokumentaci na adrese: *docs.unity3d.com/Manual/UnityRemote5.html.*



Obr. 1 Spouštění projektu

Pro vygengerování .apk souboru je nutné mít stažený Android Build Support module, Android Software Development Kit (SDK), Native Development Kit (NDK) a Java Development Kit. Podrobnější informace jsou k nalezení v dokumentaci na adrese: *docs.unity3d.com/Manual/android-sdksetup.html*.

6. Základní mechanika hry

Mechanika ovládání postavy je podobná hře Monument valley. Postava může chodit po vyznačených chodníčcích, které se skládají ze čtvercových dlaždic. Hráč klikne na dotykové obrazovce na požadovanou dlaždici a postava na ni dojde za předpokladu, že existuje mezi postavou a požadovanou dlaždicí cesta. Požadavkem pro tvorbu skriptu obstarávajícího pohyb postavy byla možnost poskytnout kódu sousedící dlaždice rozumnou formou pomocí Unity editoru při tvorbě jednotlivých levelů. Dlaždice jsou ve hře reprezentovány čtvercovou plochou nacházející se nad povrchem 3D modelu zbytku scény.

Některé dlaždice však díky využití optických klamů a pohyblivých prvků mohou spolu sousedit jen za určitých okolností. Jednou z možností bylo k těmto objektům přistupovat na základě jejich jména. To by mohlo fungovat tak, že by všechny dlaždice byly v hierarchii scény potomkem jednoho objektu, díky tomu bychom mohli všechny dlaždice projít v cyklu a pomocí jejich jména nebo jiných vlastností zjistit jejich sousedy. V případě využití jména by mohlo obsahovat číslice oddělené čárkou, kde by první číslice značila id dané dlaždice a všechna další čísla oddělená čárkou by byla id sousedících dlaždic. To by samo o sobě neřešilo nestálé sousedství dlaždic, které by bylo nutné řešit např. přidáním speciálního skriptu pro tyto sousedící dlaždice. Navíc by správné pojmenovávání bylo během vývoje nepřehledné a náchylné na chyby v případech zpětných úprav.

Další ze zamýšlených možností bylo proces hledání sousedů zautomatizovat. Při prvním spuštění hry by program na základě vzdálenosti jednotlivých dlaždic od sebe vyhodnotil, zdali jsou sousedící. Vytváření levelů by se tímto značně zrychlilo a případné změny by nevyžadovaly složité úpravy v systému zaznamenávání sousedů. Tento přístup by musel být doplněn řešením zabývajícím se sousedícími dlaždicemi, které spolu sousedí jen při splnění určitých podmínek. Navíc by bylo nutné zajistit, aby např. u pohyblivých prvků, které obsahují dlaždice od začátku hry sousedící s okolními dlaždicemi, nedocházelo k nastavení jejich trvalých sousedů s těmito prvky. Hlavní nevýhodou by byla částečná ztráta kontroly a možnost navázání nechtěných sousedství mezi dlaždicemi ve specifických místech.

Problém zaznamenávání sousedství jsem řešil použitím pole, které je možné v okně inspektoru upravovat a vkládat do něj objekty dlaždic. Jelikož v jednotlivých úrovních se vyskytují různě rozvětvené chodníčky a větší otevřené plochy (např. čtvercová plocha čtyř dlaždic) jsou výjimkou, hodí se do pole dlaždice vkládat v pořadí, v jakém na sebe navazují. Aby bylo možné přidat více takových „řetězců dlaždic“, bylo by ideální mít možnost dynamicky přidávat tato pole, což bohužel prostředí Unity neumožňuje. Nespeciální dlaždice se všechny vkládají do jednoho pole, konce sousedících úseků se značí vložením prázdného prvku do pole mezi dva souvislé úseky dlaždic v poli.

Pro dlaždice, které sousedí jen za specifických okolností, existuje oddělený game object, který obsahuje pro každý speciální spoj instanci skriptu s odkazy na dané dlaždice s nastavením zahrnujícím id booleanu určujícího platnost tohoto spoje nebo odkaz na třetí pomocnou dlaždici pro druh speciálního spoje, který vyžaduje teleportaci. Spoje, u kterých dlaždice spolu v 3D prostoru skutečně sousedí, zapnout funkci teleportace nepotřebují. V případech, kdy je zapotřebí postavu teleportovat v rámci hloubky z pohledu izometrické kamery, je možné do spojení přidat zmíněnou třetí dlaždici, která zpravidla překrývá dlaždici umístěnou níže (dále od kamery). Tato třetí dlaždice navíc opravdu sousedí s dlaždicí blíže ke kameře v 3D prostoru. Skript obsahuje i možnost nastavit, kterou z původních dlaždic pomocná dlaždice překrývá. Původně jsem se pokoušel problém s teleportací vyřešit tím, že by se postava na nesousedící dlaždici pohybovala stejně jako na jakoukoliv jinou dlaždici s možností nastavit pro tento přechod jinou rychlost a směr, kterým by měla postava hledět. Problém spočíval, že postava se viditelně zabořovala do 3D modelů ve svém okolí a řešení v podobě třetí dlaždice a teleportace se jevilo jako jednodušší než tvorba vlastního shaderu, který by byl vždy na vrchu před ostatními.

O seznam hodnot platnosti daných spojení se stará skript navržený pro každý level zvlášť, tyto proměnné jsou public pole booleanů ve skriptu level controller. Jde o jediné programování nutné k vytvoření nové úrovně. Možnost nahradit nutnost psaní tohoto skriptu pomocí několika univerzálních skriptů by byla možná, ale značně by to znepřehlednilo pravidla fungování v dané scéně, jelikož by nejspíše pro každou položku v poli musel být samostatný skript jako komponent nějakého objektu. Samotný skript řídící sousednost dlaždic zmíněný níže by byl dostatečný jako šablona obsahující většinu potřebné logiky pro tvorbu nekomplexních úrovní.

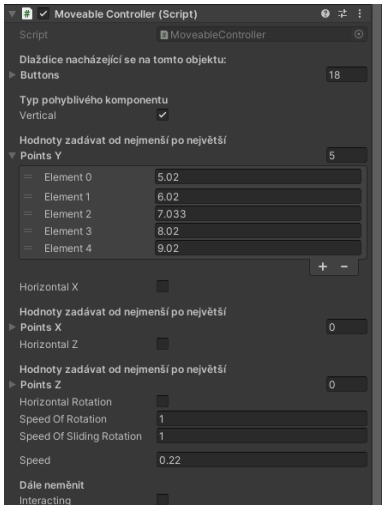
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class <NazevControlleru> : MonoBehaviour {
    LevelCotroller controller; // Skript objektu obsahující pole s booleanů
    public GameObject slider;  // Vytvoří v inspektoru kolonku na vložení GameObject
    public GameObject rotate;  // Objekt, který se otáčí

    void Start() {     // Nalezneme ve scéně skript kontrolující sousednost
        controller = GameObject.Find("LevelController").GetComponent<LevelCotroller>();
    }
    void Update() {
        float lvlRot = rotate.transform.rotation.eulerAngles.y; // Získání rotace
        float slideY = slider.transform.position.y; // Získání rotace
/* Sousednost platí v případě, že rotace je 90°. Rozsah od 89° do 91° je z důvodu
   použití vyhlazovací funkce, kde dosažení plných 90° trvá delší dobu */
        if(lvlRot > 89 && lvlRot < 91) {
            controller.IDOkolnosti[0] = true;
        } else {
            controller.IDOkolnosti[0] = false;
        } if(slideY > 6.9f && slideY < 7.1f) { // Sousednost platí, je-li Y = 7
            controller.IDOkolnosti[1] = true;
        } else {
            controller.IDOkolnosti[1] = false;
        }
    }
}
```

7. Rozhraní tvorby levelů

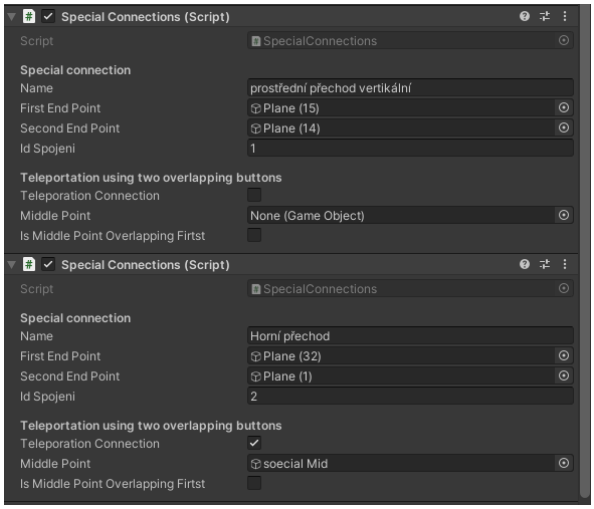
Každý level ve hře je reprezentován jako samostatná scéna, která musí obsahovat objekt Main Camera s možností využití komponentu Post-process, v takovém případě je nutné ke kameře přidat také skript PostProcessingController, který po spuštění nastaví své hodnoty a umožňuje animované přechody mezi scénami. Dále musí mít scéna nulový objekt nazvaný „LevelController“ se skriptem LevelController obsahujícím odkaz na nulový objekt s komponenty skriptů se speciálními spojeními. Nesmí chybět ani game object s názvem „Player“ a skriptem nazvaným „Player“. Pokud chceme ve scéně používat skript MoveableController, musíme mít ve scéně také skript DragController aplikovaný jako komponent třeba u stejnojmenného nulového objektu.

Do scény vložíme 3D modely, které nejprve naimportujeme do projektu v okně Assets controller a následně jej vložíme do scény. Pomocí komponentů Mesh Renderer můžeme nahradit naimportované materiály vlastními vytvořenými přímo v Unity. Tento model opatříme objekty plane, které tvoří dlaždice, na které může postava vstoupit, pokud jsme jimi model nevybavili již při modelování. Tyto objekty musí mít komponent některý collider komponent (Box Collider je zcela dostačující) a skript nazvaný „Button“, do kterého nahrajeme odkaz na prefab objekt s animací, který se na dané dlaždici objeví při kliknutí (používáme tento: Assets\Models\Click\ Click.prefab).

Do scény umístíme postavu nad jednu z dlaždic a vložíme odkaz na tuto dlaždici do skriptu LevelController. Dále tomuto skriptu do pole Buttons dáme odkazy na všechny sousedící dlaždice. Sousedící úsek dlaždic se do pole zadává ve stejném pořadí, tedy sousedící dlaždice v poli by spolu měly sousedit i ve skutečnosti. Úseky sousedících dlaždic se v poli oddělují vložením prvku s hodnotou null. Speciální spoje přidáme každý jako samostatnou instanci skriptu SpecialConnections s vlastním nastavením jako komponent vytvořeného nulového objektu, na který nastavíme odkaz ve skriptu level controller. Nakonec nezapomeneme nastavit správnou délku pole s id okolností, za jakých je sousedství platné. U hráče nastavíme aktuálního předka, nachází-li se pod některým.



Obr. 2 Moveble Controller



Obr. 3 Special Connection

Pokud chceme ve scéně používat pohyblivé části, musíme do ní jako komponent některého z objektů vložit skript DragController. Objekt, kterým chceme pohybovat, vybavíme colliderem vhodného typu a skriptem MoveableController, který bude mít v seznamu Buttons odkaz na všechny dlaždice, které se na něm nachází. Hra tak bude moci posouvat hráče i s daným objektem, bude-li na něm stát. Dále je nutné zaškrtnout jednu z možností, jak by se měl daný objekt pohybovat. Na výběr jsou posuny ve všech osách a rotace kolem osy y. Pokud vybereme posun, vyplníme do příslušného pole hodnoty pozic, ve kterých se může daný objekt nacházet, seřazené vzestupně vzhledem k indexům v poli. To nastaví rozsah možného pohybu a po ukončení posunu objektu uživatelem objekt plynule „doklouže“ do nejbližšího stanoveného bodu. Dále je zde možné nastavit rychlost posunu v závislosti na dotykovém vstupu uživatele. Tato hodnota je závislá na velikosti daných objektů, respektive na měřítku stanoveném hlavní kamerou.

Ukončení levelů je možné udělat implementací skriptu pojmenovaném „LevelEnder“, který je nutné nastavit jako komponent některého objektu ve scéně, jako např. objektu level controller. Do tohoto skriptu je nutné nastavit odkaz na dlaždici, na kterou když hráč vstoupí, tak má dojít k ukončení dané úrovně. Další možnost vstupu je textové pole, do kterého je možné zadat název scény, která se má načíst po ukončení stávající. Pokud toto pole zůstane nevyplněné, nastane návrat do hlavního menu, jehož jméno je uloženo jako statická proměnná ve skriptu pojmenovaném „GlobalVariables“.

Skript LevelController automaticky přidá UI tlačítko do pravého horního rohu s ikonkou domečku pro návrat do hlavního menu. Tuto funkci ovládá možnost zaškrtávacího políčka Home button, jehož výchozí hodnota je true. Po kliknutí na toto tlačítko se načte scéna hlavního menu. Hlavní menu je určeno hodnotou statické proměnné menuSceneName ve skriptu GlobalVariables . Důležité je také poznamenat, že aby bylo možné požadovanou scénu načíst, musí být uvedena v seznamu Scenes in build, který je možné nalézt v okně Build settings v záložce File.

7.1. Ukládání herního postupu

Ukládání herního postupu obstarává skript LevelController, který umožňuje tuto funkci vypnout odškrtnutím zaškrtávacího políčka Auto Save. Při povolení této možnosti ukládání levelů probíhá zcela automaticky každých 30 sekund, během kerých se uživateli objeví v levém horním rohu obrazovky točící kolečko značící načítání. Ukládání probíhá do souboru savedGame.bin, ve kterém je serializovaný objekt obsahující následující informace:

- Název levelu
- Objekt obsahující pozici a rotaci hráče
- Id dlaždice, na které se hráč právě nachází
- Pole booleanů sousednosti dlaždic
- Objekt obsahující pozici a rotaci kamery
- Pole objektů pozic všech Game objects majících jako komponent MoveableControler
- Id zničených Game objects majících jako komponent BlockForDestruction
- Pole additionalData typu float pro případné další informace

Pokud je tato funkce povolena, v GlobalVariables je loadFromSave nastaveno na true a jméno uloženého levelu se shoduje se jménem otevřené scény, dojde k načtení dat zmíněných v seznamu výše do aktuálního levelu. Když dojde po třiceti sekundách k uložení, původní data se přepíšou.

7.2. Camera controller

Pohyby hlavní kamery lze ovládat pomocí skriptu pojmenovaném „CameraController“, ten umožňuje několik nastavení pohybu kamery na základě pozice postavy, kterou kamera ve vybraných osách následuje s nastaveným offsetem. V jakém rozsahu má kamera postavu následovat lze nastavit pomocí pole dlaždic, na kterých když se hráč nachází, tak je pohyb kamery aktivní. Pohyb je možný pouze po jedné ze zvolených os najednou. Pro pokročilejší pohyby kamery nabízí tento skript funkci, která dostane jako vstup aktivační dlaždici a skript, který má v sobě zadané souřadnice, kam se kamera po stoupnutí postavy na danou dlaždici kamera posune.

Kamera má na sobě také skript starající se o postprocessing. Z možností postprocessingu v našem projektu nejvíce využíváme funkce vignette a bloom, které tento skript nastavuje. Vignette způsobuje tmavnutí v oblastech krajů obrazovky. Bloom může způsobovat záření světlých barev a materiálů používajících emmision chanel. Těchto vlastností se také užívá při přechodech mezi scénami.

7.3. Appearing block

V levelu pojmenovaném „Řetěz“ jsme se rozhodli pro možnost překonat vzdálenost jednoho bloku mezi dlaždicemi jako schopnost opačnou ke schopnosti rozbít blok. Při přiblížení postavy k danému přechodu by mělo dojít k částečnému zobrazení tohoto „mostu“ v závislosti na vzdálenosti od postavy. K implementaci této

funkce je zapotřebí mít vytvořený 3D objekt reprezentující most v plně viditelném stavu, ten musí mít jako komponent skript AppearingBlock. Ten obsahuje několik možných nastavení. První je možnost zadání vlastního textu, který se objeví po kliknutí na tento objekt, jelikož se v průběhu testování ukázalo, že přijde hráčům intuitivní klikat na tento most, jelikož předpokládali, že obsahuje dlaždice, což tak být nemusí (a v našem použití není). Další možností je omezit viditelnost bloku v závislosti na vzdálenosti hráče, což by mělo pomoci myšlence, že postava vytváří tento objekt ve snu pomocí své fantazie. Tuto vzdálenost rozhraní okna Inspektor umožňuje také měnit. Poslední možností, kterou by měl tvůrce levelu měnit, je možnost nastavit omezení viditelnosti v závislosti na hodnotě pole sousednosti dlaždic pod zadaným indexem.

Aby fungovala grafická stránka, musí mít objekt nastavený materiál v komponentu Mesh Renderer na FadedMat nalezitelný ve složce: assets/Materilals. Aby fungovala interakce s hráčem, musí mít na sobě objekt hráče jako komponent připojený skript Mask Controller, který umožňuje nastavit velikost viditelného okruhu kolem hráče a velikost rozhraní mezi viditelnou a neviditelnou částí.

7.4. Sound manager script

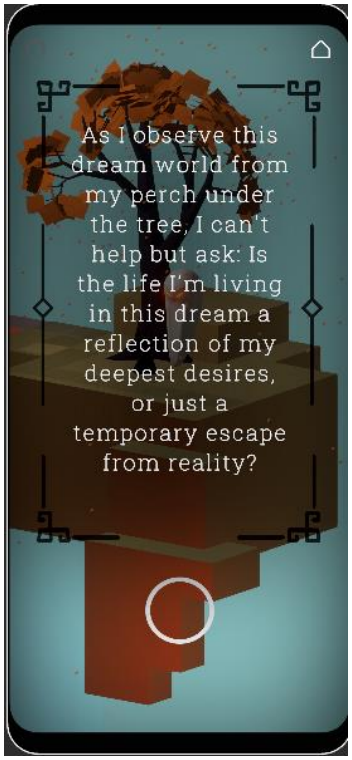
Do projektu je zvuková složka implementována pomocí skriptu nazvaném sound manager, který je možné přidat do úrovně stejně jako jiné skripty (LevelController apod.). Skript umožňuje vložení jednoho zvukového souboru, který slouží jako podkladová hudba, která se bude přehrávat dokola po spuštění levelu. Dále je možné vložit pole volitelné délky krátkých zvuků, které se budou přehrávat po kliknutí na tlačítka, která jsou součástí UI. Tyto zvuky také spouští kliknutí na dlaždici, na kterou může vstoupit postava. Z tohoto pole se zvuky vybírají náhodně, aby nedocházelo k opakování stále stejných zvuků.

Možnost přehrát zvuk dostávají i ostatní skripty pomocí statické funkce tohoto objektu, tento skript se ale pro přehrání zvuku se musí v dané scéně nacházet. Poslední pole umožňující vložení zvukových souborů přehrává tyto zvuky při pohybech pohyblivých součástí levelu. To funguje vypočítáváním vzdálenosti u každého z těchto objektů od poslední pozice, ve které přehrály zvuk, který byl také náhodně vybrán z daného pole.

7.5. Okno pro vyprávění příběhu

Jako primární způsob komunikace příběhu jsme zvolili formu jednoduchého dialogového okna, které umožňuje vyprávění v podobě textu, který je možné procházet tlačítkem na spodní části obrazovky. Jde pouze o jednosměrný způsob komunikace bez možnosti interakce hráče. Zobrazení tohoto okna může být aktivováno vstoupením na konkrétní dlaždici, kterou lze zadat do tohoto skriptu. Je možné tuto funkci do scény přidat importem prafeb, který se nachází v projektu v následující složce: Assets/Resources/RozhovorCanvas/Rozhovor Canvas.prefab. Kořenový objekt tohoto prefab objektu obsahuje skript Dialog Canvas Controller, ve kterém je možné nastavit aktivační dlaždici. Při stoupnutí na tuto dlaždici dojde ke zobrazení tohoto okna a

částečnému překrytí zbytku hry. Druhá vstupní kolonka obsahuje odkaz na objekt typu Text Mesh, který je potomkem tohoto kořenového objektu a měl být již nastaven. Nastavení slouží pro možnost snazšího vytvoření tohoto okna s odlišným formátováním textového pole. Položka Duration obsahuje číslo typu float, které udává ve vteřinách dobu, kdy dochází ke kontrole, stojí-li hráč na aktivační dlaždici. Poslední největší položka je pole textových polí, které umožňuje zadat texty objevující se na jednotlivých stranách tohoto okna v zadaném pořadí. Po přečtení posledního textu stisknutí posledního tlačítka, které jinak posouvá na další text, dojde k zavření tohoto okna a jeho opětovná aktivace je možná pouze opětovným stoupnutím na aktivační dlaždici po jejím opuštění.



Obr. 4 Ukázka okna pro vyprávění příběhu

7.6. Blocks for destruktion

Bloky určené k destrukci jsou objekty ve hře, které je nutné odstranit, aby bylo možné se přes ně dostat pryč. To je možné učit kliknutím na červený kruh, který je animovaný a mění svou velikost, když se hráč přiblíží dostatečně blízko. Používat model jako speciální blok k destrukci je možné přidáním skriptu BlockForDestruction do jeho seznamu komponentů. Jelikož je hlavním účelem tohoto bloku blokovat cestu, první vstupní pole určuje id okolnost sousednost dlaždic řídícího skriptu, které se po zničení daného objektu nastaví na true. Další pole umožňuje nastavit maximální vzdálenost od hráče, kdy je červený kruh stále viditelný.

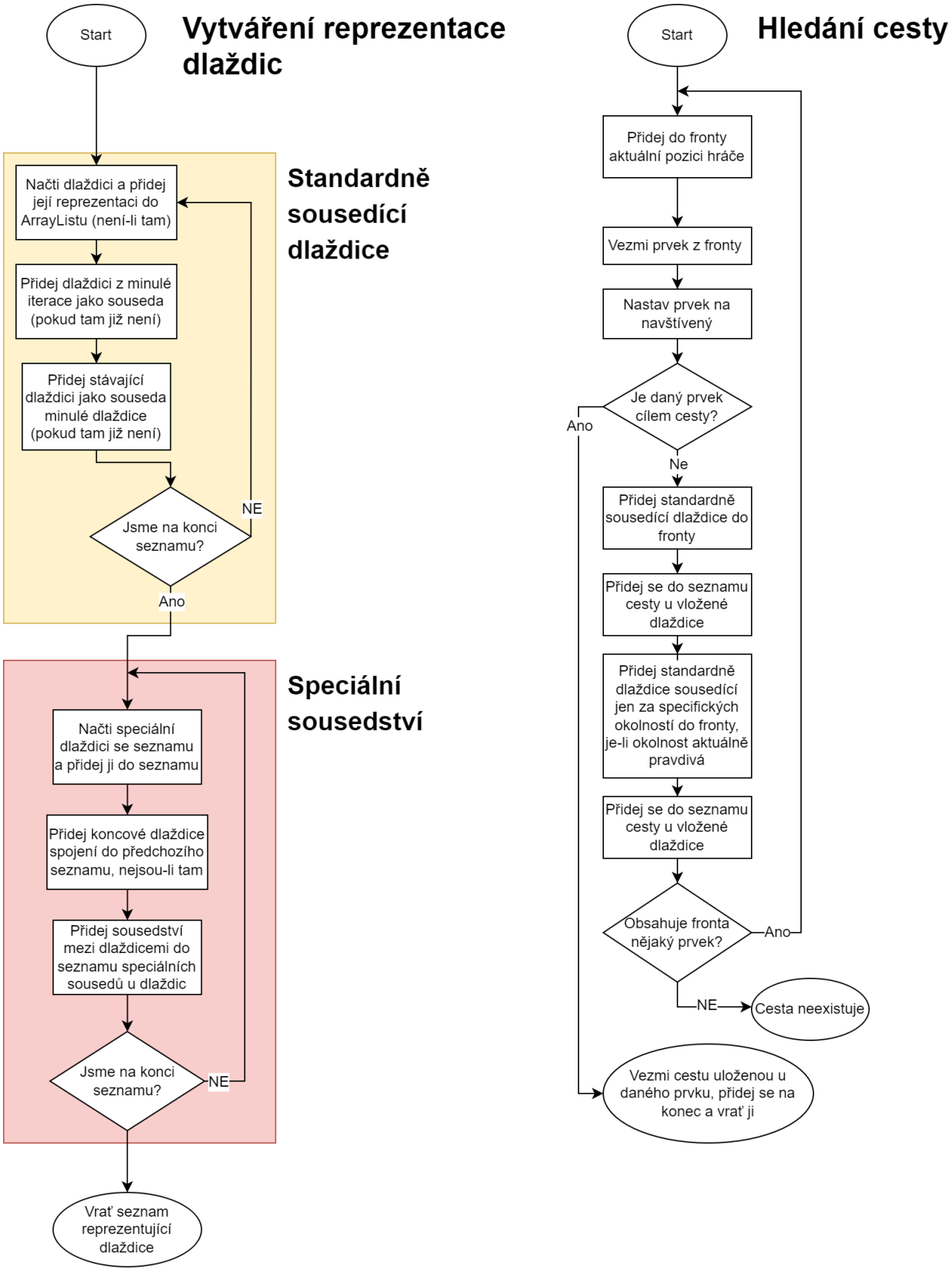
8. Hledání cesty

Informace o sousedících dlaždicích program získává z pole s odkazy na dlaždice ve skriptu LevelController. Po prvotním spuštění si program uloží objekty reprezentující dlaždice do ArrayListu, kde do seznamu každého tohoto objektu přidá odkaz na jeho sousedy. Mimo seznam standardních sousedů má každý vrchol grafu také seznam speciálních sousedů. Při kliknutí na dlaždici, která má na sobě skript Button, se program pokusí nalézt cestu z pozice, na které se nachází hráč. Při hledání této cesty se uvažují aktuálně platné speciální spoje. Problém může vzniknout, pokud dojde ke změně přístupnosti dlaždic v průběhu chůze postavy. Zvažovaným řešením bylo po dobu přesouvání postavy znemožnit hráči s pohyblivými částmi hýbat. To by znamenalo značné omezení hráče, jelikož se postava přesouvá pomalu a mohlo by také dojít k zmatení hráče ohledně mechaniky hry. Hráč tedy v naší verzi může v průběhu přesouvání pohybovat s pohyblivými prvky a postava si musí vždy před vkročením na další speciální dlaždici zkontrolovat její přístupnost. Problém by se stále mohl vyskytnout v případě, že by k posunu došlo bezprostředně po zkontrolování přístupnosti cesty. Postava by v takovém případě klouzala za cílovou dlaždicí mimo vyznačené cesty. Výsledný kompromis je, že právě v případě přesouvání postavy mezi těmito dvěma dlaždicemi je pohybování s pohyblivými prvky hráči znemožněno.

Pozice postavy se nastaví na příslušnou dlaždici až po dokončení přesunu na ni, počáteční dlaždici, na které stojí postava po spuštění hry, je zapotřebí nastavit v inspektoru ve skriptu LevelController. Změna cíle je během chůze postavy umožněna v případě, je-li k zadanému cíli platná cesta, pak dojde k nahrazení postupu cesty novým postupem.

Samotný proces hledání cesty potřebuje jako vstupní informace zhotovený graf dlaždic. Ten vyhotoví skript LevelController ze seznamu sousedících dlaždic (viz diagram níže). Výsledkem je ArrayList objektů pojmenovaných Point, které obsahují odkaz na příslušnou dlaždici, obsahují také seznam sousedů (ArrayList s odkazy na další objekty Point). Taktéž obsahuje seznam speciálních dlaždic a další proměnné, jako je přiřazené id dané dlaždice, seznam na ukládání cesty z dlaždic apod.

8.1. Vizualizace hledání cesty

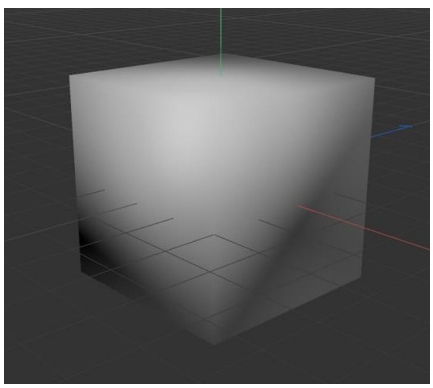


9. Specifika modelování za účelem tvorby her

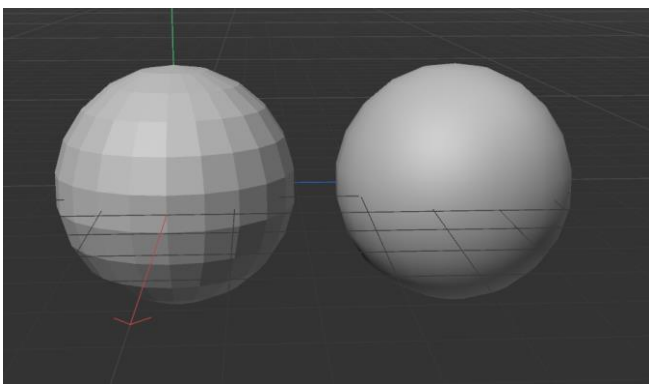
Pro tvorbu modelů jsme využili software Maxon Cinema 4D r25, Autodesk Maya 2022 a částečně Blender v 3.4. Jelikož cílem bylo použít modely v Unity 3D, bylo nutné přizpůsobit proces modelování možnostem tohoto herního engine. Modely určené pro tvorbu her se modelům pro jiné účely liší v několika oblastech.

Topologie se u modelů pro hry výrazně liší od ostatních modelů zejména kvůli nutnosti optimalizace, pro kterou je nejvýhodnější trojúhelníková topologie, která by neměla být detailní s tím, že se detaily dodají pomocí dodatečných textur. Jelikož cílíme na jednoduchou grafiku bez snahy o fotorealismus, tak jsme detaily tímto způsobem nepřidávali. Ačkoliv pravidla správné topologie nejsou striktně sepsána a ohledně správné topologie existuje mnoho různých názorů založených na osobních zkušenostech. Existuje shoda na několika základních pravidlech, které by měla topologie splňovat. Jedním z těchto pravidel je striktní nedoporučení používání n-gonů (polygon s více jak čtyřmi hranami), které některé programy jako Pixologic Zbrush neumožňují. Za nejideálnější topologii je možné považovat mesh tvořenou pouze čtyřúhelníkovou topologií respektující přirozené hrany svým edge flow s vrcholy, které netvoří více jako pět hran. Nejnáročnější na topologii jsou modely určené na rigging a následnou animaci, kde topologie hraje klíčovou roli pro podobu deformace při animaci. U statických modelů topologie hraje klíčovou roli při vyhlazování a také usnadňuje práci popř. úpravy modelu.

Vyhlazování u modelů může být dosaženo několika způsoby. Při tvorbě 3D modelů je prioritou udržet množství geometrie co nejnižší pro snadnou práci. Metoda tvorby subdivizového povrchu využívá Catmull-Clarkův algoritmus, který upravuje geometrii přidáváním další geometrie s tím, že výsledná topologie je vždy čtyřúhelníková. K dosažení dokonale hladké geometrie by bylo nutné přidat nekonečné množství nové geometrie. Aplikujeme-li Catmull-Clarkovo vyhlazování na krychli, vyhladíme ji až do podoby koule. Phongovo stínování vyhlazuje interpolací normály polygonu, což má za následek změnu v odrážení světla, kdy dojde k „zahlazení“ hran. Jde ale jen o iluzi, tudíž tímto způsobem nevyhladíme krychli na kouli. Navíc tato metoda funguje dobře, pokud úhel mezi sousedícími polygony není příliš velký, proto se tyto dvě metody v praxi kombinují.

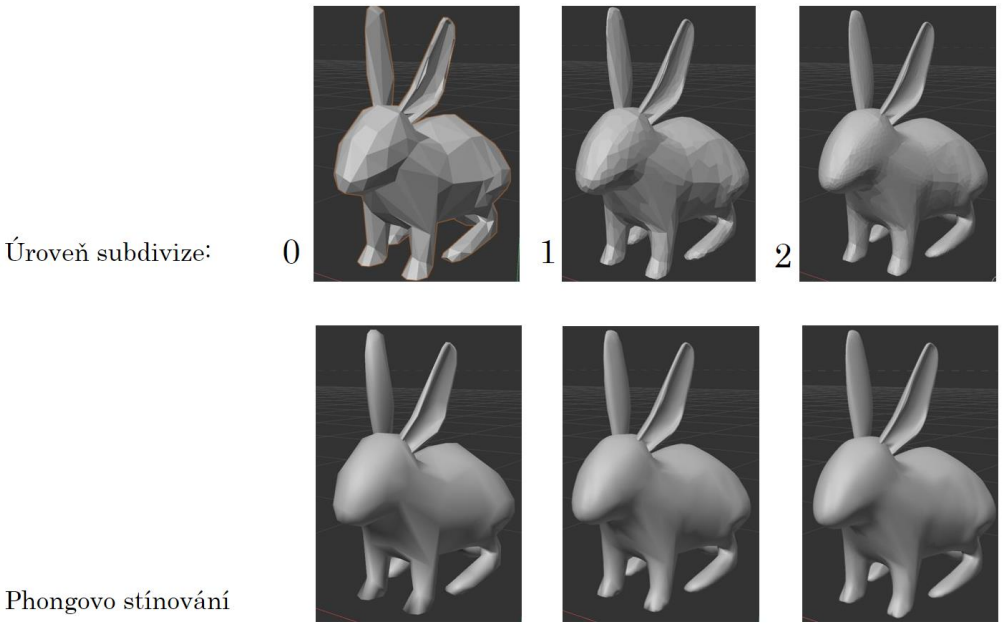


Obr. 5 Phongovo stínování na krychli



Obr. 6 Porovnání koulí s Phongovým stínováním a bez

Subdivizovaný povrch se na model obvykle aplikuje v plné míře výhradně při renderování kvůli náročnosti na vykreslování. Unity v základu subdivizovaný povrch vytvářet neumí, modely je nutné subdivizovat již před exportem. Phongovo stínování Unity nabízí s několika základními nastaveními jako je nastavení limitního úhlu.



Obr. 7 Znázornění kombinace subdivizovaného povrchu a Phongova stínování

Materiály jsou omezeny možnostmi real-time renderování. Nejvíce se tato omezení projevují u fotorealistických materiálů vyžadujících ray tracing, jedná se hlavně o částečně průhledné materiály lámající světlo, kdy není možné používat vlastnost materiálů jako IOR (index of refelction) nebo SSS (subsurcace scattering). Materiály používané pro potřeby real-time renderování využívají některé vlastnosti navíc, jako jsou AO mapy (ambient occulsion map) nebo parallax mapy. [1]

10. Možnosti a limitace Unity 3D

10.1. Import 3D souborů

Unity podporuje načítání 3D dat dvou rozdílných typů souborů. Prvním typem jsou 3D exportní soubory, konkrétně .fbx, .dae, .3ds, .dxf a .obj. Výhodou tohoto typu je možnost vybrat jen část existujícího projektu na export, který obsahuje jen vybrané množství nezbytných informací, soubor je tedy menší. Nevýhodou je, že není možné již importovaný model upravovat a změny je v něm možné udělat jen jeho opětovným importováním s provedenými změnami. [2]

Druhý typ jsou soubory projektů 3D aplikací. Unity podporuje formáty programů: Autodesk 3ds Max, Autodesk Maya, Blender a Modo. Unity tyto formáty automaticky převádí do formátu .fbx pokaždé, když dojde ke změně v souboru projektu. Výhodou je snadná možnost modely v průběhu produkce upravit, nevýhodou je ukládání přebytečných dat do projektu a nutnost nainstalovaného licencovaného software na daném počítači k provedení konverze.

Do verze 2019.3 Unity podporovalo projekty programu Maxon Cinema 4D, od této verze Unity nativně .c4d nepodporuje a odkazuje na rozšíření do Unity s názvem Cineware by Maxon, který podle osobních zkušeností a reviews v Asset storu zhruba od roku 2021 nefunguje. Z první kategorie je nejpoužívanějším formátem v případech, kdy nejde o holou geometrii, .fbx (Filmbox – formát vlastněný společností Autodesk od roku 2006), který podporuje:

- Geometrii (normály, UV souřadnice)
- Textury
- Barvy
- Světla
- Kosti a jejich vliv na vrcholy geometrie
- Efekty (např. mlha)
- NURBS (non-uniform rational B-spline)
- Více stop animací

Formát .obj je pak používaný pro import základní geometrie (pozice vrcholů, UV souřadnice, polygony a normály) s omezeným přesahem do barev a materiálů. [3]

10.2. import animací pomocí formátu .fbx

Animace můžeme rozdělit na několik typů podle parametrů modelu, které animujeme. Asi nejjednodušší je animovat základní vlastnosti modelu jako je jeho pozice nebo rotace. Pokud bychom chtěli měnit jeho tvar, můžeme využít několik přístupů. Jedním z nich je vytvoření Ik chain (řetězec kostí řídící se inverzní kinematikou – nejčastější způsob animace končetin postav a živočichů), který propojíme s modelem tak, že každá kost bude mít jasně definovaný svůj vliv na vrcholy modelu. Když dojde k animaci kostry pomocí základních parametrů pozice a rotace,

animace se převede podle vlivu na zbytek modelu. Dalším z přístupů je využití některého z modifikátorů v používaném 3D software nebo ruční animaci jednotlivých vrcholů. Modifikátory .fbx nepodporuje, a tak je před exportem nutné je převést na animaci vrcholů (bake animation). PLA (Point Level Animation) je částečně podporována formátem .fbx s tím, že informace o samotné animaci (point chahe files) jsou uložené mimo soubor .fbx a unity tento způsob animace nativně nepodporuje [3]. Existují rozšíření umožňující import PLA pomocí point chahe files k již naimportovanému modelu, jejichž cena se pohybuje v rozmezí od 45 do 150 Eur.

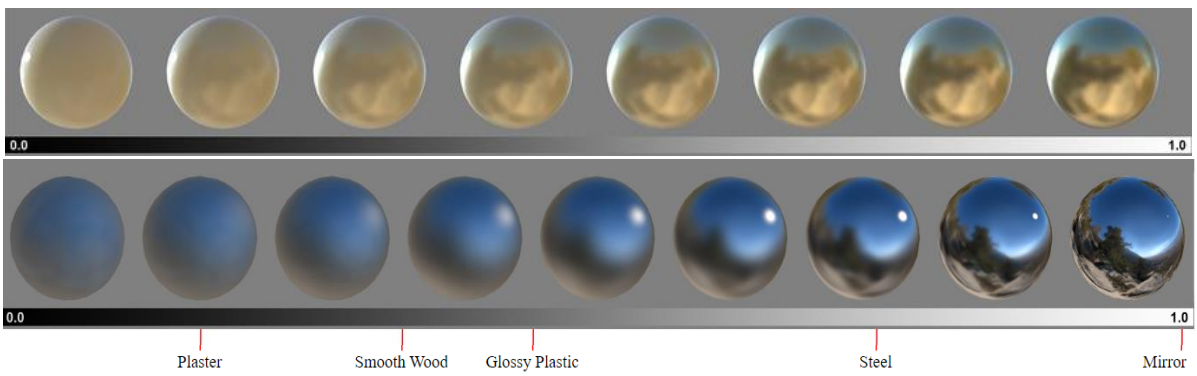
Za zmínku stojí také animace tvořené pomocí blend shapes, kdy existuje několik kopií modelu v různých verzích a animace spočívá v kombinování rozdílných tvarů různých verzí na originálním modelu. Tato funkce je formátem .fbx podporována, ale unity ji importovat nedokáže. [4] [5]

Naše možnosti importu animací zůstaly u animace základních vlastností modelu a užívání kostí k deformacím, což je značná limitace omezující možnosti importu mograph animací (viz kapitola mograph). K animaci základních vlastností Unity nabízí svůj vlastní jednoduchý animační editor, který nelze použít k animaci vrcholů, jelikož Unity neumožňuje přístup k jejich editaci.

10.3. Materiály

Unity nabízí několik základních shaderů pro specifické účely (UI, skybox atd.). Jako materiál pro mesh slouží Standard shader, který nabízí následující mapy:

- **Albedo** – Základní barva materiálu nezahrnující vlivy osvětlení. V Unity není možné přidat samostatně alpha mapu, informace o alpha kanálu se získávají z albedo textury a průhlednost samotná je aplikována na základě aktuálního nastavení rendering mode daného materiálu. Tyto nastavení jsou čtyři:
  - **Opaque** je defaultní hodnota nepodporující průhlednost.
  - **Cutout** umožňuje buď 100%, nebo 0% průhlednosti.
  - **Transparent** učiní neviditelnou barvu materiálu v průhledných oblastech, ale zachová ostatní vlastnosti materiálu, jako je odrazivost.
  - **Fade** učiní průhledné všechny kanály materiálu v daných oblastech. [6]
- **Metallic** – Určuje odrazivost materiálu. Obsahuje dodatečné nastavení parametru smoothness, které má vliv na hladkost odrazové plochy. [7]



Obr. 8 Znázornění vlivu kanálů metalic a smoothness na materiál

Normal map – Mapa obsahující informace o směru vektoru normály v daném místě za účelem tvorby umělého efektu plasticity objektu. Hodnoty RGB každého pixelu normálové mapy reprezentují směr normálového vektoru v osách x, y, z, kde osa z reprezentuje směr vzhůru, což nebývá zvykem, jelikož v programech jako Autodesk Maya, Blender, Cinema 4D, Unity 3D, Autodesk 3d studio max zastává směr vzhůru osa y. Hodnoty RGB se pohybují v rozmezí 0 – 1 a výpočet na směr normálového vektoru spočívá ve vynásobení hodnoty dvěma a odečtení jedničky. Díky domu může vektor nabývat i záporných hodnot v rozmezí od -1 do 1. Hodnota původního směru vektorové normály je (0, 0, 1). Barva pro takovýto vektor by odpovídala hodnotám (0.5, 0.5, 1), což je právě fialová barva typická pro normálové mapy. [8]

Obr. 9 Ukázka normálové mapy

- **Height map** – Mapa známá také jako parallax map je podobná normálové mapě se kterou se často používá v kombinaci. Jedná se o jednoduchou černobílou texturu, kde světlejší odstíny značí vyvýšeninu na povrchu a tmavé odstíny prohlubně. Mapa vytváří efekt, kdy části bližší ke kameře se budou zdát mírně roztažené a zadní části budou zmenšeny a zakryty předními částmi. [9]
- **Occlusion** – Mapa okluze je tvořena texturou ve stupních šedi a slouží k označení míst, která by měla být zasažena přímým osvětlením (bílá) a míst, která by neměla být těmito světly ovlivněna, povětšinou jde o místa v záhybech modelu. [10]
- **Emission** – Jedná se o kanál, který přidává materiálu možnost stát se zdrojem světla, má parametry barva, intenzita a nastavení globální iluminace, která určuje, jestli popř. jak budou tímto světlem ovlivněny objekty v okolí. Buď s možností none nebudou objekty v okolí nijak ovlivněny nebo s možností realtime se bude každý snímek odrazivost světla od okolních modelů přepočítávat. Poslední volba je baked, kdy dojde předem k vypočítání vlivu světla na okolí, které se poté použije a v případě změny ve scéně se nezmění. [11]
- **Secondary maps** – Možnost vybrat detailnější a kvalitnější albedo mapu a normálovou mapu, které se použijí v případě, že je kamera ve větší blízkosti objektu. [12]

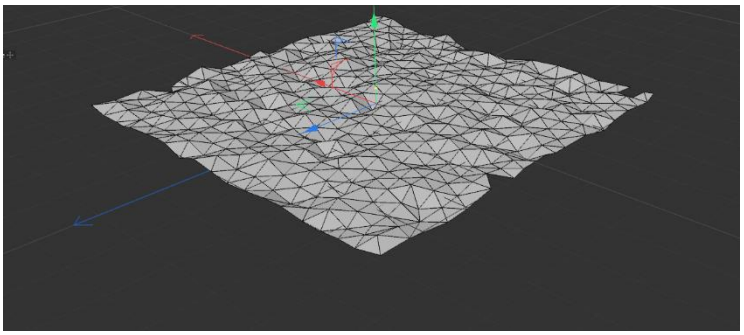
11. Mograph

11.1. Animované moře

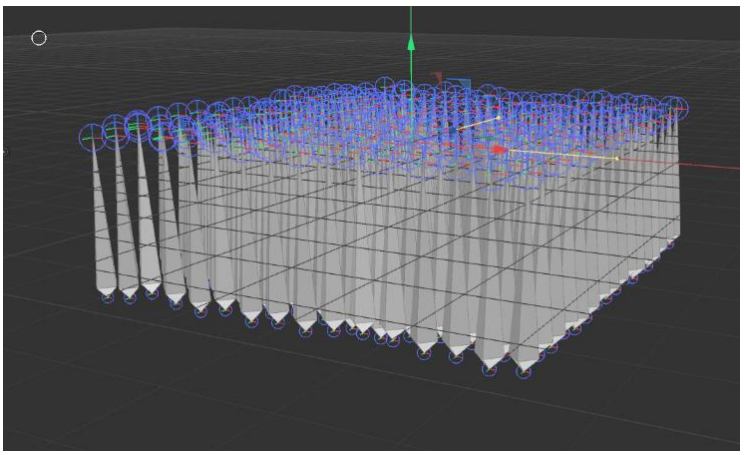
Animované low poly moře se skládá ze základní mesh s požadovanou topologií, na kterou se aplikuje modifikátor pozice vrcholů v ose y, který se řídí shaderem šumu. Pro náš případ jsem trianguloval čtyřúhelníkovou mesh z primitivního tvaru plane v programu Cinema 4D, mezi jehož přednosti patří tvorba mograph. Na výslednou mesh jsem dále aplikoval modifikátor displacer, který dostal jako parametr standardní šum upraven nastavením měřítka a nastavením jeho animace. K tomu bylo u animace několika typů šumů, včetně základního, možné nastavit periodu opakování animace šumu, kdy je možné pouštět animaci stále dokola s tím, že na sebe jednotlivé dané cykly navazují. Toto nastavení omezuje použití dalších funkcí, jako je pohyb šumu, který by v případě moře mohl naznačovat pohyb vln jedním směrem a tím naznačovat přítomnost větru. Opakování v cyklaech by ale nenavazovalo, jelikož by byl daný shader posunu. Celá animace byla 3 sekundy dlouhá, což při nastavení 30 fps dělá celkových 90 snímků.

Tuto animaci by bylo standardně možné exportovat jako PLA, což bohužel Unity nepodporuje. Jediným způsobem deformace modelů je využití kostí. Cílem tedy bylo převést animaci pohybu moře na kosti, které by následně deformovaly model. Do toho se postavily omezení programu Cinema 4D, který nemá pro tyto účely zabudované funkce, jelikož mograph pomocí kostí není standardní. Mým plánem bylo vytvořit vertikálně postavené kosti rozprostřené po povrchu plane, na které bych mohl aplikovat změnu jejich pozice na základě shaderu. Vytvořil jsem nejprve tuto kost, kterou jsem pomocí mograph nástroje Clone pravidelně rozprostřel po povrchu plane. Nejprve jsem z instancí jedné kosti udělal samostatné kosti, které jsem propojil s povrchem, ale pak nastal problém s aplikováním shaderu šumu. Lepší se ukázalo nechat kosti jako instance nástroje Clone a shader vložit jako jeho efektor. Díky tomu se podařilo naanimovat pohyby kostí na základě opakujícího se šumu. Instance kostí se ale nedají použít k propojení s vrcholy a při snaze zbavení se procedurálního klonování vznikly samostatné kosti v pozici, kde se v danou chvíli nacházely bez animace. Zkoušel jsem i funkci bake animation, která vracela stejný výsledek. Nakonec jsem zjistil, že požadovaného výsledku lze dosáhnout exportem celé scény s animacemi do formátu .fbx a následným importováním tohoto souboru. Tímto způsobem jsem získal samostatně animované kosti, které jsem spojil s vrcholy. Nutné bylo jen nastavit počáteční stav shaderu jako počáteční deformaci mesh, jinak by povrch moře v každém cyklu prošel fází dokonale rovné hladiny.

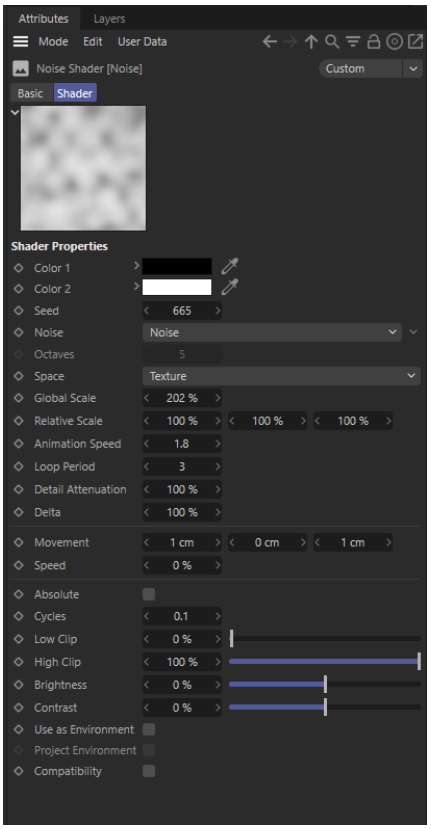
Unity vzniklou animaci dokázalo zpracovat, ale z nějakého důvodu nedocházelo k přepočítávání normály během pohybu, tudíž se odstín barvy částí moře během pohybu neměnil, a to i přes pokusy měnit různá nastavení renderování daného modelu. Změny nejsou tak výrazné a podle mého to nepředstavuje výrazný problém, který by se měl řešit např. tvorbou vlastního shaderu.



Obr. 10 Aplikovaný displacer hladiny moře



Obr. 11 Kosti ovlivňující vrcholy povrchu hladiny moře

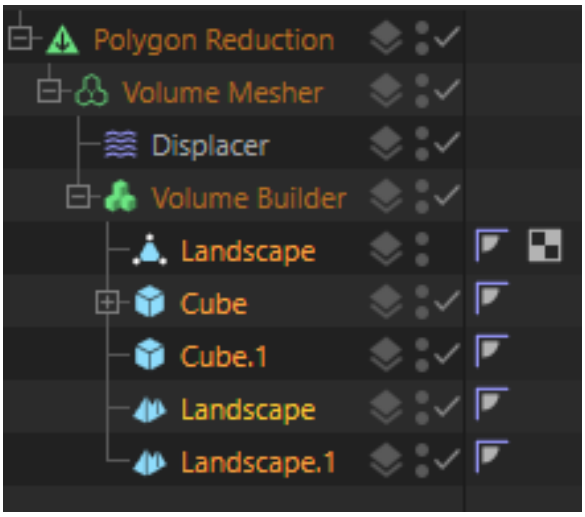


Obr. 12 Shader hladiny moře

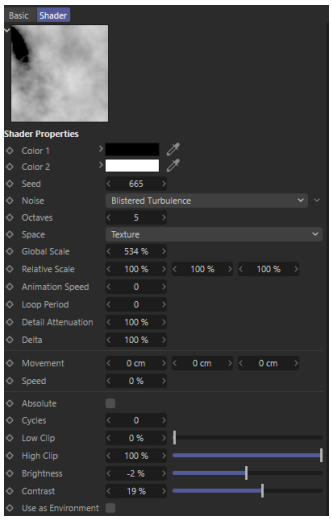
11.2. Generování pohoří

Metody procedurálního modelování na rozdíl od standardních postupů spočívají v tvorbě systému (většinou ve formě nodes), který vyrábí daný model podle zvolených parametrů. V porovnání se standardními postupy se procedurální modelování může zdát obtížnější a více časově náročné, proto je jeho využití výhodné v situacích, kdy je daný model rozsáhlý a je velká pravděpodobnost, že bude docházet k jeho úpravám. Právě ve flexibilitě procedurálně generovaných modelů spočívá jejich hlavní výhoda, jelikož není obtížné se vrátit o několik kroků zpět, něco poupravit a nechat model znovu vygenerovat.

Pro tento projekt jsem se rozhodl vytvořit systém generující pohoří v programu Cinema 4D. Cílem není generování terénu, ale úprava jeho povrchu. Tento systém tedy potřebuje jako vstup model nebo několik modelů, které jako první spojí převedením na voxely. Ty se v dalším kroku převedou zpět na jednu polygonovou síť, na kterou se aplikuje deformátor displacer, který na základě generovaného šumu zdeformuje povrch. Výsledná mesh projde na konci polygonovou redukcí, jejímž výsledkem je trojúhelníková topologie, která by měla zachovávat hlavní hrany.



Obr. 13 Object manager generátoru pohoří



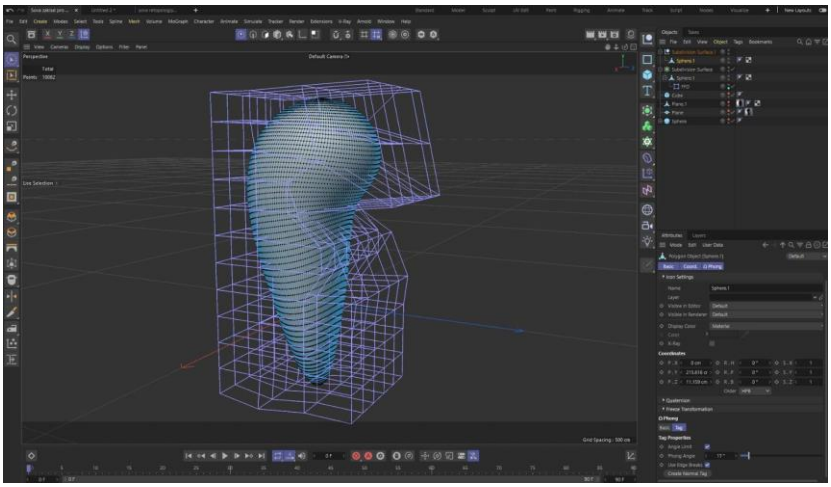
Obr. 14 Shader generátoru pohoří

12. Ukázky

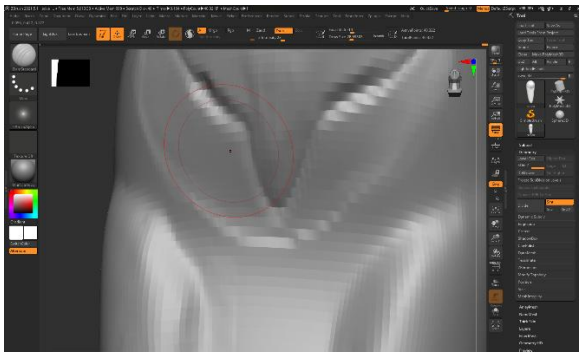
12.1. Modelování vedlejší postavy

Podle předlohy jsem začal vedlejší postavu sovu modelovat z primitivního tvaru koule, kterou jsem pomocí FFD deformátoru (Vytvoří kvádr kolem objektu, kdy deformace daného kvádru se aplikují na objekt.) upravil na základní hlavní tvary modelu. Poté jsem model editoval v programu Pixologic Zbrush, kde jsem provedl subdivizi povrchu a pomocí nástrojů na sculpting do modelu přidat detaily. Jelikož výsledkem byla hustá a příliš podrobná geometrie, retopologizoval jsem ji v programu Autodesk Maya.

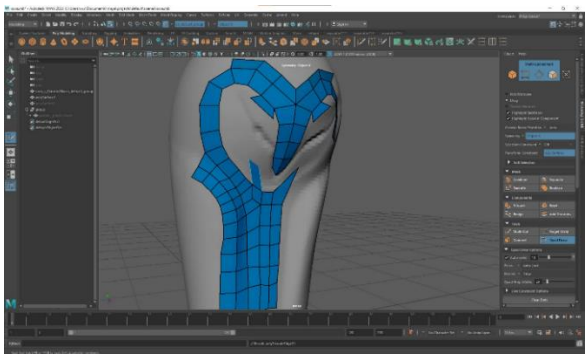
Obr. 15 Předloha



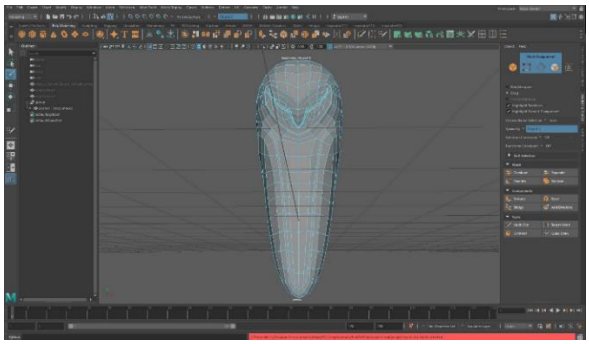
Obr. 16 Užití FFD modifikátoru



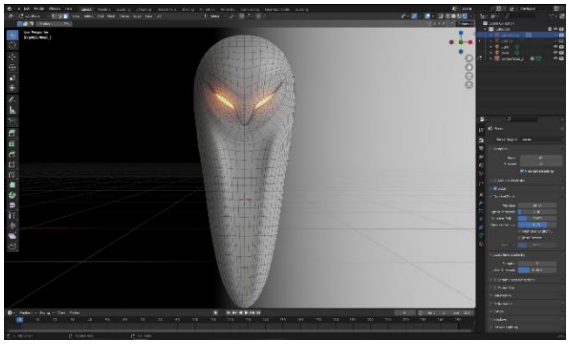
Obr. 17 Scuplting základního tvaru v ZBrush



Obr. 18 Retopologizace v Autodesk Maya



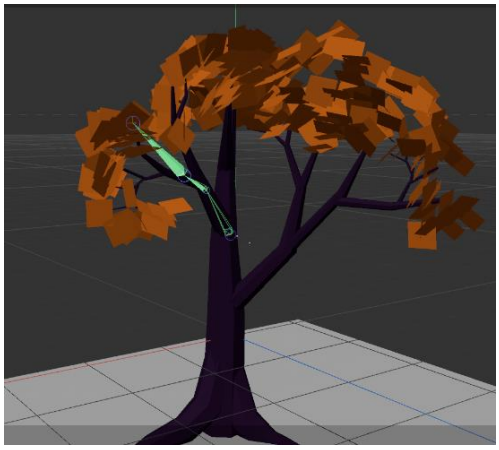
Obr. 19 Výsledná topologie po retopologizaci



Obr. 20 Výsledný render v Blenderu

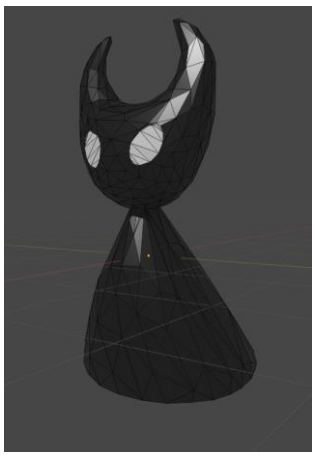
12.2. Animovaný strom

Obr. 21 Model stromu

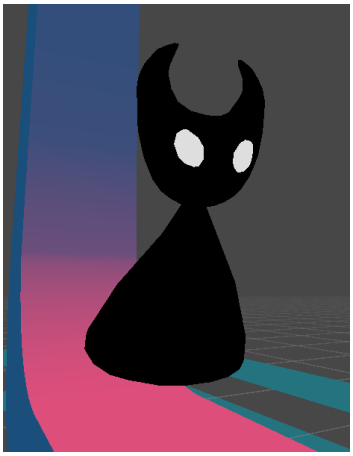


Obr. 22 Rigging stromu za pčelem animace

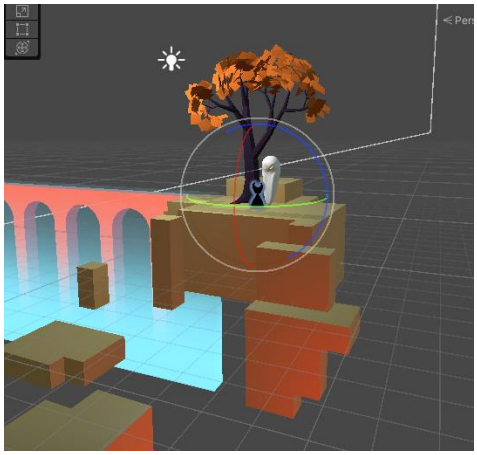
12.3. Hlavní postava



Obr. 23 Postava wireframe



Obr. 24 Postava ve hře

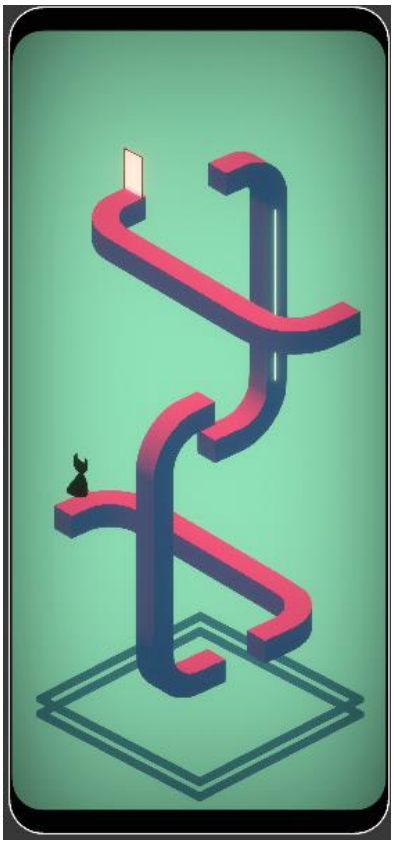


Obr. 25 Postava při rozhovoru

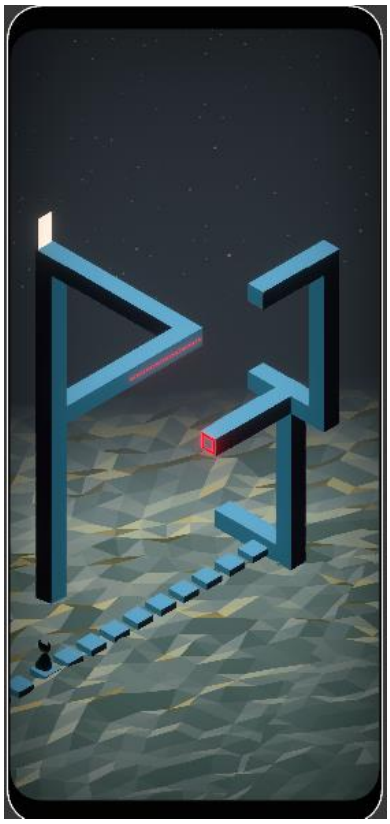
12.4. Ukázky herních scén

Obr. 26 1. level – pod stromem

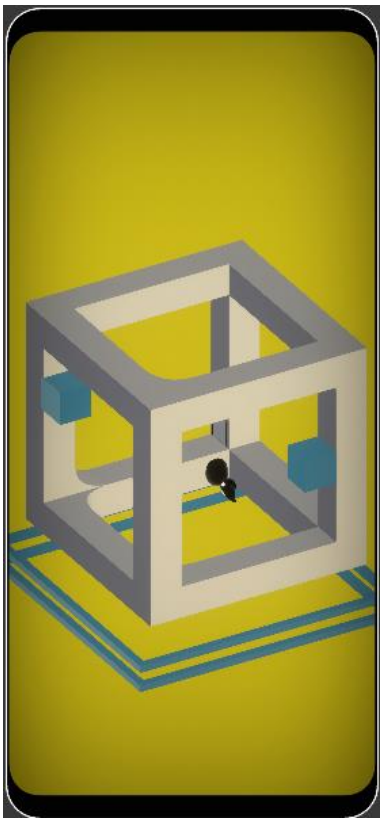
Obr. 27 1. level - most



Obr. 28 Level Most

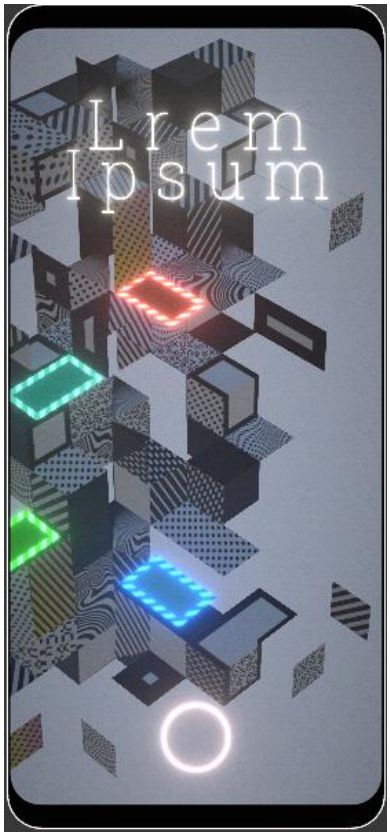


Obr. 29 Level Trojúhelníky



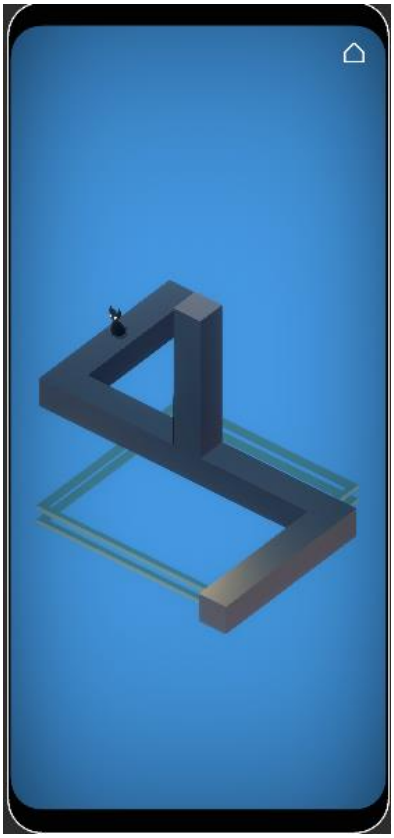
Obr. 30 Level Cube

Obr. 31 Level Podzemi



Obr. 32 Hlavní menu

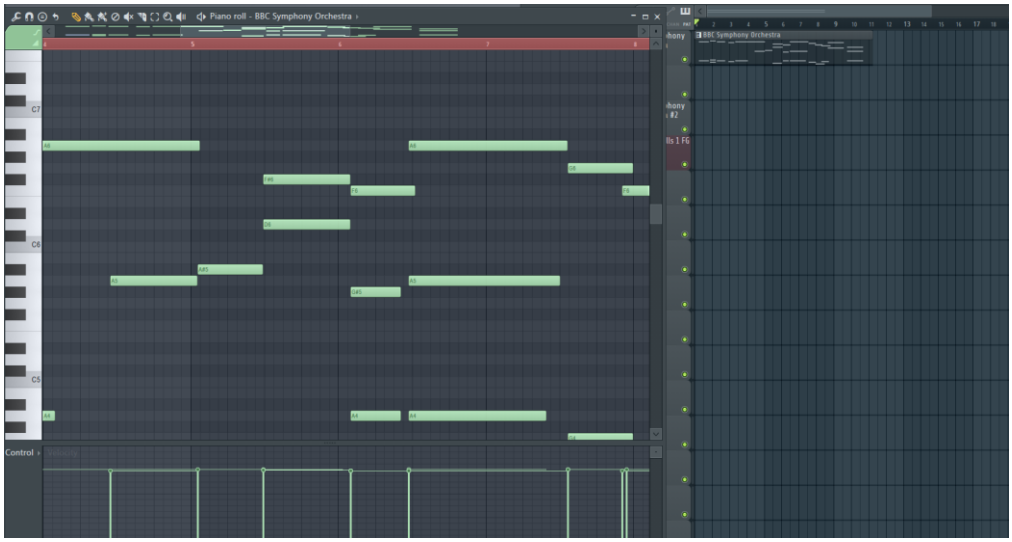
Obr. 33 Rozhovor s vránou



Obr. 34 Level Spirála

13. Sound design

K vytváření doprovodné hudby jsme postupně použily dvě velmi odlišné metody a ve hře jsou implementovány výsledky obou těchto přístupů. Ze začátku jsme použili programu FL studio, ale potom jsem přešel na nahrávání hudby naživo a jejího následného upravování. FL studio umožňuje uživateli jednoduše skládat hudbu, v našem případě pomocí funkce piano roll (Obr. 22). V FL studiu jsme navíc použili dvě rozšíření. BBC orchestra pro zvuky nástrojů a LABS pro organické zvuky, jako například zpěv ptáků.

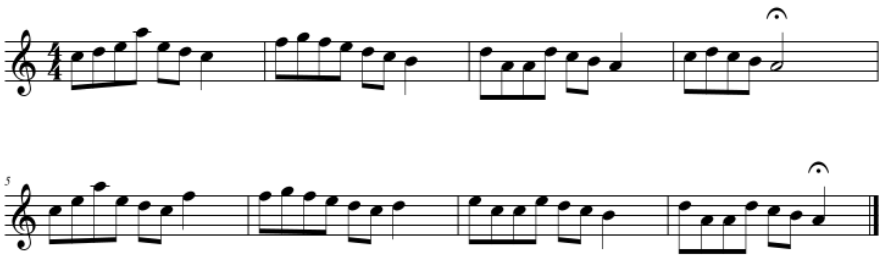


Obr. 35 Fl studio, piano roll skladby nazvané Ominous forest

Druhá zmíněná metoda spočívala v tom, že jsem nejdříve hudbu nahrál a následně upravil. Melodii jsem buďto zapsal do not a poté zahrál na housle, nebo jsem rovnou hrál a tedy žádný zápis ani neexistuje. Pro nedostatek lepší techniky jsem k nahrávání použil mobilní telefon, což byl také hlavní důvod, proč nahrávka potřebovala úpravy. Úpravy jsem prováděl v programu PowerDirector365 a byly následující

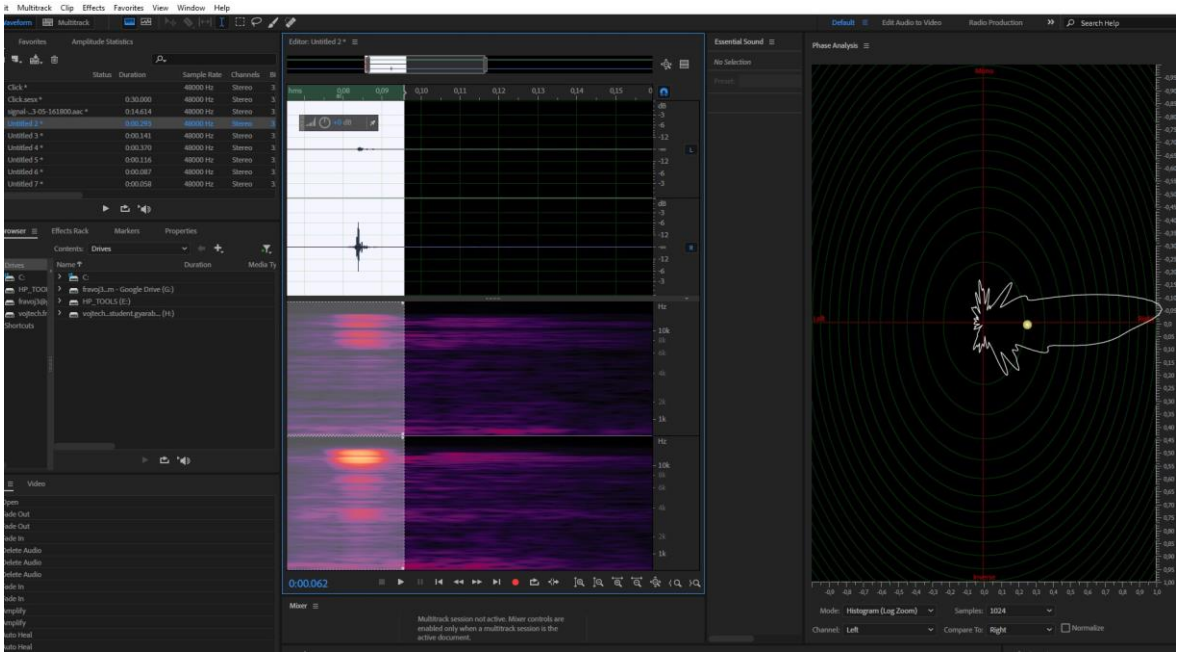
- Odstranění hluku
- Úprava výraznosti určitých frekvencí funkcí *Equilizer*
- Přidání ozvěny

Všechny upravené nahrávky jsem pak spojil do jednoho souboru *Finalviolin.wav.*

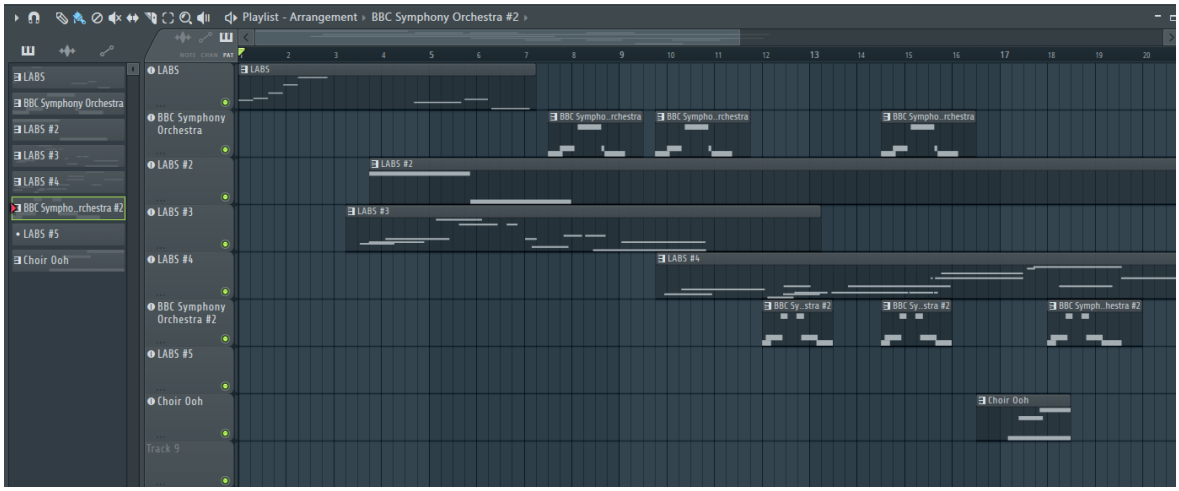


Obr. 36 notace části soundtracku

Mimo tvorby hlavní podkladové hudby je do projektu možné přidat také zvuky kliknutí na některé objekty umožňující interakci a reakci na posun pohyblivých částí levelu. K tomu jsem se rozhodl použít mechanický zvuk kliknutí, jelikož zvuky z bezplatných zvukových bank na internetu byly nekvalitní a do prostředí hry se nehodily, rozhodl jsem si nahrát si zvuky vlastní. K tomu jsem použil staré natahovací hodinky a mobilní telefon. Nahrál jsem sérii klikání, ze které jsem vybral šest nejvhodnějších zvuků, které jsem následně v Adobe Audition očistil od šumu.



Obr. 37 Adobe Audition – editace zvuku kliknutí

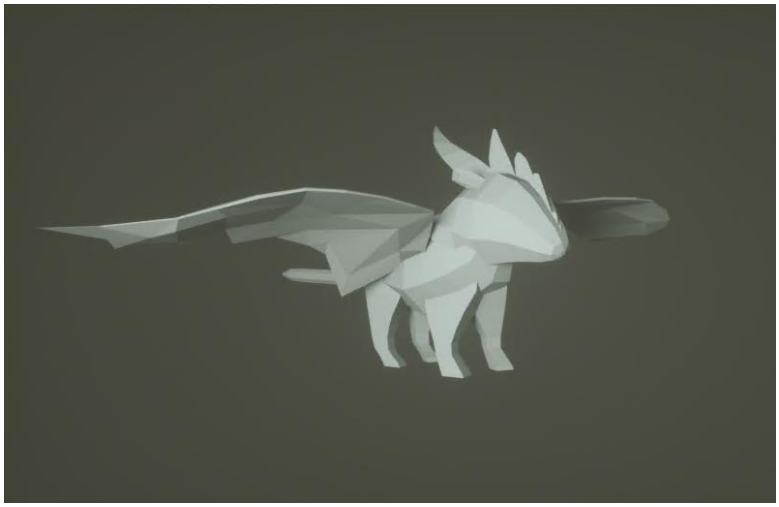


Obr. 38 PLaylist arrangement v FL studiu

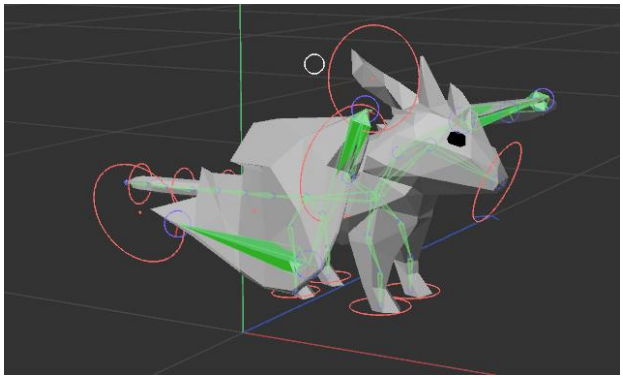
14. Nepoužité návrhy

V průběhu tvorby práce došlo k zavržení spousty návrhů z různých důvodů. Některé kvůli obtížné proveditelnosti (animace hroutícího se mostu), jiné z časových důvodů, jako bylo rozhodnutí nepoužít již předpřipravenou postavu draka pro animaci, která by byla velmi časově náročná, jelikož bylo výhodnější čas investovat do věcí majících větší vliv na celkový dojem ze hry. Jedním dalších podobných příkladů může být plánované animované intro do hry, ze kterého byl vytvořen jen storyboard. Další návrhy nebyly použity, jelikož pro ně nebylo nalezeno v projektu využití, příkladem může být simulace letu hejna vran.

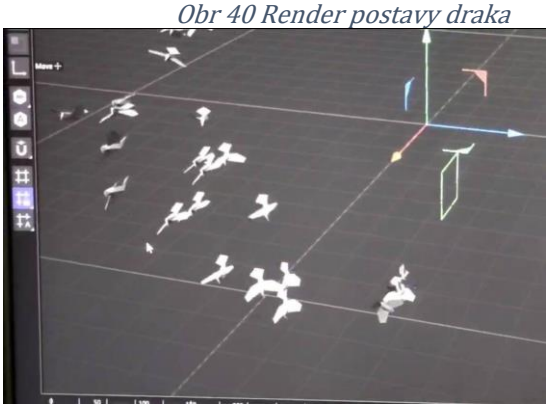
Obr. 39 Prvotní návrh podoby postavy



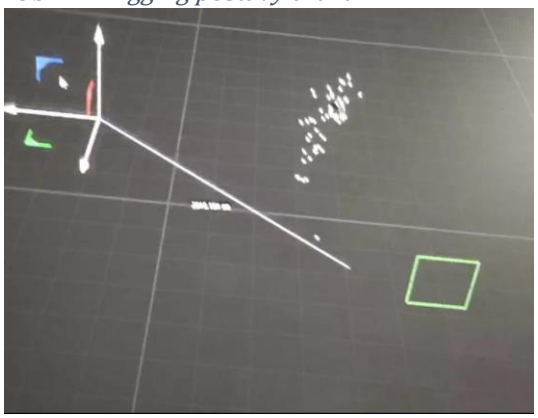
Obr 40 Render postavy draka



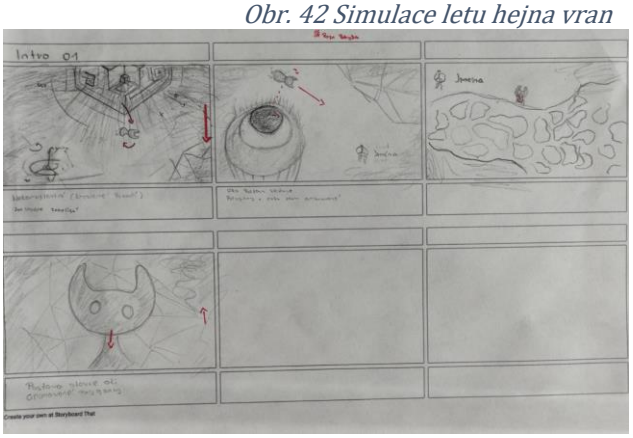
Obr. 41 Rigging postavy draka



Obr. 42 Simulace letu hejna vran



Obr. 43 Simulace letu vran reprezentovaných body



Obr. 44 Storyboard intra hry

15. Závěr

V průběhu práce na tomto projektu se brzy ukázalo, že prvotní podoba zadání byla velmi ambiciózní, výsledkem byla nutnost původní představy značně omezit. Projekt tak nebyl ochuzen z hlediska vytvořených funkcí pro tvorbu, ale utrpěl omezení na straně samotného obsahu, který bylo obtížné vytvářet. Existoval obecný rámec, jakou podobu a roli by měl daný obsah splňovat, ale do podrobnějších forem se se dostával mnohem obtížněji. Jednou z hlavních potíží, kterými jsme během tvorby čelili, byla nedokonalá komunikace týmu, špatné rozdělení práce a rozdílný přístup členů týmu k plnění svých úkolů na přidělené části. To mělo za následek omezení množství vyprodukovaného obsahu, což je nejvíce patrné na příběhové části projektu.

Přesto se nám podařilo vytvořit fungující aplikaci obsahující více jak osm úrovní s vlastními 3D modely a vlastní hudbou. V porovnání s hrou Monument valley, která obsahuje v základu deset úrovní, které jsou výrazně obsáhlejší, než jsme měli my, jsme si nevedli špatně, vezmeme-li v potaz velikost týmu vyvíjející srovnávanou hru a fakt, že šlo o náš první velký projekt v dané oblasti.

16. Bibliografie

[1] Unity technologies, „Secondary Maps (Detail Maps) & Detail Mask," Unity Technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterDetail.html. [Přístup získán 9 2 2023].

[2] Unity technologies, „Supported Model file formats," Unity technologies, 24 2 2021. [Online]. Available: https://docs.unity3d.com/2020.1/Documentation/Manual/3D-formats.html#Exported3DFiles. [Přístup získán 9 2 2023].

[3] J. O'Connell, „FBX File Format (CAD): Simply Explained," All3DP, 22 12 2020. [Online]. Available: https://all3dp.com/2/fbx-file-format-simply-explained/. [Přístup získán 9 2 2023].

[4] Snap Inc., „Vertex Animation," Snap Inc., [Online]. Available: https://docs.snap.com/lens-studio/references/guides/adding-content/3d/animation/vertex-animation#3ds-export. [Přístup získán 9 2 2023].

[5] I. Faber, „BLENDSHAPES - FROM MAYA TO UE4 EASY!," 26 9 2021. [Online]. Available: https://www.youtube.com/watch?v=Tww_LSQEVd8. [Přístup získán 9 2 2023].

[6] RichardKain, „Import blendshapes keyframes from maya into unity," 29 7 2014. [Online]. Available: https://forum.unity.com/threads/import-blendshapes-keyframes-from-maya-into-unity.259488/. [Přístup získán 9 2 2023].

[7] Unity technologies, „Rendering Mode," Unity technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterRenderingMode.html. [Přístup získán 9 2 2023].

[8] Unty technologies, „Metallic mode: Metallic Parameter," Unty technologies, 2023 2 3. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterMetallic.html. [Přístup získán 9 2 2023].

[9] Unity technologies, „Normal map (Bump mapping)," Unity technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html. [Přístup získán 9 2 2023].

[10] Unity technologies, "Heightmap," Unity technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterHeightMap.html. [Accessed 9 2 2023].

[11] Unity technologies, „Occlusion Map," Unity technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterOcclusionMap.html. [Přístup získán 9 2 2023].

[12] Unity technologies, „Emission map examples," Unity technologies, 3 2 2023. [Online]. Available:

https://docs.unity3d.com/Manual/StandardShaderMaterialParameterEmission.html. [Přístup získán 9 2 2023].

[13] Unity technologies, „Secondary Maps (Detail Maps) & Detail Mask," Unity technologies, 3 2 2023. [Online]. Available: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterDetail.html. [Přístup získán 9 2 2023].

[14] kennypu, „way to get all object with a certain component/script attached," 4 2 2011. [Online]. Available: https://answers.unity.com/questions/46283/way-to-get-all-object-with-a-certain-componentscri.html. [Přístup získán 17 2 2023].

16.1. Zdroje využité pro tvorbu tohoto projektu

Fade of UI element script
forum.unity.com/threads/real-fade-of-text-mesh-pro.620833/

Editing Text Area object
forum.unity.com/threads/how-to-make-a-multi-line-textfield-for-editor.153995/

App saves path in Android
answers.unity.com/questions/539339/saving-data-in-to-files-android.html

Text mesh pro script for changing text
forum.unity.com/threads/access-textmeshpro-text-through-script.527992/

instancing prefab assets
answers.unity.com/questions/389702/how-to-instantiate-prefab-with-resourcesload-.html

Screen coordinates to 3D space script
www.youtube.com/watch?v=7XVSLpo97k0

Audio
gamedevbeginner.com/how-to-play-audio-in-unity-with-examples/

Acess saved files
stackoverflow.com/questions/44419658/location-of-application-persistentdatapath-in-a-build

Playing recorded animations
https://docs.unity3d.com/ScriptReference/Animator.Play.html

Saving and loading game
www.red-gate.com/simple-talk/development/dotnet-development/saving-game-data-with-unity/

Swiping gestures
pressstart.vip/tutorials/2019/05/15/95/swiping-pages-in-unity.html

Změna průhlednosti shaderu v závislosti na vzdálenosti od hráče
Chat GPT se zadáním „Unity create shader that changes opacity based on distance from GameObject.".

17. Seznam obrázků

17.1. Zdroje obrázků

Obr. 7 - docs.unity3d.com/Manual/StandardShaderMaterialParameterSmoothness.html
Obr. 8 - docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html
Obr. 14 - cz.pinterest.com/pin/60418610627792262/