Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj 3D hry v enginu Unity

Ivan Usachev

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ivan Usachev

Informatika

Název práce

Vývoj 3D hry v enginu Unity

Název anglicky

Development of a 3D game in the Unity engine

Cíle práce

Hlavním cílem práce je navrhnout a vytvořit herní aplikaci 3D akční adventury s plošinovými prvky v prostředí Unity. Dílčím cílem je seznámení čtenáře s celým procesem vývoje a představení nejdůležitější součásti vývojového prostředí Unity včetně postupů pro vytvoření originálních modelů a lokací pro hru.

Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných informací budou v teoretické části popsány základní prvky a principy Unity, které jsou nezbytné v rámci vývoje hry. V praktické části bude popsána analýza, návrh a implementace herních komponent. Implementace bude prováděna v prostředí Unity a programovacím jazyce C#.

Doporučený rozsah práce

35-40 stran

Klíčová slova

Unity, C#, 3D, Herní aplikace, Blender

Doporučené zdroje informací

BLACKMAN, Sue, 2014. Unity for Absolute Beginners. New York: Apress Media. ISBN 987-1-4302-6778-2

Blender Foundation, 2021. Blender – Manual: Blender 2.93 Reference Manual [online]. Dostupné z: https://docs.blender.org/manual

BOND, Jeremy Gibson, 2018. Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#. Second edition. Upper Saddle River, NJ: Addison-Wesley. ISBN 978-0-13-465986-2

HOCKING, Joseph, 2018. Unity in Action: Multiplatform game development in C#. Second edition. Shelter Island: Manning. ISBN 978-1617294969

Unity Technologies, 2020. Unity – Manual: Unity User Manual (2020.3) [online]. Dostupné z: https://docs.unity3d.com/Manual/index.html

Unity Technologies, 2020. Unity – Scripting API [online]. Dostupné z: https://docs.unity3d.com/ScriptReference/index.html

---

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Tomáš Benda

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 02. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj 3D hry v enginu Unity" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2022 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Tomáši Bendovi za odborné vedení této bakalářské práce, věnovaný čas a cenné rady.
Dále bych chtěl poděkovat své rodině za podporu během celého studia.

Vývoj 3D hry v enginu Unity

Abstrakt

Tato bakalářská práce se věnuje návrhu a implementaci herní aplikace 3D akční adventury s plošinovými prvky v prostředí Unity za použití programovacího jazyka C# a programu 3D modelování Blender.

V teoretické části se práce zaměřuje na popis základních prvků a principů enginu Unity, které jsou nezbytné v rámci vývoje hry. Nejprve je čtenář seznámen s vývojovým prostředím Unity, včetně rozhraní a základních komponent enginu. Dále jsou popsány základní nástroje a modifikátory programu Blender. V praktické části práce je představen proces tvorby originálních herních modelů, animací, lokací a také postupy implementace herních mechanik.

Celkovým výsledkem praktické části je funkční prototyp hry. Prototyp se skládá ze dvou herních lokací, kde se hráč setkává s nepřáteli. Hra může skončit výhrou nebo prohrou. Prototyp hry je otevřený a připravený na další vylepšení a změny.

Klíčová slova: Unity, C#, 3D, Herní aplikace, Blender

Development of a 3D game in the Unity engine

Abstract

This bachelor thesis focuses on the design and implementation of a 3D action-adventure game application with platform elements in Unity Engine using the C# programming language and the 3D modelling program Blender.

In the theoretical part, the thesis focuses on describing the basic elements and principles of the Unity engine that are essential in the development of the game. First, the reader is introduced to the Unity development environment, including the interface and the basic components of the engine. Next, the basic Blender tools and modifiers are described. In the practical part of the thesis, the process of creating original game models, animations, and locations is presented, as well as the procedures for implementing game mechanics.

The overall result of the practical part is a working prototype of the game. The prototype consists of two game locations where the player encounters enemies. The game can end with a win or a loss. The game prototype is open and ready for further improvements and changes.

Keywords: Unity, C#, 3D, Game application, Blender

Obsah

1 Úvod

Videohry jsou rok od roku populárnější. O počítačové hry se zajímá stále více lidí. A to jak běžní hráči, tak i vývojáři. Ale nebylo to tak vždy.

Na konci sedmdesátých let 20. století, kdy videoherní průmysl teprve vznikal, byl proces vytváření videoher zcela specifický a vyžadoval speciální znalosti a dovednosti. Ne každý si mohl dovolit přispívat do herního průmyslu. Naučit se to mohlo trvat velmi dlouho a vyžadovalo to nákladné vybavení. Koncept herního enginu začal získávat širokou popularitu v osmdesátých letech 20. století (Buttle, 2020). Standardy kvality a požadavky na nové projekty každým rokem rostou, ale zároveň s tím i počet technologií a nástrojů, které usnadňují přímý proces tvorby videoher.

V současné době se začínající vývojář může seznamovat s procesem vývoje her pomocí herních enginů. Herní engine poskytuje vývojářům nástroje k vytvoření většiny součástí hry a umožňuje je vzájemně propojit. Herní engine umožňuje implementaci mnoha herních komponent, od vykreslování, fyziky, zvukového designu, skriptování, tvorby umělé inteligence až po síťové funkce. Herní prvky, které nelze vytvořit přímo v enginu, mohou být vytvořeny ve specializovaném programu a poté importovány do vývojového prostředí. Unity Engine je jednou z mnoha možností, jak začít s procesem vývoje herních aplikací. Blender zase umožňuje tvořit nejen primitivní, ale i profesionální modely a animace, které dokážou naplnit herní svět vytvořený v Unity.

Tato bakalářská práce se věnuje návrhu a vytvoření herní aplikace 3D akční adventury s plošinovými prvky v prostředí Unity. A také slouží k seznámení čtenáře s celým procesem vývoje a představení nejdůležitější součásti vývojového prostředí Unity včetně postupů pro vytvoření originálních modelů a lokací pro hru. Práce může sloužit jako pomocná příručka pro začínající vývojáře.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této práce je navrhnout a vytvořit herní aplikaci 3D akční adventury s plošinovými prvky v prostředí Unity za použití programovacího jazyka C# a programu 3D modelování Blender. Dílčím cílem je seznámení čtenáře s celým procesem vývoje a představení nejdůležitější součásti vývojového prostředí Unity včetně postupů pro vytvoření originálních modelů a lokací pro hru.

2.2 Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných informací budou v teoretické části popsány základní prvky a principy Unity, které jsou nezbytné v rámci vývoje hry. V praktické části bude popsána analýza, návrh a implementace herních komponent. Implementace bude prováděna v prostředí Unity a programovacím jazyce C#.

3 Teoretická východiska

3.1 Unity

Unity je herní engine profesionální kvality používaný k vytváření videoher zaměřených na různé platformy. Nejen, že se jedná o profesionální vývojový nástroj, který denně používají tisíce zkušených herních vývojářů, je to také jeden z nejdostupnějších moderních nástrojů pro začínající herní vývojáře. Až donedávna čelil nováček ve vývoji her spoustě nemalých překážek, ale Unity usnadňuje začít se učit tyto dovednosti (Hocking, 2018). Unity engine byl vyvinut společností Unity Technologies v roce 2005. Unity se stal jedním z nejznámějších enginů používaných giganty herního průmyslu a nezávislými vývojáři.

Jedním ze zajímavých projektů vytvořených v enginu Unity nezávislým týmem vývojářů je hra „SUPERHOT“. „SUPERHOT“ je neobvyklý First-person shooter, ve kterém se čas pohybuje pouze spolu s pohybem hlavního hrdiny. Hru vytvořil malý tým z Polska. Nejprve to byl prototyp vytvořený na „7 Day FPS Challenge“. Projekt se poté přesunul na Kickstarter, kde získal potřebné finance a následně se stal plnohodnotnou hrou (Superhot, 2022).

Engine Unity však nevyužívají jen malé herní společnosti, ale také velké společnosti s dlouhou historií a vlastními technologiemi. Například společnost Blizzard, která vyvinula karetní online hru na motivy Warcraftu. *„Hearthstone je rychlá strategická karetní hra od Blizzard Entertainment.“* (Blizzard, 2022). Podstata hry spočívá ve virtuálních soubojích mezi sebou pomocí balíčků karet s tahovým systémem přenášení tahů mezi soupeři během zápasu.

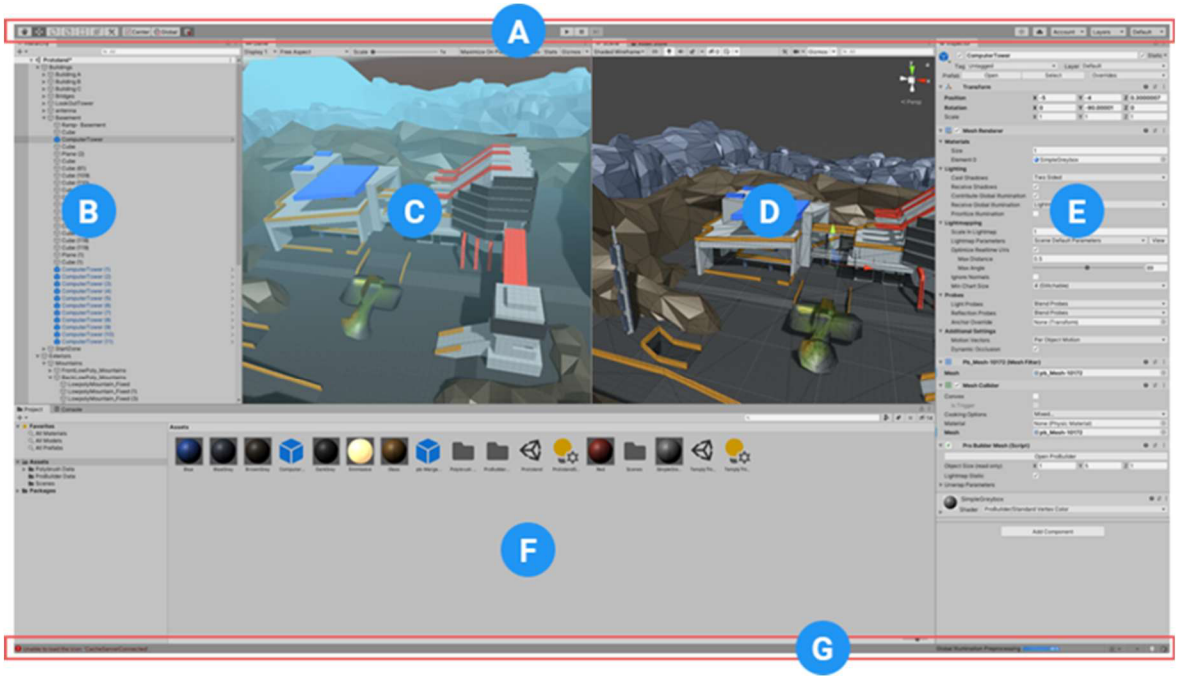
Herní engine Unity umožňuje vytvářet nejen aplikace pro zábavu, ale také specializované vzdělávací aplikace. Například projekt „SARGE“. Cílem projektu „SARGE“ je vytvořit videohru, ve které si hráči osvojí dovednosti potřebné k ovládání několika typů robotů, které se obvykle používají při pátrání a záchraně a při vymáhání práva. Síťová součást hry navíc vyžaduje, aby hráči pracovali v týmech při prohledávání oblasti katastrofy. Hra bude sledovat výkon hráče během několika herních relací a poskytne mu zpětnou vazbu ohledně zlepšení jeho dovedností. Unity engine zjednodušil vývoj „SARGE“ a umožnil vývojářům soustředit se na vývoj obsahu místo starostí s integrací různých open source fyzikálních a renderovacích komponent nebo bojů s chybnými editory prostředí (Craighead et al., 2008).

Pro pochopení základů práce ve vývojovém prostředí Unity je nutné se seznámit se strukturou editoru Unity a základními komponentami, se kterými je potřeba pracovat při tvorbě jednoduché hry.

3.1.1 Uživatelské rozhraní

Rozhraní Unity má několik sekcí, které jsou zodpovědné za různé vývojové prvky - assety, herní objekty, nastavení jejich vlastností a parametrů. Základními sekcemi prostředí Unity jsou:

- (A) Toolbar
- (B) Hierarchy Window
- (C) Game View
- (D) Scene View
- (E) Inspector Window
- (F) Project Window
- (G) Status Bar



Obrázek 1: Rozhraní Unity (Unity, 2020).

Scene View

„Scene View“ umožňuje prohlížet trojrozměrné scény a vybírat, přesouvat, otáčet a měnit velikost objektů. Scénou může být například úroveň hry, hlavní menu, úvodní obrazovka (Bond, 2018).

Panel v horní části okna scény umožňuje vybrat režim zobrazení a také ovládat osvětlení a zvuk. Je také možné přizpůsobit zobrazení „Gizmo“. „Gizmos“ se používají pro kreslení tvarů v pohledu scény. Tyto tvary lze použít k nakreslení dalších informací o herních objektech.

Game View

„Game View“ odpovídá za výsledný obrázek, který uživatel uvidí během hry. Tři tlačítka v horní části okna Unity jsou zodpovědná za ovládání náhledu hry. Tlačítko „Play“ umožní přehrát aktuální scénu. Tlačítko „Pause“ pozastaví provádění aktuálně spuštěného zobrazení hry. Tlačítko „Step“ funguje, když je zobrazení hry pozastaveno a způsobí, že hra spustí jeden snímek hry. Umožňuje to efektivně „procházet“ hrou pomalu a ladit jakékoliv problémy.

V okně „Game“ je možné vybrat požadovaná nastavení pro zobrazení hry. Lze například vybrat poměr stran obrazu, aby bylo jasné, jak bude hra vypadat na různých monitorech, vypnout zvuk ve hře nebo zobrazit Gizmo.

Hierarchy Window

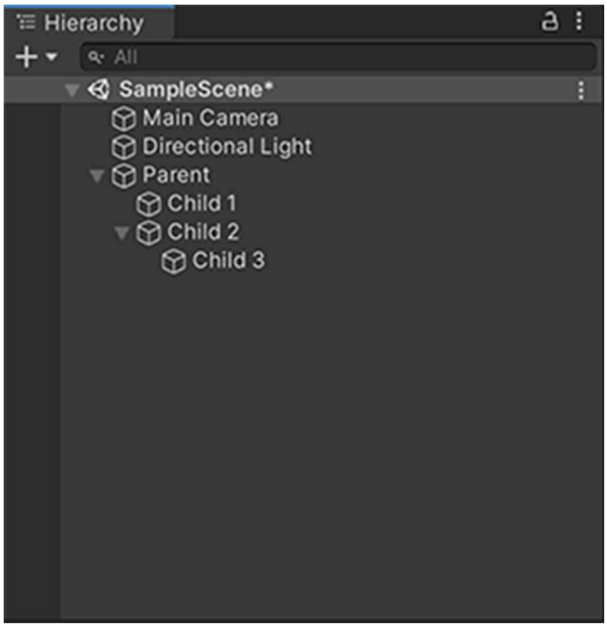
Hierarchie obsahuje všechny objekty (GameObject) otevřené scény. Objekty, které jsou přidány do scény nebo z ní odstraněny, se zobrazují nebo naopak přestanou zobrazovat v „Hierarchy“.

Dědičnost (Parenting)

V hierarchii Unity lze objekty dědit. Jakýkoli objekt může být potomkem jiného. *„Objekty mohou dědit vlastnosti a činnosti od jiných objektů.“* (Hanák, 2008). Pro vytvoření podřízeného odkazu je nutné přetáhnout objekt na „rodiče“ v okně „Hierarchy“.

Objekty jsou organizovány stromovým způsobem, kdy objekty jednoho typu mohou dědit od druhého, přebírat jejich schopnosti, ke kterým pouze přidávají svá vlastní rozšíření (Žoltá, 2022).

Dědičnost umožňuje implementovat složitá schémata s jasnou hierarchií. To usnadňuje pochopení a škálování kódu. Není nutné přepisovat stejné vlastnosti mnohokrát v různých objektech. Tyto objekty stačí zdědit od jednoho „rodiče“ a vlastnosti „rodiče“ se použijí automaticky.



Obrázek 2: Hierarchy Window (Unity, 2020).

Project Window

„Project“ je složka, která obsahuje všechny materiály projektu. Každý projekt obsahuje složku „Assets“. Obsah této složky je zobrazen v oblasti „Project View“. Jedná se o zdroje Hry: soubory-skripty, 3D modely, textury, zvukové soubory, prefaby (objekty, které lze klonovat).

Herní projekty se skládají z jednoho nebo více souborů scén. Každá samostatná scéna je samostatnou úrovní hry. Scény jsou také uloženy ve složce „Assets“ a zobrazeny v „Project View“.

Obrázek 3: Project Window (Unity, 2020).

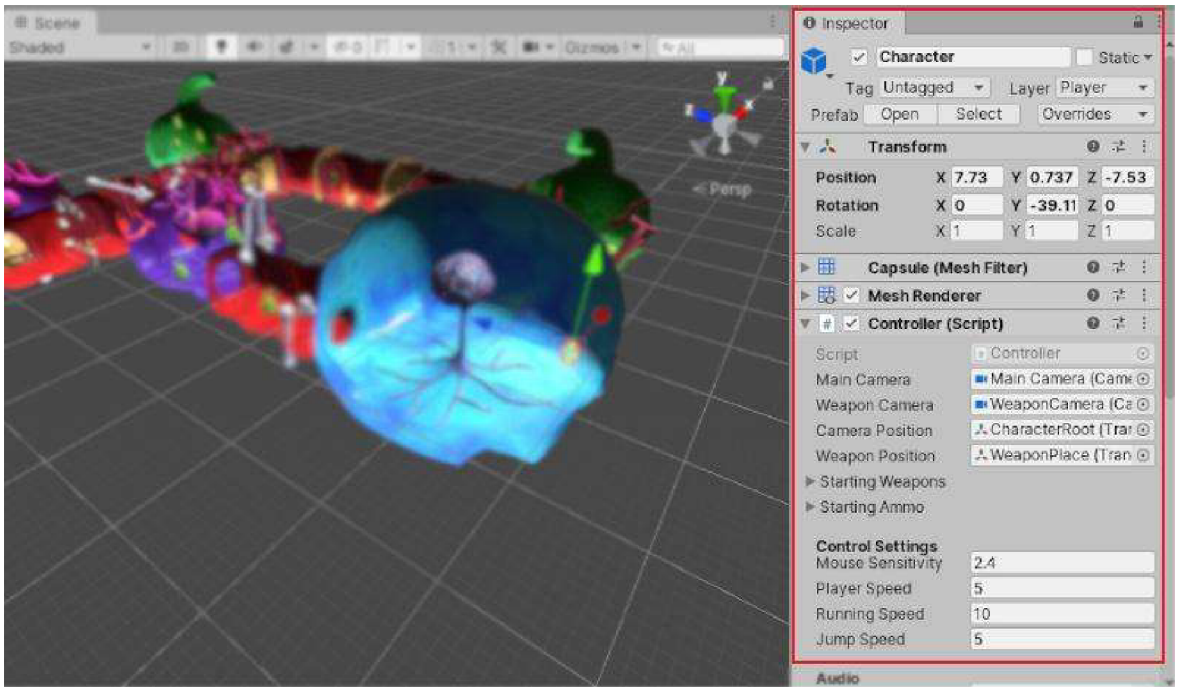
Console Window

„Console“ je konzole, kde se budou během hry zobrazovat zprávy, zejména zprávy popisující chyby v kódu. Konzole může také zobrazovat uživatelské zprávy, které uživatel použil ve skriptu k identifikaci a odstranění určitých chyb. Například pro kontrolu správného fungování komponenty „Nav Mesh Agent“ je možné do skriptu přidat zobrazení zprávy v případě jakéhokoliv konfliktu.

```
if (agent == null)
{
    Debug.LogError("The NMA Component is not attached to " +
    gameObject.name);
}
```

Inspector Window

Okno „Inspector“ zobrazuje podrobnější informace o vybraném herním objektu, včetně všech připojených komponent a jejich vlastností, a umožňuje měnit funkčnost herních objektů ve scéně. Inspektor lze použít k zobrazení a úpravě vlastností a nastavení v editoru, včetně herních objektů, assetů, materiálů a nastavení samotného editoru.



Obrázek 4: Inspector Window (Unity, 2020).

3.1.2 Game Object

„Game Objects“, neboli herní objekty, jsou základní objekty v Unity, které představují například postavy, rekvizity a scenérie (Unity, 2020). Samy o sobě jsou tyto objekty neviditelné a nemají nic jiného než jméno. Jejich chování je určeno komponentami. Každý herní objekt obsahuje alespoň jednu komponentu, kterou nelze odstranit. Toto je komponenta „Transform“. Tato komponenta je zodpovědná za nastavení umístění, rotace a měřítka objektu.

Komponenty umožňují herním objektům promítat geometrii, od jednoduché krychle po složitější 3D modely, vyzařovat světlo, fungovat jako kamera nebo vytvářet složité chování prostřednictvím skriptů. *„Objekty si pamatují svůj stav a navenek poskytují operace přístupné jako metody pro volání.“* (Žoltá, 2022).

Herní objekty mohou být potomky jiných objektů. V tomto případě jsou charakteristiky komponenty „Transform“ podřízených objektů kombinovány s odpovídajícími charakteristikami jejich nadřazených objektů. Pokud objekt nemá rodiče, pak nulové charakteristiky odpovídají nulovým charakteristikám souřadnic scény.

Tagy

Tagy jsou pojmenované značky, které lze použít k identifikaci herních objektů v projektu. Tag je možné použít na jeden nebo více herních objektů, ale herní objekt bude mít vždy pouze jeden tag. Ve výchozím nastavení se tag „Untagged“ používá pro označení herního objektu, který nebyl specificky označen.

Prefaby

Prefab v Unity je běžný herní objekt se všemi jeho komponentami a vlastnostmi v nich nakonfigurovanými a je uložený jako asset, který lze kopírovat. V podstatě je to šablona, kterou je možné znovu použít ve scénách. Lze například vytvořit prefab nepřítele a později jej použít podle potřeby.

Jedním z možných způsobů použití prefabů je „Ragdoll“. To je metoda počítačové animace, ve které pohyb postavy (hlavně při letu a volném pádu) není sadou předrenderovaných animací, ale snahou o reprodukci fyziky v reálném čase.

Implementace ragdoll fyziky umožnila u nepřátel dynamické a fyzicky realistické chování ve chvíli smrti (Zendle et al., 2018).

3.1.3 Komponenty

Každý herní objekt v Unity se skládá z komponent. Jakákoli komponenta jasně implementuje specifickou sadu chování vyžadovanou pro provedení „GameObject“. Proč objekt má určité vlastnosti? Je to díky přispění komponent, které tvoří tento objekt. Komponenty v Unity definují chování objektů ve scéně. Komponenty jsou cokoli připojené k hernímu objektu, ať už je to „Collider“, „Mesh Renderer“ nebo „Rigidbody“.

Komponenty dědí ze základní třídy „MonoBehavior“. K propojení komponent se často používá přímý odkaz. To znamená, že v jedné komponentě se k získání dat z jiné komponenty používá metoda „GetComponent<…>()“. Například:

```
void Start()
{
    animator = GetComponent<Animator>();
}
```

Komponenty mohou být vytvořeny vývojáři pomocí skriptování, ale Unity již poskytuje mnoho komponent, zejména těch nejzákladnějších a nejdůležitějších, které poskytují určitou funkčnost.

Transform

„Transform“ se používá k uložení pozice, rotace, měřítka a rodičovského stavu objektu „GameObject“, a je proto velmi důležitá. Herní objekt bude mít vždy připojenou komponentu „Transform“, není možné odstranit Transform nebo vytvořit GameObject bez ní (Unity, 2020).

Transformace lze upravovat jak v okně scény, tak v okně inspektoru úpravou jeho vlastností. Transformace v okně scény lze upravit pomocí nástrojů z panelu nástrojů.

Rigidbody

Komponenta „Rigidbody“ dělá z objektu fyzické tělo, na které budou působit různé síly, včetně gravitace. Ale sám o sobě „Rigidbody“ nedává představu o fyzických rozměrech objektu. K tomu slouží různé Collidery.

Collider

Komponenty „Collider“ mají dva hlavní typy funkčnosti. Výchozím nastavením je blokování průchodu objektů založených na fyzice objemem definovaným komponentou „Collider“. Druhý, pokud je zapnutý parametr „Is Trigger“ v nastavení Collideru, umožňuje objektům projít jiným objektem a zaregistrovat událost pro další vyhodnocení a možnou akci. Jakýkoli objekt, který má způsobit spuštění události při výběru, kolizi, průniku nebo ray-cast (paprsek vycházející z určitého bodu, v daném směru o určité délce, dotýkající se všech Colliderů v tomto směru), musí mít nějaký typ Collideru. I když je jejich „Mesh Renderer“ vypnutý (ve scéně neviditelný), objekty jsou ve hře stále plně aktivní (Blackman, 2014).

Trigger je neviditelný objekt, který při kontaktu s definovaným objektem spustí určitou událost nebo řetězec událostí. Události tohoto druhu jsou implementovány pomocí skriptů. Například při interakci s triggerem je možné zobrazit libovolné menu.

```
void OnTriggerEnter(Collider other)
{
    WinGameMenuUI.SetActive(true);
 }
```

Character Controller

„Character Controller“ slouží k ovládání postavy v první nebo třetí osobě bez použití fyziky „Rigidbody“. Hlavní koncept Character Controlleru spočívá v tom, že poskytuje základní reakce kolidér bez jakékoli fyziky. V zásadě se postava bude pohybovat, jako by se pohnula pomocí „Transform“, ale zároveň nebude schopna projít kolidéry.

Hlavní výhodou použití této komponenty je míra kontroly nad chováním postavy, nevýhodou však je, že tato komponenta bude vyžadovat více práce při psaní skriptů.

Nav Mesh Agent

Cílem umělé inteligence (AI) je v podstatě to, aby počítače byly schopny myslet a rozhodovat se jako živé organismy pro provádění specifických operací (Aversa et al., 2018).

Někdy je nutné, aby se AI postavy pohybovaly herním světem po náhodné nebo přesně definované cestě. Aby agenti AI vypadali chytře, musí být schopni určit, co dělají, a pokud nedosáhnou požadovaného bodu, musí být schopni vytyčit nejúčinnější trasu a změnit svou cestu, v případě že se na cestě objeví překážky.

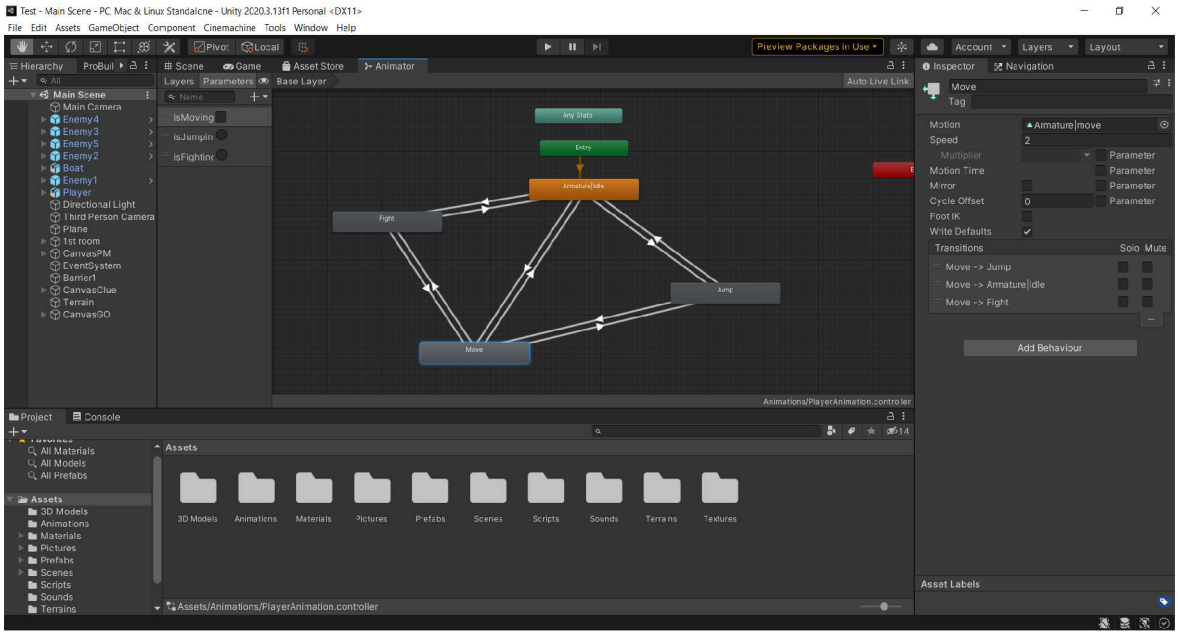
Aby umělá inteligence našla cesty, musí být scéna prezentována v určitém formátu. Agenti umělé inteligence potřebují vědět, kde jsou překážky, zejména ty statické. Unity má vestavěný nástroj pro generování „NavMesh“, který vykresluje scénu v kontextu, jenž je pro agenty umělé inteligence vhodný k dosažení cíle.

Komponenta „Nav Mesh Agent“ zase umožňuje agentovi používat informace, které byly nastaveny ve scéně pomocí „NavMesh“.

Animator

Komponenta „Animator“ se používá k přiřazení animace k hernímu objektu ve scéně. Potřebuje odkaz na „Animator Controller“, který určuje, jaké klipy animace se mají použít, a řídí, kdy a jak se protínají a přepínají mezi sebou (Unity, 2020).

„Animátor Controller“ umožňuje uspořádat a udržovat sadu animačních klipů a souvisejících přechodů animace pro postavu nebo objekt. Ve většině případů je v pořádku mít více animací a přepínat mezi nimi, když nastanou určité herní podmínky. Lze například přepnout z klipu animace pohybu na klip animace skoku pokaždé, když je stisknut mezerník (Unity, 2020).



Obrázek 5: Animator Controller (vlastní zpracování).

3.1.4 Osvětlení scény

Osvětlení lze do hry implementovat několika způsoby. Existují zejména metody „Realtime“ a „Baked“. Výpočty v reálném čase probíhají za běhu aplikace, ale zatěžují zařízení. Naproti tomu zapečené osvětlení je vytvořeno před spuštěním aplikace a nemá vliv na výkon za běhu aplikace, ale v tomto případě nebude osvětlení tak realistické. V případě vývoje jednoduché hry má smysl využít zapečené osvětlení.

„Lightmapping“ je technologie, která ukládá informace o osvětlení do textury, což umožňuje uvolnit výpočetní zdroje pro jiné potřeby. Unity má vestavěný nástroj pro vytváření světelných map, který je plně integrován do Unity. To znamená, že nástroj peče (Bake) světelné mapy pro scénu na základě toho, jak je scéna nastavena v Unity, jaké mesh, materiály, textury a světla obsahuje. Znamená to také, že světelné mapy jsou nyní nedílnou součástí vykreslování – jakmile jsou vytvořeny, není potřeba dělat nic jiného, automaticky se aplikují na objekty.

3.1.5 Asset Store

Assety jsou komponenty, které představují grafiku, zvuk nebo skripty. Připevňují se k objektům a tvoří důležitou součást hry. „Unity Asset Store“ je oficiální obchod, kde lze najít placené i bezplatné komponenty pro hry, jako jsou 3D modely, zvuky / hudba, sady uživatelského rozhraní, shadery / částice, sady sprite a také nástroje. V „Asset Store“ je možné vybrat kategorie assetů a to, co je součástí sady, a také zkontrolovat, zda je podporována v požadované verzi Unity. *“Unity Asset Store je úžasné místo pro výměnu peněz za čas. Internetový obchod Asset Store nabízí tisíce různých zdrojů (a šetří spoustu času), včetně modelů, animací a softwarových knihoven.”* (Bond, 2018).

3.1.6 Skripty

Skripty jsou instrukční soubory v určitém programovacím jazyce, popisují logiku chování objektů, jak si ji vývojář přeje. Unity podporuje několik programovacích jazyků, z nichž nejoblíbenější jsou JavaScript a C#.

Jazyk C# byl vyvinut společností Microsoft na konci devadesátých let. Spojil sílu Java a jednoduchost C++. Tento jazyk se používá k vytváření aplikací na platformě Microsoft.NET Framework. C# byl vyvinut především jako nástroj na aplikační úrovni CLR (Sharp, 2018). Unity používá produktivní programovací jazyk C#. Díky C# a Mono, multiplatformní implementaci .NET, mohou projekty Unity stavět a běžet stejně stabilně na různých operačních systémech.

MonoBehaviour

MonoBehaviour je základní třída pro veškeré skriptování v Unity. MonoBehaviour definuje neviditelný základ pro to, jak se komponenty připojují k herním objektům, a dědění z ní poskytuje několik automaticky spouštěných metod, které lze implementovat. Mezi tyto metody patří „Start()“, volaná jednou, když se objekt stane aktivním (což nastává obecně, jakmile se načte úroveň s daným objektem), a „Update()“, která se nazývá každý snímek (Hocking, 2018).

Metoda „Awake“ je volána ihned po vytvoření instance objektu ve scéně a toto je první příležitost ke spuštění kódu ve skriptu. „Awake“ je volána pouze jednou za dobu existence instance.

Metoda „Start“ je volána jednou těsně před prvním voláním metody „Update“ objektu, za předpokladu, že je objekt dostupný a je aktivní související herní objekt. Metody „Start“ všech objektů jsou volány až po dokončení všech volání metody „Awake“.

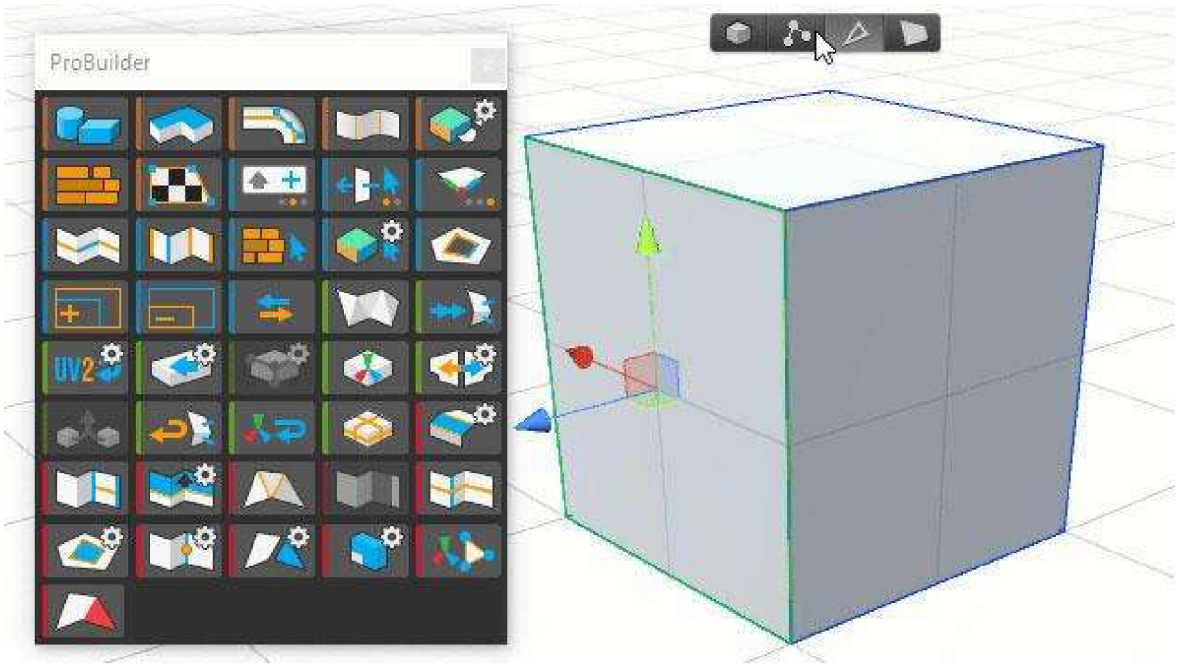
Metoda „Update“ je volána na každém snímku, dokud je komponenta dostupná a je aktivní herní objekt, ke kterému je komponenta připojena.

3.1.7 Rozšiřující balíčky

Balíček obsahuje funkce, které vyhovují různým potřebám projektu. To může zahrnovat jakékoli základní funkce Unity, které se instalují spolu s Editorem, nebo jiné balíčky, které lze nainstalovat podle potřeby (Unity, 2020).

ProBuilder

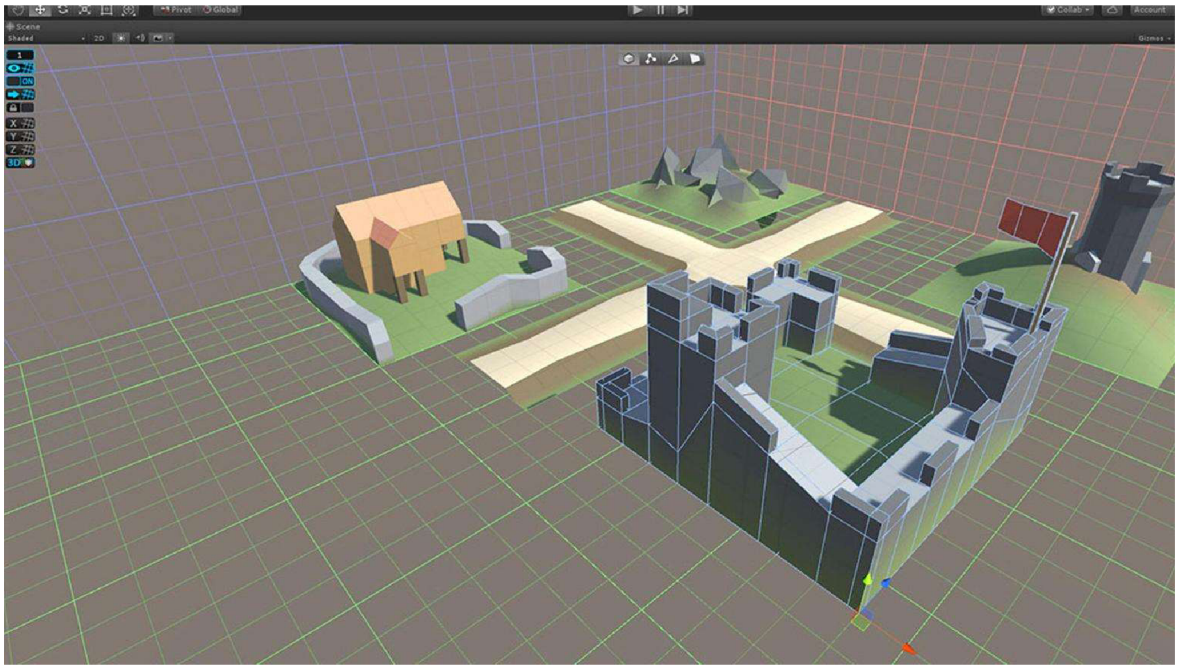
„ProBuilder“ je unikátní hybrid 3D modelování a nástroje pro návrh úrovní, optimalizovaný pro vytváření jednoduchých geometrií, včetně detailních úprav. „ProBuilder“ umožňuje upravovat tvary primitivních objektů, čímž se tvoří složitější herní objekty.



Obrázek 6: ProBuilder (Unity, 2022).

ProGrids

„ProGrids“ je dynamicky umístěná mřížka dostupná na všech třech osách a na libovolné pozici ve scéně. Mřížky „ProGrids“ zachycují objekty ve světovém prostoru a zajišťují konzistentní umístění objektů ve scéně (Unity, 2020).



Obrázek 7: ProGrids (Unity, 2022).

Cinemachine

„Cinemachine“ je sada nástrojů pro dynamické, chytré kamery bez kódu, které umožňují vzniknout nejlepším záběrům na základě kompozice scény a interakce, umožňuje ladit, opakovat, experimentovat a vytvářet chování kamery v reálném čase (Unity, 2022).

Tento jednoduchý a pohodlný nástroj umožňuje velmi flexibilně a přesně upravit polohu a princip interakce kamery jak s postavou samotnou, se kterou kamera interaguje z pohledu třetí osoby, tak s objekty, které tuto postavu obklopují.

3.1.8 Shrnutí

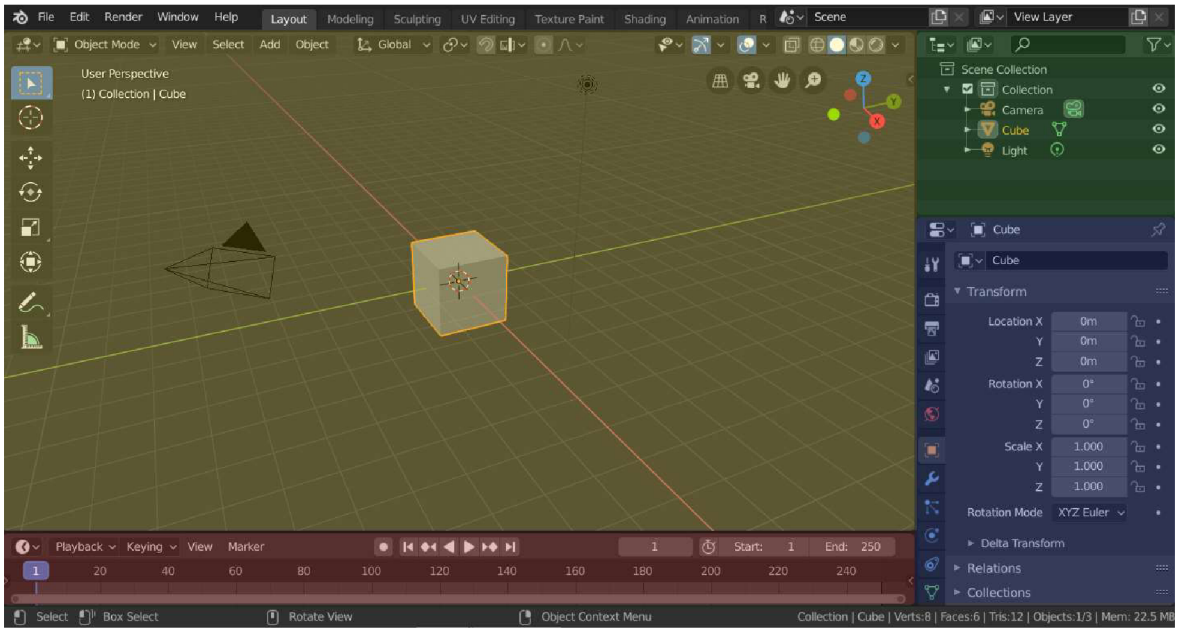
Herní engine Unity má velkou sadu nástrojů a komponent pro vytvoření herní aplikace jakéhokoli druhu a složitosti. Sada nástrojů Unity není omezena, lze ji obohatit o balíčky rozšíření, vytváření vlastních komponent a také přidávání materiálů třetích stran. Flexibilita Unity umožňuje přizpůsobit engine vlastním potřebám.

3.2 Blender

Blender je bezplatný softwarový produkt určený k vytváření a úpravě trojrozměrné grafiky. Program je distribuován na všech populárních platformách a má otevřený zdrojový kód. Blender je vyvíjen neziskovou nadací Blender Foundation. Program umožní se seznámit s hlavními funkcemi 3D modelování a také nabídne použití přehledných nástrojů pro tvorbu nebo úpravu modelů.

3.2.1 Uživatelské rozhraní

Uživatelské rozhraní Blenderu je stejné pro všechny platformy. Rozhraní lze přizpůsobit konkrétním úkolům. Okno aplikace je rozděleno do 4 oblastí. Jejich počet a velikost lze měnit. Každá oblast obsahuje jeden editor. Každý editor je určen k provádění určitého úkolu. Ve 3D zobrazení (žlutá oblast na obrázku níže) se například provádějí základní transformace objektů. Toto je zobrazení aktuální scény. Outliner (zelená oblast na obrázku níže) slouží jako správce objektů, Properties (modrá oblast na obrázku níže) jsou zodpovědné za nastavení objektů a scén. Pomocí časové osy (červená oblast na obrázku níže) lze vytvořit jednoduchou animaci.



Obrázek 8: Rozhraní Blenderu (Blender, 2021).

3.2.2 Mesh Objects

Mesh jsou jedním typem objektu Blender. Jedná se o trojrozměrná geometrická primitiva, jejichž změnou pomocí základních transformací a dalších modifikátorů vznikají další, obvykle složitější tvary.

Složité trojúhelníkové sítě se hojně vyskytují v počítačové grafice jako výsledek operací geometrického modelování, simulací globálního osvětlení a rekonstrukcí z 3D skenů (Hoppe et al., 1999).

3.2.3 Modifikátory

V Blenderu modifikátory používají v sobě vložený algoritmus k úpravě objektu, aniž by ho bylo třeba upravovat v režimu úprav. Fungují tak, že mění způsob zobrazení a vykreslení objektu, nikoli však geometrii, kterou lze upravovat přímo. K jednomu objektu je možné přidat více modifikátorů, čímž bude vytvořen stack modifikátorů, a aplikovat modifikátor, pokud je třeba, aby jeho změny byly trvalé (Blender, 2021).

Subdivision Surface.

Modifikátor „Subdivision Surface“ umožňuje vytvářet složité hladké povrchy bez komplikované geometrie. Není potřeba ukládat a udržovat velké množství dat, a přesto objekty získají hladký vzhled.

„Subdivision Surface“ rozděluje každý polygon na čtyři části, jedná se o první úroveň dělení, která je specifikována v parametru „Levels View“. Pokud v tomto parametru bude nastavená úroveň 2, pak se všechny nové polygony opět rozdělí na čtyři části. Parametr „Levels View“ nastavuje úroveň dělení objektu v okně 3D pohledu a parametr „Render“ nastavuje úroveň dělení objektu při vykreslování. To umožní pracovat s low-poly modelem v okně 3D pohledu a při vykreslování lze vidět stejný model, ale s velkým počtem polygonů.

Skutečnost, že se objekt zaokrouhlí, dojde-li ke zvýšení úrovně dělení, je dílem algoritmu „Catmull-Clark“. Při přepínání na algoritmus „Simple“ se objekt bude také dále dělit, ale již nebude zaoblen.

Mirror

Funkce modifikátoru „Mirror“ je vytvořit zrcadlový obraz objektu kolem nějaké lokální souřadnicové osy, která prochází počátkem objektu. Ve výchozím nastavení má modifikátor již parametr Axis X povolený.

Modifikátor odráží síť Mesh podél jedné (nebo více) lokálních os (X, Y, Z), které procházejí středem objektu. Jako střed lze také použít jakýkoli jiný objekt (Mirror Object) a jeho lokální osy.

3.2.4 Object Modes

Režimy v Blenderu jsou úrovně objektově orientovaných funkcí (nástrojů) a dostupné režimy se liší v závislosti na typu vybraného aktivního objektu – mnoho z nich obsahuje pouze jeden výchozí režim, režim objektu (např. kamery, světla atd.). Každý režim je určen pro úpravu určitých vlastností vybraného objektu (Blender, 2021).

Object Mode

V objektovém režimu nelze vybrat samostatné skupiny prvků: vrcholy (vertex), hrany (edge) a plochy (face). Všechny manipulace se tedy provádějí s celým objektem, když je zapnutý objektový režim. Transformace v objektovém režimu platí pro celý objekt. Například operace přesunutí přesune celý objekt na nové místo.

Edit Mode

V režimu úprav ovlivňují změny jednotlivé prvky. Lze například přesunout jeden vrchol nebo změnit velikost více ploch, čímž se změní tvar objektu. Prvky lze také odstranit.

3.2.5 Animace

Blender umožňuje vytvářet nejen 3D grafiku. Program zahrnuje rozsáhlou sadu nástrojů moderní počítačové animace. V Blenderu lze animovat nejen jednoduchý pohyb objektů v prostoru, ale také měnit jejich tvar, lze využít kostní systém, vytvořit cyklický pohyb, pohybovat se po dráze atd.

Nejdůležitější věcí v animaci je koncept klíčového snímku. Není nutné kreslit všechny snímky animace, ale pouze ty vybrané, tedy klíčové snímky. Vše mezi tím si program vypočítá sám.

4 Vlastní práce

4.1 Návrh hry

Hra „Goldboy's Adventure" je klasická 3D adventura s plošinovými prvky a bojovým systémem. Hráč ovládá hlavního hrdinu z pohledu třetí osoby s možností volného pohybu po lokaci. Na cestě bude hráč narážet na různé překážky, které bude muset překonat, zejména pomocí skoku. Hráč se také může setkat s nepřítelem, který může být poražen. Hráč má ale také omezenou dávku zdraví, a pokud ji vyčerpá hra skončí. Konečným cílem hráče je uniknout z ostrova.

Důležité součásti hry, které je nutné implementovat:

- Hráč
  - Model
  - Animace
  - Herní mechaniky (pohyb, skok, útok, zdraví)
- Nepřítel
  - Model
  - Animace
  - Herní mechaniky (pohyb, útok, pronásledování)
- Rozhraní hry
  - Hlavní menu (pozadí, tlačítka)
  - Menu pauzy (pozadí, tlačítka)
  - Menu výhry / prohry (pozadí, tlačítka)
- Herní prostor
  - Malá uzavřená lokace
  - Velká otevřená lokace (krajina, textury, vegetace)
  - Skybox
  - Osvětlení

4.2 Implementace hry

4.2.1 Hlavní menu

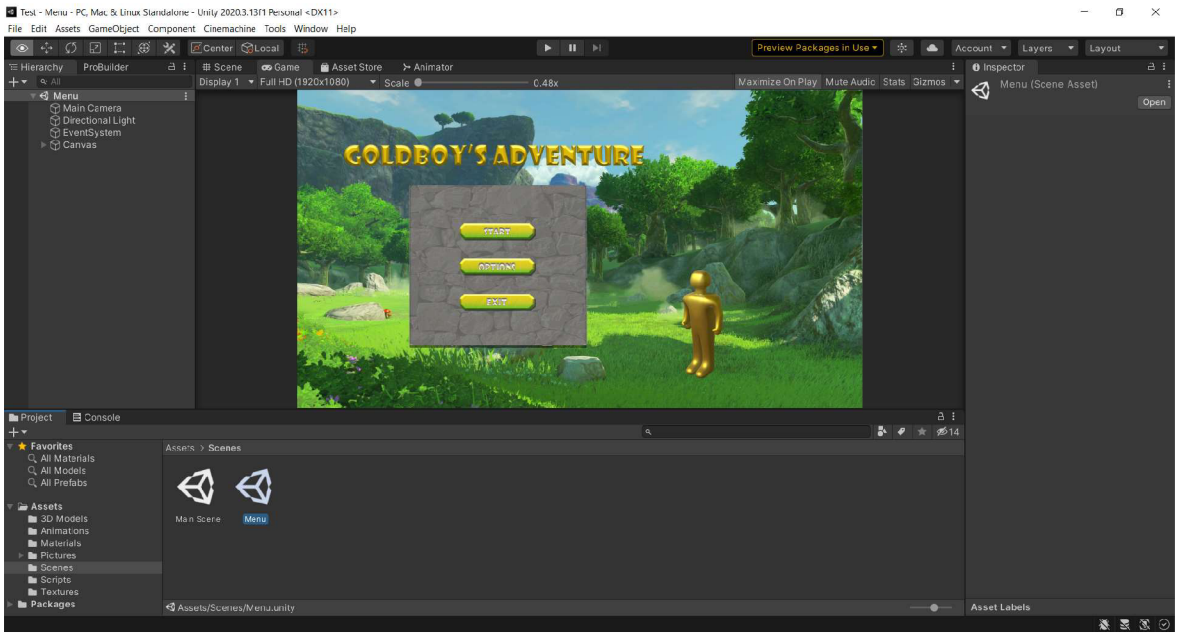
Aby hráč mohl ovládat hru, musí s ní určitým způsobem interagovat. K tomu slouží různá herní menu. Interakce s hrou začíná z hlavního menu.

Pro vytvoření hlavního menu je nutné vytvořit novou scénu. Nejprve se vytvoří objekt „Image“. V nadřazeném objektu „Canvas“ je nutné změnit nastavení komponenty „Canvas Scale“. V nastavení „UI Scale Mode“ je potřeba nastavit „Scale With Screen Size“ a v „Reference Resolution“ vybrat požadované rozlišení obrazovky. Objekt „Image“ bude pozadím (BackGround). Ve vlastnostech objektu „Image“ je potřeba nastavit požadovanou výšku a šířku obrázku. Obrázek, který bude použit jako pozadí, je umístěn v okně „Source Image“. Všechny obrázky používané k vytváření menu musí mít „Sprite (2D and UI)“ ve vlastnosti „Texture Type“. Dále se vytvoří tlačítka pomocí prvku „Button“, který má také požadovanou velikost a připojí požadovaný obrázek.

Po vytvoření vizuální části hlavního menu je přidána funkčnost. Aby tlačítka fungovala, je potřeba vytvořit skript, který bude mít na starosti přepnutí na jinou scénu nebo ukončení aplikace po stisku tlačítka.

```
public void StartGame()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}

public void ExitGame()
{
    Application.Quit();
}
```



Obrázek 9: Hlavní menu (vlastní zpracování).

4.2.2 Menu pauzy

Menu pauzy je vytvořeno podobně jako hlavní menu s několika rozdíly. Hlavním z nich je, že herní svět je pozastaven, zatímco je aktivní menu pauzy, a místo obrázku na pozadí je použit průhledný objekt „Canvas“.

```
if (Input.GetKeyDown(KeyCode.Escape))
{
    if (GameIsPaused)
    {
        PauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }
    else
    {
        PauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
    }
}
```

4.2.3 První lokace

Hra začíná v malé uzavřené lokaci s několika místnostmi, kde se hráč poprvé setká s nepřítelem. Když hráč vyjde z této lokace, vstoupí do velké otevřené lokace.

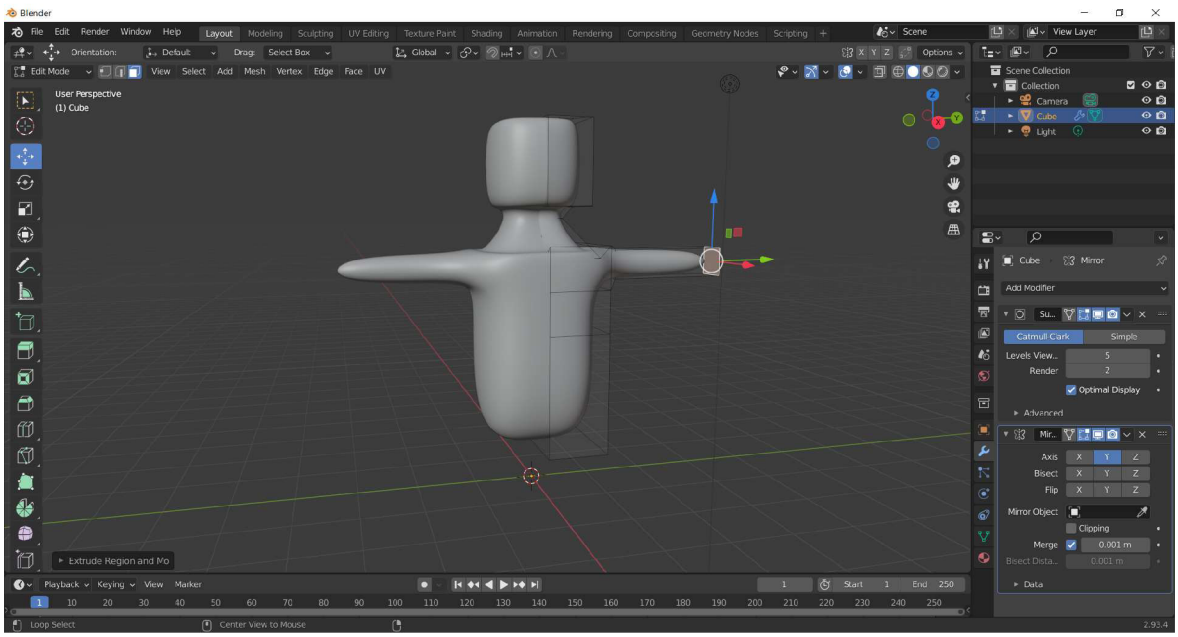
Prototypy takových lokací lze velmi snadno implementovat pomocí balíčků rozšíření „ProBuilder“ a „ProGrids“. „ProGrids“ poskytuje mřížku, která pomáhá přizpůsobit a

zarovnat objekty vůči sobě navzájem. „ProBuilder“ pro změnu umožňuje vytvářet a upravovat objekty. Pro vytvoření objektu je nutné kliknout na „NewShape“ v okně „ProBuilder“ a následně vybrat typ objektu a jeho parametry. Dále je objektu dán požadovaný tvar pomocí klávesy Shift, je možné změnit tvar objektu vzhledem k jedné z rovin, ploch nebo vrcholů, aby se objekt změnil bodově. Tímto způsobem lze vytvořit libovolnou místnost.

4.2.4 Hráč

Model

Pro vytvoření jednoduchého modelu postavy v Blenderu slouží dva modifikátory. Pro přidání modelu zaobleného vzhledu se používá modifikátor „Subdivision Surface“. Dále pomocí klávesy E se model natahuje, pomocí klávesy S se model od okrajů zužuje a rozšiřuje, aby se vytvořil např. krk modelu. Po vytvoření těla, krku a hlavy lze začít tvořit ruce a nohy. Stisknutím Ctrl + R se vytvoří uprostřed modelu čára rozdělující poloviny modelu. Dále jsou odstraněny krajní vrcholy jedné z polovin. Poté se použije modifikátor „Mirror“. To umožní vytvořit paži nebo nohu na jedné straně a okamžitě získat symetrickou končetinu na druhé straně.



Obrázek 10: Modelování (vlastní zpracování).

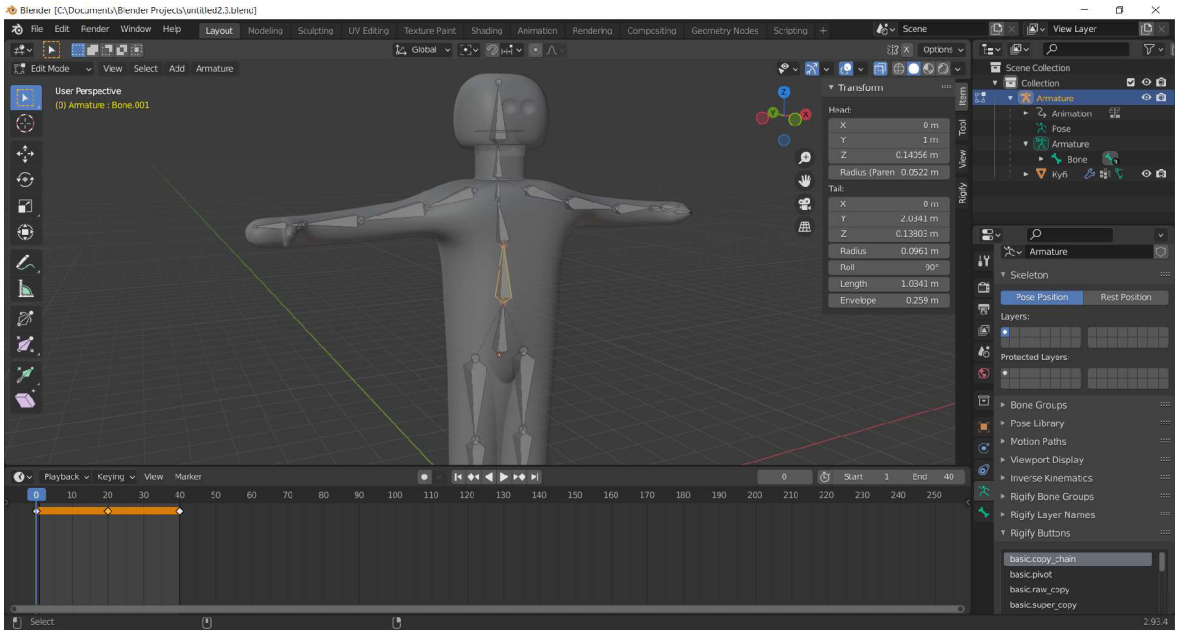
Rigging

„Rigging“ je vytvoření armatury, kostry modelu. Armatura má několik spojených kostí, ke kterým lze připojit vrcholy, aby se pohybovaly, když se kost posouvá. Před vytvořením kompletní struktury je důležité pochopit, jak je potřeba ji nakonfigurovat, aby byla kompatibilní s humanoidním avatarem enginu Unity. Vzhledem k povaze spojování kostí s vlastním systémem Unity jsou vyžadovány některé klíčové kosti.

Unity předpokládá minimálně 15 kostí, což jsou:

- Hips (root bone) - pánev
- Lower spine - spodní část páteře
- Upper spine - horní část páteře
- Neck - krk
- Head - hlava
- Two upper arms - dvě ramena
- Two lower arms - dvě předloktí
- Two hands - dvě ruce
- Two upper legs - dvě stehna
- Two lower legs - dvě holeně

Pro vytvoření kostry je nutné stisknout Shift + A a ze zobrazeného menu vybrat Armature → Single Bone. Dále se zapíná režim úprav bez odstranění výběru z vytvořené kosti. To lze provést rychle stisknutím klávesy Tab. Výběr horního nebo spodního uzlu kosti je proveden kliknutím pravým tlačítkem myši. Stisknutím klávesy E a posunem kurzoru myši libovolným směrem se vytvoří nová kost. Až bude výsledek uspokojivý, lze dokončit tvorbu kosti levým kliknutím myši. Nyní může být vytvořena nová kost z libovolného uzlu opakováním stejných kroků.



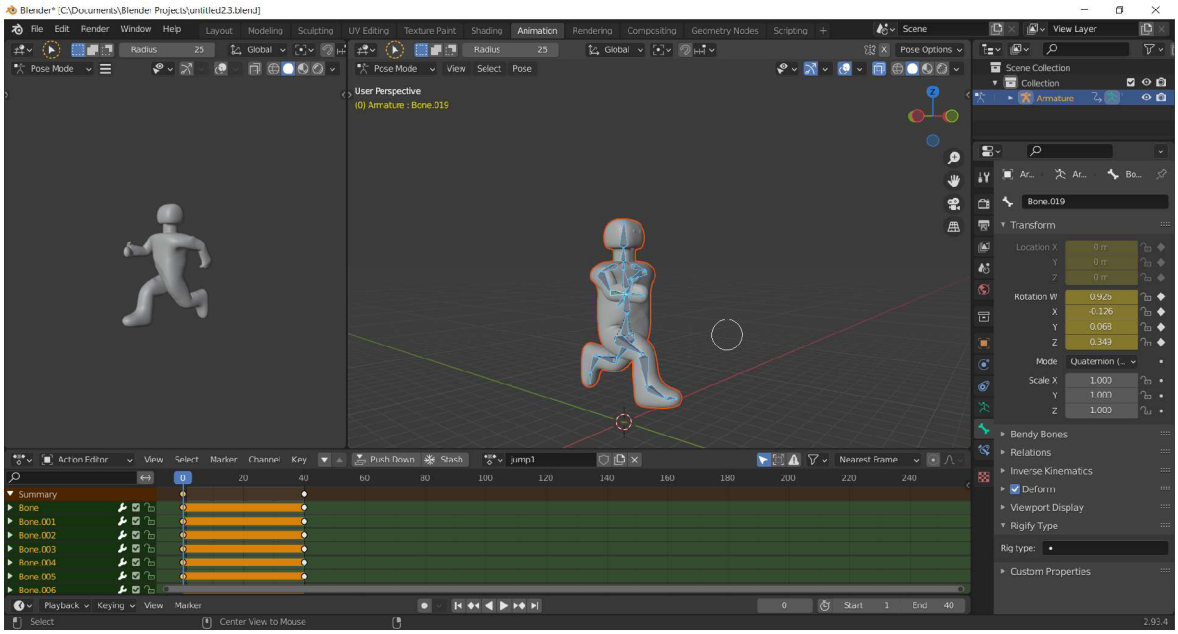
Obrázek 11: Rigging (vlastní zpracování).

Animace

Aby ve hře prováděné akce byly viditelné, musí mít model řadu animací. Hlavní animace pro hru jsou primárně animace nečinnosti (Idle), která se přehrává ve výchozím nastavení, a také animace skoku (Jump) a pohybu (Move), která se přehrává, když se postavy pohybují. Důležitá je také animace útoku (Fight), která se začne přehrávat po stisku tlačítka útoku.

Pro vytvoření animaci v Blenderu je nutné se přepnout do režimu animace. Dále je vybrána požadovaná kost a pomocí klávesy R se upravuje úhel rotace. Když model zaujme požadovanou pozici, musí být snímek animace uložen. Nejprve se vyberou všechny kosti stisknutím klávesy A, poté se stiskne klávesa I → „Location & Rotation“. Snímek animace se tak uloží v požadovanou sekundu. Takto jsou vytvořeny všechny následující snímky animace.

Aby se animace za určitých podmínek navzájem měnily a správně fungovaly po importu do Unity, je třeba v „Animator Controller“ nastavit přechody.



Obrázek 12: Animace (vlastní zpracování).

Zvláštnost exportu

Blender používá pravostranný souřadnicový systém s osou Z směřující nahoru. Téměř všechny CAD systémy používají tento souřadnicový systém. Unity Engine, jako téměř každý jiný herní engine, používá levostranný souřadnicový systém. Když je Y nahoře, X je bokem, Z je dopředu.

Pohyb

Pohyb je prováděn pomocí komponent „Character Controller“ a „Cinemachine“. Pomocí proměnných `float horizontal` a `float vertical` se zaznamenává stisk pohybových kláves vodorovně a svisle. Proměnná `Vector3 direction` obsahuje směr pohybu. Pohyb nastává za podmínky posunu v libovolném směru (`if (direction.magnitude >= 0.1f)`). Proměnná targetAngle nám umožňuje získat úhel natočení postavy pomocí matematické funkce `Mathf.Atan2`, jejíž hodnota se převede z radiánů na stupně (`Mathf.Rad2Deg`) a k tomu se přičte hodnota natočení kamery (`cam.eulerAngles.y`). Funkce `transform.rotation` otočí postavu v závislosti na hodnotě proměnné `angle`, která je součtem hodnoty úhlu natočení a hodnot plynulosti otáčení. Nakonec se model postavy přesune pomocí funkce `controller.Move`.

```
if (direction.magnitude >= 0.1f)
{
    float targetAngle = Mathf.Atan2(direction.x, direction.z) *
    Mathf.Rad2Deg + cam.eulerAngles.y;
    float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
    targetAngle, ref turnSmoothVelocity, turnSmoothTime);
    transform.rotation = Quaternion.Euler(0f, angle, 0f);

    Vector3 moveDir = Quaternion.Euler(0f, targetAngle, 0f) *
    Vector3.forward;
    controller.Move(moveDir.normalized * speed * Time.deltaTime);
 }
```

Animace pohybu se aktivuje voláním komponenty „Animator“.

```
void Start()
{
    animator = GetComponent<Animator>();
}
void Update()
{
    bool hasHorizontalInput = !Mathf.Approximately(horizontal, 0f);
    bool hasVerticalInput = !Mathf.Approximately(vertical, 0f);
    bool isWalking = hasHorizontalInput || hasVerticalInput;
    animator.SetBool("isMoving", isWalking);
}
```

Skok

Skok je realizován pomocí komponent „Character Controller“ a „Animator“. Aby byl skok proveden, musí být splněny dvě podmínky: postava stojí na zemi (grounded) a dojde ke stisknutí mezerníku. Když jsou obě podmínky splněny, animace skoku se začne přehrávat, hodnota gravitace se sníží a postava se posune nahoru.

```
void Update()
{
    grounded = controller.isGrounded;
    animator.ResetTrigger("isJumping");

    if (grounded && velocity.y < 0)
    {
        velocity.y = -2f;
    }

    if (Input.GetButton("Jump") && grounded)
    {
        animator.SetTrigger("isJumping");
        velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
    }

    velocity.y += gravity * Time.deltaTime;
    controller.Move(velocity * Time.deltaTime);
}
```

Kamera 3. osoby

Herní kamera je implementována pomocí rozšiřujícího balíčku „Cinemachine“. V nastavení Cinemachine stačí určit, koho má kamera sledovat (Follow) a na koho se dívat (Look at). Dále v sekci „Orbits“ lze vybrat požadovanou pozici kamery. A aby kamera neprocházela objekty, je potřeba přidat rozšíření „Cinemachine Collider“, kde v položce „Strategy“ je vybráno „Push Camera Forward“ a také v položce „Ignore Tag“ je vybrán tag „Player“. Zároveň je třeba přidat odpovídající tag modelu hráče.

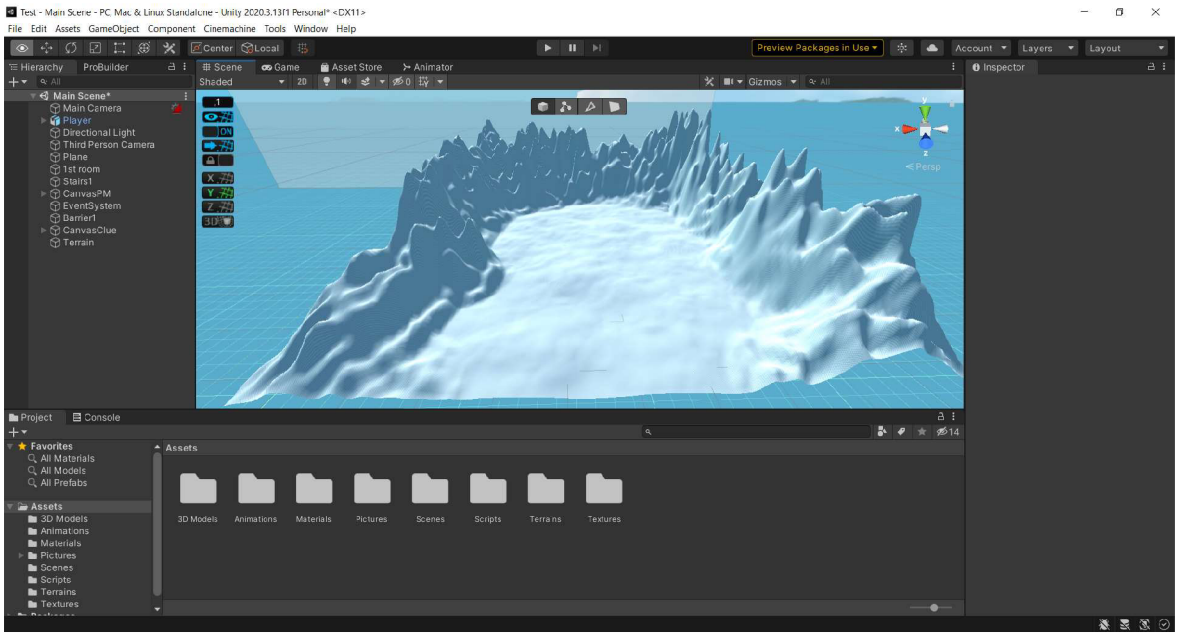
Zdraví

Hlavní hrdina má 3 životy. Jakmile vstoupí do konfrontace, každý zásah nepřítele vezme hrdinovi 1 život. Pokud hrdina dostane 3 zásahy, hra skončí. Každá následující bitva začne se 3 životy hlavního hrdiny.

4.2.5 Velká lokace (Terrain)

Sculpting

Hra musí mít nějaký herní prostor, kde se budou odehrávat všechny události. Takovým prostorem může být velká přírodní lokace. Pro implementaci takové lokace, musí být vytvořen objekt Terén (3D objekt → Terrain). V okně inspektoru se vybírají požadované vlastnosti objektu, jako je velikost a umístění. Dále se krajina lokace vytvoří pomocí nástroje „Brush“ (Paint Terrain → Raise or Lower Terrain). Nástroj štětec je možné přizpůsobit výběrem požadovaného tvaru, oblasti štětce a intenzity.



Obrázek 13: Sculpting (vlastní zpracování).

Textury

Aby terén získal správný vzhled a barvu, musí mít texturu. Textury lze vytvořit buď ručně, nebo je importovat z „Asset Store“, jak tomu bylo v případě tohoto projektu (byl stažen bezplatný balíček materiálů „Outdoor Ground Textures“). Pro import do svého projektu je nutné přidat potřebné materiály do svého účtu Unity na webové stránce „Unity Store“ a poté je stáhnout a importovat do aktuálního projektu (Window → Package Manager).

Proces mapování textur na terén je téměř stejný jako práce s výškou terénu, používá se nástroj štětec. Nejprve je ale potřeba přejít do sekce „Paint Terrain“ → „Paint Texture“. Poté je nutné vytvořit potřebné vrstvy (Layers) „Edit Terrain Layers“ → „Create Layer“. Nyní pomocí nástroje štětec a vybrané vrstvy (Layer) lze na objekt „Terrain“ aplikovat požadovanou texturu.

Vegetace

Aby přírodní lokace působila živě a nikoli prázdně, je potřeba ji zaplnit vegetací, tedy stromy a trávou. Pro vytvoření stromů je nutné přejít na záložku „Paint Trees“ a poté na „EditTree“ → „AddTree“, kde je přidán požadovaný model stromu. Nyní s pomocí nástroje štětec je lokace pokryta stromy. Pro vytvoření trávy je nutné přejít na záložku „Paint Details“

a poté na „EditDetails“ → „AddGrassTexture“, kde se přidá požadovaná textura trávy. Nyní je lokace pomocí nástroje štětec pokryta trávou a celá lokace je tak pokryta vegetací.

4.2.6 Nepřítel

Model

Model nepřítele a animace byly vytvořeny podobně jako při tvorbě modelu postavy, až na rozdíl ve volbě proporcí modelu (viz kapitola 4.2.4). Aby model nepřítele správně interagoval s prostředím, musí mít komponentu „Capsule Collider“.

Model chování

Model chování nepřítele spočívá v tom, že se nepřítel začne pohybovat směrem k hráči, jakmile se model postavy objeví v jeho zorném poli, a také začne útočit, jakmile se přiblíží na dostatečnou vzdálenost.

Nepřítel bude hráče pronásledovat díky komponentě „Nav Mesh Agent“ a skriptu, který umožní nastavit nepříteli cíl pronásledování.

```
void Start()
{
    agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
}

void Update()
{
    agent.SetDestination(target.position);
 }
```

Aby se pronásledovatel pohyboval správně a komponenta „Nav Mesh Agent“ správně fungovala, musí být rovina, po které se objekt pohybuje, označena jako statická (Static) a musí být zapečená.

Zaškrtnutím políčka „Navigation Static“ budou zahrnuty vybrané objekty do procesu pečení „NavMesh“. Bake neboli „zapéct“ je proces vytváření samotného „NavMesh“ pro scénu.

Zorné pole nepřítele musí být omezeno triggerem, aby postavu nekonečně nepronásledoval a hned si jí nevšiml. K tomu je vytvořen podřízený objekt, ve kterém je „Mesh Collider“ vypnuto a „Collider“ je nakonfigurován v souladu s požadovaným zorným polem a má zapnuto „Is Trigger“. Nakonec je vytvořen skript, který mění rychlost komponenty „Nav Mesh Agent“ na 3.5, aby se nepřítel mohl pohybovat, když je hráč v zorném poli nepřítele, a mění rychlost teto komponenty na 0, aby se nepřítel nemohl

pohybovat, když se hráč přesune z tohoto zorného pole. Připojena je i komponenta „Animator“, takže se animace spouští při pohybu nepřítele.

Výsledkem je, že nepřítel bude pronásledovat model postavy, jakmile se model dostane do jeho zorného pole, a naopak přestane pronásledovat, jakmile model postavy zmizí z dohledu.

```
void Update()
{
    if (ExitTrigger==true)
    {
        Enemy.GetComponent<Animator>().SetBool("isMoving", false);
        Enemy.GetComponent<NavMeshAgent>().speed = 0f;
    }
    else
    {
        Enemy.GetComponent<Animator>().SetBool("isMoving", true);
        Enemy.GetComponent<NavMeshAgent>().speed = 3.5f;
    }
 }
```

Jakmile se nepřítel dostane dostatečně blízko k postavě, zastaví se a začne útočit pomocí dalšího triggeru. Nepřítel přestane útočit a začne znovu pronásledovat, jakmile se postava vzdálí na krátkou vzdálenost.

```
if (other.tag == "Player")
{

    Enemy.GetComponent<NavMeshAgent>().speed = 0f;
    Enemy.GetComponent<Animator>().SetBool("isFighting", true);
    EnWeapon1.SetActive(true);
    EnWeapon2.SetActive(true);
 }
```

4.2.7 Bojový systém

Bojový systém hry je založen na triggerech. Když se neviditelné objekty připevněné k pažím postavy dotknou neviditelného objektu, který je připevněn k modelu nepřítele a zobrazuje přibližné rozměry tohoto modelu nepřítele, s komponentou „Rigidbody“, která má pro korektní fungování vypnuté nastavení gravitace (Use Gravity → Off), model nepřítele se změní na „Ragdoll“, čímž se zobrazí tělo bez života.

Pro vytvoření Ragdolla se zkopíruje objekt, kterým je postava. Všechny komponenty připojené k objektu jsou odstraněny. Nyní je vytvořen Ragdoll (Game Object → 3D Object → Ragdoll). V okně nastavení „Ragdoll“ jsou připojeny potřebné kosti modelu (Bones), pro které je Ragdoll vytvořen. Po rozložení všech kostí je potřeba nastavit Collider tak, aby

model Ragdoll správně interagoval s prostředím. Když je Ragdoll hotový, lze vytvořit prefab tohoto modelu jednoduchým přetažením modelu z okna Hierarchie do okna „Assets“.

```
void OnTriggerEnter(Collider other)
{
    if (other.tag == "PlayerWeapon")
    {
        Enemy.SetActive(false);
        Ragdoll.SetActive(true);
        Instantiate(Ragdoll, transform.position, transform.rotation);
    }
 }
```

Podobně funguje i útok nepřítele. K vítězství však musí nepřítel postavu zasáhnout třikrát. Pokud se tak stane, model postavy se také změní na Ragdoll a hra skončí prohrou.

```
void OnTriggerEnter(Collider other)
{
    if (other.tag == "EnemyWeapon")
    {
        HitsCount++;
    }

    if (HitsCount == 3)
    {
        Player.SetActive(false);
        Ragdoll.SetActive(true);
        Instantiate(Ragdoll, transform.position, transform.rotation);
        GameOverMenuUI.SetActive(true);
    }
}
```

4.2.8 Skybox

„Skybox“ je panoramatická textura, která se prostírá za všemi objekty ve scéně a simuluje tak oblohu, velké město nebo jakoukoli jinou perspektivu na velké vzdálenosti. Jinými slovy, Skybox je jakýmsi pozadím herního světa.

Pro vytvoření Skyboxu musí být předem připraven požadovaný obrázek a poté v editoru Unity je obrázek připraven k použití. Dále je vytvořen nový materiál s nastavením shaderu „Skybox/Cubemap“. Nakonec se připravený obrázek vloží do okna „Cubemap“ a Skybox je připraven k použití.

4.2.9 Osvětlení

V Unity jsou všechny objekty rozděleny na dynamické (dynamic) a statické (static). Statické objekty jsou takové, které vždy zůstanou na místě a nikam se nepohybují. Právě pro ně probíhá „pečení“ osvětlení, proto musí být všechny objekty tohoto typu označeny jako „Static“. Dynamické objekty jsou ty, které jsou naopak v pohybu.

Pro automatické vytvoření světelné mapy stačí kliknout na „Generate Lighting“ v okně „Lighting“. Použití GPU ke generování světelné mapy značně urychlí proces.

4.2.10 Konec Hry

Výhra

Pokud se hráč po překonání všech překážek dostane na konec lokace, má možnost hru dokončit. Při interakci s objektem „Loď“ („Boat“) bude fungovat trigger a objeví se menu „WinGame“, kde si hráč může vybrat jednu ze tří akcí: Spustit hru znovu („Start“), Hlavní menu („Main menu“), Ukončit hru („Exit“).

Popup menu „WinGame“ bylo implementováno podobným způsobem jako menu pauzy (viz kapitola 4.2.2). Jediný rozdíl je v tom, že se menu „WinGame“ objeví při aktivaci triggeru.

```
void OnTriggerEnter(Collider other)
{
    WinGameMenuUI.SetActive(true);
    Time.timeScale = 0f;
 }
```

Prohra (Game Over)

Pokud jsou ztraceny všechny životy postavy, hra skončí prohrou a objeví se menu „Game Over“, kde si hráč může vybrat jednu ze tří akcí: Spustit hru znovu („Start“), Hlavní menu („Main menu“), Ukončit hru („Exit“).

5 Výsledky a diskuse

Výsledný prototyp hry má všechny plánované funkce a také stabilně fungující komponenty popsané v teoretické části práce a implementované v praktické části. Před začátkem tvorby práce měl autor velmi malé znalosti týkající se teorie a téměř žádné zkušenosti s prací v enginu Unity a programu Blender.

Nejnáročnější bylo na začátku práce zvládnutí základních funkcí 3D modelovacího programu Blender. Další komplikací v procesu vývoje hry byla implementace pohybu modelu postavy. Problém byl vyřešen hlubším zkoumáním komponenty „Character Controller“. Ale bohužel skok hlavního hrdiny při pohybu po nerovném povrchu zůstal nepřesný. Rozdíl mezi souřadnicovými systémy Blender a Unity se také stal malou komplikací, která způsobovala chybný import modelů do Unity. Problém byl vyřešen výběrem správného nastavení exportu.

Systém pronásledování hlavního hrdiny nepřítelem realizovaný pomocí „NavMesh“ a komponenty „Nav Mesh Agent“ funguje nečekaně korektně. Bezchybně fungují i všechny herní mechanismy založené na triggerech, včetně vyskakovacích herních menu.

V průběhu práce si autor osvojil počáteční dovednosti modelování, animace a také práce ve vývojovém prostředí Unity. Prototyp hry je otevřený a připravený na další vylepšení a změny.

6 Závěr

Cílem bakalářské práce bylo navrhnout a vytvořit herní aplikaci v prostředí Unity za použití programovacího jazyka C# a programu 3D modelování Blender a také seznámit čtenáře s celým procesem vývoje.

Teoretická část práce se věnovala základním prvkům a principům Unity, které jsou nezbytné v rámci vývoje hry. Byly popsány komponenty a nástroje, které byly použity v procesu vývoje aplikace v prostředí Unity, a také nástroje a modifikátory 3D modelovacího programu Blender. Na začátku praktické části práce byly stanoveny požadavky na vytvářenou hru. Dále byly popsány způsoby implementace herních prvků a také principy fungování komponent enginu Unity.

Výsledkem této práce je hra, kterou hráč může spustit, pozastavit a dokončit. Hráč se může volně pohybovat po lokaci a vstoupit do konfrontací s nepřáteli. Hra má všechna potřebná menu a prvky rozhraní pro komunikaci s hráčem. Hra je plně funkční, ale má své nevýhody. Herní modely hlavní postavy i nepřítele nejsou příliš detailní. Herní svět také neobsahuje mnoho detailů. S většími zkušenostmi bude možné v budoucnu vytvářet složitější a detailnější modely. Měl by být opraven jeden z klíčových nedostatků, a to skok, který ne vždy funguje správně. Ve hře chybí sprint a také systém sbírání předmětů pro plnění questů a zkomplikování bojového systému. Hru lze později rozšiřovat a doplňovat.

Tato práce může pro začínajícího vývojáře sloužit jako ukázka možných problémů v určitých fázích vývoje hry, které by bylo dobré brát v zvláštní úvahu.

7 Seznam použitých zdrojů

AVERSA, Davide, Aung Sithu KYAW a Clifford PETERS, 2018. Unity Artificial Intelligence Programming: Add powerful, believable, and fun AI entities in your game with the power of Unity 2018!. 4th Edition. Birmingham: Packt Publishing. ISBN 9781789533910.

BLACKMAN, Sue, 2014. Unity for Absolute Beginners [online]. Berkeley, CA: Apress [cit. 2022-02-20]. ISBN 978-1-4302-6779-9. Dostupné z: doi:10.1007/978-1-4302-6778-2

Blender Foundation, 2021. Blender - Manual: Blender 2.93 Reference Manual [online]. [cit. 2022-02-21]. Dostupné z: https://docs.blender.org/manual

Blizzard Entertainment, 2022. Hearthstone Official Game Site. Hearthstone [online]. [cit. 2022-02-27]. Dostupné z: https://playhearthstone.com/

BOND, Jeremy Gibson, 2018. Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#. Second edition. Upper Saddle River, NJ: Addison-Wesley. ISBN 978-0-13-465986-2.

BUTTLE, Paul, 2020. The Power Behind Video Games: A Look at Game Engines. We The Players, Medium [online]. [cit. 2022-02-21]. Dostupné z: https://medium.com/wetheplayers/the-power-behind-video-games-a-look-at-game-engines-2731315086e0

CRAIGHEAD, J., R. GUTIERREZ, J. BURKE a R. MURPHY, 2008. Validating the Search and Rescue Game Environment as a robot simulator by performing a simulated anomaly detection task. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems [online]. IEEE, 2008, s. 2289-2295 [cit. 2022-03-10]. ISBN 978-1-4244-2057-5. Dostupné z: doi:10.1109/IROS.2008.4650746

HANÁK, Ján, 2008. Objektovo orientované programovanie v jazyku C# 3.0: (príručka pre vývojárov, programátorov a softvérových expertov). Brno: Artax. Microsoft (Artax). ISBN 978-80-87017-02-9.

HOCKING, Joseph, 2018. Unity in action: multiplatform game development in C#. Second edition. Shelter Island: Manning. ISBN 978-1617294969.

HOPPE, H. a Derya Güleç ÖZER, 1999. New quadric metric for simplifying meshes with appearance attributes. Proceedings Visualization '99 (Cat. No.99CB37067) [online]. IEEE, 9(7), 59-510 [cit. 2022-03-13]. ISBN 0-7803-5897-X. ISSN 2332-1091. Dostupné z: doi:10.1109/VISUAL.1999.809869

SHARP, John, 2018. Microsoft Visual C# Step by Step. 9th Edition. Redmond, Wash.: Microsoft Press. ISBN 978-1509307760.

Superhot Team, 2022. SUPERHOT - The FPS where time moves only when you move [online]. [cit. 2022-02-27]. Dostupné z: https://superhotgame.com/superhot

Unity Asset Store. Unity Asset Store [online]. Copyright © 2022 Unity Technologies [cit. 13.03.2022]. Dostupné z: https://assetstore.unity.com/

Unity - Manual: Unity User Manual 2020.3 (LTS). [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 2022-03-13]. Dostupné z: https://docs.unity3d.com/Manual/index.html

Unity - Scripting API: . [online]. Copyright ©2022 Unity Technologies. Publication Date [cit. 2022-03-13]. Dostupné z: https://docs.unity3d.com/ScriptReference/index.html

Unity Real-Time Development Platform | 3D, 2D VR & AR Engine [online]. Copyright © 2022 Unity Technologies [cit. 2022-03-13]. Dostupné z: https://unity.com/

ZENDLE, David, Daniel KUDENKO a Paul CAIRNS, 2018. Behavioural realism and the activation of aggressive concepts in violent video games. Entertainment Computing [online]. 24, 21-29 [cit. 2022-02-21]. ISSN 18759521. Dostupné z: doi:10.1016/j.entcom.2017.10.003

ŽOLTÁ, Lucie, 2022. Objektově orientované paradigma [online]. [cit. 2022-02-21]. Dostupné z: http://lucie.zolta.cz/index.php/softwarove-inzenyrstvi/141-objektove-orientovane-paradigma

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů

8.4 Seznam použitých zkratek

RMB - Right Mouse Button (Pravé tlačítko myši)
CAD - Computer Aided Design (Počítačově podporovaný design)
GPU - Graphics Processing Unit (Grafický procesor)
AI - Artificial intelligence (Umělá inteligence)

9 Přílohy

9.1 Ukázkové snímky ze hry

Obrázek 14: Skok postavy (vlastní zpracování).

Obrázek 15: Herní menu „Win Game" (vlastní zpracování).