Komponenty v Unity

Nikola Valvodová

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Nikola Valvodová**
Osobní číslo:	**A20526**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Komponenty v Unity**
Téma práce anglicky:	**Components in Unity**

Zásady pro vypracování

1. Vypracujte literární rešerši na zadané téma.
2. Navrhněte edukační prototyp hry se zaměřením na demonstraci použití komponent v Unity.
3. V rámci návrhu zohledněte i vhodnou interakci uživatele s herními objekty, a také vhodné zobrazení informací o komponentách, které jsou v nich použity.
4. Implementujte edukační prototyp hry dle návrhu.
5. Vhodně popište vytvořenou aplikaci a její ovládání.

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

Seznam doporučené literatury:

1. WALKER, Philip. Unity Certified Programmer: Exam Guide. Birmingham, UK: Packt, 2020. ISBN 9781838828424.
2. SMITH, Matt a Shaun FERNS. Unity 2021 Cookbook. 4. Birmingham, UK: Packt, 2021. ISBN 9781839217616.
3. WELLS, Robert. Unity 2020 By Example. 3. Birmingham, UK: Packt, 2021. ISBN 9781800203389.
4. BORROMEO, Nicolas Alejandro. Hands-On Unity 2021 Game Development. 2. Birmingham, UK: Packt, 2021. ISBN 9781801071482.
5. Unity User Manual 2021.3 (LTS) [online]. San Francisco, USA: Unity Technologies, 2022 [cit. 2022-11-28]. Dostupné z: https://docs.unity3d.com/Manual/index.html.

Vedoucí bakalářské práce: **Ing. Tomáš Vogeltanz, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

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

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 23.5.2023

Nikola Valvodová, v.r.
podpis studenta

ABSTRAKT

Tato bakalářská práce je zaměřena na komponenty v herním enginu Unity. V teoretické části je stručný popis prostředí Unity, za kterým následuje charakteristika komponent a jejich nejdůležitější parametry.

V praktické části je navržen a implementován prototyp 3D hry, na kterém jsou následně popisované jednotlivé komponenty. K popisu komponent zde byl využit nástroj Unity in-Editor Tutorials.

Klíčová slova:

Unity, 3D hra, C#, komponenty, in-Editor Tutorials.

ABSTRACT

This bachelor's thesis is focused on components in the Unity game engine. The theoretical part contains a brief description of the Unity environment, followed by the characterization of the components and their most important parameters.

In the practical part, a 3D game prototype is designed and implemented, on which individual components are described. The Unity in-Editor Tutorials tool was used to describe the components.

Keywords:

Unity, 3D game, C#, components, in-Editor Tutorials.

Na tomto místě bych chtěla poděkovat vedoucímu práce Ing. Tomáši Vogeltanzovi za všechny cenné rady a jeho trpělivost.

Dále bych ráda poděkovala rodině, přátelům a mému příteli za podporu a toleranci během studia.

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

OBSAH

ÚVOD

Cílem této práce je navržení a následné vytvoření prototypu hry. Ten bude zaměřen na představení a popis komponent v Unity.

V teoretické části bude stručně popsán Unity Engine a zároveň bude tato část charakterizovat nejpodstatnější komponenty a jejich parametry.

Praktická část bude rozdělena na dvě fáze. V první části bude navržen a implementován prototyp 3D hry, který bude obsahovat terén, ve kterém bude vytvořena pohybující se postava reagující na vstupy uživatele.

Druhá část se zaměřuje na vytvoření edukativních návodů, zaměřené na nejpodstatnější Unity komponenty.

Pro docílení efektivní interakce uživatele nejen s herními objekty, ale i s Unity Editorem jako takovým, byl zvolen poměrně nový nástroj Unity in-Editor Tutorials, který poskytuje možnost tvorby tutoriálů přímo uvnitř samotného Unity Editoru.

Díky funkcím in-Editoru se může uživatel interaktivně zapojit při tvorbě herních objektů, na které jsou pak navázané jednotlivé komponenty. Za pomocí vytvořených tutoriálů budou tak funkce jednotlivých komponent vysvětleny, přičemž s nimi bude uživatel během celého procesu interagovat. To zajistí jeho přímou participaci v celém procesu a uživatel tak získá informace o jednotlivých komponentách přímo z praktické ukázky.

Vytvořené manuály budou sloužit pro začínající herní vývojáře, přičemž bude možné projekt využít v předmětu Vývoj počítačových her.

I. TEORETICKÁ ČÁST

1 UNITY ENGINE

Unity je jedním z více multiplatformních herních enginů na trhu, který umožňuje vývoj virtuální a rozšířené reality, simulátorů, 2D a 3D her a mnoho dalšího.

Unity nejenže je uživatelsky velmi přívětivé, ale zároveň má kolem sebe silnou komunitu, čímž se tak (především pro začínající vývojáře) stává velmi atraktivním. Není tak problém najít jakýkoliv tutoriál, dokumentaci nebo odpověď na časté dotazy či problémy.

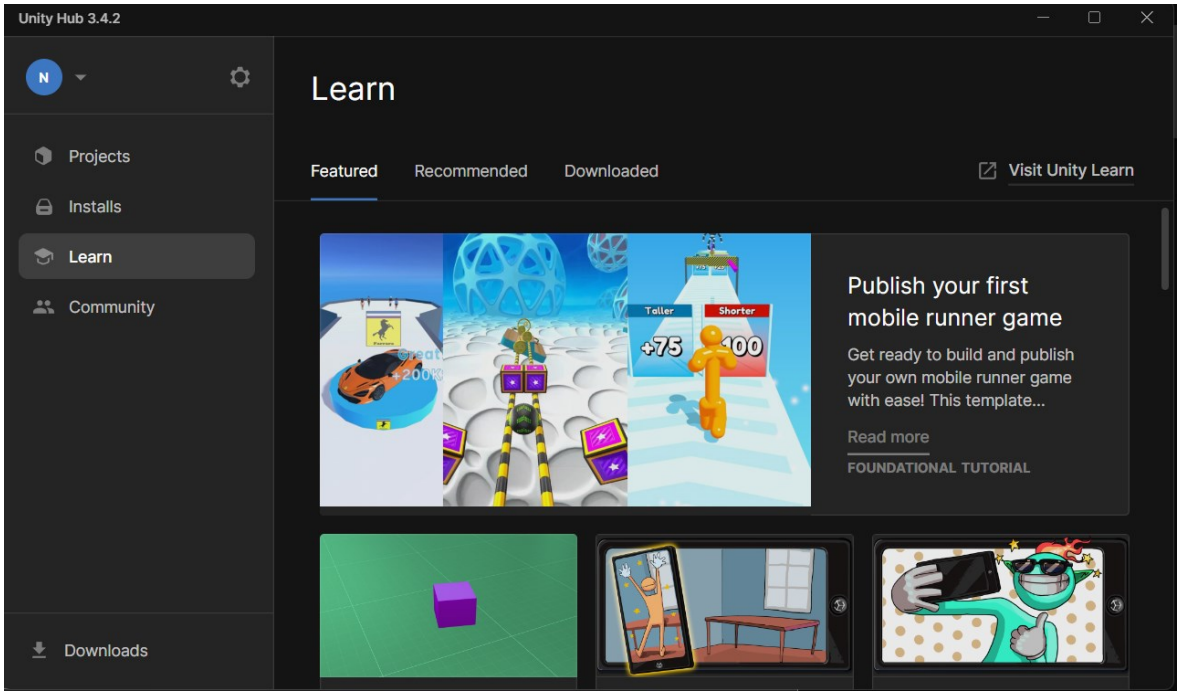
Nedílnou součástí herních enginů jsou také volně dostupné assety, které můžou ulehčit práci při vývoji hry. Co se Unity týče, je zde k dispozici především Unity Asset store, kde lze najít hotové herní objekty, grafické efekty, zvuky, scripty apod. Dalším takovým příkladem může být Mixamo od Adobe, kde můžeme najít postavy a jejich animace.

Během vývoje her je potřeba vytvářet i vlastní skripty, které ovlivňují chování herních objektů, k čemuž Unity Engine využívá programovací jazyk C#.

1.1 Unity Hub

Podstatným prvkem rodiny Unity je Unity Hub. Jedná se o užitečný management nástroj, který nám umožňuje správu a aktualizaci jak našich projektů, tak i samotného Unity enginu. Máme zde k dispozici 4 záložky s následujícím obsahem: [5]

- Projects: Souhrnný náhled na všechny vytvořené projekty.
- Installs: Všechny nainstalované verze Unity Editoru, lze je zde přidat i odinstalovat.
- Learn: Užitečné tutoriály, obsahově jsou zde vhodné návody nejen pro začátečníky, ale i pro zkušené vývojáře, kteří chtějí jít více do hloubky.
- Community: Užitečné odkazy např. do Asset Storu, Unity fóra apod.

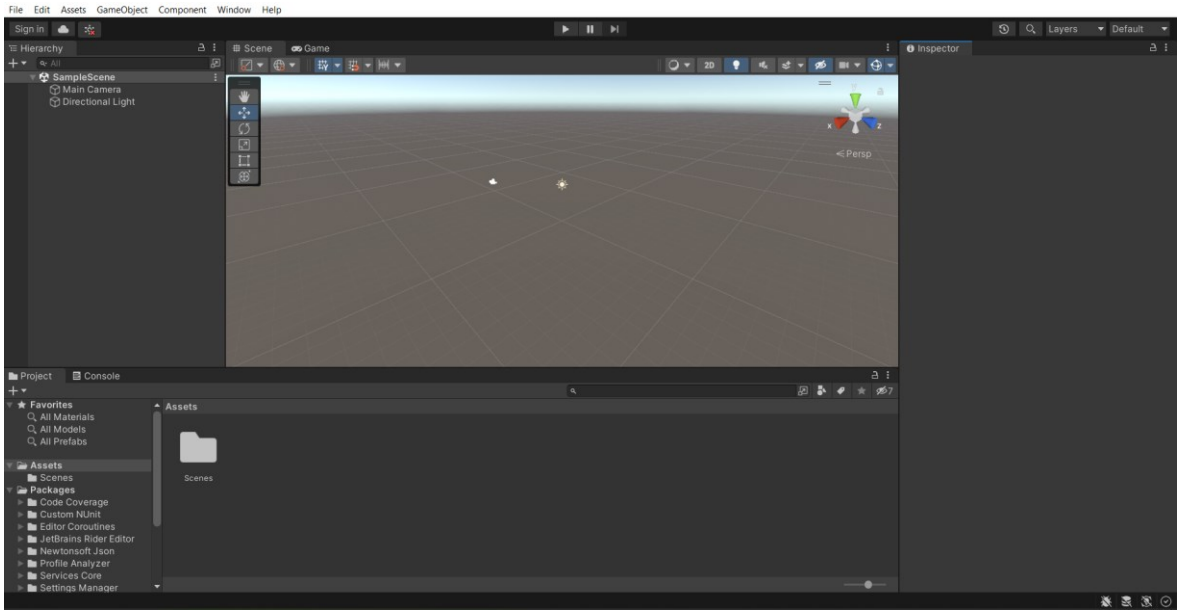


Obrázek 1: Unity Hub

1.2 Prostředí unity

Prostředí Unity se obdobně jako u jiných vývojových systému skládá z oken a panelů s variabilním rozložením, které si uživatel může upravit dle vlastních preferencí.

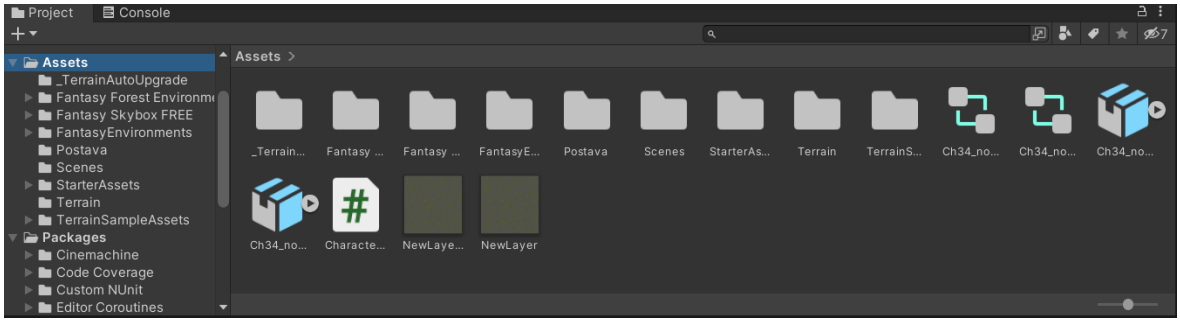
Po spuštění je defaultně dáno rozvržení do sekcí Hierarchy, Project, Scene, Inspector a Game.



Obrázek 2: Prostředí unity bezprostředně po spuštění

1.2.1 Project window

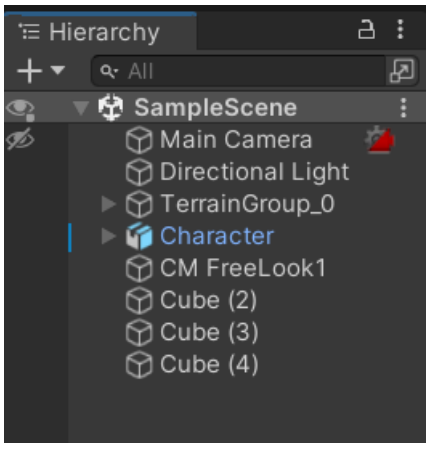
V Project window vidíme veškeré vytvořené nebo importované soubory/assety, jako jsou například textury, animace, skripty apod. Díky Project window, který funguje jako běžný souborový průzkumník, si můžeme organizovat jednotlivé objekty do složek a udržovat tak náš projekt přehledný. V případě potřeby zde lze objekt vyhledat a třeba ho i rovnou přetáhnout do scény. [2]



Obrázek 3: Project window

1.2.2 Hierarchy window

Hierarchy window je další klíčovou součástí základního rozložení oken v Unity projektu. V hierarchické struktuře zde vidíme všechny herní objekty, které se nachází ve scéně. Můžeme zde strukturovat herní objekty do struktury rodič – potomek, což nám může při vývoji pomoci například v případě, pokud potřebujeme organizovat či měnit více stejných prvků ve scéně naráz. [4]



Obrázek 4:Hierarchy window

1.2.3 Scene window

Scene window slouží k náhledu a editaci aktuální scény, ve které se hra odehrává. V tomto okně vidíme vytvořené herní objekty a zároveň i menší lištu s nástroji, které nám s objekty umožňují základní operace.

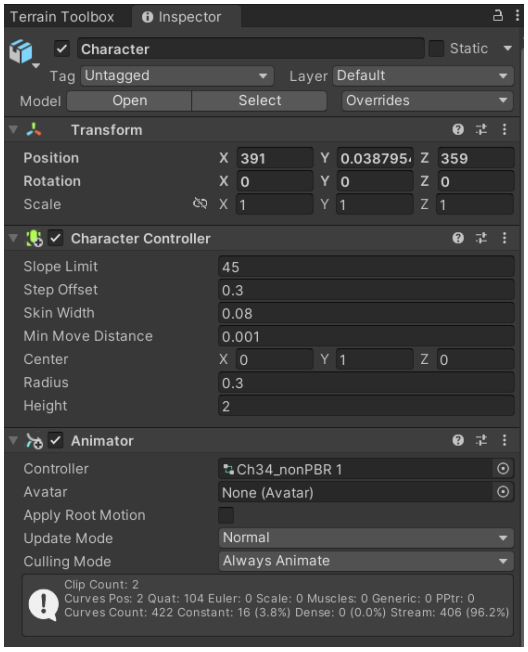
Obrázek 5:Scene Window

Herní objekty zde tedy můžeme vytvářet, upravovat nebo mazat. Díky již zmíněným nástrojům jsou zde k dispozici jejich úpravy jako jsou rotace, změna pozice nebo velikosti daného objektu. Při prvním spuštění lze v tomto okně vidět pouze dva herní objekty – kameru a světlo. [1]

1.2.4 Inspector window

Okno Inspectoru nám poskytuje zobrazení podrobných informací o vybraném herním objektu. Zobrazuje se zde hierarchická struktura všech vlastností, komponent a dalších atributů, které jsou navázané na daný herní objekt a v podstatě tak lze okno inspektoru považovat za nástroj, který nám umožňuje s komponentami pracovat. [6]

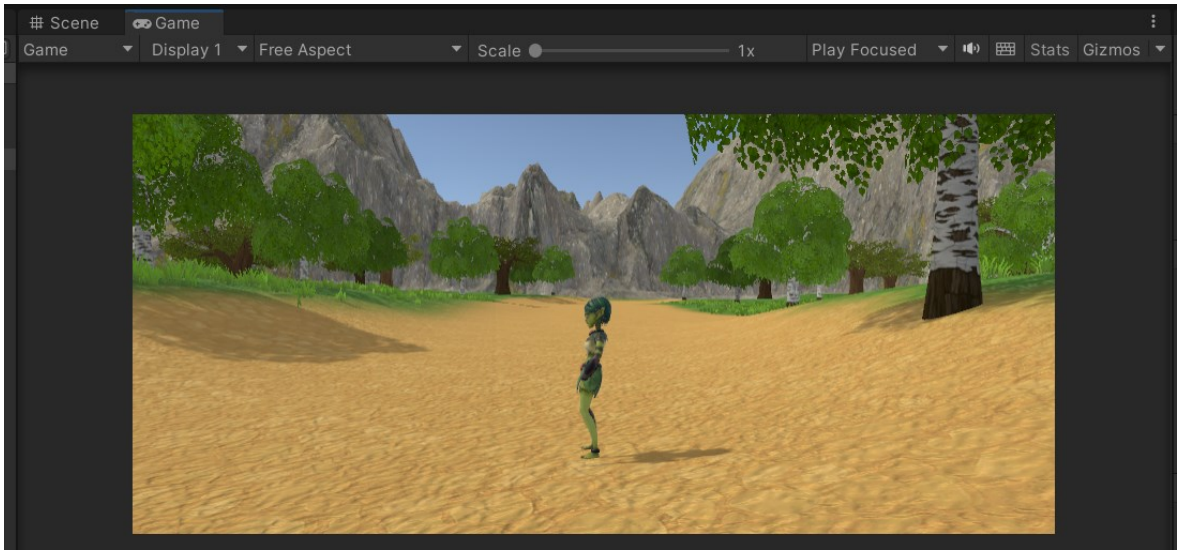
Kromě náhledu je zde také možné zvoleným objektům komponenty přidávat, mazat nebo upravovat jejich parametry a hodnoty, jako jsou velikost, pozice, barva apod.



Obrázek 6: Inspector Window

1.2.5 Game view

Díky tomuto oknu máme možnost náhledu na naši hru z pohledu nastavené kamery v projektu. Můžeme tedy průběžně sledovat či testovat stav, chování a celkovou interakci prvků při spuštění hry samotné. K testování možných změn nám slouží tzv „play mode“, ve kterém jsou změny pouze dočasné a je tak pro vývojáře pohodlnější testovat možné varianty zobrazení nebo vzájemnou interakci herních objektů a jejich komponent. [3]



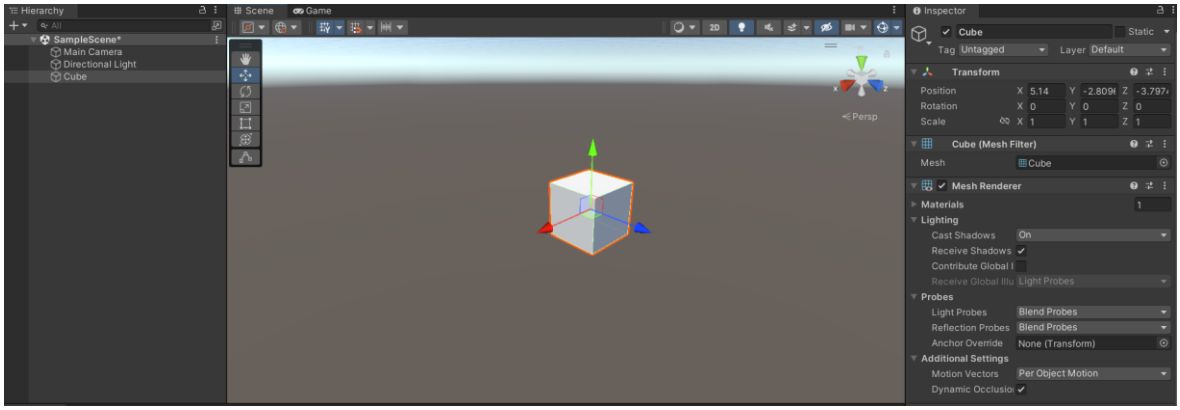
Obrázek 7: Game view

2 HERNÍ OBJEKTY A KOMPONENTY

Herní objekty (GameObjecty) jsou základním stavebním prvkem her v Unity. Je to každá nezávislá entita, kterou lze vidět v hierarchy nebo scene window – tedy nejen vytvořená postava a terén, ale i světlo, kamera, zvuk, a dokonce i prázdné herní objekty.

Herní objekty jako takové jsou ve své podstatě jen prázdné, nic nedělající schránky, a právě to, co jim dává potřebné vlastnosti, jsou komponenty.

Každá komponenta je svým způsobem unikátní a v případě potřeby je možné si pomocí skriptu vytvořit svoji vlastní. Tím lze měnit chování jednotlivých herních objektů nebo také jiných komponent, které jsou na daný GameObject navázané. Díky možnosti tvorby vlastních skriptů a možného přístupu k parametrům a metodám Unity komponent tak nejsme nijak omezeni funkčností předdefinovaného chování. V oficiální Unity dokumentaci vždy nalezneme popis komponenty a parametrů, které vidíme přímo v editoru (component reference – manual), zároveň je ale dostupná i scripting dokumentace, kde vidíme přístupné parametry v případě tvorby vlastních skriptů. [43]



Obrázek 8: GameObject s komponentami

Pokud v hierarchy window nebo přímo ve scéně vybereme herní objekt, v okně inspektoru vidíme všechny komponenty, které jsou na daný herní objekt navázané. Pomocí tlačítka „Add Component“ lze dle potřeby přidávat další komponenty.

Komponenty můžou fungovat nezávisle jako takové, ale běžně jsou na pozadí propojené a vzájemně spolu interagují. Příkladem může být komponenta Mesh Renderer, která vykresluje herní objekt v místě, na které odkazuje komponenta Transform. [6]

Obdobným způsobem jsou využívané námi vytvořené komponenty (tvořené vlastními skripty), ve kterých se odkazujeme na parametry jiných komponent a můžeme je díky tomu

měnit (aktualizovat). Příkladem může být vytvořený script pro pohyb postavy, který pravidelně aktualizuje komponentu Transform, a díky tomu se může postava v prostoru pohybovat.

Komponenty se dělí do několika logicky seskupených kategorií na základě jejich funkčností a použití, přičemž je možné najít komponenty ve variantě 2D i 3D.

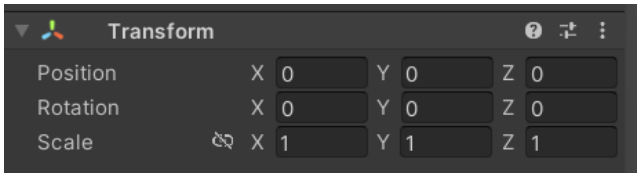
Jedná se o následující kategorie:

- Audio
- Effects
- Event
- Layout
- Mesh
- Miscellaneous
- Navigation
- Physics 2D
- Physics
- Playables
- Rendering
- Scripts
- Tilemap
- UI Toolkit
- UI
- Video
- Visual Scripting

2.1 Komponenta Transform

Základní komponentou všech herních objektů je komponenta „Transform“. Tato komponenta zobrazuje pozici, rotaci a velikost herního objektu ve scéně, a to pomocí souřadnicového systému, který je zde reprezentován osami X, Y, Z. Pozice herního objektu se určuje pomocí vzdálenosti (offsetu) od středobodu scény, který má souřadnice 0,0,0. [2]

Podstatnou charakteristikou této komponenty je především to, že se nedá samostatně přidat ani odebrat a není ani tak součástí žádné skupiny komponent. Pokud je tedy herní objekt vytvořen nebo vložen do scény, vždy automaticky obsahuje komponentu Transform. [6]



Obrázek 9. Komponenta Transform

2.2 Komponenty skupiny Physics a Physics 2D

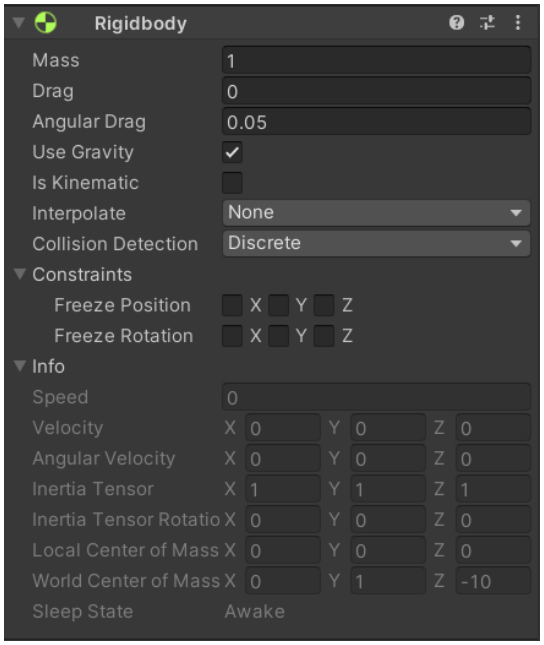
Tato skupina komponent, a celkově Unity Physics Engine, který je defaultně součástím Unity, jsou základním stavebním kamenem pro tvorbu her. Díky těmto prvkům lze v herním prostředí simulovat fyzikální zákony, což nám umožňuje tvořit herní objekty, které dokáží reagovat například na kolize, hybnost nebo být ovlivněny gravitací. To nám ve své podstatě umožňuje simulovat chování objektů jako v reálném světě. [6]

Nejdůležitějšími komponentami z této skupiny jsou Rigidbody a Collidery.

2.2.1 Rigidbody a Rigidbody 2D

Rigidbody je komponenta, která herním objektům dodává již zmíněné reakce na fyzikální zákony. Skrz nastavitelné parametry ovlivňujeme chování herního objektu. Zde jsou nejdůležitější z nich: [7]

- Mass: Parametr pro nastavení hmoty objektu, přičemž čím větší bude, tím méně bude GameObject s touto komponentou reagovat na působení sil jiných objektů.
- Drag a Angular Drag: Slouží pro simulaci odporu vzduchu. Stejně jako v reálném světě, čím větší hodnota bude, tím více bude GameObject vlivem odporu zpomalovat při vyvolaném pohybu či rotaci.
- Use Gravity: Důležitý parametr, který nastavuje objektu reakci na gravitační sílu.
- Collision Detection: Pro nastavení detekce kolizí.
- Is Kinematic: Tento parametr je užitečný ve chvíli, kdy využíváme vlastní skript pro pohyb postavy, a tím pádem je náš pohyb ovlivněn daným skriptem, nikoliv fyzikální silou.
- Constraints: Zde můžeme omezit působení fyzikálních sil v určitém směru pohybu či rotace.

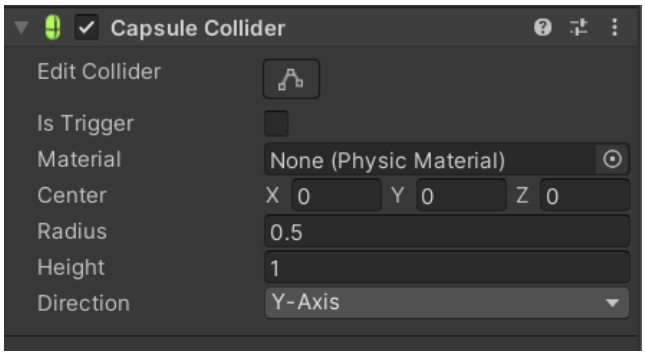


Obrázek 10: Komponenta Rigidbody

2.2.2 Collidery

Dalšími komponentami, které patří do těch nejpodstatnějších, jsou komponenty podskupiny Collider. Colliderů je několik variant, v závislosti na tvaru herního objektu, který tuto komponentu obsahuje. Základními tvary jsou zde Box, Sphere, nebo Capsule. Dalšími možnými variantami jsou například komponenta Mesh Collider, Terrain Collider nebo Wheel Collider.

Tyto komponenty nám zajišťují reálné chování objektů, co se jejich vzájemných kolizí týče. Aby například vytvořená postava neprošla zdí, je potřeba nastavit objektu zdi komponentu Box Collider a postavě samotné dodat komponentu Rigidbody v kombinaci s Capsule Collider nebo Character Controller. Z toho důvodu je také důležité, abychom Collideru co nejpřesněji nastavili parametry jako jsou například Center, Heigh, Size a Radius (pro Capsule Collider), které nám zajistí přesné ohraničení objektu, a tím i jejich bezchybnou kolizi.



Obrázek 11: Capsule Collider

Obecně platí, že čím jednodušší tvar Collideru, tím méně za to platíme výkonem hry, vzhledem k tomu, že výpočet kolizí na pozadí je poměrně náročný. [6] Nejúspornější variantou je Sphere a Circle Collider. Collidery mají 2D i 3D varianty.

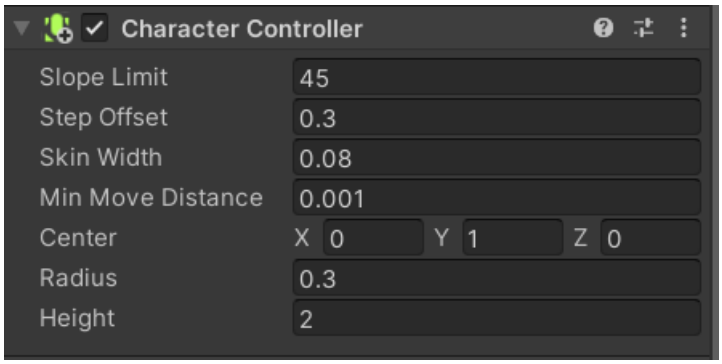
2.2.3 Character Controller

Tato komponenta slouží k ovládání pohybu postavy v kombinaci s integrovaným Capsule Colliderem. Používá se během implementací 3D her, kde využíváme pohled první nebo třetí osoby. Zjednodušeně řečeno nám tato komponenta poskytuje funkce a parametry, které můžeme využívat ve vlastním skriptu pro pohyb postavy.

Stejně jako ostatní komponenty, i zde máme řadu parametrů, které si lze nastavit dle potřeby naší herní postavy. [8]

Parametry Character Controlleru:

- Center, Height, Radius: tyto parametry slouží pro nastavení umístnění Capsule Collideru v závislosti na co nejpřesnějším ohraničení GameObjectu postavy.
- Slope Limit: určuje maximální sklon terénu, po kterém postava zvládne stoupat. Výchozí nastavení je 45 stupňů.
- Step offset: Parametr, který se využívá pro překonávání výšky překážek, přičemž takovým příkladem překážky mohou být schody. Pokud je výška schodu/překážky větší, než je námi nastavený limit, postava se před ni zastaví.



Obrázek 12: Komponenta Character Controller

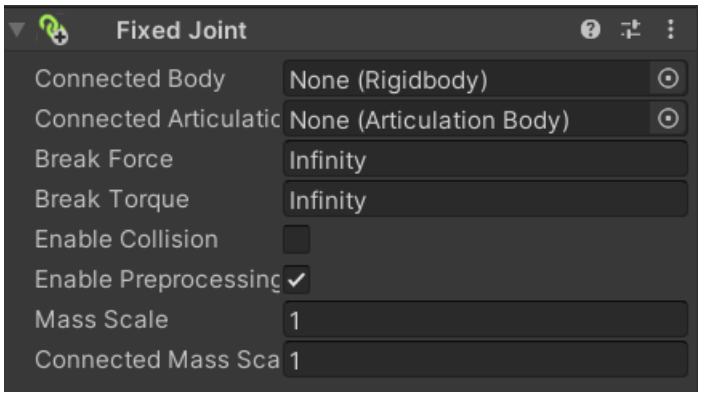
Nicméně je vždy potřeba zvážit jakou kombinaci komponent pro pohyb postavy využijeme, a to v závislosti na požadovaném chování a možnostech jejich využití. [8]

Je potřeba si dávat pozor na vzájemnou kompatibilitu komponent, přičemž u Character Controlleru si musíme dát speciálně pozor na komponentu Rigidbody. Ta se ne vždy slučuje s Character Controllerem, a může nám tak u postav způsobit velmi neočekáváné chování.

2.2.4 Joint

Stejně jako v případě skupiny colliderů, i zde existuje několik variací komponent „Joint“, spojující (většinou) více herních objektů, které mají komponentu Rigidbody. Následující varianty můžeme nalézt jak ve variantě 2D tak i 3D. [9] [10]

- Hinge Joint: Používá se například pro herní objekty dveří, kde (jak název napovídá) simulují funkci pantů. Je potřeba nastavit parametry pro otáčení kolem požadované osy (Axis) nebo třeba správně nastavit pozici kotvy (Anchor), která bude výchozím bodem pro otáčení kolem zvolené osy.
- Fixed Joint: Slouží pro fixní spojení dvou herních objektů. Toto spojení může a nemusí být trvalé. Pokud například nastavíme parametr Break Force, je možné silou jiného GameObjectu spojení přerušit.
- Spring Joint: Umožňuje nám spojení dvou herních objektů takovým způsobem, jako by byly propojené pružinou.
- Character Joint: Slouží pro simulaci kloubů herních postav, což umožňuje jejich realistický pohyb.
- Configurable Joint: Kombinuje parametry všech variant zmíněných spojů, které si uživatel může nastavit dle libosti.



Obrázek 13: Fixed Joint

Dalšími variantami pro 2D vývoj her jsou: Distance Joint, Friction Joint, Relative Joint, Slider Joint, Target Joint a Wheel Joint.

2.2.5 Cloth

Tato komponenta ve spojení s komponentou Skinned Mesh Renderer slouží pro simulaci materiálů tkanin ve hře. Má velkou řadu parametrů, díky kterým můžeme simulovat chování

různých variant GameObjectů, které tuto komponentu využívají. Příkladem může být oblečení naší postavy, ale třeba i předměty jako jsou vlajky, plachty apod. [11]

2.3 Audio komponenty

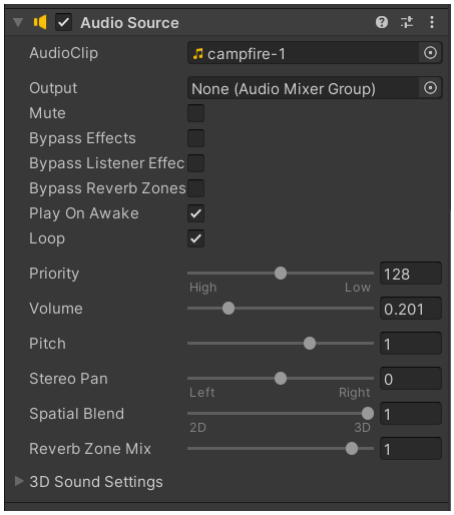
Tato skupina komponent slouží v Unity k přidání hudby do pozadí a zvuků (či zvukových efektů obecně) jednotlivým GameObjectům. Dobře zvolená hudba a zvukové efekty jsou jedním z mnoha aspektů, které zajišťují lepší herní zážitek. K tomuto účelu je zde řada komponent, přičemž nejdůležitějšími jsou Audio Source a Audio Listener. [12]

2.3.1 Audio Source

Díky této komponentě můžeme vytvářet zvuky, které vydávají herní objekty. Pokud se například postava přiblíží ke GameObjectu ohně, který má komponentu Audio Source s příslušnou audio nahrávkou, hráč uslyší zvuk, který oheň vydává. V naší scéně můžeme mít neomezené množství herních objektů s touto komponentou. [6]

Důležitými parametry této komponenty jsou:

- Play on Awake: Určuje, zda má audio automaticky začít hrát po spuštění hry.
- Loop: Zajistí hraní zvuku ve smyčce.
- Volume: Pro nastavení hlasitosti zvuku.
- Spatial Blend: Pro nastavení, zda se jedná o 2D nebo 3D audio. Pokud nastavíme tento parametr na hodnotu 0, dostaneme 2D zvuk, který uslyšíme pro různé vzdálenosti od GameObjectu pořád stejně. Oproti tomu hodnota 1 nám dodá 3D zvuk, který bude závislý na vzdálenosti GameObjectu se zvukem od posluchače.



Obrázek 14: Komponenta Audio Source

2.3.2 Audio Listener

Narozdíl od komponenty Audio Source můžeme mít v herní scéně pouze jeden Audio Listener, což znamená, že můžeme slyšet zvuk pouze prostřednictvím jednoho objektu. Objektem, na který se nejčastěji komponenta Audio Listener navazuje je buď kamera, nebo přímo naše herní postava. Díky tomu lze vytvářet reálné zvukové efekty, v závislosti na pohybu postavy či kamery, která postavu sleduje. Vzhledem k tomu, že tato komponenta funguje pouze jako přijímač zvuku, neobsahuje žádné nastavitelné parametry. [6]

Obrázek 15: Komponenta Audio Listener

2.3.3 Audio Filter

Tyto komponenty nám umožnují aplikovat zvukové efekty na zvuky v naší scéně, například pro změnu akustiky zvuku. Pokud komponentu Audio Filter přidáme na GameObject s komponentou Audio Source, bude ovlivněn pouze zvuk, který komponenta Audio Source obsahuje. Pokud ovšem komponentu Audio Filter přidáme na GameObject s komponentou Audio Listener, bude filtr působit na vše, co Audio Listener zachytí. Tuto komponentu máme opět k dispozici ve více variantách, jsou to: [44]

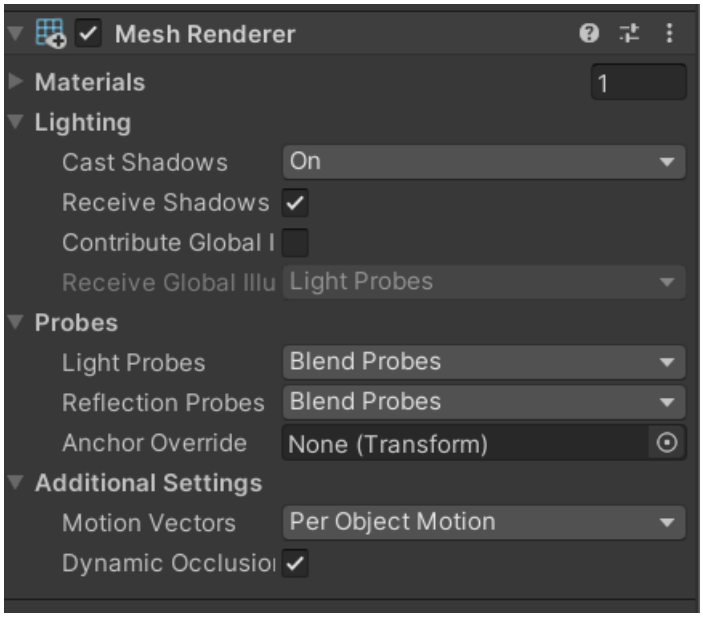
- Audio Distortion Filter: Používá se pro zkreslení zvuku – například pro vytvoření „sci-fi“ zvukových efektů.
- Audio Echo Filter: Slouží pro vytvoření efektu ozvěny.
- Audio High Pass Filter: Propouští jen vysoké frekvence.
- Audio Low Pass Filter: Propouští jen nízké frekvence – vhodné například pro audio, které má simulovat zvuk pod vodou.
- Audio Chorus Filter: Používá se pro vytvoření efektu vícenásobných zdrojů zvuků – například u efektu výbuchu, kdy se nastaví určitá prodleva a opakování, což zajistí realistický zvukový efekt.
- Audio Reverb Filter: Pro simulaci zvuku uzavřených prostor – například kostel, jeskyně.

2.4 Mesh komponenty

Do této skupiny patří komponenty, které souvisí s přímým vykreslováním GameObjectů ve scéně a jedná se tak o podstatnou kategorii, kterou nám obsluhuje Rendering Egine, který je již součástí základního Unity Editoru. [23]

2.4.1 Mesh Renderer a Mesh Filter

Tyto dvě komponenty nám spolu umožňují vykreslovat 3D GameObjecty ve scéně. Jsou na sobě závislé tím, že Mesh Filter vyfiltruje síť polygonů z 3D modelu GameObjectu a následně tuto referenci předá komponentě Mesh Renderer. Ten na základě převzaté geometrie a dalších faktorů jako jsou materiály a světlo vyrenderuje vizuál herního objektu do scény. [22]



Obrázek 16: Mesh Renderer

2.4.2 Skinned Mesh Renderer

Na stejném principu (viz předchozí podkapitola) vykreslování s Mesh Filtrem je založena i tato komponenta. Jediným rozdílem mezi klasickou komponentou Mesh Renderer a Skinned Mesh Renderer je jejich použití, přičemž Skinned Mesh Renderer, se používá pro vykreslení animovaných GameObjectů. Klíčovým prvkem jsou zde vytvořené kosti a spoje v modelu postav či zvířat, které určují jeho pohyby. Je tak potřeba zajistit správné vykreslování postavy v závislosti na pohybu modelu ve scéně, což nám tato komponenta zajišťuje. [45]

2.4.3 Text Mesh Pro

Tato komponenta umožňuje tvorbu 2D a 3D textů přímo ve scéně naší hry, což je užitečným prvkem například při tvorbě dialogů herních postav, zobrazování herního skóre, životů apod. Tato komponenta nabízí pokročilé možnosti tvorby textů a jejich nastavení, přičemž nahradila komponentu Text Mesh, která se používala v minulosti. [46]

2.5 Effects

V této skupině nalezneme různé variace komponent, které nám umožňují přidat vizuální efekty do naší scény. Obecně platí, že je potřeba tyto efekty do scény přidávat s rozvahou, jelikož je jejich vykreslování poměrně náročné a můžou tak zásadním způsobem ovlivnit výkon hry. [47]

- Trail Renderer: Tato komponenta nám umožňuje přidat „cestu“ za pohybujícím se GameObjectem, příkladem může být vizuální efekt odpáleného míčku.
- Line Renderer: Tato komponenta má poměrně široké využití, přičemž její princip je sice jednoduchý, ale velmi užitečný. Využívá se tam, kde je potřeba ve scéně vykreslit jakékoliv úsečky. Například se tedy využívá k vykreslování laserových paprsků, vizualizaci dostřelu, nebo ve hrách, kde musí uživatel kreslit do scény– např. trasu.
- Halo: Tento efekt dodává světelné záření kolem GameObjectů. Použít ji tak můžeme na ozáření portálů, nebo přímo na postavy, které chceme odlišit – například pokud z nich chceme udělat cíl.
- Lens Flare: Jak název vypovídá, tato komponenta nám vytváří efekt odrazu světla v kameře, a použít ji můžeme například pro zdůraznění ostrosti světla.
- Projector: Tato komponenta slouží k projekci vizuálních efektů, jako jsou například stíny či přímá světla.

Nejpodstatnější komponentou v této kategorii je však Particle System, která bude popsána detailněji.

2.5.1 Particle System

Díky Particle System jsme schopni v herní scéně nasimulovat pohyb částic, zde představují malé 2D obrázky, které se postupně generují a renderují do herní scény. To nám umožňuje vytvořit ve hře tekutiny, kouř, ohňostroj, exploze nebo třeba oheň. Pro velmi široké využití je zde i velmi obsáhlé nastavení. Oproti jiným komponentám se nastavení komponenty

Particle System skládá z několika modulů, kde si nastavujeme požadované chování vykreslovaných částic. [48]

Nejdůležitějšími parametry v této komponentě jsou:

- Duration: Tento parametr určuje délku života všech částic.
- Looping: Pro nastavení opakování životního cyklu částic.
- Start Delay: Určuje prodlevu, než se částice začne ve scéně vykreslovat.
- Start Lifetime: Určuje délku života částice, tentokrát ale po jejím vykreslení.
- Start Speed: Nastavuje počáteční rychlost částic.
- Start Size: Pro nastavení výchozí rychlosti částic.
- Gravity Modifier: Umožňuje nám změnit gravitaci působící na částice.
- Max Particles: Určuje maximální počet částic, které se mohou naráz vykreslit.

Obrázek 17: Particle System – hlavní modul

2.6 Event

Skupina Event je určena pro obsluhování interakcí a událostí ve hře, přičemž každá komponenta z této skupiny má své konkrétní využití. Do této skupiny patří: [27]

- Event System: Tato komponenta představuje řídící prvek událostí (Events) ve hře a úzce souvisí se všemi ostatními komponentami v této kategorii. Při příjmu vstupní událost vyhodnocuje a předává ji dále na zpracování ostatním komponentám.
- Event Trigger: Získává informace o událostech z Event System, a obstarává volání příslušných funkcí dané události. Slouží tak jako spouštěč, jak název vypovídá.
- Raycaster – Graphic: Tato komponenta slouží k detekci kolizí pouze s grafickými UI objekty – tedy například detekuje stisknutí tlačítek.
- Raycaster – Physics, Physics 2D: I tato komponenta slouží k detekcí kolizí, tentokrát však slouží k detekci interakcí/kolizí s GameObjecty.
- Touch Input Module: K detekci událostí z dotykových obrazovek – například ovládání mobilní hry.
- Standalone Input Module: Slouží k detekci událostí, které uživatel předá skrz klávesnici nebo myš.

2.7 Layout

Tato skupina komponent velmi úzce souvisí se skupinou UI, přičemž Layout skupina zajišťuje organizaci a rozložení UI prvků ve hře.

V této skupině nalezneme následující: [28] [29] [30] [31] [54]

- Aspect Ratio Fitter: Díky této komponentě lze tvořit UI prvky s funkčností simulující responzivní zobrazení.
- Canvas: Tato komponenta reprezentuje abstraktní prostor, pro všechny vykreslované UI prvky. Každý UI GameObject má tuto komponentu automaticky přiřazenou. Jedná se o základní prvek při tvorbě UI v Unity.
- Canvas Group: Tato komponenta umožňuje tvořit skupiny UI prvků v rámci UI kontejneru (abstraktního prostoru), díky čemuž lze hromadně upravovat prvky v dané skupině.
- Content Size Fitter: Zajišťuje, že se velikost UI prvku bude přizpůsobovat jeho obsahu.

- Layout Group: Máme k dispozici 3 varianty použití – Vertical, Horizontal a Grid, přičemž tyto komponenty slouží k uspořádání skupiny prvků vertikálně, horizontálně nebo dle mřížky, kterou si uživatel může nastavit.
- Layout Element: Slouží pro individuální nastavení UI prvku, což je užitečné, pokud chceme prvek odlišit například v rámci Canvas Group.
- Rect Transform: Tato komponenta umožňuje individuální nastavení UI prvku co se pozice nebo rotace týče.

2.8 Miscellaneous

V této skupině najdeme (dle vypovídajícího názvu) různé komponenty, které nemají specifické zařazení. [54] [55] [56] [57] [58]

- Grid: Jedná se o komponentu, která úzce souvisí s komponentou Tilemaps. Obdobně jako komponenta Canvas nám i Grid slouží jako kontejner, zde například pro dlaždice při tvorbě prostředí 2D her.
- Animator a Animation: Komponenta Animator slouží pro řízení animací daného GameObjectu. Je potřeba do komponenty přiřadit Animator Controller, který definuje animace a jejich tranzice mezi sebou. Animation má totožné vlastnosti, nicméně se jedná o již nevyužívanou komponentu, která je součástí Unity pouze kvůli zpětné kompatibilitě.
- Billboard Renderer: Tato komponenta nám umožňuje přidávat do 3D hry 2D prvky, jako jsou například obyčejné obrázky, které budou v herní scéně na pohled orientované vždy směrem na kameru. Tímto způsobem tak lze tvořit iluze 3D prvků pomocí 2D prvků.
- Sprite Mask a Sprite Shape Renderer: Komponenta Sprite Mask slouží pro zobrazování či naopak skrývání jen určité částí 2D objektů (Sprite) ve hře, přičemž o korektní vykreslování objektu dle nastavené masky se zde stará Sprite Shape Renderer.
- Position, Rotation, Scale Constraints: Komponenty, které určitým způsobem omezují GameObject. Například omezujeme jeho pohyb v určitém směru, rotaci po ose nebo škálovatelnost.
- Aim Constraint: Slouží ke směrování k určitému cíli ve scéně. Využívá se u GameObjectů zbraní.
- Look At Constraint: Slouží k zaměření pozornosti na určitý objekt ve scéně, používá se například pro animaci očí postavy.

- Parent Constraint: Umožňuje nám tvořit objekty, které budou kopírovat pozici, rotaci nebo měřítko rodičovského objektu.

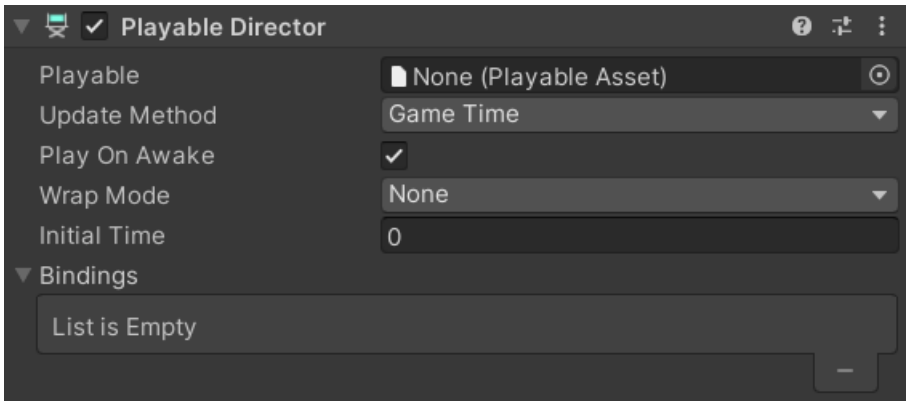
2.9 Navigation

Tato skupina komponent tvoří základ herní navigace v prostředí Unity. Pro použití navigace se nejdříve musí vytvořit navigační síť (Nav Mesh), která kopíruje vytvořený terén, dle nastavených parametrů jako jsou úhel, výška schodů apod. Navigace se v herní scéně využívá například k pohybu postavy na určené místo po kliknutí myši nebo například pro navádění nepřátel k vytvořené postavě. [53]

- Nav Mesh Agent: Komponenta, která umožňuje pohyb GameObjectu po navigační síti, přičemž hledá optimální trasu a detekuje kolize, kterým se vyhýbá.
- Nav Mesh Obstacle: Využívá se pro tvorbu překážek, které ovlivňují chování GameObjectu s komponentou Nav Mesh Agent.
- Off Mesh Link: Komponenta, která určuje, jakým způsobem bude GameObject s komponentou Nav Mesh Agent přecházet mezi dvěma různými prostředí v případě, že je přechod mimo navigační síť. Definuje samotné přechody, kterými může být například animace přechodu po žebříku, laně nebo pouhého skoku.

2.10 Playables – Playable Director

V této kategorii najdeme komponentu Playable Director. Tato komponenta spolupracuje s objektem Timeline, který přebírá a následně ho řídí. Timeline je objekt, který nám umožňuje vytvořit různé sekvence videí, zvuků, animací nebo efektů na základě načasovaných událostí či podmínek ve hře. Pokud například hráč dojde úspěšně do cíle za splněných podmínek, přehraje se vítězný zvuk. [52]



Obrázek 18: Komponenta Playable Director

2.11 Rendering

Tato skupina obsahuje všechny komponenty, které zajišťují vykreslování herních a UI prvků, světlo a speciální efekty. Nejdůležitějšími komponentami zde jsou Camera a Light. [34] [35] [60] [61] [62]

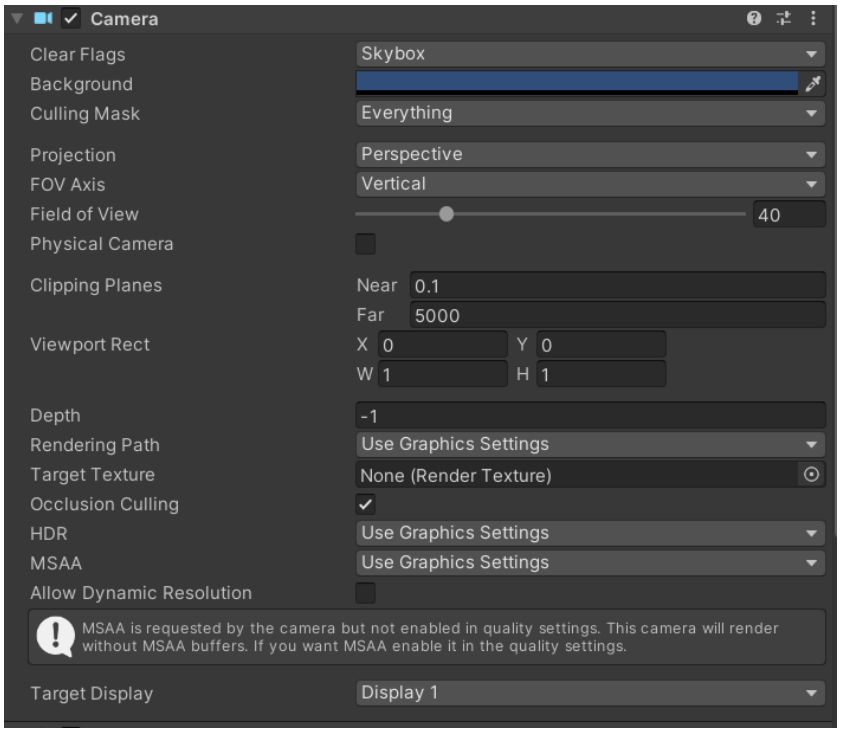
- Canvas renderer: Tato komponenta je provázána s UI prvky, jako jsou například tlačítka, které vykresluje a aktualizuje.
- Flare Layer: Tato komponenta je potřebná pro GameObject kamery, pro vykreslení světelného efektu z komponenty Lens Flare.
- Light Probe Group: Umožňuje nám přidat do scény objekty Light Probe, což jsou virtuální body, které analyzují osvětlení v určitém bodu scény. Light Probe Group spravuje umístění těchto virtuálních bodů ve scéně.
- Light Probe Proxy Volume: Tato komponenta už přímo souvisí s nastavením jednotlivých bodů Light Probe, přičemž zde lze ovládat osvětlení, například u vnitřního prostoru herní scény.
- LOD Group: V této komponentě můžeme nastavit úroveň vykreslování detailu (Level Of Detail) GameObjectů ve scéně na základě přiblížení kamery ke GameObjectům. Pokud jsme například od objektu ve hře daleko, vykreslí se objekt v menší kvalitě a s přibližováním kvalita stoupá.
- Occlusion Area: Slouží nám k označení oblasti ve scéně, na kterou bude namířená kamera, a pouze tato specifická oblast tak bude vykreslená. Ve své podstatě to slouží k optimalizaci a menší zátěži na GPU.
- Occlusion Portal: Stejně jako předchozí komponenta, i tato slouží k vydefinování oblasti, která bude vykreslena. Používá se v souvislosti s průchody z místnosti do místnosti, přičemž vykreslí místnost, ve které se aktuálně nachází kamera, a zároveň přestane vykreslovat scénu z přechozí místnosti.
- Reflection Probe: Tato komponenta se využívá v souvislosti s vykreslováním odrazů ve scéně.
- Skybox: Díky této komponentě máme možnost nastavit si pozadí ve scéně, které hráč vidí jako nebe.
- Sorting Group: Slouží pro nastavení vykreslování a vzájemné překrývání GameObjectů ve scéně. Například pokud ve 2D hře potřebujeme vykreslit postavu před pozadím.

2.11.1 Camera

Jedná se o velmi důležitou komponentu, která nám umožňuje zobrazovat herní scénu hráči. V případě potřeby je možné mít ve scéně více kamer, které můžou mít jiná nastavení, například pokud potřebujeme ve hře změnit perspektivu.

Je tak potřeba věnovat velkou pozornost parametrům této komponenty, jako jsou: [41]

- Field Of View: Zde nastavujeme zorný úhel kamery, čímž nastavujeme, co vše uvidíme v aktuálním záběru ve hře.
- Backround: Pro nastavení pozadí kamery ve scéně. V podstatě se jedná o pozadí, které uživatel vnímá jako nebe.
- Projection: Pro nastavení perspektivy.
- Clipping Planes: určuje vzdálenost viditelnosti GameObjectů ve scéně.



Obrázek 19: Komponenta Camera

V případě potřeby zde existuje balíček Cinemachine, který nám dodává rozsáhlejší nastavení pro kamery.

2.11.2 Light

Tato komponenta umožňuje nastavení světel v herní scéně, přičemž se jedná o důležitý prvek k dosažení lepšího herního zážitku. Tato komponenta v sobě obsahuje nastavení několika podstatných parametrů, jako jsou: [43]

- Type: Pro nastavení typu světla. Můžeme zvolit mezi variantami Spot, Directional, Point a Area v závislosti na jejich použití.
- Color: Nastavuje barvu světla.
- Shadow Type: Určuje, jakým způsobem budou ve scéně vykreslovány stíny.
- Intensity: Pro nastavení intenzity světla.

2.12 Tilemap

Tato skupina komponent slouží ve vývoji 2D her k tvorbě herního prostředí pomocí systému dlaždic. Pokud vytvoříme předdefinovaný GameObject Tilemap, automaticky se nám u GameObjectu vytvoří i komponenty Tilemap, Grid, Tilemap Collider a Tilemap Renderer. Touto vizuální formou je tvořena například hra Mario 2D.

Komponentami v táto skupině jsou: [33]

- Tilemap: Jedná se o komponentu, která představuje vrstvu herního prostředí, vytvořenou z jednotlivých dlaždic, které jsou tvořeny z malých 2D obrázků (Sprite). Vizuálně jsou dlaždice srovnány na základě definované mřížky, kterou zde představuje komponenta Grid.
- Tilemap Collider: Stejně jako jiné collidery, i tato komponenta detekuje kolize.
- Tilemap Renderer: Stará se o vykreslování vizuálu dlaždic v herní scéně.

2.13 UI

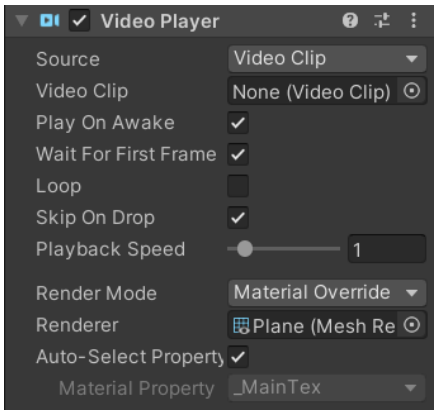
Tato kategorie je zaměřena na tvorbu UI v prostředí hry. Díky těmto prvkům můžeme uživateli zobrazovat text nebo vkládat tak ovládací prvky jako jsou například tlačítka. Lze vytvořit například dialogy nebo menu hry. [36] [37] [49]

- Button: Komponenta, která slouží pro tvorbu interaktivních tlačítek. Lze nastavit vlastnosti jako jsou například text a definování události po kliknutí.
- Dropdown – TextMeshPro: Umožňuje nám tvořit rozbalovací seznamy s nabídkou, ze které uživatel vybírá.

- Effects: Podskupina komponent, které slouží pro vytvoření vizuálních efektů UI prvků, jako je například stín.
- Image: Umožňuje nám tvořit vizuál UI prvků, jako je například vzhled tlačítka.
- Mask a Rect Mask 2D: Tyto komponenty nám umožňují definovat oblast, kde se bude nacházet například Scroll Rect, přičemž vše mimo oblast nebude viditelné.
- Raw Image: Pro vložení obrázků nebo animací do prostředí hry. Například pro vizualizaci zbývajících životů v podobě obrázku srdce.
- Scroll Rect: Vytváří UI prvek v herní scéně, který můžeme posouvat myší nebo prstem. Pro představu lze takto vytvořit náhled mapy ve hře, kterou lze v tomto UI prvku posouvat.
- Scrollbar: Slouží k vytvoření klasického posuvníku (scrollbar), například v případě vytvoření dlouhých textových polí ve Scroll Rectu.
- Selectable: Tato komponenta detekuje interakci uživatele s UI prvky.
- Slider: Umožňuje uživateli výběr hodnot z definovaného rozsahu.
- TextMeshPro – Input Field: Umožňuje zadání textového vstupu od uživatele.
- TextMeshPro – Text (UI): Má stejné použití jako komponenta TextMeshPro, s tím rozdílem, že tato komponenta nabízí rozsáhlejší možnosti.
- Toggle: UI prvek představuje přepínač mezi dvěma stavy.
- Toggle Group: Pro seskupování Toggle přepínačů.

2.14 Video – Video Player

V této skupině nalezneme pouze jednu komponentu, kterou je Video Player. Jak název napovídá, jedná se o komponentu, která nám umožňuje ve hře přehrávat videa. Díky této komponentě máme možnost v rámci hry přehrávat například krátké videotutoriály. [51]



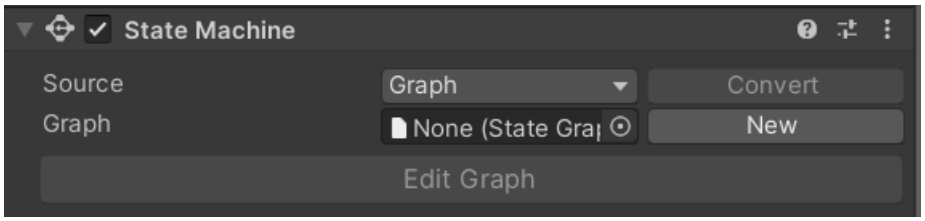
Obrázek 20: Komponenta Video Player

2.15 Visual Scripting

Visual Scripting nám v Unity poskytuje možnost tvorby logiky hry, aniž by uživatel musel psát kód. K tomuto účelu slouží tvorba grafů (Script Graph a State Graph), do kterých uživatel může přidat bloky, reprezentující například stavy nebo funkce a vzájemně je propojovat. Tím definuje například smyčky, podmínky a další operace potřebné k implementaci logiky hry. [32] [50]

K tomuto účelu zde nalezneme 3 komponenty:

- Script Machine: Tato komponenta obstarává provedení akcí jako jsou podmínky, nebo smyčky, které jsme vizuálně definovali prostřednictvím grafu Script Graph.
- State Machine: Tato komponenta obstarává provedení akcí, které jsme vizuálně definovali v grafu State Graph. Jednoduchým příkladem zde může být přechod postavy z jednoho stavu do druhého po stisknutí určité klávesy – například po stisknutí klávesy W nastane změna stavu z nečinnosti do běhu.



Obrázek 21: Komponenta State Machine

- Variables: Tato komponenta zobrazuje a udržuje veškeré proměnné, které vytvoříme v jednom ze zmíněných grafů. Pokud přidáme GameObjectu komponentu Script Machine či State Machine, přidá se komponenta Variables automaticky.

II. PRAKTICKÁ ČÁST

3 NÁVRH A VYTVOŘENÍ PROTOTYPU

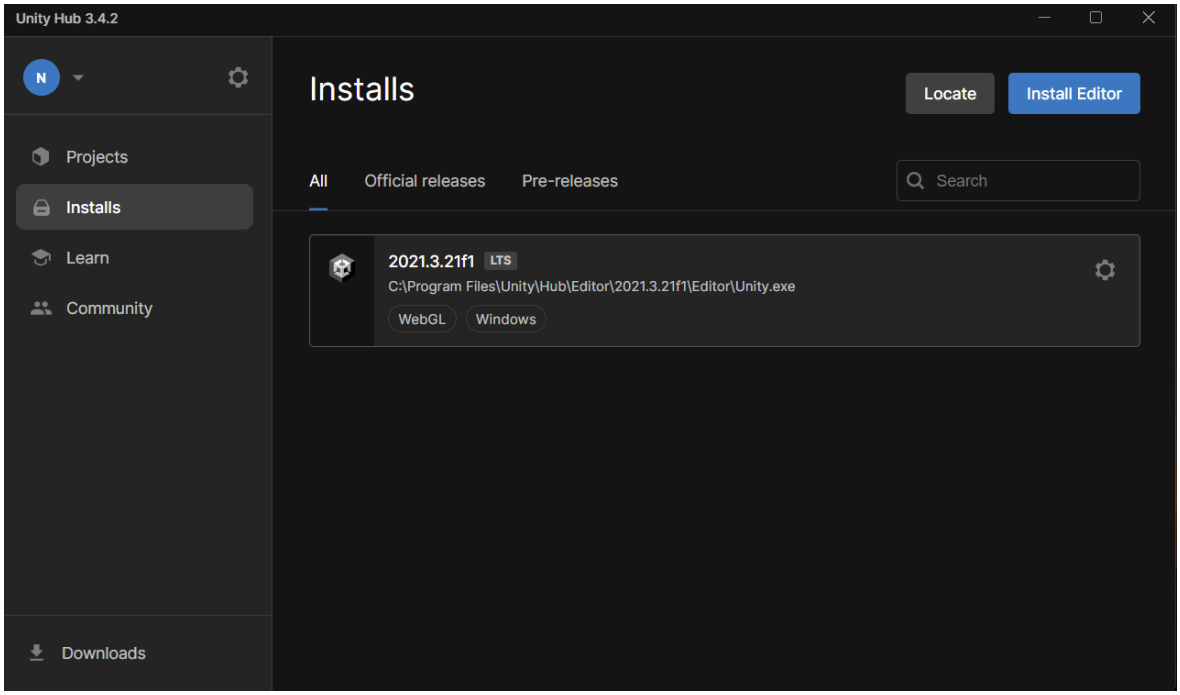
Cílem bude vytvoření základní scény, s vytvořenou postavou, se kterou se bude moci uživatel pohybovat. Prototyp tak získá povahu klasické 3D plošinové hry.

Vzhledem k tomu, jak složité a časově náročné by bylo vytváření vlastní postavy a jiných prvků, které budou v demo scéně použity, budou využity volně dostupné Assety z oficiálního Asset storu.[1] K vytvoření postavy a jejich animací budou využity zdroje ze stránky Mixamo od Adobe.

3.1 Instalace a příprava prostředí

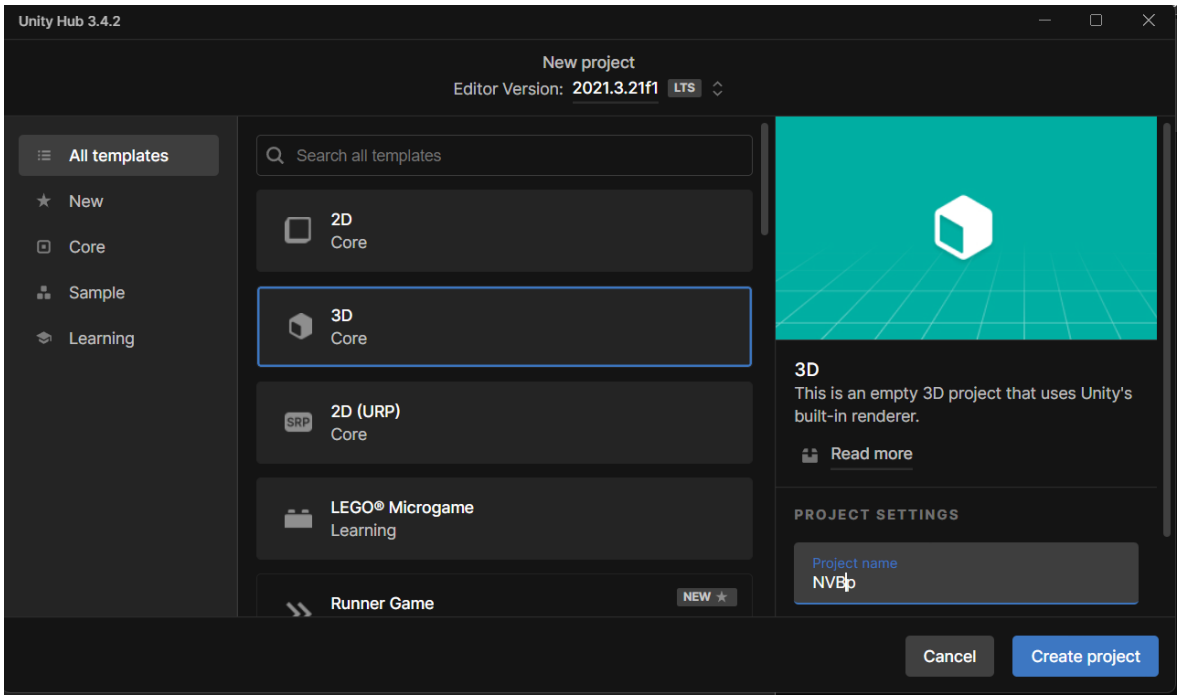
Pro pohodlné spravování projektu a verze samotného enginu Unity si jako první stáhneme nástroj Unity Hub z oficiálních stránek.[2] Po stažení spustíme instalační soubor – instalací nás provedou pokyny na obrazovce a celý proces instalace je velmi intuitivní. Po dokončení instalace spustíme Unity Hub a vytvoříme si účet Unity ID, bez kterého nelze dále pokračovat. Po vytvoření účtu se Unity Hub automaticky spáruje s naším účtem, čímž se přihlásíme do nainstalované desktopové aplikace. Následně se nám zobrazí dialog, který nám umožňuje instalaci nejaktuálnější verze Unity Editoru, přičemž při vytváření této bakalářské práce je nejaktuálnější verze Unity 2021.3.21f1. Po instalaci editoru je pak možné vidět všechny nainstalované verze v záložce „Installs“.

[1] Unity Asset Store - The Best Assets for Game Making
[2] https://unity.com/download



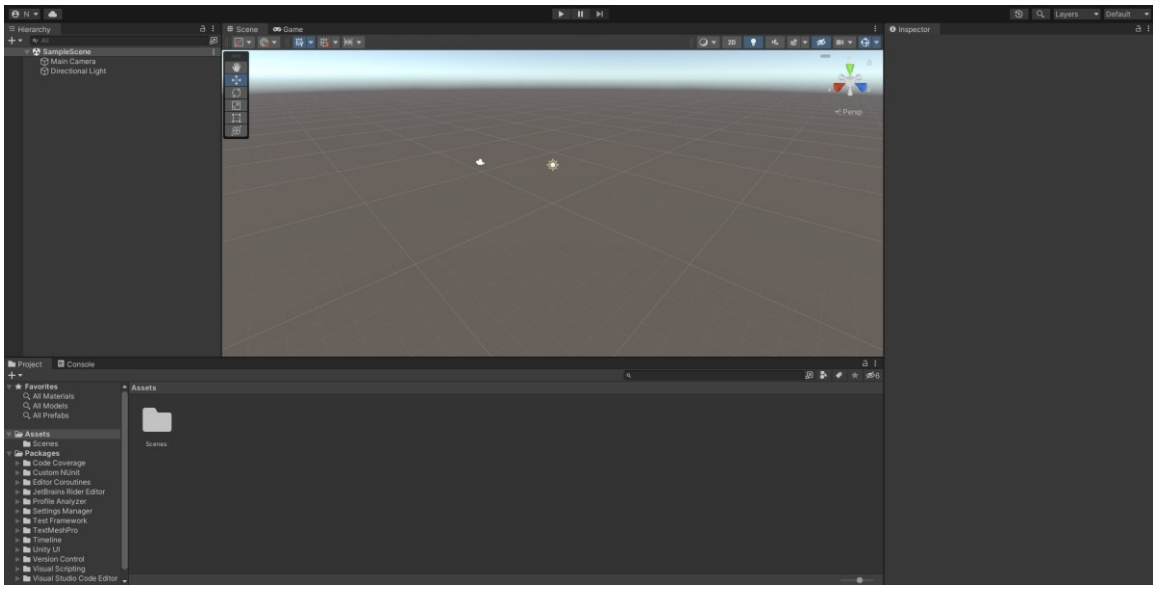
Obrázek 22: Nainstalovaná verze Unity Editoru

Nyní máme nainstalovaný potřebný editor a můžeme si vytvořit projekt, což uděláme na záložce Project, přes tlačítko „New project“. Zobrazí se nabídka, ve které si vybereme „3D Core“ šablonu, zvolíme nainstalovaný editor (defaultně je zde vybrán ten nejaktuálnější) a nastavíme název projektu a jeho umístění.



Obrázek 23: Vytvoření nového projektu

Po nastavení vytvoříme projekt pomocí tlačítka „Create project“. Tím se začne spouštět Unity Editor, a začnou se importovat všechny potřebné prvky výchozího projektu. Po dokončení importu se nám otevře Unity Editor, který obsahuje pouze dva herní objekty – hlavní kameru a světlo.

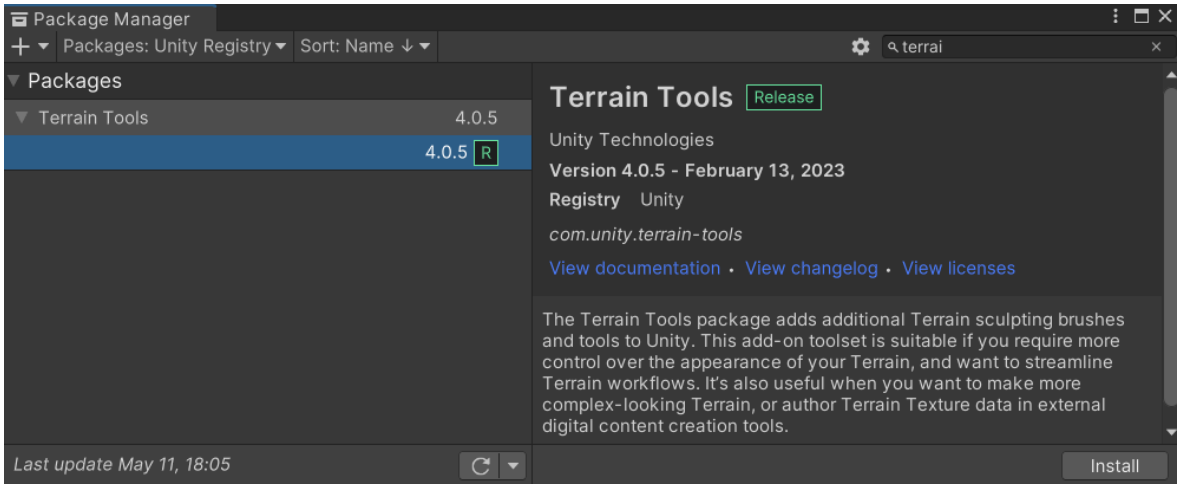


Obrázek 24: Nově vytvořený projekt

Nyní je potřeba nastavit Script Editor, který bude používaný k úpravě a vytváření skriptů v projektu. To lze přes záložku Edit> Preferences. Následně v záložce External tools, v sekci External Script Editor zvolíme preferovaný editor – pro tento projekt Microsoft Visual Studio 2022.

3.2 Terén

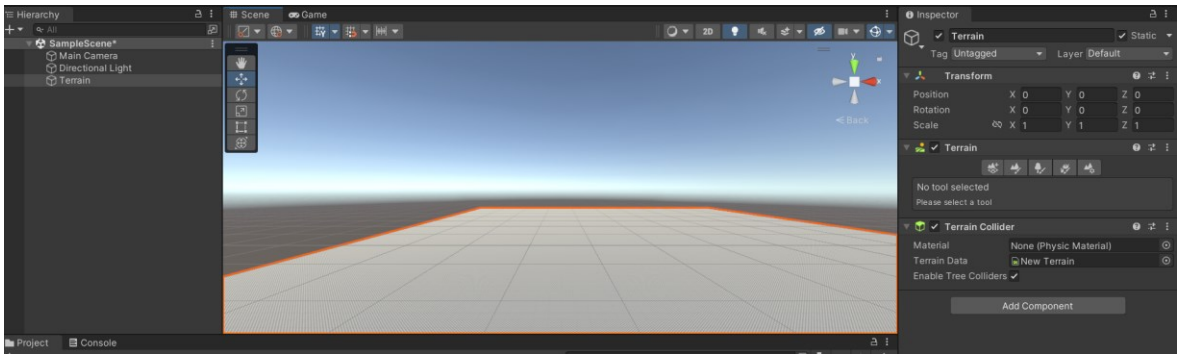
Pro vytvoření terénu je potřeba přidat si Terrain Tools, které získáme prostřednictvím Package Managera v sekci Unity Registry.



Obrázek 25: Terrain Tool package

Po instalaci balíčku Terrain Tool si rovnou stáhneme nabízený Terrain Sample Asset Pack z Unity Asset Store, který obsahuje modely, materiály, textury a štětce potřebné k vytvoření terénu. Po stažení tento asset naimportujeme do projektu a tím je vše připravené pro modelování terénu.

Otevřeme Terrain Toolbox přes záložku Window> Terrain> Terrain Toolbox a v okně „Create New Terrain“ klikneme na tlačítko Terrain, čímž se nám do scény vygeneruje nový GameObject Terrain, představující náš terén. V tuto chvíli je to pouze rovný povrch, který má komponenty Transform, Terrain a Terrain Collider.

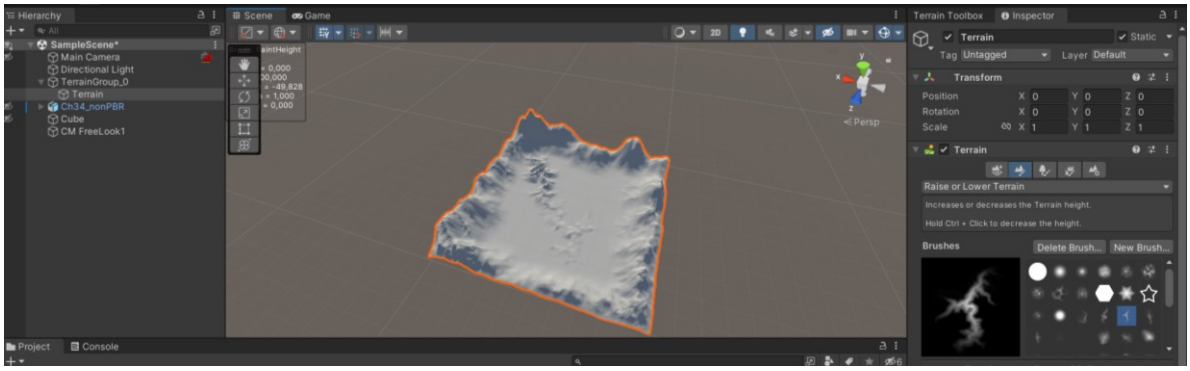


Obrázek 26: Vygenerovaný GameObject Terrain

3.2.1 Modelace terénu

Nyní přejdeme na modelaci povrchu terénu, k čemuž využijeme samotnou komponentu Terrain a její nástroje. Začneme funkcí Paint Terrain> Set Height, kde zvolíme tvary štětce, které jsme si naimportovali v assetu. Pomocí této funkce zvyšujeme povrch vygenerovaného terénu až do požadovaného tvaru hor po okolí a lehce i středu terénu.

Aby nebyl povrch terénu všude kromě hor nepřirozeně rovný, pomocí nástroje Paint Terrain> Sculpt> Noise a následně nástrojem Paint Terrain> Smooth Height vytvoříme lehce nejednotný, ale vyhlazený terén.

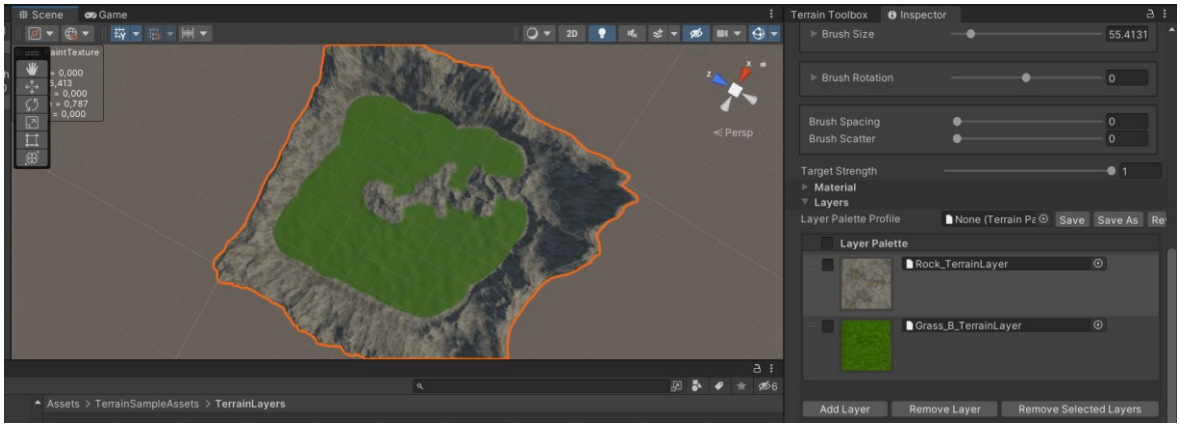


Obrázek 27: Vymodelovaný povrch terénu

3.2.2 Textura terénu

Ve chvíli, kdy je terén vymodelovaný, přidáme vzhled, který vytvoříme opět pomocí nástroje komponenty Terrain. Zvolíme nástroj Paint Terrain> Paint Texture a vytvoříme první vrstvu pomocí tlačítka „Add Layer“ v panelu „Layers“. Tato vrstva bude reprezentovat kamenný vzhled hor, proto do vrstvy přidáme texturu „Rock_TerrainLayer“, kterou jsme získali, stejně jako štětce, při importu assetu. Nyní se nám tato vrstva nastaví na celý terén. Vrstvě upravíme parametry Tilling Setting> Size v panelu Layer Properties. Hodnotu tohoto parametru nastavíme na 70x70, čímž upravíme opakování textury ve vzhledu terénu.

Aby scéna nebyla pouze kamenná, pro další vrstvu zvolíme texturu „Grass_B_TerrainLayer“ a pomocí štětců vykreslíme texturu trávy tam, kde je nižší povrch terénu. Stejně jako v předchozím případě je potřeba upravit Tilling Setting> Size, přičemž u této textury nastavíme hodnotu na 10x10, aby se samotná textura trávy zobrazovala korektně.



Obrázek 28: Terén s texturou vrstev

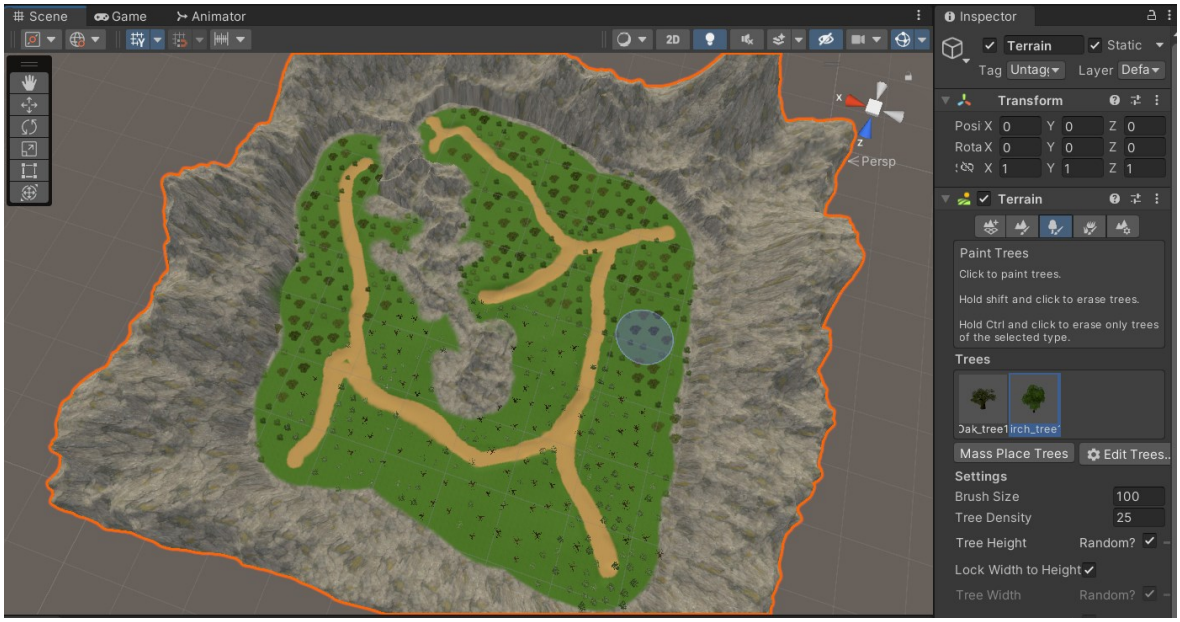
Pomocí stejného principu vykreslíme do terénu cestu. Přidáme tedy další vrstvu s texturou Sand_Terrain_Layer a pomocí štětce cestu vykreslíme. Následně zvolíme nástroj Paint Terrain> Set Height a tam, kde máme cestu, snížíme terén. Tím vytvoříme snížený povrch pro realisticky vypadající vzhled. Následně pomocí Paint Terrain> Smooth Height vyhladíme okraje, aby nebyl přechod mezi trávou a cestou na pohled moc ostrý.

Obrázek 29: Terén s vykreslenou cestou

3.2.3 Detaily terénu

Nyní máme vymodelovaný terén s příslušnou texturou a můžeme přidat detaily. K tomu bude potřeba stáhnout si další asset, konkrétně Fantasy Landscape. Z tohoto assetu do scény jako první přidáme stromy, opět pomocí komponenty Terrain, kde najdeme nástroj Paint Trees. Zde klikneme na tlačítko Edit Trees> Add Tree a zvolíme prefab stromu Oak_Tree1 z importovaného assetu.

Nyní máme nastavené stromy, které budeme do terénu přidávat. Je potřeba nastavit parametr „Tree density“ na menší hodnotu – např. 25, aby stromy nebyly moc blízko u sebe. Nyní do terénu rozmístíme stromy jednoduchým klikáním na požadovaná místa. Stejným způsobem postupujeme při přidání a rozmístění dalšího druhu stromu Birch_Tree1.



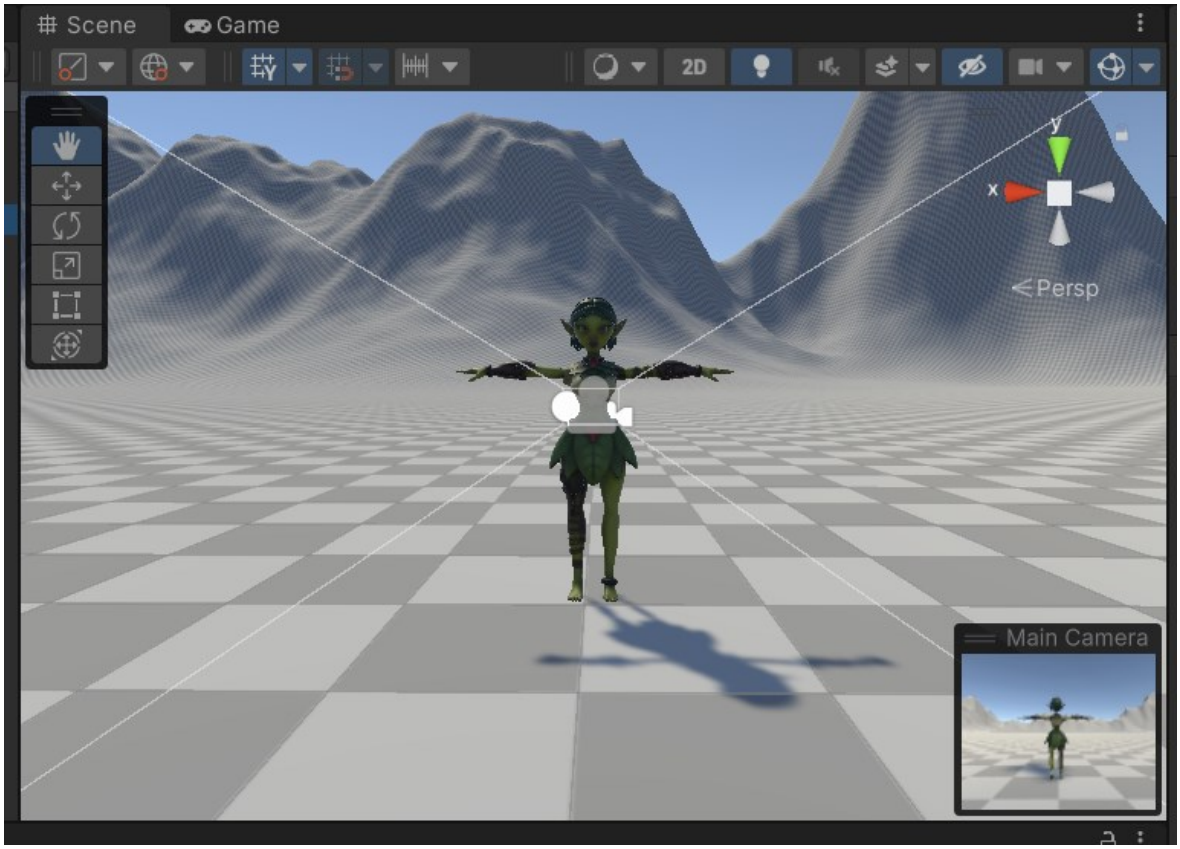
Obrázek 30: Terén se stromy

Nyní pro lepší efekt přidáme do terénu větší detail, pomocí nástroje Paint Details, kde klikneme na Edit details> Add Detail Mesh a vybereme prefab Grass1. Po přidání modelu trávy do štětce postupujeme stejným způsobem jako při přidávání stromů, dokud nejsou pokryté všechny zelené plochy ve scéně, představující trávu.

3.3 Postava

Pro vzhled postavy využijeme stránku Mixamo od Adobe, která nabízí jak vzhled postav do her, tak jejich animace. Do našeho projektu tak stáhneme postavu „Jolleen“, přičemž při stažení je potřeba zvolit formát FBX for Unity, Without Skin. [13] [64]

V Unity přidáme postavu do scény přetažením z Project (konkrétně ze složky, kam jsme ji uložili), přičemž aby se vzhled postavy vykresloval korektně, je potřeba v nastavení modelu extrahovat textury a materiály, které jsme si společně s postavou importovali do projektu. Tím se nám naše postava vykreslí přesně jak je zobrazována na stránce Mixamo. Abychom lépe viděli na naši postavu, je potřeba nastavit si pohled kamery. To uděláme pomocí komponenty Transform u GameObjectu Main Camera, kde změníme parametry Position tak, abychom při spuštění hry mohli testovat změny, které u postavy provedeme.

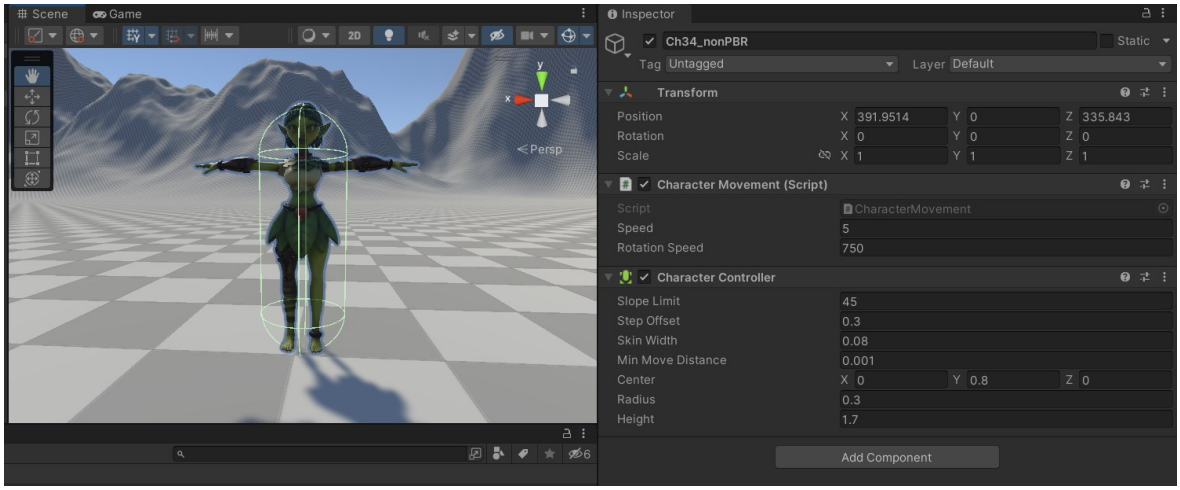


Obrázek 31: Importovaná postava s nastavenou kamerou

Nyní máme připravenou postavu, která ale aktuálně nic nedělá a je potřeba přidat ji pohyb.

3.3.1 Pohyb a skok postavy

Pro pohyb postavy využijeme komponentu Character Controller. Zvolíme tedy náš GameObject postavy a přidáme ji komponentu Character Controller pomocí tlačítka „Add Component“ v okně inspektoru. Upravíme parametry Center, Radius a Hight, aby Collider, který je součástí Controlleru, co nejlépe obklopoval postavu.



Obrázek 32: Nastavení parametrů

Samotná komponenta ovšem není dostačující, a je potřeba vytvořit vlastní skript pro pohyb a skok postavy. Pomocí Add Component> New Script vytvoříme script „Character Movement“. Po vytvoření se nám uloží script jako komponenta u dané postavy a je možné ho vidět v okně inspektoru.

Skript otevřeme v již dříve nastaveném Script Editoru (Microsoft Visual Studiu 2022). Po otevření máme vytvořenou metodu Start(), která se volá při spuštění hry a metodu Update(), která je volaná při vykreslování každého snímku. [14]

Abychom mohli ke komponentě přistupovat, upravíme tuto referenci v metodě Start() a následně začneme implementací pohybu postavy úpravou metody Update().

```
void Start()
  {
      characterController = GetComponent<CharacterController>();
  }
```

Pokud chceme pohybovat postavou na základě vstupu uživatele, je potřeba tyto vstupy získat a přiřadit je do příslušných proměnných. Tyto proměnné – horizontalInput a verticalInput získávají vstupy v horizontálním (doprava, doleva) a vertikálním (nahoru, dolů) směru. Tyto hodnoty následně předáme do Vector3 (3D vektor) movementDirection, který reprezentuje směr pohybu. Následně nastavíme vektor pohybu a jeho rychlost.

K pohybu postavy použijeme SimpleMove metodu z Character Controller komponenty. [19]

```
float horizontalInput = Input.GetAxis("Horizontal");
float verticalInput = Input.GetAxis("Vertical");

Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);
float magnitude = Mathf.Clamp01(movementDirection.magnitude) * speed;
movementDirection.Normalize();
characterController.SimpleMove(movementDirection * magnitude);
```

Následně nastavíme na postavě otáčení ve směru pohybu:

```
if (movementDirection != Vector3.zero)
        {
            Quaternion toRotation = Quaternion.LookRotation(movementDirection,
Vector3.up);

            transform.rotation = Quaternion.RotateTowards(transform.rotation,
toRotation, rotationSpeed * Time.deltaTime);
        }
```

Základní pohyby postavy jsou tímto ošetřené a je potřeba přidat funkcionalitu pro skok postavy.

Do proměnné ySpeed si uložíme rychlost skoku, která je ovlivněna gravitací – bude se tedy měnit v čase v závislosti na pozici postavy (při pádu dolů).

```
ySpeed += Physics.gravity.y * Time.deltaTime;
```

Následně upravíme funkci pro pohyb postavy ze SimpleMove na Move(), kterou opět vezmeme z Character Controlleru. SimpleMove() na rozdíl od metody Move() využívá i pohyb po ose y.

```
Vector3 velocity = movementDirection * speed;
velocity.y = ySpeed;

characterController.Move(velocity * Time.deltaTime);
```

Zbývá upravit podmínky skoku, kde využijeme proměnné pro uložení času posledního stisknutí tlačítka skoku (jumpButtonPressedTime), posledního dotyku země (lastGroundedTime), a časové okno, ve kterém je možné provést skok (jumButtonGracePeriod), aby skoky a pády šly simulovat kontinuálně za sebou. Zároveň využijeme proměnnou bool isGrounded z Character Controlleru, která nám určuje, zda je postava aktuálně na zemi a tím povolit či zakázat skok postavy. Tím zabráníme nekontrolovatelnosti skoků, které by mohly stoupat stále výš po každém stisknutí klávesy skoku. [18] [20]

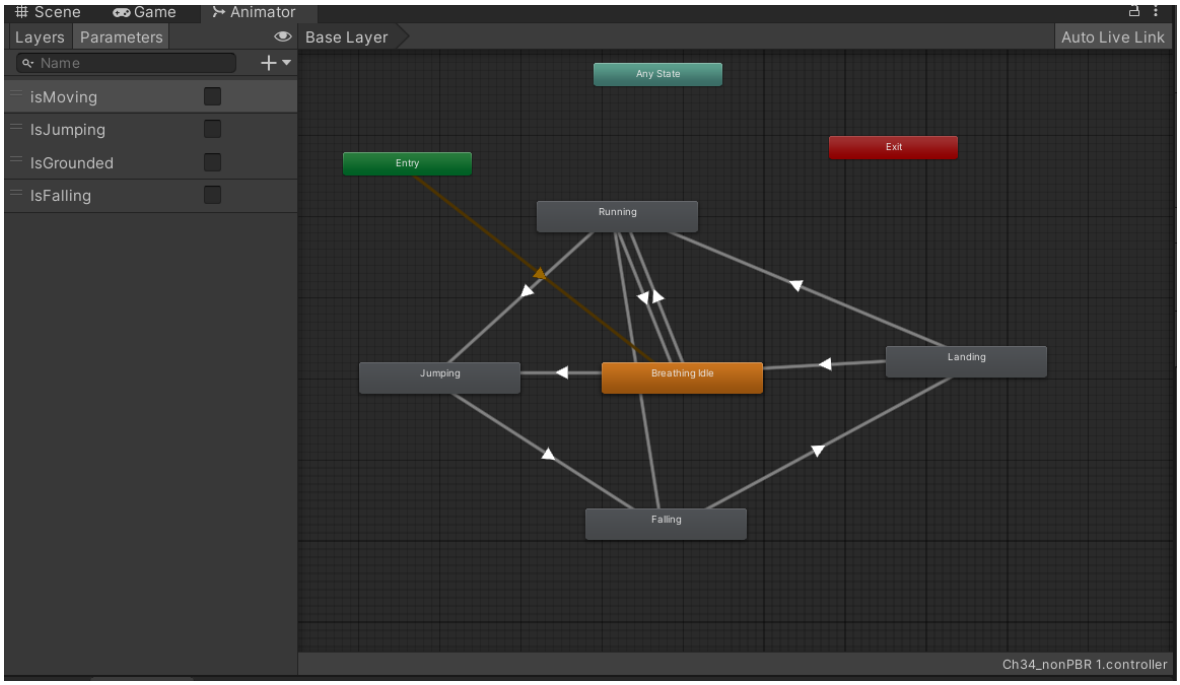
```
if (characterController.isGrounded)
        {
            lastGroundedTime = Time.time;
        }

        if (Input.GetButtonDown("Jump"))
        {
            jumpButtonPressedTime = Time.time;
        }

        if (Time.time - lastGroundedTime <= jumpButtonGracePeriod)
        {
            characterController.stepOffset = originalStepOffset;
            ySpeed = -0.5f;

            if (Time.time - jumpButtonPressedTime <= jumpButtonGracePeriod)
            {
                ySpeed = jumpSpeed;
                jumpButtonPressedTime = null;
                lastGroundedTime = null;
            }
        }
        else
        {
            characterController.stepOffset = 0;
        }
```

3.3.2 Animace

K vložení animací do projektu použijeme opět stránku Mixamo, kde jsme našli vloženou postavu. Stáhneme si animace, které budou reprezentovat stavy postavy v různé fázi pohybu.

Pro postavu, která se aktuálně nepohybuje a jen v klidu stojí využijeme animaci „ Breathing Idle“, kterou si stáhneme ve formátu FBX for Unity. Po importu animace do projektu je potřeba nastavit v animaci parametr „Loop Time“ a „Loop pose“, aby se animace nezastavila a změnu potvrdíme tlačítkem „Apply“. Následně animaci přetáhneme na GameObject naší postavy, čímž se k postavě automaticky přidá komponenta „Animator“. Zároveň se vytvoří i Animation Controller, který se do komponenty Animator automaticky přiřadí. Další animace bude pro pohyb. Zvolíme ze stránky Mixamo animaci „Running“ a postupujeme přesně jako v předchozím kroku. Následně je potřeba přidat poslední animace – skoku nahoru, průběh skoku a přistání zpět na zem. Pro tyto tři případy najdeme animace „Jumping up“, „Falling Idle“ a „Falling To Landing“. Všechny tři animace naimportujeme do našeho projektu, přičemž pouze u animace „Falling Idle“ zapneme parametr „Loop Time“ a „Loop Pose“. U Animacce „Falling To Landig“ nyní potřebujeme pouze část animace, kde postava přistane na zem, a proto nastavíme počátek animace na hodnotu 8. [65]

Nyní je potřeba nastavit přechody mezi jednotlivými animacemi, to provedeme v okně Animator, kam se dostaneme otevřením Animation Controlleru. Při přetahování stáhnutých animací na GameObject postavy se automaticky vytvořili stavy Running, Jumping, Breathing Idle, Falling a Landing, které odpovídají importovaným animacím. Tyto prvky reprezentují stavy, ve kterých se postava nachází. V sekci Parameters vytvoříme 4 parametry s datovým typem bool, které budou reprezentovat přechody mezi jednotlivými stavy. Nyní tyto přechody vytvoříme přímo mezi stavy. To uděláme kliknutím pravým tlačítkem myši na příslušný stav a následně na Make Transition. Tím můžeme začít propojovat prvky mezi sebou, postupujeme logicky dle průběhu pohybu postavy – například z výchozího stavu „Breathing Idle“ vytvoříme přechod (šipku) na stav Running, čímž zajistíme, aby postava, která je v klidu, mohla přejít do stavu Running. Logiku přechodů implementujeme dle obrázku 33. [65]



Obrázek 33: Stavy a přechody mezi animacemi postavy

Stavy, parametry a logiku přechodů máme vytvořené, nyní je potřeba nastavit přechodům (šipkám) podmínky. Tím zajistíme, že se budou animace měnit v závislosti na pohybu postavy (přechodu z jednoho vytvořeného stavu, reprezentující animaci, do druhého), přičemž tranzice z výchozího stavu „Entry“ do „Breathing Idle“ se nastaví automaticky jako výchozí.

Začneme tranzicí z „Breathing Idle“ do „Running“. Po kliknutí na šipku se v okně inspektoru otevře nastavení pro tranzici mezi animacemi. V sekci Condition nyní využijeme vytvořené parametry, a pro tuto tranzici přidáme parametr „isMoving“ a nastavíme ho na hodnotu true.

Následně nastavíme opačný směr, tedy tranzici z „Running“ do Breathing Idle“, přičemž opět nastavíme parametr „isMoving“, tentokrát ale na hodnotu false. Nastavíme podmínky pro všechny přechody mezi stavy/animacemi, dle níže uvedené tabulka (tabulka 1). [24] [25]

Tabulka 1: Tranzice mezi stavy pohybu

Počátek	**Cíl**	**Parametr**	**Hodnota parametru**
Breathing Idle	Running	isMoving	true
Running	Breathing Idle	isMoving	false
Breathing Idle	Jumping	IsJumping	true
Landing	Breathing Idle	isMoving	false
Landing	Running	isMoving	true
Falling	Landing	IsGrounded	true
Jumping	Falling	IsFalling	true
Running	Jumping	IsJumping	true
Running	Falling	IsFalling	true

Abychom mohli skákat (přejít do stavu Jumping) v průběhu stavu Running, a zároveň například skočit víckrát za sebou, je potřeba v přechodu Landing>Running a Landing>Breathing Idle nastavit parametr Interruption Source na hodnotu Next State, což umožní přerušení jedné animace na úkor jiné.

Nyní je potřeba upravit skript CharacterMovement pro přechod mezi animacemi v závislosti na ovládání pohybu, přičemž budeme vycházet z navržené logiky.

Jako první potřebujeme zajistit přístup ke komponentě Animator, což stejně jako v případě komponenty Character Controller provedeme v metodě Start():

```
animator = GetComponent<Animator>();
```

Nyní v sekci kódu, kde zjišťujeme, zda je postava v pohybu, přidáme pomocí vytvořených parametrů (v komponentě Animator) podmínku. Pokud se tedy postava pohybuje (isMoving = true), postava se bude pohybovat s animací Running (v komponentě Animator tento stav reprezentuje šipka, která ukazuje z Breathing Idle do stavu Running). V opačném případě

(else statement) nastavíme podmínku na hodnotu false, čímž se postava dostane do klidového stavu, což odpovídá animaci „Breathing Idle“. Navržená logika zde musí souhlasit s tím, co bylo navrženo v komponentě Animator:

```
if (movementDirection != Vector3.zero)
        {
            animator.SetBool("isMoving", true);
            Quaternion toRotation = Quaternion.LookRotation(movementDirection,
Vector3.up);

            transform.rotation = Quaternion.RotateTowards(transform.rotation,
toRotation, rotationSpeed * Time.deltaTime);
        }
        else
        {
            animator.SetBool("isMoving", false);
        }
```

Nyní máme ošetřený pohyb ze stavu „Breathing Idle“ do „Running“ a opačně. Při spuštění hry tak postava plynule přechází do animace běhu a naopak, dle vstupu uživatele. [25]

Stejným způsobem pokračujeme při úpravě pohybu, kde chceme využít animací Falling, Landing a Jumping. Tento proces je složitější, v závislosti na stavech, které se skokem souvisí.

Ve chvíli, kdy postava vyskočí, nastavíme parametr isJumping na hodnotu true, tuto úpravu provedeme v části, kde ošetřujeme skok postavy. Pokud je postava na zemi, parametr isGrounded bude mít hodnotu true, všechny ostatní parametry naopak false. Změny provedeme v metodě Update(), kde ověřujeme, zda je postava na zemi:

```
    if (Time.time - lastGroundedTime <= jumpButtonGracePeriod)
        {
            characterController.stepOffset = originalStepOffset;
            ySpeed = -0.5f;
            animator.SetBool("IsGrounded", true);
            isGrounded= true;
            animator.SetBool("IsJumping", false);
            isJumping = false;
            animator.SetBool("IsFalling", false);

            if (Time.time - jumpButtonPressedTime <= jumpButtonGracePeriod)
            {
                ySpeed = jumpSpeed;
                animator.SetBool("IsJumping", true);
                isJumping= true;
                jumpButtonPressedTime = null;
                lastGroundedTime = null;
            }
        }
```

Následně v sekci podmínky else nastavíme parametr Isgrounded na hodnotu false a provedeme vnořenou podmínku pro animaci Falling. Pokud je ySpeed (rychlost skoku ve směru osy y) na hodnotě menší než 0 a zároveň pokud přecházíme ze stavu isJumping, znamená to, že naše postava padá na zem – nastavíme tedy parametr IsFalling na hodnotu true. Zároveň přidáme podmínku, při které zajistíme přehrávání animace, pokud bude postava padat dolů – například z překážky.

```
else
        {
            characterController.stepOffset = 0;
            animator.SetBool("IsGrounded", false);
            isGrounded = false;

            if((isJumping && ySpeed <0) || ySpeed<-2)
            {
                animator.SetBool("IsFalling", true);
            }
        }
```

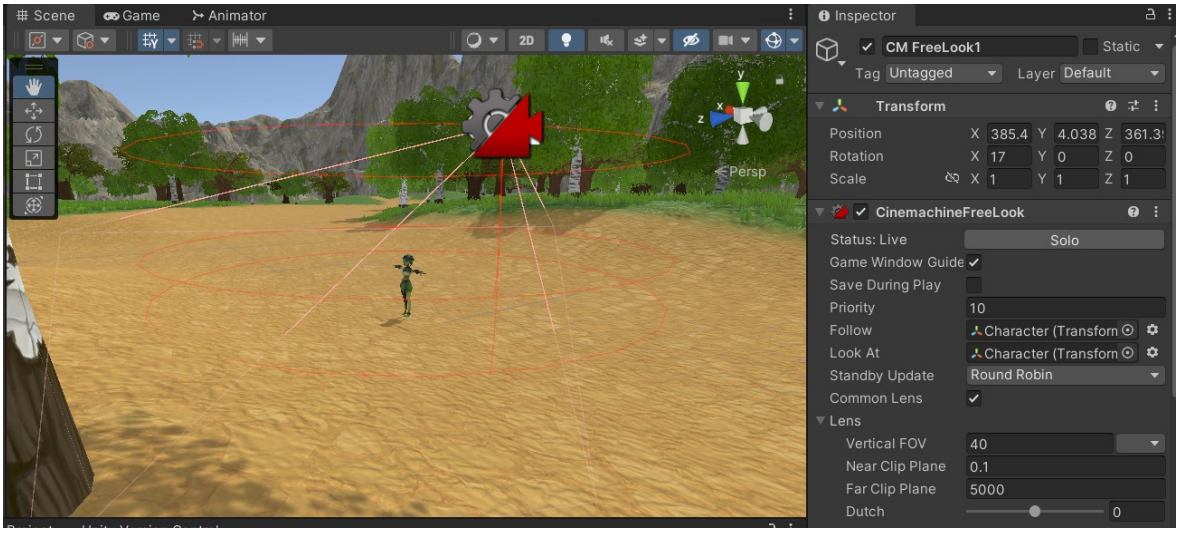
Animace postavy jsme ošetřili, nyní vše vypadá mnohem lépe a postava se pohybuje dle očekávání.

3.3.3 Nastavení kamery

Pro nastavení kamery využijeme komponentu Cinemachine. Tu Unity ve výchozím nastavení neobsahuje a je potřeba si ji přidat přes Window> Package Manager> Unity Registry. Ve chvíli, kdy máme balíček Cinemachine nainstalovaný, vložíme novou kameru, která bude rozšiřovat GameObject Main Camera. To uděláme přes Component> Cinemachine> CinemachineFreeLook.

Nyní máme přidaný nový GameObject CM FreeLook1, což představuje kameru, která sleduje naši postavu a ovládá hlavní kameru v naší scéně. Zároveň nám automaticky do objektu Main Camera přibyla komponenta CineMachineBrain, čímž se nám propojily tyto dva GameObjecty.

V nově vzniklém objektu kamery upravíme u komponenty CinemachineFreeLook parametr „Follow“ a „Look at“, kam přidáme referenci na komponentu transform naší postavy.



Obrázek 34: Free Look kamera

Zároveň nastavíme parametry TopRig, MiddleRig a BottomRig (Obrázek 28), které ovlivňují, jakým způsobem bude kamera natočena. Tyto parametry ve scéně vidíme jako tři červené kruhy. [15] Všem parametrům nastavíme Radius na hodnotu 6, což nám zajistí stabilitu kamery při pohybu myši. Budeme tak moci kolem postavy kameru otáčet kolem osy y, ale nebude nám to ovlivňovat vzdálenost od postavy samotné.

Obrázek 35: Nastavení parametrů kamery

Nyní upravíme skript CharacterMovement.

V metodě Update() přidáme pro změnu směru pohybu (proměnná movementDirection) funkci, aby se postava pohybovala tím směrem, kam myší natočíme kameru. [21]

Použijeme funkci Quaterninon.AngleAxis(), kterou lze aplikovat na vektory, čímž upravíme rotaci kamery. Na základě prvního parametru (výpočet úhlu rotace), proběhne rotace kamery podél osy y, což zde představuje druhý parametr funkce (Vector3.up). [16] [17]

```
movementDirection = Quaternion.AngleAxis(cameraTransform.rotation.eulerAngles.y,
Vector3.up) * movementDirection;
```

Tím je nastavena kamera a ovládání směru pohybu a rotace postavy.

4 TUTORIÁLY

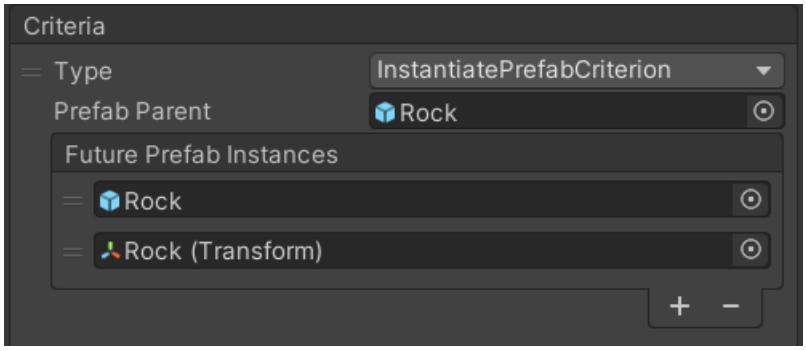
K interaktivnímu zapojení uživatele využijeme nástroj Unity in-Editor Tutorials, který bude sloužit pro tvorbu edukativních tutoriálů na nejdůležitější komponenty, a to přímo uvnitř samotného projektu.

Pro tvoru tutoriálů je zapotřebí do projektu nainstalovat Tutorial Authoring Tools a Tutorial Framework. To uděláme přes Window> Package Manager> Unity Registry.

K práci s nástrojem Unity in-Editor Tutorials je potřeba stručně nastínit funkci nástrojů Criteria, Masking, Callbacks. [63]

4.1 Criteria

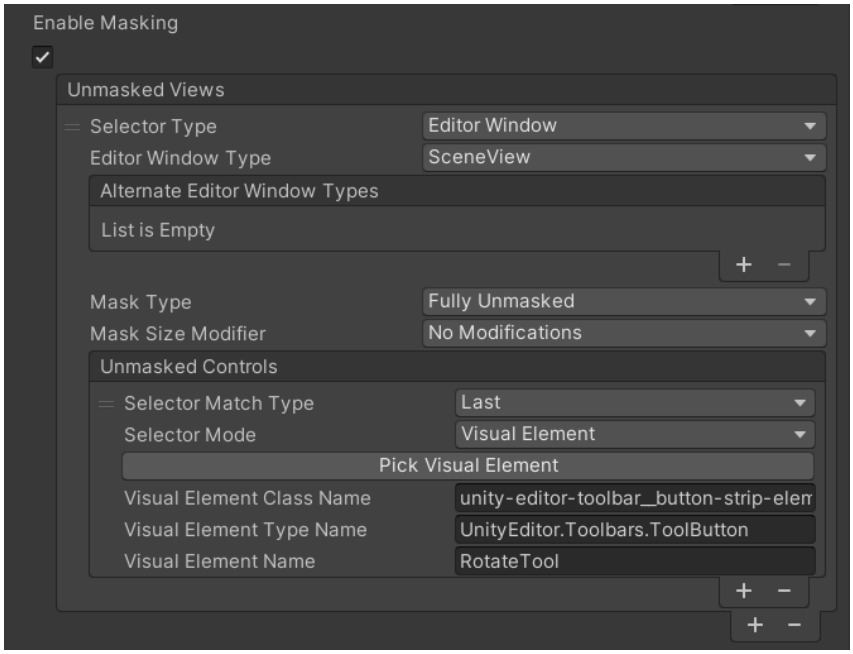
Abychom mohli v tutoriálech detekovat aktivitu uživatele a přecházet tak na další kroky, jsou nám k dispozici kritéria (Completion Criteria). Tato kritéria nám slouží k nastavení a detekci podmínek, které musí uživatel splnit, aby přešel na další krok v tutoriálu. Unity nám poskytuje rozsáhle množství kritérií, jako jsou například detekce vložení určitého GameObjectu, vybrání nástroje, a mnoho dalších. Zároveň je možné si pomocí skriptu vytvořit i svá vlastní. [63]



Obrázek 36: Kritérium pro vložení GameObjectu Rock

4.2 Masking

Pro pohodlnější navigování uživatele uvnitř Unity Editoru je možné využívat funkcionalitu Masking. Jedná se o velmi užitečný nástroj, který nám umožňuje vysvítit důležité GameObjecty, tlačítka nebo okna a zároveň zakrýt a zneaktivnit ostatní části Unity Editoru. Díky tomu můžeme uživateli pomoci s navigací a lepším dokončením instrukcí. Na Obrázku 37 lze vidět jakým způsobem je možné zviditelnit prvek RotateTool v okně Scene View. [63]



Obrázek 37: Masking nad RotateTool

4.3 Common Tutorials Callback

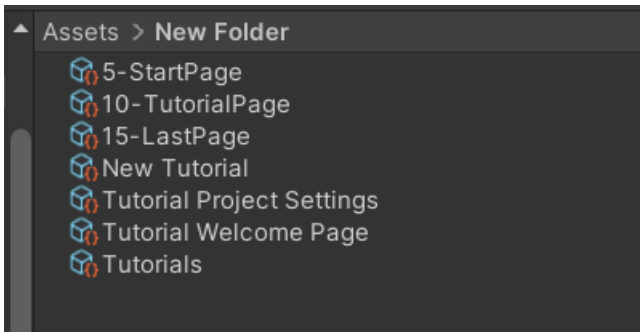
Callbacks metody nám umožňují společně s Masking lépe navigovat uživatele v prostoru Unity Editoru. Je nám k dispozici například metoda PingFolderOrFirstAsset(), kde můžeme předat referenci na GameObject. Ten při spuštění tutoriálu, kde je pro dokončení kroku potřeba přidat GameObject do scény, zobrazí v okně Project zvolený GameObject a uživatel ho tak nemusí složitě hledat. Další užitečnou metodou, která bude využívána v projektu je SelectGameObject(), pro navigování uživatele k výběru herního objektu v Hierarchy Window. [63]

4.4 Prvotní nastavení

Ve chvíli, kdy máme nainstalované potřebné balíčky, můžeme začít tvořit tutoriály. První tutoriál vytvoříme v okně Project, přes Create> Tutorials> Tutorial> Ready-to-Use Tutorial Project, přičemž se vygeneruje 7 objektů:

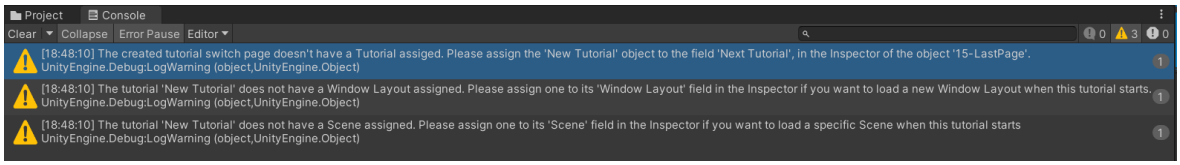
- Tutorials: Slouží jako kontejner pro hotové tutoriály, které zde můžeme seřadit dle toho, jak na sebe navazují.
- Tutorial Welcome Page: Uvítací stránka, která se zobrazí po spuštění projektu s tutoriály.
- Tutorial Project Setting: Zde nastavujeme obecná nastavení, jako je výběr Welcome Page, Scény nebo vizuál tutoriálů.

- New Tutorial – Slouží jako kontejner, tentokrát už pro jednotlivé stránky tutoriálu.
- 5 – StartPage: Jedná se o první stránku, kde zobrazujeme obecný popis daného tutoriálu. Tato stránka je pouze informativní a nezobrazuje žádné instrukce pro uživatele.
- 10 – TutorialPage: Stránka tutoriálu, kde tvoříme instrukce pro uživatele a zadáváme zde kritéria pro úspěšné dokončení kroku v tutoriálu.
- 15 – LastPage: Poslední stránka tutoriálu, která navíc od běžné stránky obsahuje přechod na další tutoriál, který je potřeba definovat.



Obrázek 38: Vygenerované objekty

Po vygenerování je potřeba vyřešit problémy, na které nás Unity upozorňuje:



Obrázek 39: Upozornění

Přes záložku Tutorials> Authoring> Layout> Save Current Layout to Asset uložíme Layout oken, který chceme mít při spuštění tutoriálů. Následně do objektu New Tutorial a Tutorial Project Settings přiřadíme scénu, ve které se budou tutoriály přehrávat. Uložený Layout přiřadíme do objektu Tutorials. Abychom odlišily tutoriály od sebe, je potřeba si tutoriály patřičně pojmenovat, první tutoriál (objekt New Tutorial) nazveme „T1“ a vytvoříme si pro něj novou složku. Poslední upozornění je na přechod mezi tutoriály, což lze vyřešit až s přidáním dalších tutoriálů.

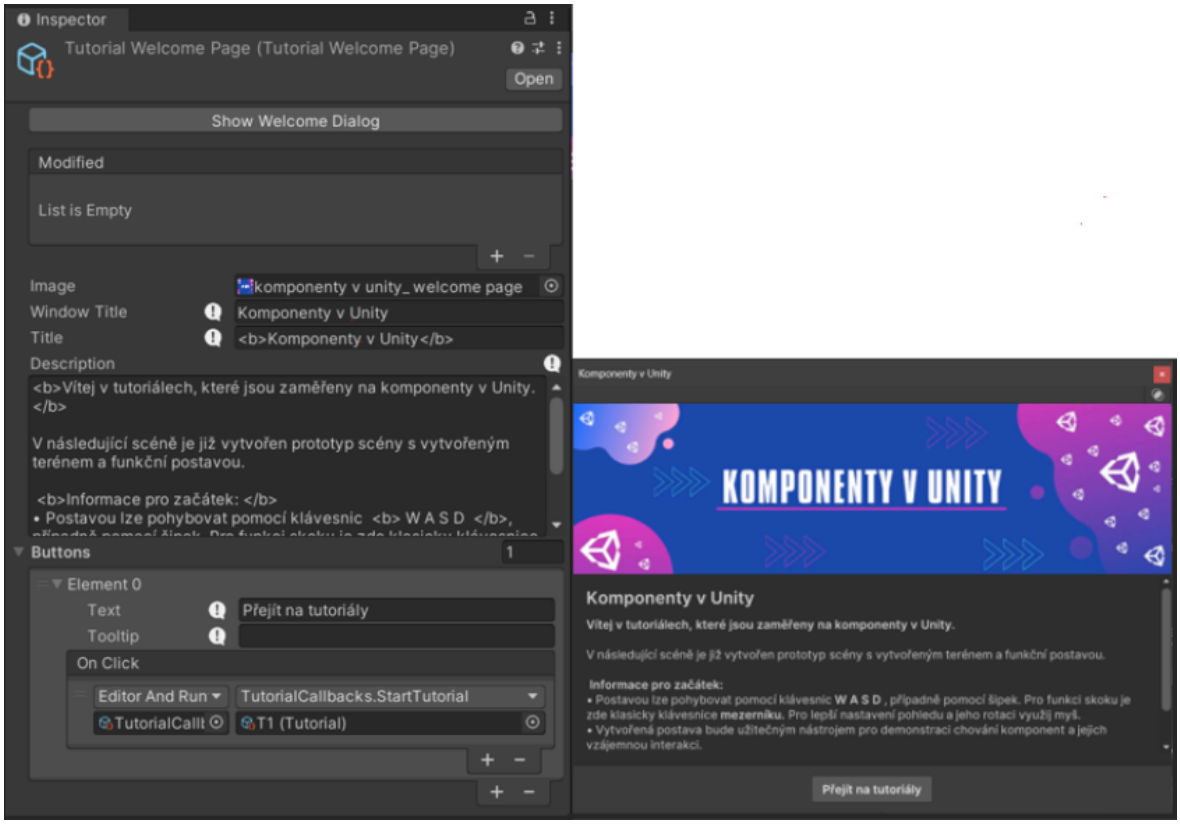
4.4.1 Welcome Page

Welcome Page je okno, které se otevře při spuštění projektu, ve kterém jsou tutoriály. Je zde možné nastavit UI, jako je titulní obrázek, nadpis, podnadpis a popis.

Zároveň je zde možné nastavit tlačítko, které uživatele přesměruje na tutoriály. Aby byl uživatel přesměrován na tutoriály, je potřeba vytvořit Callback Handler a jeho skript. Ve složce, kde máme tutoriály klikneme na Create> Tutorials> Callback Handler, čímž se nám vygeneruje objekt a script Tutorials Callback. Následně upravíme script, kam přidáme metodu StartTutorial pro spuštění tutoriálu.

```
public void StartTutorial(Tutorial tutorial)
        {
            TutorialWindowUtils.StartTutorial(tutorial);
        }
```

Následně ho přiřadíme do Welcome Page, do tlačítka, které pojmenujeme „Přejít na tutoriály“, přičemž se jako první zavolá tutoriál T1, na který předáme referenci.



Obrázek 40: Nastavení a zobrazení Welcome Page

4.5 Tutoriál T1

První tutoriál je zaměřen na komponentu Transform a Collider. Zde využijeme již na začátku stažený Asset Fantasy Environments, jehož součástí jsou GameObjecty Rock a Mushroom. U GameObjectu Mushroom odebereme koponentu Collider a prefab tak

uložíme. Uživatel tak do scény vloží dva herní objekty, přičemž GameObject Mushroom nebude mít komponentu Collider a bude moci otestovat rozdíl mezi chováním herních objektů s Collidery a bez.

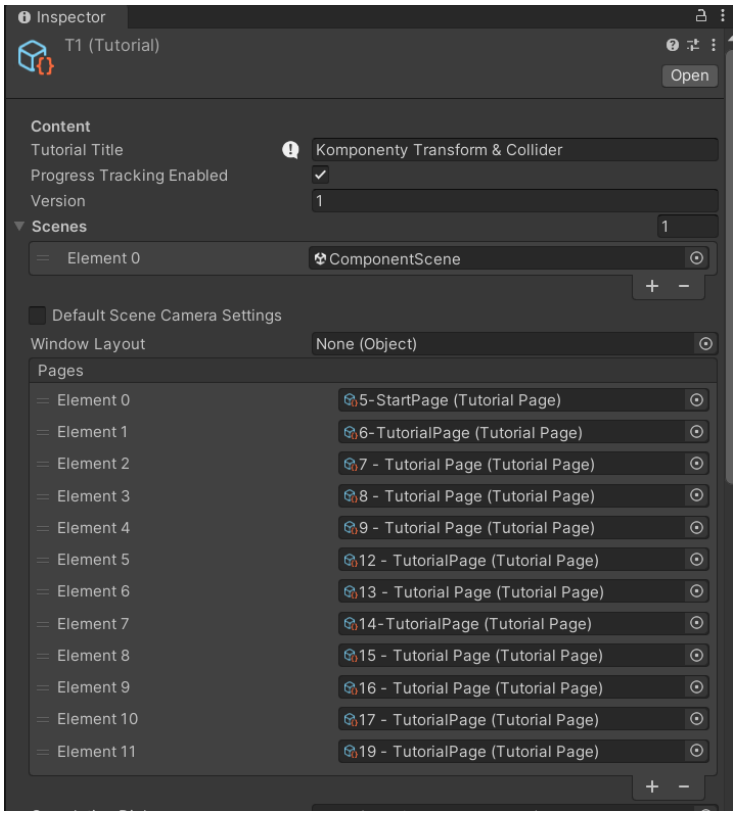
4.5.1 Scénář tutoriálu

Tabulka 2: Scénář tutoriálu T1

Krok	Popis kroku či instrukce
1	Start Page.
2	Vložení herního objektu Rock s komponentou Transform a Mesh Collider do scény. Kritérium pro dokončení kroku je InstatntiatePrefabCriterion s instancí GameObjectu Rock.
3	Popis komponenty Transform, jedná se o stránku bez instrukce, není zde tedy žádné kritérium pro dokončení (ActiveToolCriterion – Rotate).
4	Zvolení nástroje Rotate Tool, kritérium pro dokončení je kliknutí na Rotate Tool.
5	Otočení GameObjectu pro demonstraci změn v komponentě Tarnsform, kritérium pro dokončení je změna parametrů Rotation v komponentě transform (PropertyModificationCriterion).
6	Uložení scény, kritérium je zde SaveSceneCriteria – po uložení scény je instrukce splněna.
7	Popis komponent skupiny Collider, jedná se o stránku bez instrukce, není zde žádné kritérium pro dokončení.
8	Uživatel vloží herní objekt Mushroom_bez_collideru do scény. Kritérium pro dokončení kroku je InstatntiatePrefabCriterion s instancí GameObjectu Mushroom_bez_collideru.
9	Popis vloženého objektu, popis GameObjectu bez collideru.
10	Přechod do herního režimu. Kritérium pro dokončení instrukce je přechod do Play Mode (PlayModeStateCriterion – Playing).

11	Odzkoušení kolizí na GameObjectech s vytvořenou postavou. Poté co uživatel zkusí chování GameObjectů, opouští Play Mode, což je kritérium pro dokončení kroku (PlayModeStateCriterion – Not Playing).
12	Uložení scény, kritérium je zde SaveSceneCriteria – po uložení scény je instrukce splněna, zároveň má zde uživatel nabídnutý přechod na druhý tutoriál.

Následně všechny kroky tutoriálu seřadíme do vytvořeného kontejneru T1. Tento krok se opakuje u všech vytvořených tutoriálů.



Obrázek 41: Tutoriál T1

4.6 Tutoriál T2

Druhý tutoriál je zaměřen na komponenty Audio Source a Audio Listener. Pro tento tutoriál je potřeba vytvořit model herního objektu ohně, přičemž využijeme Asset Camfire & Torches Models. Uživatel vloží předem připravený GameObject, který bude mít komponentu Particle System (pro vytvoření efektu plamenů) a komponentu Audio Source se zvukem praskání ohně. [39] [66]

4.6.1 Scénář tutoriálu

Tabulka 3: Scénář tutoriálu T2

Krok	Popis kroku či instrukce
1	Start Page.
2	Vložení herního objektu campFire s komponentou Audio Source do scény. Kritérium pro dokončení kroku je InstatntiatePrefabCriterion s instancí GameObjectu campFire.
3	Popis komponenty Audio Source. Jedná se o stránku bez instrukce, není zde tedy žádné kritérium pro dokončení.
4	Uložení scény s herním objektem. K dokončení kroku je potřeba scénu uložit, což detekuje SaveSceneCriteria.
5	Přechod do herního režimu. Kritérium pro dokončení instrukce je přechod do Play Mode (PlayModeStateCriterion – Playing).
6	Odzkoušení AudioSource na GameObjectu s vytvořenou postavou. Poté co uživatel zkusí chování GameObjectů, opouští Play Mode, což je kritérium pro dokončení kroku (PlayModeStateCriterion – Not Playing).
7	Popis komponenty Audio Listener, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
8	Vybrání GameObjectu Character (herní postavy) v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání postavy (RequiredSelectionCriterion) s referencí na GameObject Character.
9	Zobrazení komponenty AudioListener, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
10	Uložení scény s herním objektem, kritérium pro dokončení je zde SaveSceneCriteria. Zároveň je zde uživateli nabídnut tutoriál T5.

4.7 Tutoriál T3

Tento tutoriál je zaměřen na komponentu Rigidbody. Pro tutoriál je zapotřebí přidat do projektu Asset Free Sports Kit, ze kterého využijeme model míče. [38]

4.7.1 Scénář tutoriálu

Tabulka 4: Scénář tutoriálu T3

Krok	Popis kroku či instrukce
1	Start Page.
2	Smazání nepotřebných GameObjectů Rock a Mushroom_bez_Collideru.
3	Uložení scény. K dokončení kroku je potřeba scénu uložit, což detekuje SaveSceneCriteria.
4	Vložení herního objektu Ball do scény. Kritérium pro dokončení kroku je InstatntiatePrefabCriterion s instancí GameObjectu Ball.
5	Zvolení nástroje Move Tool, kritérium pro dokončení je kliknutí na Move Tool (ActiveToolCriterion – Move).
6	Změna pozice míče po ose y ve scéně – k detekci zde slouží PropertyModificationCriterion.
7	Uložení scény s herním objektem. K dokončení kroku je potřeba scénu uložit, což detekuje SaveSceneCriteria.
8	Přechod do herního režimu. Kritérium pro dokončení instrukce je přechod do Play Mode (PlayModeStateCriterion – Playing).
9	Odzkoušení GameObjectu Ball v herním módu. Poté, co uživatel zkusí chování GameObjectů, opouští Play Mode, což je kritérium pro dokončení kroku (PlayModeStateCriterion – Not Playing).
10	Vybrání GameObjectu Ball v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání postavy (RequiredSelectionCriterion) s referencí na GameObject Ball.
11	Přidání komponenty rigidbody GameObjectu Ball. Pro detekci zde využijeme kritérium ComponentAddedCriterion.

12	Popis komponenty Rigidbody.
13	Přechod do herního režimu. Kritérium pro dokončení instrukce je přechod do Play Mode (PlayModeStateCriterion – Playing).
14	Odzkoušení GameObjectu Ball s komponentou Rigidbody v herním módu. Poté, co uživatel zkusí chování GameObjectů, opouští Play Mode, což je kritérium pro dokončení kroku (PlayModeStateCriterion – Not Playing).
15	Uložení scény s přechodem na další tutoriál T6.

4.8 Tutoriál T4

Čtvrtý tutoriál je zaměřen na komponentu Light a Camera, které jsou výchozí součástí Unity projektu po spuštění.

4.8.1 Scénář tutoriálu

Tabulka 5: Scénář tutoriálu T4

Krok	Popis kroku či instrukce
1	Start Page.
2	Vybrání GameObjectu Main Camera v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání postavy (RequiredSelectionCriterion) s referencí na GameObject Main Camera.
3	Popis komponenty Camera, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
4	Vybrání GameObjectu Directional Light v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání objektu (RequiredSelectionCriterion) s referencí na GameObject Directional Light.
5	Popis komponenty Light, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
6	Stránka po přechod na tutoriál T3.

4.9 Tutoriál T5

Tento tutoriál je zaměřen na komponentu Particle System, k čemuž využijeme předem připravený GameObject campFire z tutoriálu T2.

4.9.1 Scénář tutoriálu

Tabulka 6: Scénář tutoriálu T5

Krok	Popis kroku či instrukce
1	Start Page.
2	Vybrání GameObjectu campFire v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání postavy (RequiredSelectionCriterion) s referencí na GameObject campFire.
3	Popis komponenty Particle System, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
4	Změna parametru Start Lifetime pro demonstraci změny chování. K detekci dokončení zde slouží PropertyModificationCriterion.
5	Změna parametru Start Speed pro demonstraci změny chování. K detekci dokončení zde slouží PropertyModificationCriterion.
6	Změna parametru Gravity Modifier pro demonstraci změny chování. K detekci dokončení zde slouží PropertyModificationCriterion.
7	Změna parametru Rate Over Time pro demonstraci změny chování. K detekci dokončení zde slouží PropertyModificationCriterion.
8	Uložení scény s přechodem na další tutoriál T4.

4.10 Tutoriál T6

Šestý tutoriál je zaměřen na komponentu Nav Mesh Agent, přičemž budeme potřebovat Asset Kawaii Slimes, ze kterého využijeme prefab, který pojmenujeme BunnyPet. [40]

Zároveň je potřeba připravit script, který přiřadíme GameObjectu BunnyPet s komponentou Nav Mesh Agent. Script přebírá do parametru „Player Target“ herní objekt Character. Díky tomu pak bude GameObject BunnyPet postavu ve hře sledovat.

```
public class pet : MonoBehaviour
{
    private NavMeshAgent petZ;

    public Transform PlayerTarget;
    // Start is called before the first frame update
    void Start()
    {
        petZ = GetComponent<NavMeshAgent>();
    }

    // Update is called once per frame
    void Update()
    {
        petZ.SetDestination(PlayerTarget.position);
    }
}
```

4.10.1 Scénář tutoriálu

Tabulka 7: Scénář tutoriálu T6

Krok	Popis kroku či instrukce
1	Start Page.
2	Vložení herního objektu BunnyPet s připraveným scriptem do scény. Kritérium pro dokončení kroku je InstatntiatePrefabCriterion s instancí GameObjectu BunnyPet.
3	Otevření okna Navigation.
4	Popis systému navigace.
5	Vybrání GameObjectu Terrain v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání objektu (RequiredSelectionCriterion) s referencí na GameObject Terrain.
6	Kliknutí na tlačítko Bake pro nastavení navigační sítě v terénu.
7	Vybrání GameObjectu BunnyPet v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání postavy (RequiredSelectionCriterion) s referencí na GameObject BunnyPet.

8	Zavření okna navigace.
9	Přidání komponenty Nav Mesh Agent do GameObjectu BunnyPet. Kritérium pro dokončení je ComponentAddedCriterion.
10	O komponentě Nav Mesh Agent.
11	Upravení parametru Player target v komponentě scriptu.
12	Upravení parametru Target Value. K detekci dokončení zde slouží PropertyModificationCriterion.
13	Uložení scény. K dokončení kroku je potřeba scénu uložit, což detekuje SaveSceneCriteria.
14	Přechod do herního režimu. Kritérium pro dokončení instrukce je přechod do Play Mode (PlayModeStateCriterion – Playing).
15	Testování komponenty Nav Mesh Agent. Následné vypnutí Play mode (PlayModeStateCriterion – Not Playing) a přechod na další tutoriál.

4.11 Tutoriál T7

Tento tutoriál je zaměřen na komponentu Terrain, k čemuž potřebujeme prefab Flower1 z již přidaného Assetu Fantasy Environments.

4.11.1 Scénář tutoriálu

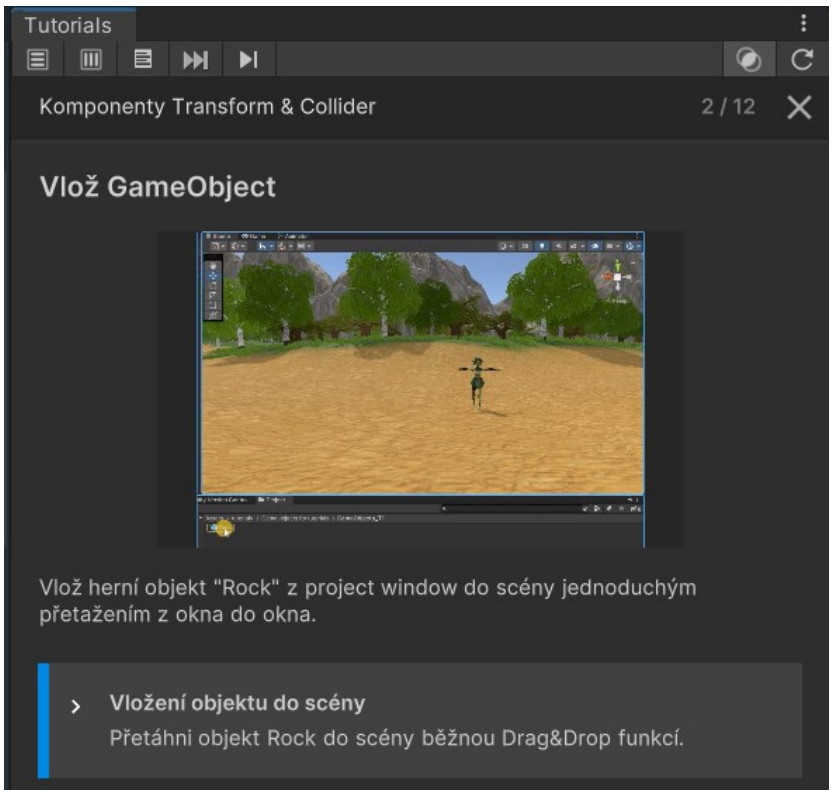
Tabulka 8: Scénář tutoriálu T7

Krok	Popis kroku či instrukce
1	Start Page
2	Vybrání GameObjectu Terrain v Hierarchy okně. Kritérium pro dokončení kroku je zde vybrání objektu (RequiredSelectionCriterion) s referencí na GameObject Terrain.
3	Popis komponenty Terrain, jedná se o stránku bez instrukce, není tedy žádné kritérium pro dokončení.
4	Zvolení nástroje Paint Details.

5	Přidání detailů do terénu pomocí nástroje Paint Details. Spuštění herního módu (PlayModeStateCriterion – Playing).
6	Pozorování změn v terénu, opuštění herního módu (PlayModeStateCriterion – Not Playing).
7	Uložení scény. K dokončení kroku je potřeba scénu uložit, což detekuje SaveSceneCriteria.

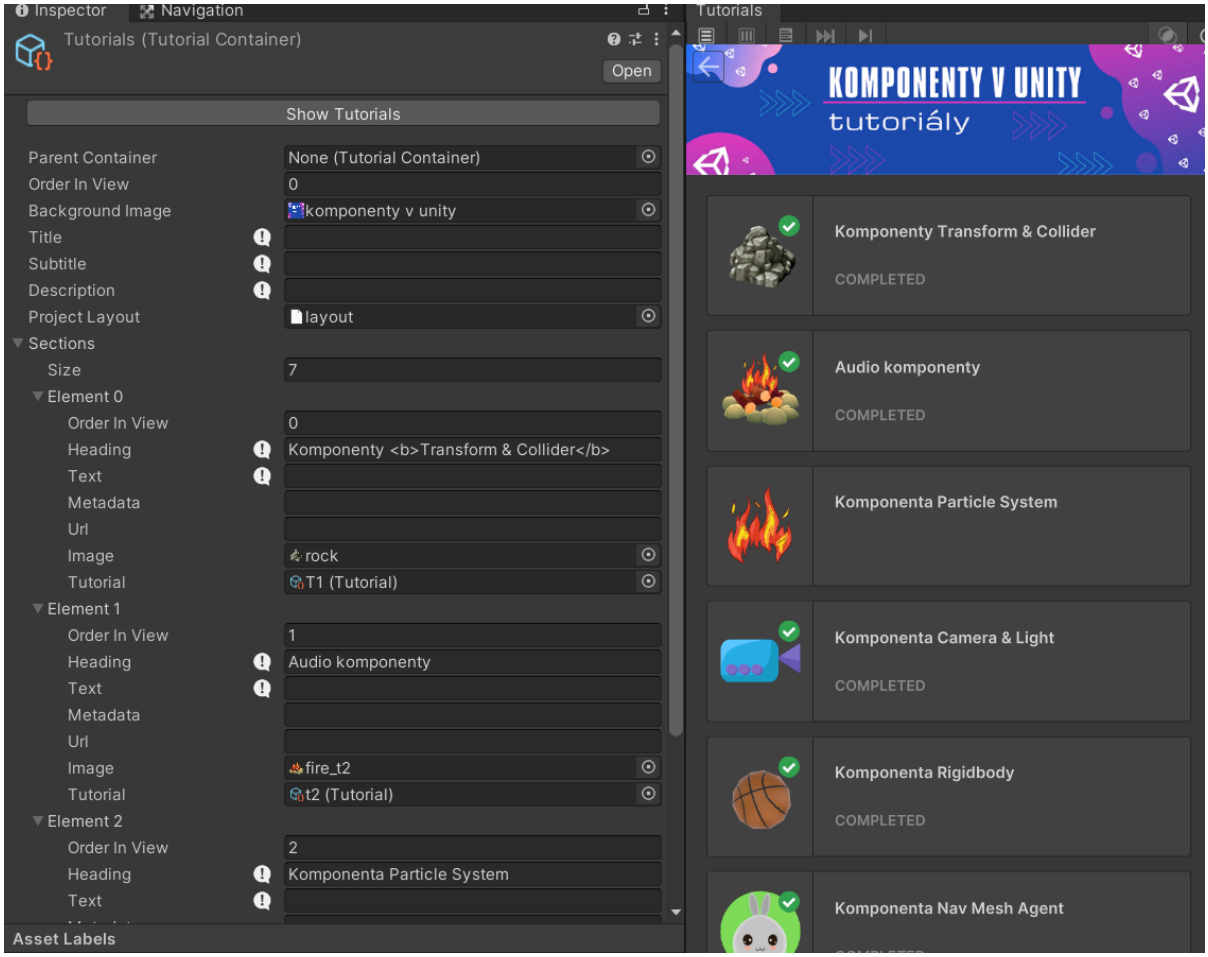
4.12 Finální úpravy

Pro lepší vizuál a zároveň i pro lepší navigaci instrukcí uvnitř tutoriálů je potřeba vytvořit i obrázky a videa, které uživateli mohou napovědět v každém kroku s instrukcí. K tomuto účelu každý tutoriál obsahuje sadu obrázků a videí.



Obrázek 42: Instrukce s videonávodem

Zároveň je také potřeba seřadit tutoriály dle jejich návaznosti, to uděláme v kontejneru pro tutoriály (objekt Tutorials). Tam zároveň nastavíme i vzhled jednotlivým tutoriálům, který se bude zobrazovat v Tutorial Window po spuštění projektu.



Obrázek 43: Hotové tutoriály

V neposlední řadě je potřeba po otestování funkčnosti všech tutoriálů resetovat jejich stav, což provedeme přes záložku Tutorials> Authoring> Debug> Progress Tracking> Clear All Statuses. Tím je projekt připraven pro nového uživatele ihned po spuštění, přičemž se mu jako první zobrazí uvítací dialog (Welcome page) s následným přesměrováním na tutoriály. Následně je uživatel pomocí instrukcí naveden na práci s Unity Editorem a komponentami, a postupně tak plní jednotlivé kroky a dokončuje danou oblast tutoriálů. Vždy je v posledním kroku nastaven pomocí tlačítka přechod na další tutoriál.

Práce s tutoriály je uživatelsky velmi přívětivá a intuitivní.

ZÁVĚR

Cílem této bakalářské práce bylo popsání Unity Editoru, charakteristika komponent a jejich nejdůležitějších parametrů. Dále bylo cílem navrhnout a implementovat prototyp 3D hry a následně efektivně představit práci s komponentami.

V první, teoretické části, byl stručně popsán Unity Editor a byly zmíněny i další podstatné prvky, jako je například Unity Hub nebo Unity Asset Store. Následně byly popsány všechny skupiny Unity komponent, včetně detailního záběru na každou komponentu a jejich parametrů, které se v dané skupině nachází.

Praktická část sestává ze dvou částí. V první části byl navrhnut a implementován protoyp 3D hry, kde byla vytvořen scéna s terénem a funkční postavou, která se po terénu může pohybovat. V druhé části bylo pomocí nástroje Unity in-Editor Tutorials vytvořeno sedm tutoriálů. Tutoriály se zaměřují na komponenty Transform, Collider, Audio Listener, Audio Source, Particle System, Camera, Light, Rigidbody, Nav Mesh Agent a Terrain. Díky nástroji in-Editor Tutorials bylo možné vytvořit interaktivní tutoriály, které se nabídnou uživateli ihned po spuštění projektu. Uživatel se tak zapojí přímo ve vytvořeném projektu do práce s komponentami, které mu jsou pomocí instrukcí jednotlivých stránek tutoriálu představeny.

Při vypracování samotných tutoriálů bylo potřeba vytvořit i scénář instrukcí, které uživatele zaujmou, ale zároveň mu dodají i potřebnou znalost o zmíněných komponentách.

Dále byl doplněn vizuál pro jednotlivé stránky tutoriálu, přičemž pro každý tutoriál byla vytvořena sada videí a obrázků. Díky tomu tutoriály nemají jednotný vzhled a udrží tak pozornost uživatele.

Důležité je také zdůraznit, že in-Editor nástroj je poměrně nový a v současné době stále nemá podchycené všechny možnosti, jako je například větší množství dostupných kritérií pro dokončení. Nicméně se jedná o velmi užitečný nástroj, který může pomoci začínajícím herním vývojářům.

Celkově jsem při tvorbě prototypu i tutoriálů získala cenné zkušenosti, především co se práce v Unity Editoru, komponent, jazyku C# a in-Editoru týče. Věřím, že práce bude užitečným prostředkem v hodinách předmětu Vývoj počítačových her.

SEZNAM POUŽITÉ LITERATURY

[1] Unity - Manual: The Scene view. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 12.03.2023]. Dostupné z: https://docs.unity3d.com/Manual/UsingTheSceneView.html

[2] WELLS, Robert. Unity 2020 By Example: A project-based guide to building 2D, 3D, augmented reality, and virtual reality games from scratch, 3rd Edition. Third Edition. Birmingham: Packt Pub, 2020. ISBN 9781800203389.

[3] Unity - Manual: The Game view. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 10.04.2023]. Dostupné z: https://docs.unity3d.com/Manual/GameView.html

[4] Davis, A., Baptiste, T., Craig, R., & Stunkel, R. (2022). *Unity 3D Game Development: Designed for passionate game developers Engineered to build professional games*. Packt Publishing.

[5] Grey Spencer. 2021. Mind-Melding Unity and Blender for 3d Game Development : Unleash the Power of Unity and Blender to Create Amazing Games. Birmingham: Packt Publishing.

[6] Borromeo, N. A. (2022). Hands-On Unity 2022 Game Development: Learn to use the latest Unity 2022 features to create your first video game in the simplest way possible (3rd ed.). Packt Publishing.

[7] Unity - Manual: Rigidbody component reference. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 11.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Rigidbody.html

[8] Unity: CHARACTER CONTROLLER vs RIGIDBODY | by IronEqual | IronEqual | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483

[9] Unity - Manual: Hinge Joint component reference. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 11.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-HingeJoint.html

[10] Intro to Unity Physics - Part 4 Joints and Chain Physics | Marc Hewitt. Marc Hewitt [online]. Dostupné z: https://content.marchewitt.com/intro-to-unity-physics-part-4-joints-and-chain-physics

[11] Unity - Manual: Cloth. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 11.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Cloth.html

[12] Unity - Manual: Audio overview. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 11.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/AudioOverview.html

[13] Unity for beginners in 2021- importing Mixamo characters #1 - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 13.05.2023]. Dostupné z: https://www.youtube.com/watch?v=0QA2O7juuWQ&ab_channel=BluefeverSoftware

[14] Lekce 2 - 3D střílečka v Unity - Terén a pohyb hráče. itnetwork.cz - Učíme národ IT [online]. Copyright © 2023 itnetwork.cz. Veškerý obsah webu [cit. 13.05.2023]. Dostupné z: https://www.itnetwork.cz/csharp/unity-3d/3d-hry/3d-strilecka-v-unity-teren-a-pohyb-hrace

[15] Creating a Third Person Camera using Cinemachine Free Look in Unity that Avoids Obstacles (Tutorial) - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 13.05.2023]. Dostupné z: https://www.youtube.com/watch?v=jPU2ri4ZwxM

[16] Unity - Scripting API: Quaternion.AngleAxis. [online]. Copyright ©2023 Unity Technologies. Publication Date [cit. 13.05.2023]. Dostupné z: https://docs.unity3d.com/ScriptReference/Quaternion.AngleAxis.html

[17] Unity - Scripting API: Quaternion.eulerAngles. [online]. Copyright ©2023 Unity Technologies. Publication Date [cit. 13.05.2023]. Dostupné z: https://docs.unity3d.com/ScriptReference/Quaternion-eulerAngles.html

[18] Making a Character Jump (Unity Tutorial) - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 13.05.2023]. Dostupné z: https://www.youtube.com/watch?v=ynh7b-AUSPE&ab_channel=KetraGames

[19] Controlling a Character - Unity Game Development Tutorial. Ketra Games [online]. Dostupné z: https://www.ketra-games.com/2020/08/unity-game-tutorial-controlling-a-character.html

[20] Improve Annoying Jump Controls With Coyote Time and Jump Buffering - Unit Game development Tutorial. Ketra Games [online]. Dostupné z: https://www.ketra-games.com/2021/08/coyote-time-and-jump-buffering.html

[21] Creating a Third Person Camera (Unity Tutorial) - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 13.05.2023]. Dostupné z: https://www.youtube.com/watch?v=jiyOZbKRfaY

[22] Unity - Manual: Mesh Renderer component. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 13.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-MeshRenderer.html

[23] Understand Real-Time Rendering In Both 3D & 2D with Unity. Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine [online]. Copyright © 2023 Unity Technologies [cit. 14.05.2023]. Dostupné z: https://unity.com/how-to/real-time-rendering-3d

[24] Animated Character Jump (Unity Tutorial) - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 14.05.2023]. Dostupné z: https://www.youtube.com/watch?v=sJvWmFYSQFY

[25] How to use Animation Transitions (Unity Tutorial) - YouTube. YouTube [online]. Copyright © 2023 Google LLC [cit. 14.05.2023]. Dostupné z: https://www.youtube.com/watch?v=2_Hn5ZsUIXM&ab_channel=KetraGames

[26] Event System Manager | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-EventSystem.html

[27] Event System | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/EventSystem.html

[28] Canvas | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-Canvas.html

[29] Canvas Group | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-CanvasGroup.html

[30] Unity - Manual: Aspect Ratio Fitter. [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity.cn/2023.1/Documentation/Manual/script-AspectRatioFitter.html

[31] Content Size Fitter | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 15.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-ContentSizeFitter.html

[32] Script Machines and State Machines | Visual Scripting | 1.8.0 . [online]. Copyright © 2022 Unity Technologies [cit. 17.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.visualscripting@1.8/manual/vs-graph-machine-types.html

[33] Unity - Manual: Tilemap component reference. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 17.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Tilemap.html

[34] Unity - Manual: LOD Group. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 18.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-LODGroup.html

[35] Unity - Manual: Occlusion culling. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 18.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/OcclusionCulling.html

[36] UI Components - Unity Learn. Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn [online]. Copyright © 2023 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://learn.unity.com/tutorial/ui-components#5c7f8528edbc2a002053b4d2

[37] Visual Components | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/comp-UIVisual.html

[38] Free Sports Kit | 3D Characters | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/characters/free-sports-kit-239377

[39] Campfires & Torches Models and FX! | 3D Environments | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/environments/campfires-torches-models-and-fx-242552

[40] Kawaii Slimes | 3D Creatures | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/characters/creatures/kawaii-slimes-221172

[41] Unity - Manual: Camera component. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 20.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Camera.html

[42] The Light component | Universal RP | 11.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 20.05.2023]. Dostupné z: https://docs.unity.cn/Packages/com.unity.render-pipelines.universal@11.0/manual/light-component.html

[43] What is a Unity GameObject, and How Do You Fit It Into Your Game? | Starloop Studios. Video Game Development Outsourcing Company. 3D & Art design Studio [online]. Copyright © 2022, Starloop Studios. All rights reserved. [cit. 21.05.2023]. Dostupné z: https://starloopstudios.com/what-is-a-unity-gameobject-and-how-do-you-fit-it-into-your-game/

[44] Unity - Manual: Audio Filters. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-AudioEffect.html

[45] Unity - Manual: Skinned Mesh Renderer. [online]. Copyright © 2020 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity.cn/2021.1/Documentation/Manual/class-SkinnedMeshRenderer.html

[46] Unity - Manual: TextMeshPro . [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/com.unity.textmeshpro.html

[47] Unity - Manual: Visual Effects Components. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/2021.1/Documentation/Manual/comp-Effects.html

[48] Unity - Manual: Particle System. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-ParticleSystem.html

[49] Interaction Components | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIInteractionComponents.html

[50] Graphs | Visual Scripting | 1.7.8 . [online]. Copyright © 2022 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity.cn/Packages/com.unity.visualscripting@1.7/manual/vs-graph-types.html

[51] Unity - Manual: Video Player component. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-VideoPlayer.html

[52] Unity - Manual: Playable Director component. [online]. Copyright © 2017 Unity Technologies. Publication [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/2017.1/Documentation/Manual/class-PlayableDirector.html

[53] Unity - Manual: Navigation System in Unity. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/nav-NavigationSystem.html

[54] Unity - Manual: Create user interfaces (UI). [online]. Copyright © 2020 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity.cn/2022.2/Documentation/Manual/UIToolkits.html

[55] Unity - Manual: Animator component. [online]. Copyright © 2023 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/2023.2/Documentation/Manual/class-Animator.html

[56] Unity - Manual: Grid. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Grid.html

[57] Unity - Manual: Billboard Renderer component. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-BillboardRenderer.html

[58] Unity - Manual: Constraints. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/Constraints.html

[59] Unity - Manual: Sprite Masks. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-SpriteMask.html

[60] Canvas Renderer | Unity UI | 1.0.0 . [online]. Copyright © 2020 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-CanvasRenderer.html

[61] Unity - Manual: Lighting. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/LightingOverview.html

[62] Unity - Manual: Skybox component reference. [online]. Copyright © 2021 Unity Technologies. Publication Date [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Manual/class-Skybox.html

[63] Using Tutorial Authoring Tools | Tutorial Authoring Tools | 1.2.2 . [online]. Copyright © 2023 Unity Technologies [cit. 21.05.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.learn.iet-framework.authoring@1.2/manual/authoring-guide.html

[64] Mixamo. Mixamo [online]. Dostupné z: https://www.mixamo.com/#/?page=1&type=Character

[65] Mixamo. Mixamo [online]. Dostupné z: https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack

[66] Campfires & Torches Models and FX! | 3D Environments | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 23.05.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/environments/campfires-torches-models-and-fx-242552

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UI User Interface

GPU Graphics Processing Unit

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Příloha P1 USB flash disk obsahující adresář „Project“ se spustitelným projektem v Unity 2021.3.10f1 a soubor „fulltext.pdf“ ve formátu PDF/A.