Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Tvorba 3D platformové hry pomocí Unity Visual Scripting

Ondřej Rykl

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ondřej Rykl

Informatika

Název práce

Tvorba 3D platformové hry pomocí Unity Visual Scripting

Název anglicky

Creating a 3D platform game using Unity Visual Scripting

Cíle práce

Cílem teoretické části práce je analyzovat funkce a možnosti herního engine Unity pro tvorbu her za pomocí nástroje Visual Scripting a představit prostředí, jak programovací, tak prostředí pro vytváření, a vkládání 3D grafiky. V teoretické části zároveň bude provedena analýza her stejného žánru. Po analýze podobných her bude následovat srovnání podobností a rozdílů s hrou vyvíjenou v rámci této bakalářské práce.

Cílem praktické části práce je vytvořit platformovou 3D hru s First-Person Shooter (FPS) prvky v herním engine Unity za pomocí nástroje Visual Scripting. Hra bude primárně vytvořena pro operační systém Windows.

Metodika

Na základě dostupných materiálů budou v teoretické části popsány nástroje, které využijeme pro tvorbu v praktické části. V praktické části bude popsán návrh a implementace hry. Hra bude tvořena programovacími bloky z Visual Scripting balíčku a doplněna grafikou. Grafika prostředí bude buď vytvořena přímo v modelovacím nástroji Unity a nebo stažena z dostupných online zdrojů s volnou licencí.

Doporučený rozsah práce

30-60 stran

Klíčová slova

3D počítačová hra, Visual Scripting, Unity

Doporučené zdroje informací

Bertolini, L 2018, Hands-On Game Development Without Coding: Create 2D and 3D Games with Visual Scripting in Unity, Packt Publishing, Limited, Birmingham. ISBN 9781789537987

Unity Learn. Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn [online]. Copyright © 2022 Unity Technologies [cit. 04.06.2022]. Dostupné z: https://learn.unity.com/

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 24. 02. 2023

Čestné prohlášení

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

V Praze dne 15.3.2023 _____

Poděkování

Rád bych touto cestou poděkoval své vedoucí bakalářské práce Ing. Daně Vynikarové, Ph.D., za odborné vedení mé bakalářské práce, za pomoc a ochotu, cenné rady a připomínky. Dále bych rád poděkoval mojí rodině a své přítelkyni, která mě nesmírně podporuje po celou dobu mých studií.

Tvorba 3D platformové hry pomocí Unity Visual Scripting

Abstrakt

Tato práce se zaměřuje na analýzu funkčnosti herního engine Unity a jeho nástroje Visual Scripting pro tvorbu her včetně vkládání 3D grafiky. V teoretické části je provedena analýza engine Unity a her stejného žánru. Praktická část se soustředí na vytvoření platformové 3D hry s FPS (First-Person Shooter) prvky pomocí Unity a nástroje Visual Scripting pro operační systém Windows. Výsledkem je prototyp hry, která demonstruje, že hry lze programovat i bez programování v kódu. Hra je pojmenována Jump for Life a cílem hry je překonat překážky v podobě platforem a nepřátel a dosáhnout zelené platformy, která značí vítezství.

Klíčová slova: 3D počítačová hra, FPS, Visual Scripting, Unity

Creating a 3D platform game using Unity Visual Scripting

Abstract

This thesis focuses on the analysis of the functionality of the Unity game engine and its Visual Scripting tool for game development, including the insertion of 3D graphics. In the theoretical part, an analysis of the Unity engine and games of the same genre is performed. The practical part focuses on the creation of a platform 3D game with FPS (First-Person Shooter) elements using Unity and the Visual Scripting tool for the Windows operating system. The result is a prototype of the game that demonstrates that games can be developed without programming in code. The game is named Jump for Life and the aim of the game is to overcome obstacles in the form of platforms and enemies to reach the green platform that marks victory.

Keywords: 3D video game, FPS, Visual Scripting, Unity

Obsah

1 Úvod

Vývoj počítačových her je stále oblíbenější oblastí, která se neustále rozvíjí. Jedním z nástrojů, které pomáhají vývojářům vytvářet hry rychleji a efektivněji jsou herní engine, které poskytují vývojářům základní funkce jako jsou například fyzikální vlastnosti. Jedním z těchto engine je právě Unity.

Tato bakalářská práce se bude zaměřovat na vývoj hry pomocí Unity Visual Scripting. Tento nástroj je stále poměrně málo využíván, neboť není tak dobře znám. Toto je jedna z motivací tvorby této práce. Tento nástroj není dostatečně využíván a není dostatek podkladů pro tvorbu jako při klasickém způsobu vývoje pomocí psaného kódu. V této bakalářské práci bude tento nástroj prozkoumán a otestována jeho efektivita v praxi.

Unity Visual Scripting je vizuální programovací nástroj, který umožňuje vytvářet logiku hry pomocí grafického rozhraní a blokového programování. Tento nástroj je vhodný pro vývojáře s různými úrovněmi programování a může značně zjednodušit proces vytváření her.

V teoretické části bude popsán úvod o hrách a žánrech, následně bude provedena analýza prostředí Unity, jeho komponent a Unity Visual Scripting, které následně bude využito v praxi v praktické části, ve které bude popsán proces tvorby hry a ukázka jednotlivých skriptů použitích pro vytvoření logiky hry. Zároveň bude uvedena ukázka použitých grafických prvků. Na závěr bude provedeno uživatelské testování.

V rámci vývoje této hry jsou využity grafické prvky z Unity Asset Store. Tento online obchod nabízí širokou škálu nejen grafických modelů a textur, které lze využít v různých typech her. Pro účely této bakalářské práce budou využity bezplatné grafické prvky, které jsou k dispozici ve zmíněném obchodě.

Cílem bakalářské práce bude ukázat, že použití Unity Visual Scripting a grafických prvků z Unity Asset Store může být efektivním způsobem, jak vytvořit hru rychleji a s menšími náklady. Výsledkem práce bude funkční hra, která by mohla ukázat potenciál tohoto nástroje a pomoci vývojářům při tvorbě dalších her. Hra bude pojmenována Jump for Life a cílem hry bude překonat překážky v podobě platforem a nepřátel a dosáhnout zelené platformy, která značí vítezství.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem teoretické části práce je analyzovat funkce a možnosti herního engine Unity pro tvorbu her za pomocí nástroje Visual Scripting a představit prostředí, jak programovací, tak prostředí pro vytváření, a vkládání 3D grafiky. V teoretické části zároveň bude provedena analýza her stejného žánru. Po analýze podobných her bude následovat srovnání podobností a rozdílů s hrou vyvíjenou v rámci této bakalářské práce.

Cílem praktické části práce je vytvořit platformovou 3D hru s First-Person Shooter (FPS) prvky v herním engine Unity za pomocí nástroje Visual Scripting. Hra bude primárně vytvořena pro operační systém Windows.

2.2 Metodika

Na základě dostupných materiálů budou v teoretické části popsány nástroje, které využijeme pro tvorbu v praktické části. V praktické části bude popsán návrh a implementace hry. Hra bude tvořena programovacími bloky z Visual Scripting balíčku a doplněna grafikou. Grafika prostředí bude buď vytvořena přímo v modelovacím nástroji Unity anebo stažena z dostupných online zdrojů s volnou licencí.

3 Teoretická východiska

3.1 Hry a herní žánry

Video hry jsou digitální formou interaktivní zábavy, která se stala jedním z nejpopulárnějších způsobů trávení volného času. Tyto hry využívají počítačové grafiky, zvuk a vstupních zařízení, aby umožnily hráčům prožít příběh, plnit úkoly nebo soupeřit s ostatními hráči.

Video hry se dělí do různých žánrů, každý z nich nabízí odlišné herní mechaniky a zážitky. Některé z nejpopulárnějších žánrů videoher jsou akční hry, RPG hry, sportovní hry, strategické hry a simulační hry.

Akční hry se zaměřují na boj a dobrodružství, kde hráči musí projít úrovněmi, bojovat s nepřáteli a plnit úkoly, aby dosáhli vítězství. RPG hry se zaměřují na rozvoj postavy a plnění úkolů v rámci příběhu. Sportovní hry simulují různé sporty, jako jsou fotbal, baseball, hokej nebo basketbal, a umožňují hráčům hrát jako profesionální sportovci nebo týmy. Strategické hry se zaměřují na plánování a taktiku a simulační hry simulují skutečné situace nebo procesy. [1]

Z průzkumů lze pozorovat, že hry jsou široce používány jak pro zábavu, tak pro vzdělávání a trénink. Například, simulační hry mohou být použity k tréninku lékařů nebo pilotů, zatímco edukační hry mohou být použity ke zlepšení matematických nebo jazykových dovedností.

Přestože video hry nabízejí mnoho pozitivních vlastností, jako je rozvoj myšlení a koordinace, mohou také mít negativní dopady, jako je zvýšený sedavý způsob života a sociální izolace. Proto je důležité, aby rodiče a vývojáři spolupracovali na zajištění zdravé rovnováhy mezi hraním her a fyzickou aktivitou, stejně jako sociální interakcí. V odborné literatuře lze nalézt řadu studií, které se zabývají tématem videoher, od analýzy žánrů a herních mechanik po dopady na lidskou psychologii a chování. [2]

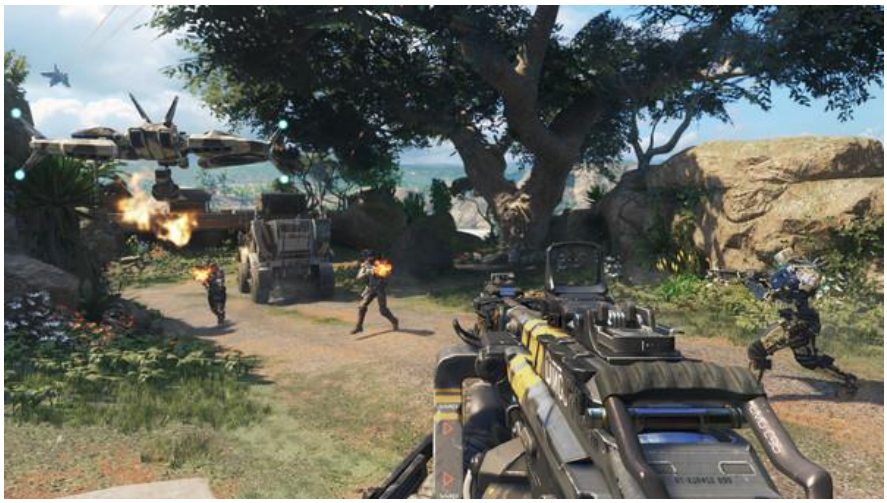
3.1.1 Použité žánry v bakalářské práci

Tato bakalářská práce je zaměřena na tvorbu hry, která kombinuje prvky ze dvou populárních herních žánrů, platformových a FPS (First-Person Shooter).

Platformové hry jsou jedním z nejstarších a nejznámějších herních žánrů. Tyto hry se skládají ze série platforem, na kterých se hráč snaží projít přes různé úrovně plné překážek, nepřátel a pokladů. Hlavním cílem platformových her je dostat se na konec úrovně, kde čeká konečný cíl. Platformové hry jsou často charakterizovány jednoduchým ovládáním, které umožňují hráči snadno ovládat svou postavu, a jsou obvykle zaměřené na akci a rychlost. Některé z nejznámějších platformových her jsou Super Mario Bros, Sonic the Hedgehog a Mega Man.

Obrázek 1 - Ukázka ze hry Mega Man [10]

FPS (First-Person Shooter) je herní žánr, který se zaměřuje na akci a střílení kde hráči se dívají na svět hry prostřednictvím očí své postavy. Tyto hry se často odehrávají ve futuristických nebo sci-fi světech a hráči bojují proti nepřátelům, kteří se objevují v různých úrovních. FPS hry jsou známy svou silnou vizuální a zvukovou stránkou a pro svou realistickou grafiku. Některé z nejznámějších FPS her jsou DOOM, Half-Life a Call of Duty.



Obrázek 2 - Call of Duty: Black Ops III [11]

3.2 Porovnání her

Hra vyvíjena v rámci této bakalářské práce je kombinací dvou žánrů, a to her platformových a FPS (First-Person Shooter).

Hra A Story About My Uncle je platformová adventura, která se zaměřuje na poutavý příběh a atmosféru. Na rozdíl od hry vyvíjené v rámci této bakalářské práci, bude mít prvky First-Person Shooter (FPS) a bude se soustředit na bojové scény. Zatímco hra A Story About My Uncle se zaměřuje na hledání a řešení hádanek, hra vyvíjená v rámci této práce bude mít více akční prvků.

Na rozdíl od hry Call of Duty tato hra neobsahuje složité a propracované bojové scény a taktické plánování a spíše kombinuje jednoduché prvky střelby se skákáním na platformách a překonávání překážek. Tato hra má také jednodušší grafiku a ovládání, což ji činí více přístupnou pro běžné a příležitostné hráče.

3.3 Tvorba her

Vytváření počítačových her je náročný proces, zahrnující roky programování, tvorbu grafických, zvukových a hudebních prvků a rozsáhlé testování. I přesto, dnes by si téměř každý chtěl vytvořit vlastní hru. Pro mnohé hráče je to natolik přitažlivá idea, že jsou ochotni zaplatit třetí straně anebo se pokusit o vývoj sami.

Vývoj her vyžaduje pečlivé plánování a použití určitých metod a pravidel. Existuje několik ucelených postupů a metod, kterých se vývojáři her často drží. Jedním z nich je tzv. vodopádový model. Tento model se skládá z několika fází, jako jsou analýza požadavků, návrh,

implementace a testování. Každá fáze musí být dokončena před tím, než se začne pracovat na další fázi. Tento postup je ucelený a může pomoci při efektivním plánování a organizaci vývoje her. [1]

Další ucelenou metodou pro vývoj her je tzv. agilní vývoj. Tato metoda klade důraz na průběžné testování a iterativní vývoj. Vývojáři pracují v krátkých cyklech a v každém cyklu se zaměřují na dokončení určitého úkolu. Tato metoda může být vhodná pro menší týmy a projekty, které potřebují rychle reagovat na změny a přizpůsobit se požadavkům zákazníka. [1]

Při vývoji her je také důležité dodržovat určitá pravidla a zásady. Jedno z takových pravidel je tzv. KISS (Keep it Simple, Stupid) pravidlo, které se snaží minimalizovat složitost hry a udržet ji co nejjednodušší. Další je tzv. 80/20 pravidlo, které říká, že 80 % výsledků pochází z 20 % práce. To znamená, že vývojáři by se měli zaměřit na nejdůležitější prvky hry a ty nejdůležitější problémy, které musí být vyřešeny. [1]

Jak již bylo zmíněno, tvorba her je náročná činnost, a proto existuje celá řada nástrojů, které celý proces výrazně usnadňují. Jedním z těchto nástrojů jsou právě herní engine.

3.3.1 Herní engine

Herní engine je prostředí pro vývoj softwaru, označované také jako „herní framework“, s nastaveními a konfiguracemi, které optimalizují a zjednodušují vývoj (nejen) videoher v různých programovacích jazycích. Herní engine může zahrnovat 2D nebo 3D grafický render engine, který je kompatibilní s různými formáty importu (souborů z grafických softwarů), fyzikální engine, který simuluje činnosti v reálném světě, detektor kolizí, umělou inteligenci (AI), která automaticky reaguje na akce hráče, zvukový engine, který ovládá zvukové efekty, animační engine a řadu dalších funkcí. [3]

3.3.2 Příklady herních engine

Existuje celá řada druhů herních engine, každý vyniká v něčem jiném, tedy je velice důležité zvážit potřeby a cílový stav hry před začátkem tvorby a zvolit nejvhodnější herní engine. Ku příkladu jsou uvedeny tři z nepopulárnějších herních engine.

3.3.2.1 Godot

Godot je moderní herní engine, který poskytuje všechny funkce, které jsou k vývoji her potřebné. Jeho velkou předností je jednoduchá tvorba 2D her. Je také zcela zdarma a open source, vydaný pod licencí MIT. Godot není vyvíjen žádnou společností jako komerční produkt. Namísto toho věnuje svůj čas a odborné znalosti budování engine, testování a opravování chyb, tvorbě dokumentace a dalším činnostem komunita nadšených vývojářů. [4] Godot používá svůj vlastní skriptovací jazyk GDScript, ale je také možné použít C#.

3.3.2.2 Unreal Engine

Unreal Engine herní engine, který byl vytvořen společností Epic Games. Je to multiplatformní engine, který lze použít pro vývoj her pro různé platformy, jako jsou PC, konzole nebo mobilní zařízení. Unreal Engine nabízí širokou škálu nástrojů pro vývoj her, jako jsou například editor scén, nástroje pro animaci a fyziku, nástroje pro tvorbu sítí a mnoho dalších. Tuto technologii lze použít jak pro tvorbu her, tak také například pro tvorbu filmů. Nutné podotknout že Unreal Engine je nejlepší engine pro vývoj AAA her (her nejvyšší kvality). Unreal Engine používá k vývoji jazyk C++. [5]

3.3.2.3 Unity

Unity je multiplatformní herní engine. Je oblíbený zejména pro vývoj mobilních her pro systémy iOS a Android a je považován za snadno použitelný pro začínající vývojáře a oblíbený pro vývoj nezávislých her. Unity lze použít k vytváření 3D a 2D her, stejně jako interaktivních simulací a nebo i virtuální realitu (VR). Unity engine si oblíbila i jiná odvětví než jen videoherní průmysl, například filmové, automobilové, architektonické, strojírenské a stavební odvětví, a dokonce i ozbrojené síly Spojených států. [6]

3.3.3 Visual scripting

Visual scripting je metoda pro tvorbu herní logiky, která umožňuje vývojářům tvořit skripty pomocí grafických bloků místo kódu. Tento přístup je vhodný pro vývojáře, kteří nejsou znalí programovacího jazyka nebo pro rychlé prototypování herních nápadů.
Tento nástroj je často k dispozici jako rozšíření nebo zabudovaný přímo do herních engine, jako je Unity nebo Unreal Engine. [7]

3.3.4 Porovnání Unity a Unreal Engine

Unity i Unreal Engine nabízejí podporu pro visual scripting, což umožňuje vývojářům tvořit herní logiku pomocí grafických bloků místo kódu. Oba engine poskytují širokou škálu nástrojů pro vývoj her a umožňují vytvořit hry pro různé platformy.

Unity podporuje visual scripting za pomocí své služby Unity Visual Scripting, která byla zabudována přímo do Unity od roku 2021. Dříve bylo možné využít externí službu Bolt. Tato funkčnost poskytuje intuitivní grafické rozhraní a je snadno použitelná pro začátečníky. Unity je také cenově dostupný a nabízí širokou komunitu, což umožňuje snadný přístup k návodům a příkladům.

Unreal Engine nabízí Blueprints, což je systém visual scripting vytvořený přímo v engine. Blueprints je flexibilní a snadno použitelný a poskytuje vysokou míru kontroly nad herní logikou. Unreal Engine také nabízí vysokou úroveň grafického zpracování a efektů, což je vhodné pro vytváření realistických her nebo projektů v oblasti architektury, designu, vzdělávání nebo pro virtuální a rozšířenou realitu.

Oba Unity a Unreal Engine jsou dobré volby pro vývoj her s využitím visual scripting, neboť oba engine nabízejí přehledné uživatelské prostředí a rozsáhlou komunitu. Záleží na konkrétních potřebách a preference vývojáře, který engine bude vyhovovat lépe.

V této bakalářské práci bylo zvoleno Unity a jeho Unity Visual Scripting z důvodů lepší uživatelské dokumentace a stabilnějšího prostředí. Zároveň Unity je vhodnější pro méně zkušené uživatele.

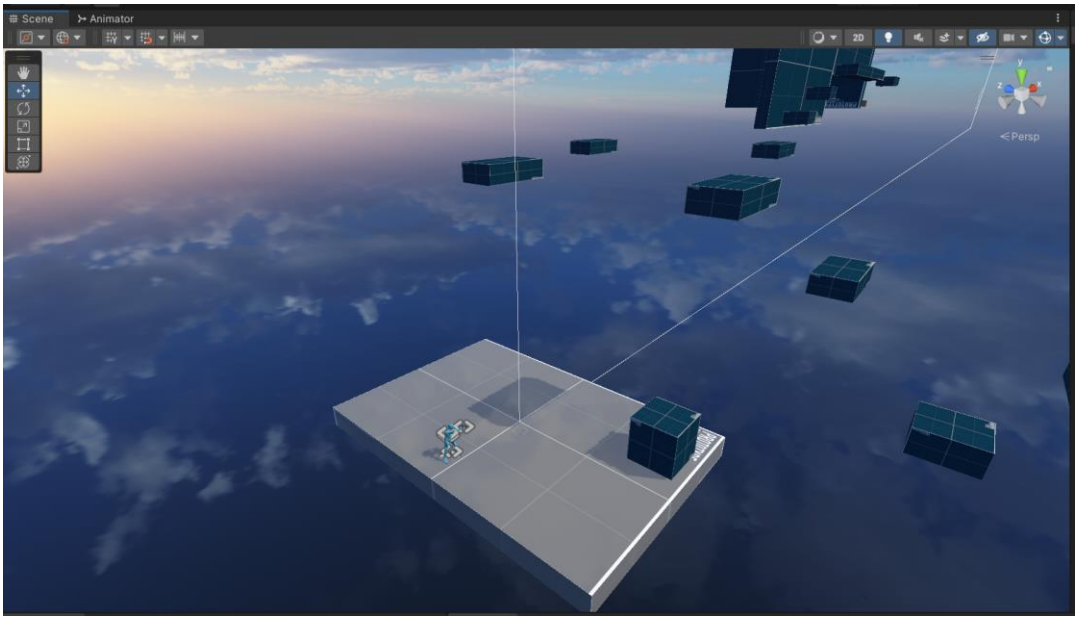
3.4 Unity

Unit poskytuje k tvorbě her své rozhraní a velké množství nástrojů. Je potřeba znát co k čemu slouží. Popsány budou pouze ty prvky, které byly použity při tvorbě hry k této bakalářské práci.

3.4.1.1 Scene View

Scéna v Unity je hlavním místem, kde se odehrává tvorba hry. Může obsahovat různé objekty jako jsou modely, světla, kamery, UI atd. Scénu lze tvořit pomocí Unity editoru, kde lze

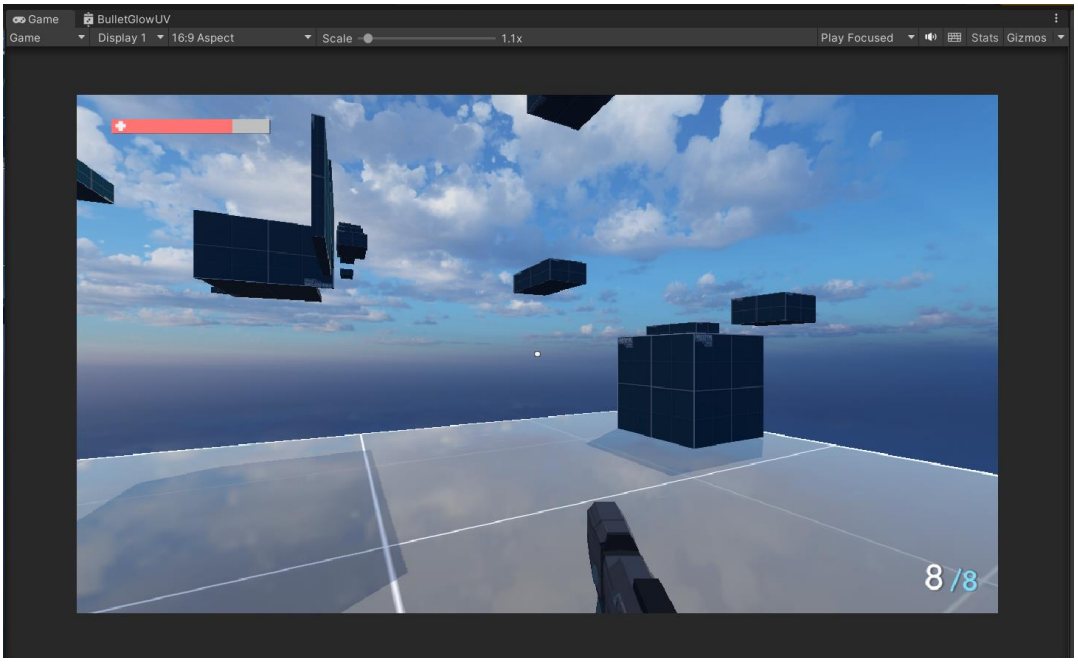
přidávat, upravovat a odebírat objekty. Scéna se skládá z objektů, které jsou organizovány do hierarchie. Scénu lze exportovat a importovat do jiných projektů. [8]



Obrázek 3 - Scene View

3.4.1.2 Game View

Game View je jedno z hlavních rozhraní v Unity, které umožňuje vývojářům zobrazit a testovat svou hru v reálném čase. Game View se automaticky aktualizuje při každé změně v projektu a umožňuje vývojářům sledovat a ladit herní mechaniku a vzhled. [8]

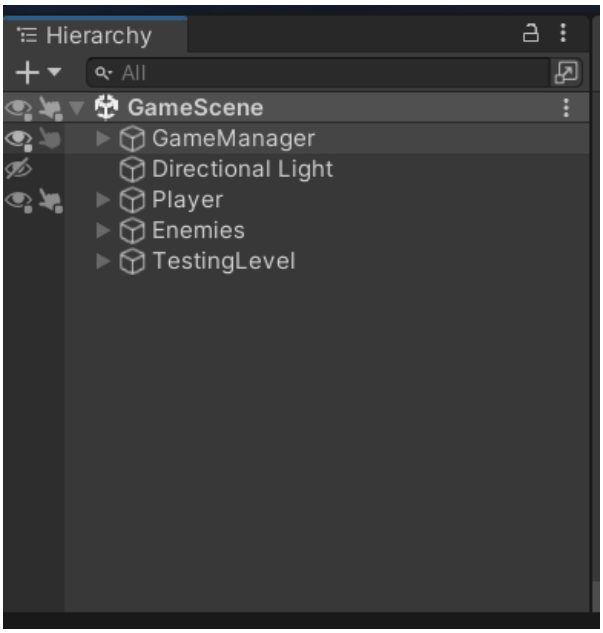


Obrázek 4 - Game View

Scene View a Game View jsou oba důležitými rozhraními v Unity, která slouží k různým účelům. Scene View umožňuje vývojářům procházet a upravovat scénu, zatímco Game View umožňuje zobrazit a testovat hru v reálném čase. Zatímco Scene View poskytuje přehled o celé scéně a umožňuje vývojářům pracovat s jednotlivými objekty, Game View poskytuje výhled na hru tak, jak by se zobrazila hráči. Obě rozhraní jsou důležité pro úspěšný vývoj hry a spolupracují při tvorbě a ladění herních mechanik.

3.4.1.3 Hierarchy

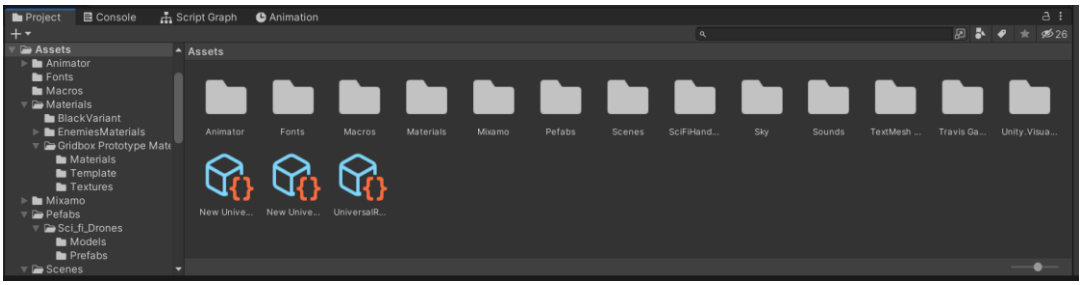
Hierarchie v Unity umožňuje organizovat objekty (GameObject) ve scéně. Každý objekt v hierarchii má svého rodiče (parent) a může mít i potomky (child). Tato struktura umožňuje snadnou orientaci ve scéně a udržování přehlednosti. Objekty v hierarchii lze také snadno přesouvat, seskupovat a upravovat. Je to jedna z nejdůležitějších součástí Unity pro správu objektů ve scéně. Je důležité, aby byly objekty v hierarchii organizovány logicky a srozumitelně pro snadnější práci s nimi. [8]



Obrázek 5 – Hierarchy

3.4.1.4 Project

Projekt v Unity obsahuje všechny soubory a složky (assets) potřebné pro vytvoření hry. Tyto soubory zahrnují 3D modely, textury, zvuky, skripty a další soubory potřebné pro vytvoření hry. Project panel umožňuje snadno přidávat a odebírat soubory z projektu a organizovat je do složek. [8]



Obrázek 6 – Project files a assets

3.4.1.5 Assets

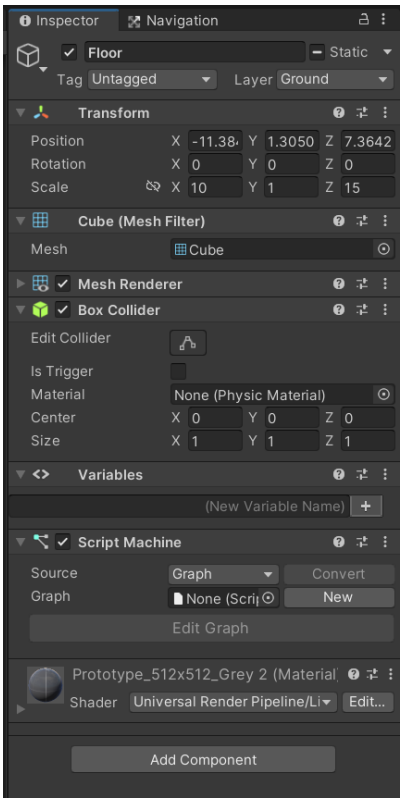
Assets jsou soubory nebo složky, které jsou uložené ve vytvořeném projektu Unity. Tyto soubory mohou obsahovat modely, textury, zvuky, skripty, animace atd. Assets jsou používány k tvorbě her a aplikací v Unity. Soubory mohou být importovány nebo vytvořeny přímo v Unity a pak jsou uložené ve složce Assets. Tyto soubory mohou být také sdíleny mezi různými scénami. Assets jsou důležitým prvkem pro tvorbu her a aplikací v Unity, protože poskytují všechny potřebné prvky pro vytvoření herního světa nebo aplikace.

3.4.1.6 GameObjects

GameObjects jsou základními stavebními kameny v Unity. Jsou to objekty, které jsou umístěny v hierarchii scény a mohou obsahovat různé komponenty, jako jsou například modely, světla, fyzikální tělesa nebo skripty. GameObject může také obsahovat další GameObject jako potomky, což umožňuje vytvářet složité struktury v hierarchii. GameObjects se také používají pro označování objektů ve scéně, což umožňuje snadnou identifikaci a manipulaci s nimi pomocí skriptů. [8]

3.4.1.7 Inspector Window

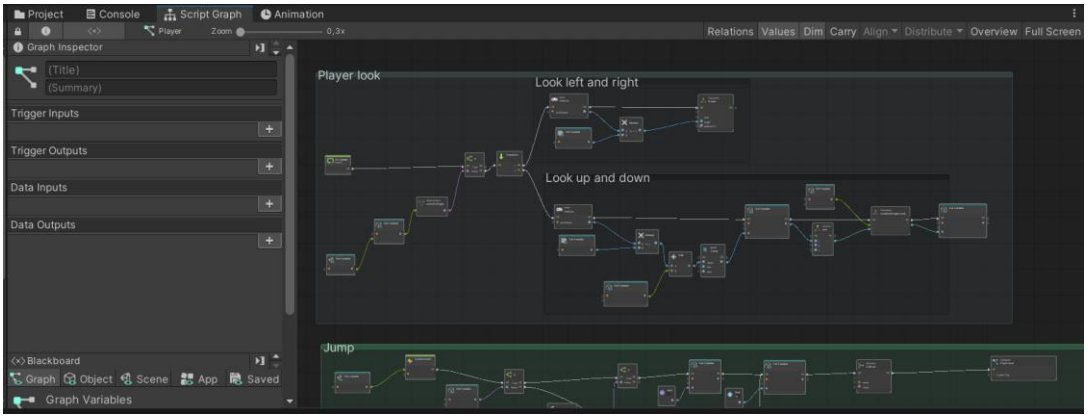
Inspector je jeden z hlavních prvků v Unity, které slouží k úpravě vlastností objektů ve scéně. V okně Inspector lze upravovat různé vlastnosti objektu jako jsou například transformace (pozice, rotace, velikost), materiály, komponenty nebo skripty.
Dále lze v okně Inspector nastavit komponenty jako jsou například komponenty pro fyziku, zvuky, světlo nebo animace. V okně Inspector lze také nastavovat hodnoty proměnných v připojených skriptech. Všechny změny, které provedeme v okně Inspector jsou okamžitě aplikovány na objekt ve scéně. Je vhodné používat Inspector pro detailní úpravu objektů a pro ladění herních mechanik. [8]



Obrázek 7 – Inspector

3.4.1.8 Visual Scripting Graph

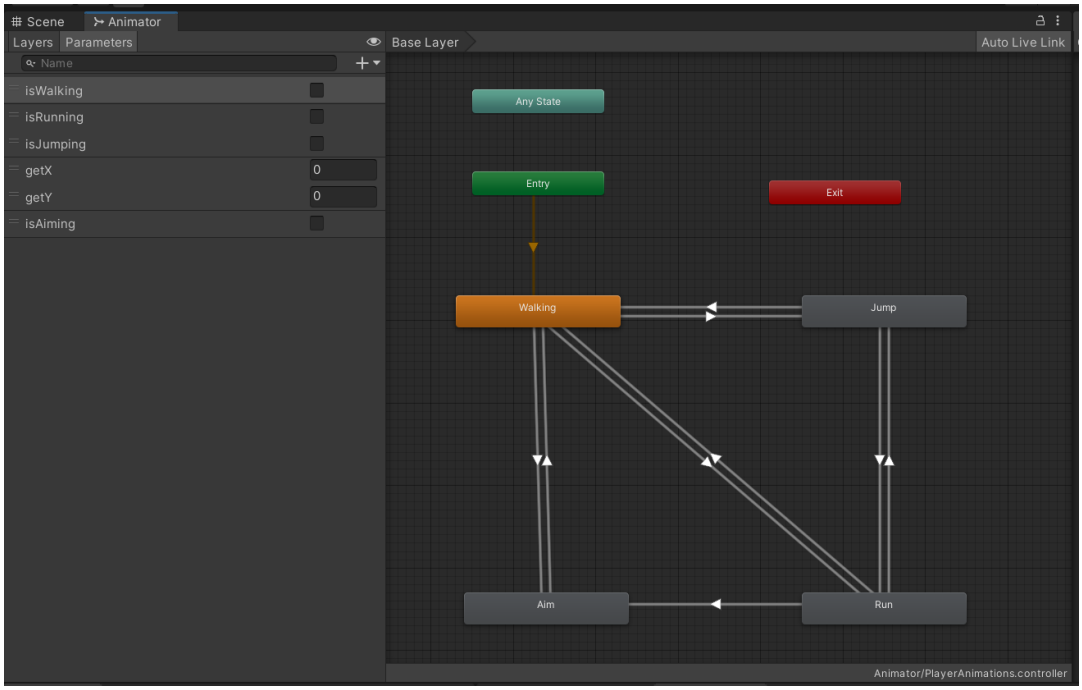
Visual Scripting Graph je rozhraní, které se používá k editaci a tvorbě scriptů v Unity pomocí grafického rozhraní. Tento editor umožňuje vývojářům vytvářet skripty pomocí jednoduchých grafických prvků nazývané uzly, které se propojují, což umožňuje rychlejší a snazší tvorbu skriptů. Editor obsahuje nástroje pro tvorbu událostí, funkcí a proměnných, stejně jako možnosti pro ladění a testování skriptů.



Obrázek 8 - Visual Scripting Graph

3.4.1.9 Animator

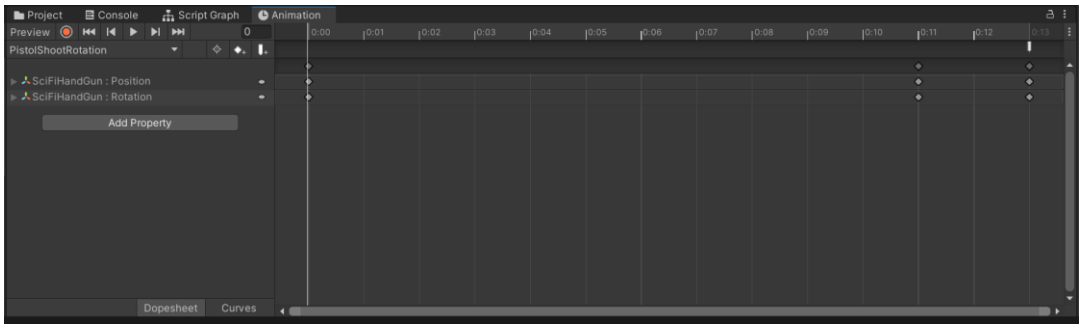
Animátor je nástroj pro tvorbu animací v Unity. Umožňuje spojit jednotlivé animační klipy do animačního stromu a řídit je pomocí parametrů a přechodů. Lze také použít pro animaci kosterních meshů a blendování mezi animacemi. [8]



Obrázek 9 – Animator

3.4.1.10 Animation

Animation v Unity umožňuje animovat objekty vložené do Unity. Lze animovat pozici, rotaci a velikost objektů, stejně tak i jiné vlastnosti jako například barvu nebo intenzitu světla. Animace se mohou tvořit pomocí nástrojů jako jsou animační klipy nebo křivky. Animační klipy jsou jednotlivé sekvence animace, které lze přiřadit k objektu. Tyto klipy se mohou skládat z jednotlivých animačních křivek, které se používají k animaci různých vlastností objektu, jako je pozice, rotace nebo velikost. [8]

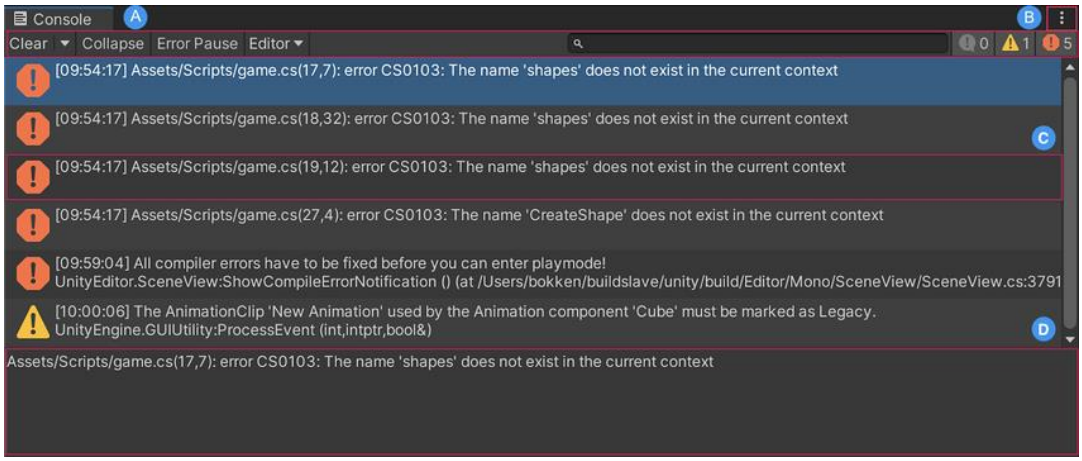


Obrázek 10 – Animation

Pozor, neplést Animator a Animation. Animator slouží k řízení a spravování animací pomocí animačních stavů a přechodů mezi nimi. Animation zase slouží k přímé animaci jednotlivých vlastností objektu.

3.4.1.11 Console

V okně Console se zobrazují chyby, varování a další zprávy, které se mohou vyskytnout při vytváření hry či aplikace. Tyto chyby a varování pomáhají najít problémy v projektu, například chyby při kompilaci skriptů. Upozorňují také na akce, které Unity provedlo automaticky, například nahrazení chybějících meta souborů, což by mohlo způsobit problém někde jinde v projektu. [8]



Obrázek 11 – Console [12]

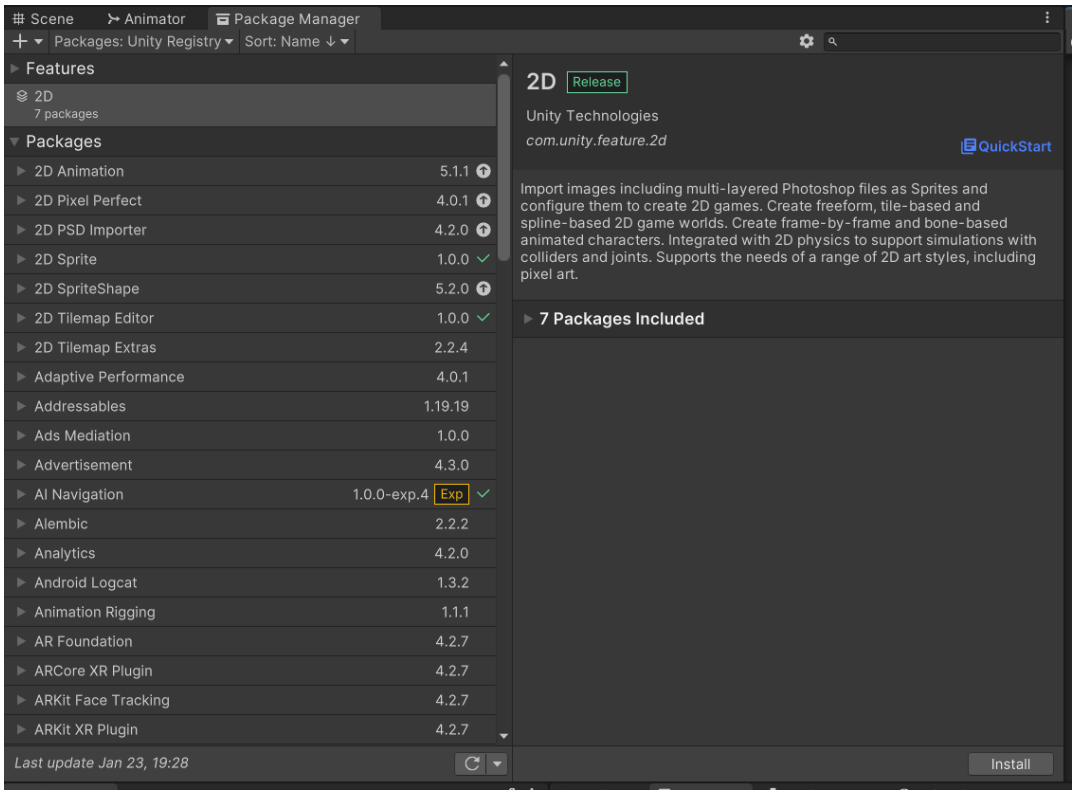
3.4.1.12 Asset Store

Asset Store je online obchod, který je součástí Unity rozhraní. Umožňuje vývojářům stahovat a nakupovat různé prvky, jako jsou modely, textury, efekty, skripty a další nástroje potřebné pro tvorbu her. Tyto prvky mohou být vytvořené komunitou nebo profesionálními

vývojáři s dlouholetou zkušeností a jsou k dispozici za poplatek nebo zdarma. Asset Store také nabízí možnost pro vývojáře prodávat své výtvory.

3.4.1.13 Package Manager

Package Manager je další nástroj, který je součástí Unity rozhraní. Umožňuje vývojářům instalovat a spravovat různé balíčky, jako jsou knihovny, nástroje a další rozšíření například prvky zakoupené v Unity Asset Store. Package Manager umožňuje vývojářům snadno instalovat, aktualizovat a odinstalovat balíčky bez nutnosti ručního stahování a nastavování.



Obrázek 12 - Package Manager

3.5 Unity Visual Scripting

Unity Visual Scripting je grafické rozhraní pro tvorbu skriptů v Unity. Umožňuje vývojářům vytvářet skripty pomocí grafického rozhraní, aniž by museli psát kód. Unity Visual Scripting umožňuje vývojářům vytvořit skripty rychleji a snadněji než při použití tradičních programovacích jazyků. Je to dobré pro vývojáře, kteří nejsou zkušení v programování nebo pro ty, kteří chtějí urychlit vývoj svých projektů. [9]

3.5.1 Nodes

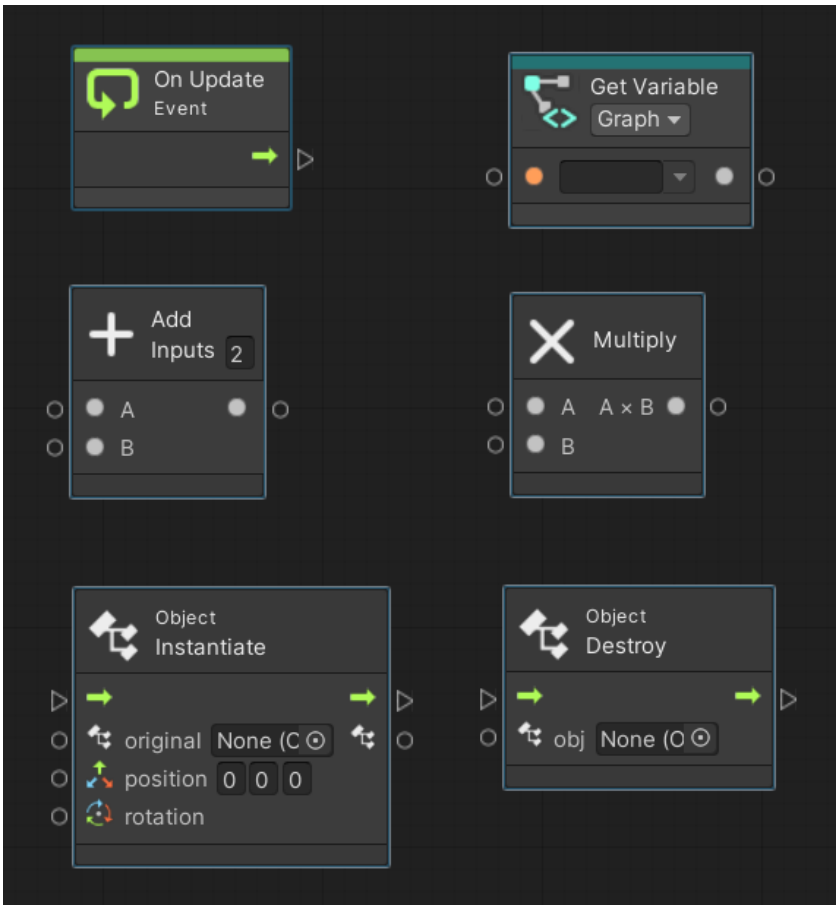
Nodes (uzly) jsou základní stavební jednotky Unity Visual Scripting. Jsou to jednotlivé komponenty, které slouží k definování akcí a hodnot v Script Graphs a State Graphs. Každý Node má své vstupy a výstupy, které jsou spojovány s ostatními Nody pomocí hran (edge) a tím se definuje logika aplikace. [9]

Nodes mohou sloužit k různým účelům, například k vytvoření události, změny hodnoty, spuštění animace, vytvoření proměnné apod. Tyto Nodes jsou dostupné přes Fuzzy vyhledávač a je možné je přidávat do Graph, upravovat je a mazat je.

Existují různé typy Nodes, jako například Event Nodes, Action Nodes, Flow Control Nodes, Operator Nodes a další. Každý typ Node má své specifické vlastnosti a slouží k specifickým účelům.

Například Event Node slouží k vytvoření události, která může být spojena s ostatními uzly. Může být například On Update, který vytvoří událost při každém updatu hry, nebo On Collision, který vytvoří událost při kolizi s jiným objektem. Dále existují uzly pro práci s proměnnými, jako je Set Variable nebo Get Variable, které slouží k nastavení nebo získání hodnoty proměnné. Pro práci s matematickými výrazy existují uzly jako Add nebo Multiply. Pro práci s GameObjects jsou k dispozici uzly jako Instantiate nebo Destroy. Tyto jsou jen některé z mnoha typů uzlů, které jsou k dispozici v Unity Visual Scripting.

Je důležité si uvědomit, že každý uzel nabízí různé vstupy a výstupy, které lze propojit s ostatními uzly, aby se vytvořil požadovaný efekt.



Obrázek 13 - Ukázky uzlů v Unity Visual Scripting

3.5.2 Grafy (Skripty)

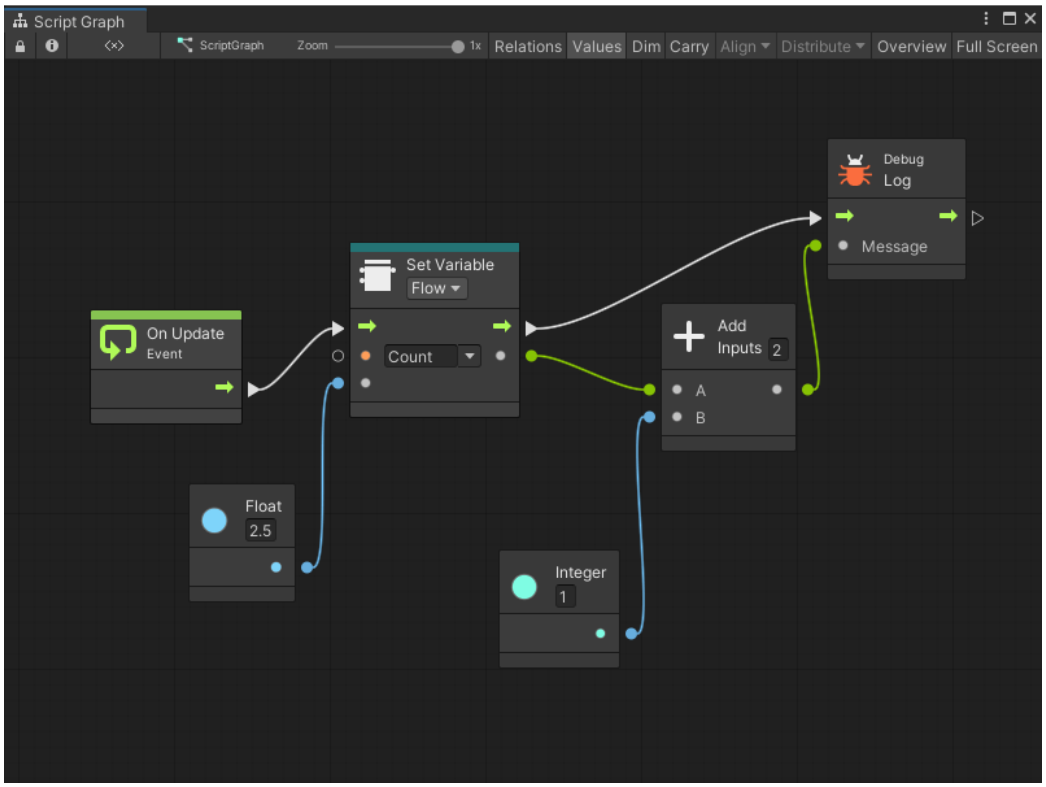
Graf je nástroj Visual Scripting, který zobrazuje grafickou reprezentaci logiky skriptu. Visual Scripting obsahuje dva různé typy grafů – Script Graph a State Graph. Oba lze použít k definování chování GameObject. Pro využití Script Graph a State Graph v projektu, musejí být vloženy do Script Machine nebo State Machine. [9]

3.5.2.1 Script Graph

Script Graph řídí a propojuje konkrétní akce a hodnoty tedy Script Graph definuje specifické chování GameObject při běhu aplikace. Akce v Script Graph probíhají v určitém pořadí. Tyto akce mohou nastat při každém snímku nebo při konkrétní události.

Visual Scripting reprezentuje akce v grafu skriptů prostřednictvím uzlů. Propojením uzlů pomocí hran (edge) je aplikaci sděleno, co má dělat a v jakém pořadí. Script Graph může

přistupovat ke kolekci uzlů, které odpovídají různým funkcím a vlastnostem v editoru Unity. Přístup k těmto uzlům je zprostředkováno pomocí Fuzzy vyhledávače. [9]



Obrázek 14 - Ukázka Script Graph [13]

3.5.2.2 State Graph

State Graph v Unity je druhým typem grafů k dispozici v Unity Visual Scripting. Jsou určeny pro řízení stavů a chování objektů ve hře. State Graph se skládá z uzlů, které reprezentují jednotlivé stavy a přechody mezi nimi.

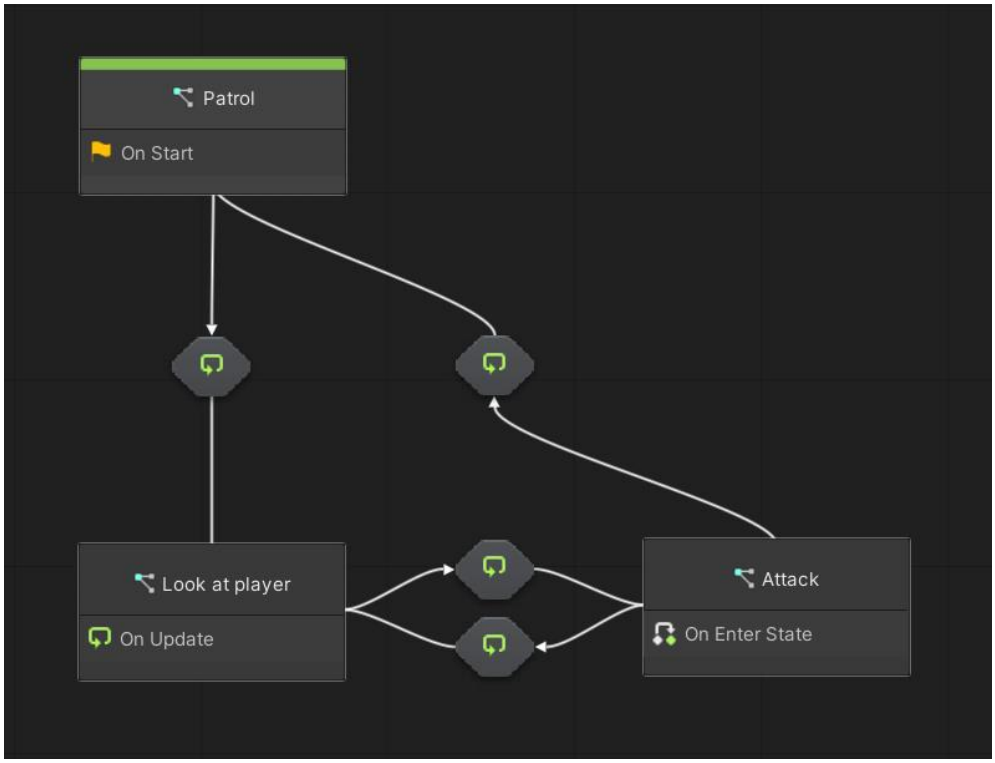
State Graph se používá k řízení stavů objektů, například pro animace postav, ovládání hráče nebo chování nepřátel. Každý stav má své vlastní vstupy a výstupy, které umožňují reagovat na události ve hře a přecházet mezi stavy. [9]

State Graph se skládá z uzlů, které reprezentují jednotlivé stavy a přechody mezi nimi. Tyto uzly lze propojit spojkami tzv. přechody, které řídí, kdy se stavy spouštějí a jak se mezi nimi přechází.

Příkladem může být nepřítel, který má stavy Hlídka a Pronásledování. Akce nepřítele se mohou změnit ze stavu Hlídka na stav Pronásledování, jakmile nepřítel detekuje hráče.

Událost detekce pro nepřítele spustí přechod mezi těmito dvěma stavy. Jednotlivé akce jsou definovány ve Script Graph.

State Graph nabízí flexibilitu a snadnost použití při řízení stavů a chování objektů ve hře. Grafické rozhraní umožňuje vidět celkovou strukturu stavů a přechodů, což umožňuje snadnou úpravu a ladění. State Graph je vhodný pro vývojáře, kteří chtějí vytvořit složitější a sofistikovanější stavové systémy pro své hry. [9]

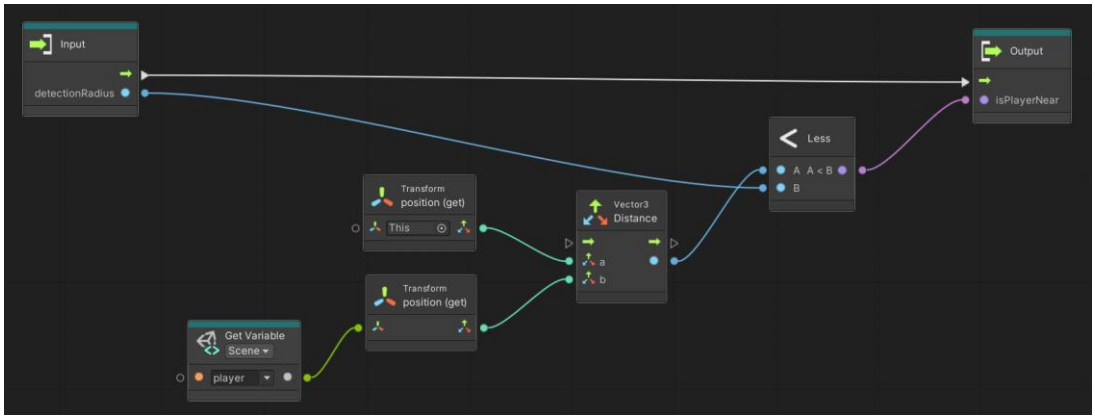


Obrázek 15 - Ukázka State Graph

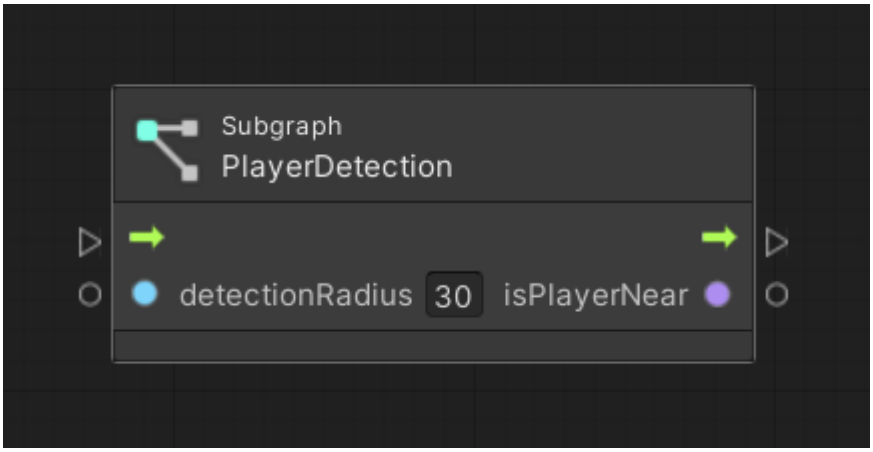
3.5.2.3 Subgraph

Subgraph je část grafu, která může být použita jako samostatný celek v rámci jiného grafu. Tyto podgrafy jsou uživatelsky definované skupiny uzlů, které mohou být použity jako jedna funkce nebo část kódu. Stejně tak obsahuje vstupní a výstupní porty pro komunikaci s hlavním grafem. To umožňuje organizovat kód do logických skupin a také umožňuje opakované použití stejné logiky v různých částech grafu. Subgraph může být také použit pro sdílení logiky mezi různými Script Graphs nebo State Graphs. [9]

Příkladem by mohlo být vytvoření podgrafu pro ověření, zda hráč dosáhl konečného cíle v hře a poté tento podgraf použít ve více Script Graphs nebo State Graphs pro různé konečné cíle v rámci hry.



Obrázek 16 - Ukázka zápisu Subgraph



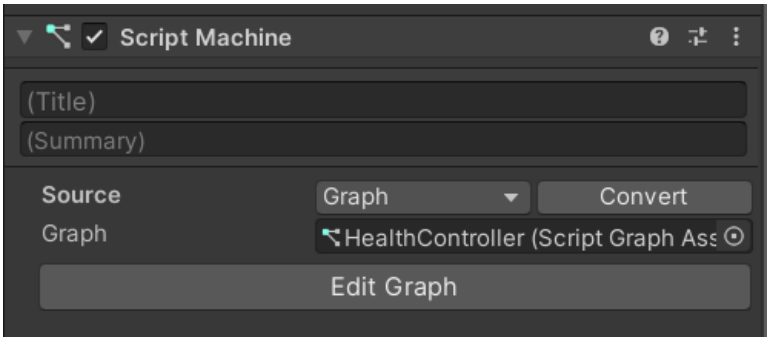
Obrázek 17 - Ukázka použití Subgraph

3.5.3 Script Machines a State Machines

Aby bylo možné vložit skripty k jednotlivým objektům (GameObject), je potřeba přidat komponentu, která umožnuje vytvořit logiku. K tomuto slouží tyto dvě komponenty Script Machine a State Machine. Obě tyto komponenty umožnují přidat logiku pomocí Visual Scripting.

3.5.3.1 Script Machines

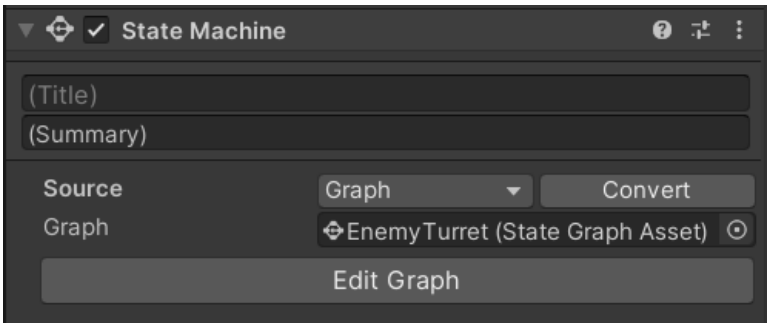
Script Machine je komponenta, která umožňuje vložení klasického skriptu k vytvoření logiky pomocí grafického rozhraní.



Obrázek 18 - Script Machine

3.5.3.2 State Machines

State Machine je komponenta, která umožňuje vytvořit různé stavy, které mohou obsahovat různé funkce a události, a definovat přechody mezi nimi. Je možné ji nastavit pomocí grafického rozhraní. State Machine komponenta umožňuje snadnou a intuitivní tvorbu složitých stavových systémů.



Obrázek 19 - State Machine

3.5.4 Typy zdrojů skriptů

V Unity Visual Scripting existují dva typy způsobů zápisu skriptů – Embed a Graph. Oba typy slouží ke stejnému účelu, pouze každý z nich funguje trochu jinak a nabízí jiná využití. Oba tyto typy vytvoří prostředí, ve kterém se tvoří struktura skriptů.

3.5.4.1 Typ Embed

Typ Embed vytvoří prostředí pro tvorbu skriptů přímo v daném objektu, ve kterém byla komponenta vložena. Pokud by se objekt smazal, smaže se i daný skript. Vhodné pro jednodušší skripty

3.5.4.2 Typ Graph

Typ Graph podobně jako typ Embed vytvoří prostředí, s rozdílem, že Graph vytvoří soubor, ve kterém je skript uložen. Tedy tento soubor je možné použít na více objektech a skript není smazán s objektem ke kterému byl přiřazen. Je vhodný pro větší a složitější skripty. Mezi hlavní rozdíl patří, že Embed je jednodušší a intuitivnější na použití, zatímco Graph nabízí více možností pro složitější logiku a navazování událostí. [9]

3.5.5 Proměnné

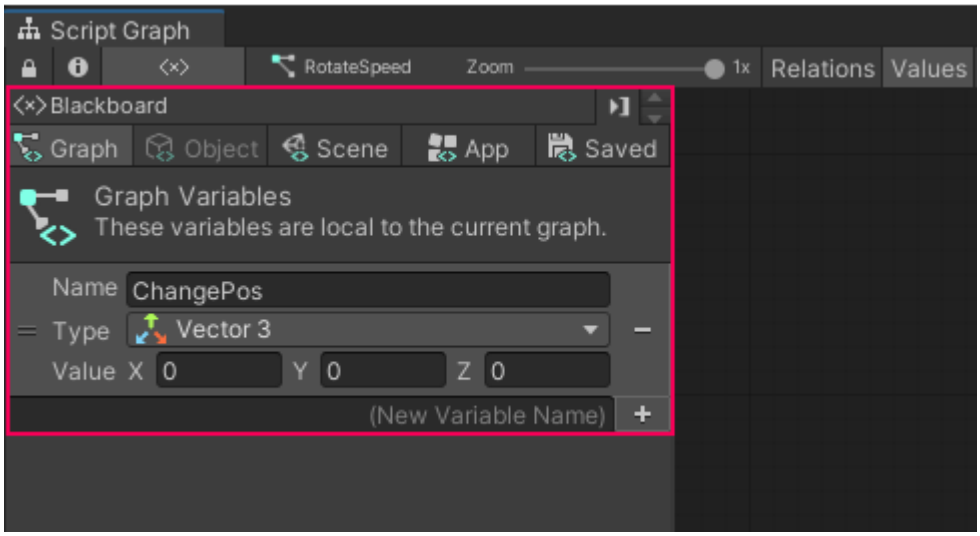
Proměnné ve Visual Scripting slouží jako sběrnice informací. Tyto informace se mohou měnit a používat v grafech a jejich uzlech.

Proměnné v Unity Visual Scripting se mohou lišit v typu. Tyto typy mohou být číslo (int/float), text (string), bool (pravda/nepravda), vektor a barva, stejně jako různé typy objektů, jako jsou například GameObject, Transform, Material a další. Tyto typy proměnných určují, jaký typ informací mohou obsahovat a jaké operace lze s nimi provádět.

V rámci grafů lze nastavit hodnoty proměnných a používat je v dalších uzlech. Je také možné vytvářet a modifikovat proměnné pomocí příslušných uzlů, jako jsou uzly pro nastavení hodnoty, matematické operace nebo textové operace.

Proměnné jsou zásadní pro funkčnost aplikace, a tím i pro funkčnost Visual Scripting. Pomáhají přenášet informace mezi jednotlivými částmi grafu a umožňují řídit chování aplikace.

Proměnné v grafu lze vytvářet a spravovat pomocí tzv. Blackboard. Blackboard má grafické rozhraní, které umožňuje jednoduše přidávat, upravovat a spravovat proměnné. To umožňuje snadné a přehledné spravování dat bez nutnosti psaní a udržování kódu. [9]



Obrázek 20 - The Blackboard, nástroj pro správu proměnných [14]

Proměnné mají také oblast působnosti (scope). Oblast působnosti proměnné určuje, které části Script Graph mohou přistupovat ke kterým proměnným a číst nebo měnit jejich hodnoty. Oblast působnosti může také rozhodovat o tom, zda k proměnné může přistupovat jiný Script Graph. [9]

3.5.5.1 Graph Variables

Graph Variables jsou proměnné, které se specifikují v rámci Script Graph. Tyto proměnné lze použít pro sdílení informací mezi uzly v rámci stejného grafu, a nelze je měnit z jiných grafů nebo z jiných součástí aplikace. [9]

3.5.5.2 Object Variables

Object Variables jsou proměnné, které jsou přidány na GameObject. Tyto proměnné se používají pro sdílení informací mezi uzly, které jsou spojeny s daným GameObject. Toto je nejběžnější kategorie proměnných. [9]

3.5.5.3 Scene Variables

Scene Variables jsou proměnné, které se specifikují na úrovni scény. Tyto proměnné lze použít pro sdílení informací mezi uzly, které jsou v rámci stejné scény, a nelze je měnit z jiných scén nebo z jiných součástí aplikace. [9]

3.5.5.4 App Variables

App Variables jsou proměnné, které se specifikují na úrovni celé aplikace. Tyto proměnné lze použít pro sdílení informací mezi všemi scénami a grafy v rámci aplikace.

Veškeré hodnoty uložené v App Variables se po ukončení aplikace vrátí na výchozí hodnoty. [9]

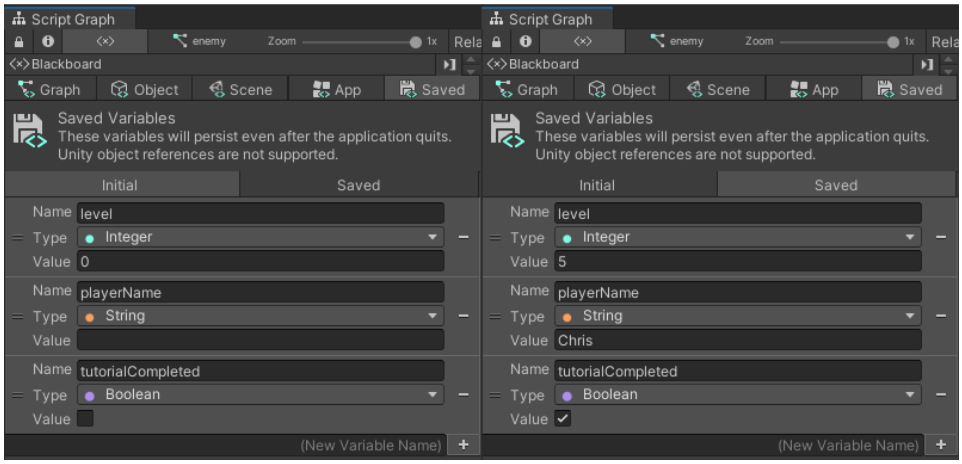
3.5.5.5 Saved Variables

Saved Variables jsou proměnné, které se ukládají v mezi paměti aplikace, aby mohly být použity i po restartu aplikace. Tyto proměnné se používají pro uložení stavu hry nebo nastavení aplikace. Například pro uložení hodnoty hlasitosti hudby.

Saved Variables mají v Blackboard dvě další záložky: Initial (Počáteční) a Saved (Uložené).

Hodnoty definované v záložce Initial se použijí pro všechny nové instance aplikace jako výchozí hodnoty.

Hodnoty definované na kartě Saved jsou poslední změněné hodnoty těchto proměnných podle toho, kdy byla aplikace naposledy spuštěna. Je možné je manuálně upravit nebo hodnoty odstranit a vrátit je tak na hodnoty definované v záložce Initial. [9]



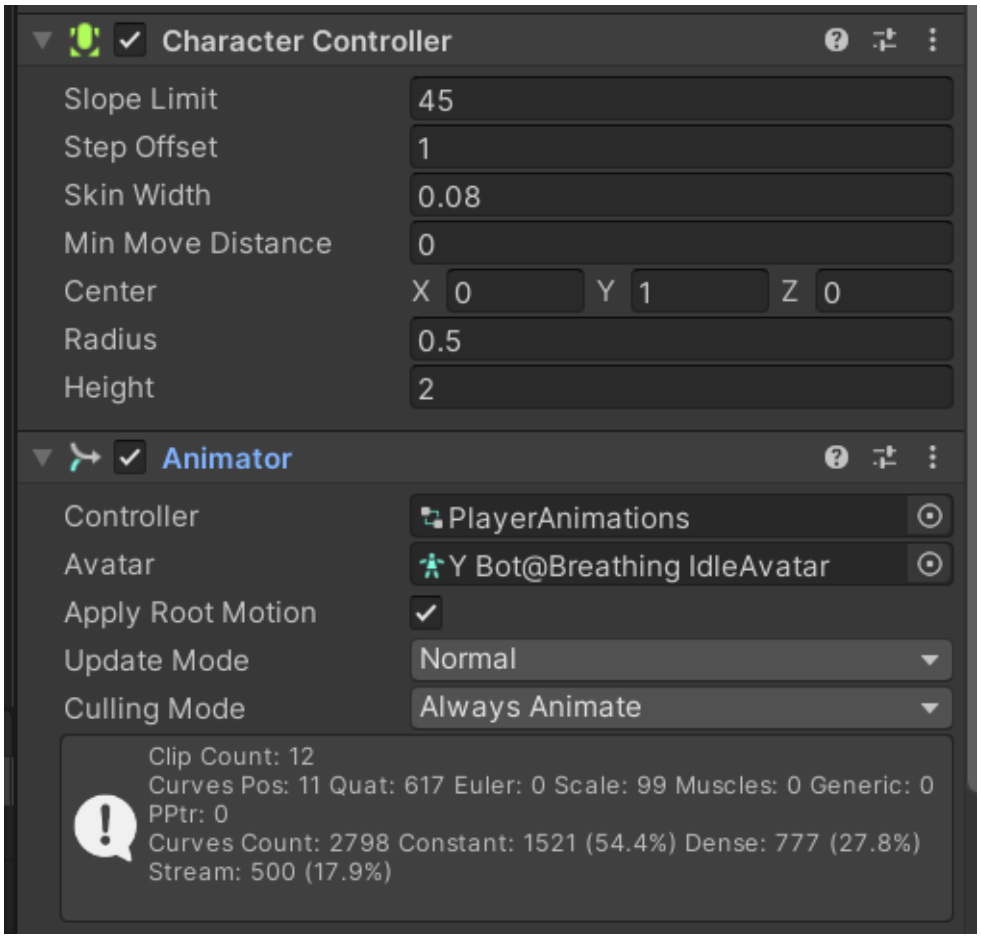
Obrázek 21 - Záložky Initial a Saved v Saved Variables [15]

3.6 Komponenty

V kontextu herního vývoje označuje pojem „komponenta“ jednu z nejzákladnějších stavebních jednotek, které se mohou přiřadit ke konkrétnímu GameObject. Jedná se o samostatný kus kódu, který rozšiřuje funkčnost objektu o danou vlastnost nebo chování. Každá komponenta poskytuje určité rozhraní, pomocí kterého lze přistupovat ke specifickým funkcím, vlastnostem nebo metodám. [8]

Komponenty mají různé účely a mohou být použity pro různé funkce, jako například pro ovládání kamery, fyzikální vlastnosti, umělou inteligenci, animaci postav a další.

Vývojáři mohou také vytvářet vlastní komponenty, které jsou specifické pro jejich hry. Tyto komponenty mohou být použity pro specifické účely, jako například pro manipulaci s umělou inteligencí nepřátelských postav, speciální efekty, specifické zvukové efekty, a další. [8]



Obrázek 22 – Komponenty

3.6.1 Transform

Komponenta Transform je jednou z nejdůležitějších komponent v Unity. Tato komponenta definuje pozici, rotaci a měřítko GameObject. Pozice určuje umístění objektu v prostoru, rotace určuje orientaci objektu a měřítko určuje velikost objektu. Transform je základním stavebním kamenem pro umístění a orientaci všech objektů v herním světě. [8]

Vlastnosti komponenty Transform jsou používány v celé hierarchii GameObject. Pokud se mění vlastnosti Transform jednoho GameObject, ovlivní to také všechny jeho potomky v hierarchii. Transform je automaticky přidáván ke každému GameObject v Unity. Transform může být přístupný přes komponentu v editoru Unity nebo přes skripty.

3.6.2 Collider

Collider představují klíčové komponenty pro detekci srážek a kolizí. Konkrétně se zde zaměříme na Box Collider, který je jedním z typů Collider a umožňuje definovat prostor, ve kterém se GameObject nachází a který by měl být brán v úvahu pro detekci srážek. [8]

BoxCollider je komponenta, která přidává kolidující těleso v podobě pravoúhlého kvádru kolem GameObject. Tento kvádr může být definován pomocí šířky, výšky a hloubky a může být ovlivněn i rotací objektu. BoxCollider umožňuje detekci srážek s ostatními kolidujícími tělesy v herním světě a na základě toho může být ovlivněna fyzika objektu. [8]

Použití Box Collider umožňuje vývojáři definovat prostor, ve kterém může hráč pohybovat svou postavu a překonávat překážky. Tato komponenta umožňuje také vytváření interakce mezi různými objekty v herním světě.

3.6.3 Character Controller

Komponenta Character Controller je jedním z typů Collider. Oproti ostatním typům Collider však nedetekuje kolize s objekty, ale umožňuje hráči pohybovat se po herním světě s různou rychlostí a zrychlením, skákat a padat. [8]

Character Controller se většinou používá v herních žánrech, jako jsou akční hry a RPG, kde je důležitá volnost pohybu postavy. Hráč může ovládat postavu pomocí klávesnice nebo gamepadu a pohybovat se v různých směrech. Tento typ Collider také umožňuje hráči překonávat překážky, jako jsou různé výškové rozdíly a překážky na zemi.

Při použití komponenty Character Controller je třeba brát v úvahu, že tato komponenta neposkytuje žádnou automatickou funkčnost pro gravitaci. Vývojář si tedy musí sám vytvořit logiku pro implementaci gravitace do hry. Při tvorbě této logiky je vhodné brát v potaz další faktory, jako například rychlost pádu nebo interakci s dalšími objekty v herním světě. Správné nastavení těchto faktorů může zásadně ovlivnit zážitek hráče ze hry a přispět k její lepší hratelnosti.

3.6.4 Animator

Animator je komponenta, která umožňuje vytváření animací v Unity. Umožňuje nastavení různých animačních křivek, které určují, jak se objekt pohybuje a jaké animace se při tom spouští. Animace mohou být řízeny pomocí různých triggerů nebo podmínek. [8]

3.6.5 NavMesh

NavMesh je komponenta v Unity, která umožňuje vytvářet navigaci v herním světě pro pohybující se objekty, jako jsou nepřátelé. Pomocí NavMesh lze vytvářet a editovat mapu pro pohybující se objekty, což umožňuje vytvářet inteligentní pohybové scénáře pro nepřátele, kteří budou reagovat na hráče a na okolní prostředí. NavMesh také umožňuje optimalizovat pohyb objektů a snižovat nároky na výpočetní výkon. [8]

4 Vlastní práce

Tato práce se zabývá analýzou a vývojem 3D hry „Jump for Life“, která kombinuje prvky žánrů FPS (First-Person Shooter) a platformových her. Hráč v této hře musí překonávat překážky, zároveň se však musí vyhnout střelám nepřátel, kteří se snaží hráče zastavit v dosažení cíle. Hráč disponuje omezeným zdravím, které je nutné chránit. Pro zneškodnění nepřátel může hráč použít svou zbraň. Hra se skládá z jedné úrovně, kterou hráč musí úspěšně projít, aby dosáhl vítězství.

Hra je vyvíjena v engine Unity za pomocí Unity Visual Scripting pro operační systém Windows. Při optimalizaci hry bylo bráno v úvahu rozmanité spektrum počítačů s různým hardwarem, aby hra mohla být spuštěna na co největším počtu různých zařízení.

4.1 Vývoj

Tato sekce bakalářské práce popisuje vývoj herních prvků, včetně charakteristik a ovládání hráče a také logiky chování nepřátel v herním prostředí.

4.1.1 Hráč

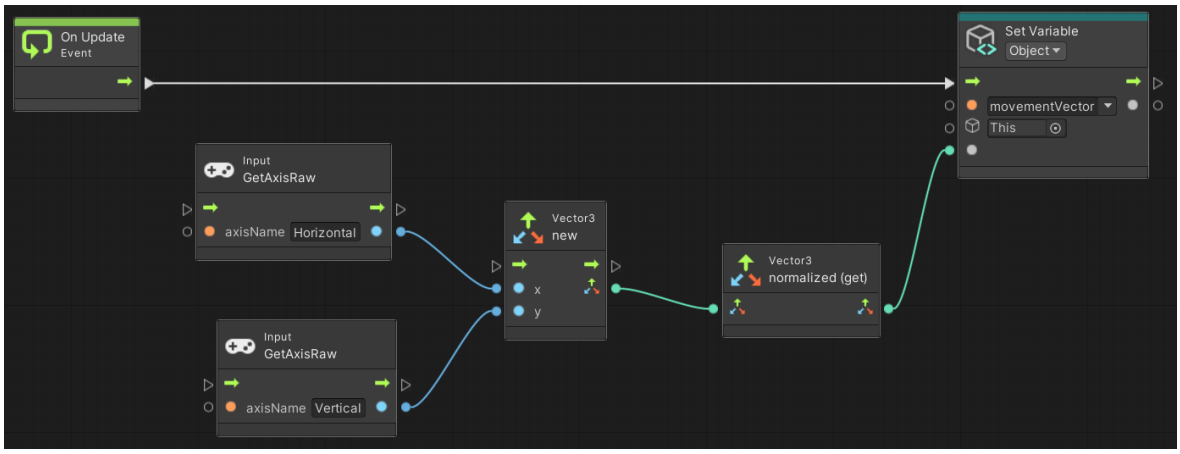
Postava hráče je soubor složený z více skriptů, který dohromady umožnuje hráči interagovat s prostředím a ovládat postavu.

4.1.1.1 Pohyb postavy

Pohyb postavy se ovládá pomocí kláves WASD, Shift pro běh a Mezerník slouží pro skok. Toto je umožněné díky komponentě Character Controller, která zpřístupňuje fyzikální vlastnosti přiřazenému GameObject. Samotná logika pohybu se ale musí naprogramovat.

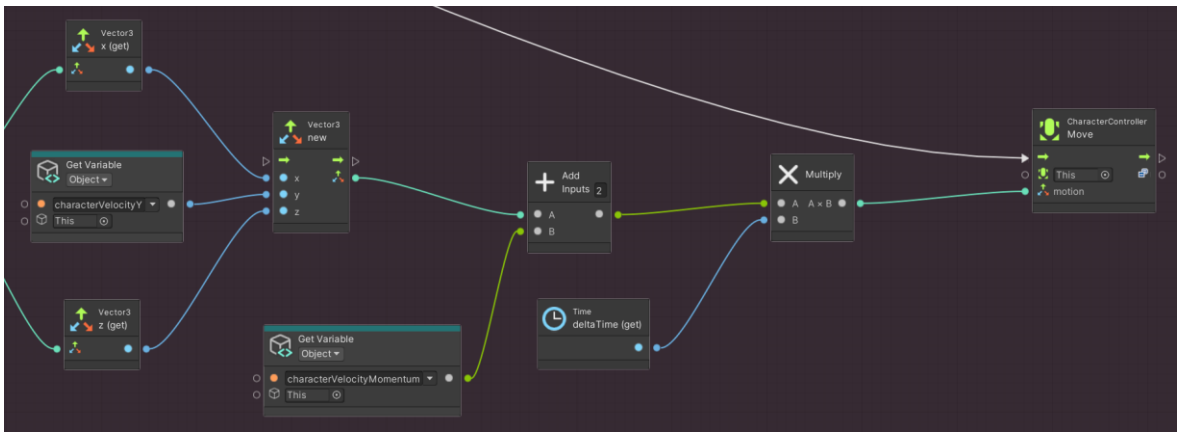
4.1.1.1.1 Chůze

Nejprve je připravena logika, která hlídá vstup z klávesnice za pomocí uzlu GetAxis. Uzel GetAxis očekává parametry Horizontal a Vertical díky kterým ví, jaké osy hlídat. Tedy při stisknutí klávesy A se spustí akce GetAxis Horizontal, která je následně předána do uzlu Vector3. Ten je následně uložený do proměnné movementVector. Hodnota se ukládá do proměnné z důvodu, aby bylo ošetřené zmáčknutí několika kláves najednou. Pokud by hráč zmáčkl klávesu A a zároveň W (pohyb doleva a vpřed), celá hodnota je pak uložena do proměnné jako typ Vector3.



Obrázek 23 - Hlídání stisku kláves

Hodnoty jsou následně vynásobeny s nastavenou rychlostí pohybu. Z těchto hodnot a hodnoty pro pohyb po ose Y (skok) je vytvořen nový Vector3, který se následně předá funkci Move z komponenty Character Controller. Jako výsledek se postava pohne po herním světě.



Obrázek 24 - Funkce Move

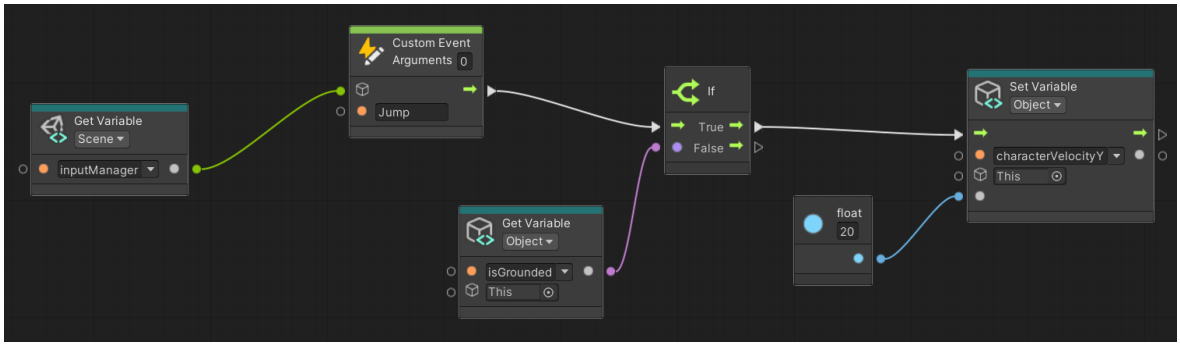
4.1.1.1.2 Běh

Pro spuštění akce běh se z části využije logika chůze. Podobně jako u chůze je připravený uzel, který hlídá stisknutí klávesy pro běh (Shift), tento uzel se nazývá On Button Input s parametrem Run. Po stisknutí klávesy se spustí signál, který navýší hodnota proměnné movementVector.

4.1.1.1.3 Skok a gravitace

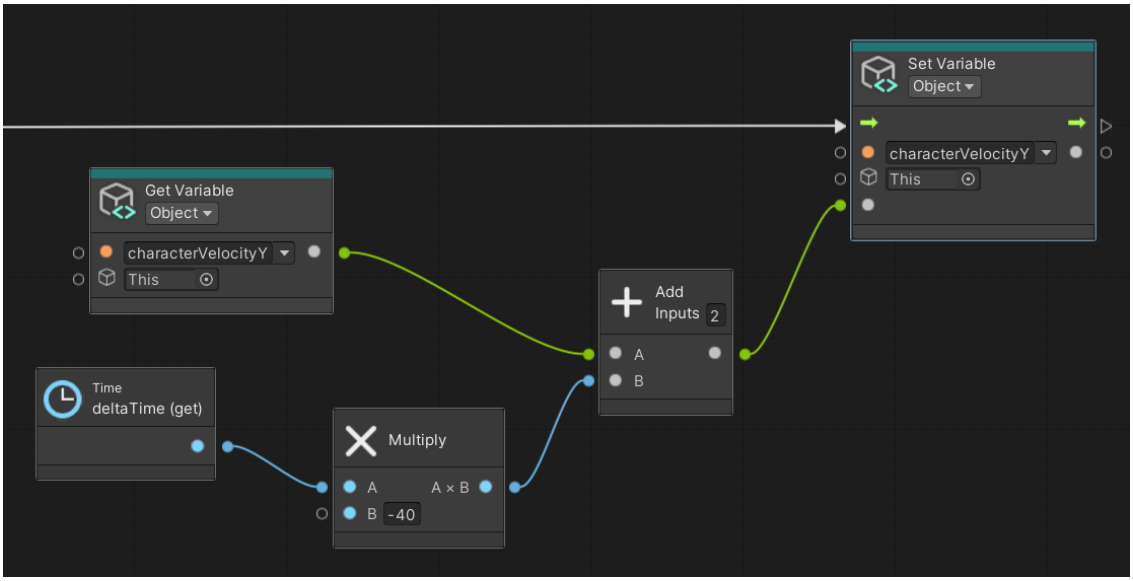
Pro zvolený žánr hry je schopnost skákání velice důležitá. Se skákáním se pojí složitější část a to gravitace. Tuto fyzikální vlastnost je potřeba naprogramovat vlastními silami, neboť komponenta Character Controller tuto vlastnost nenabízí.

Obdobně jako u předchozích funkcí pro pohyb, opět je na začátku uzel, který očekává stisknutí klávesy (Mezerník) pro odstartování logiky skoku. Při stisknutí klávesy se provede test pomocí funkce isGrounded, který ověří, zda je hráč stále na zemi, aby bylo možné předejít neomezenému množství skoků. Pokud na zemi je spustí se již samotná logika pro skok, která nastaví proměnnou characterVelocityY typu float na hodnotu 20. Toto v zásadě posune hráče o danou hodnotu po ose Y tedy nahoru.



Obrázek 25 - Zjednodušená logika pro skok

Zároveň se spustí připravená logika pro gravitaci, která v zásadě odečítá v průběhu času od hodnoty nastavené v proměnné characterVelocityY.

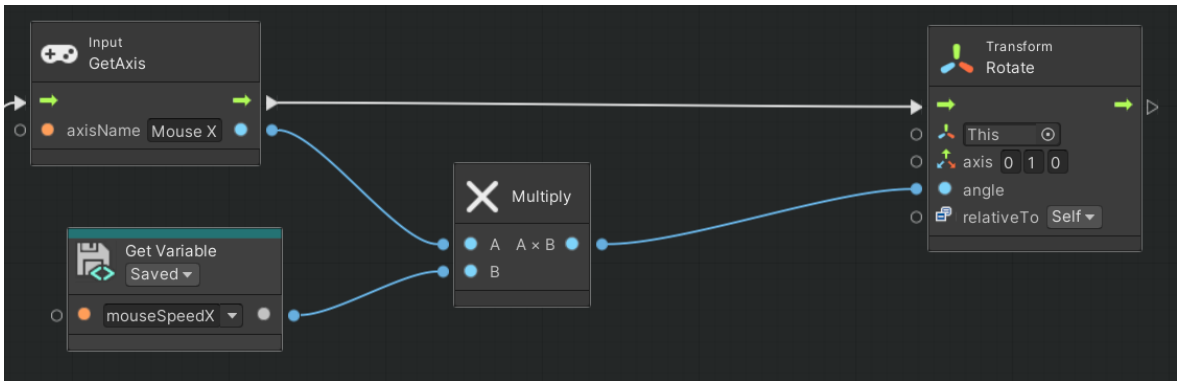


Obrázek 26 - Gravitace

4.1.1.2 Kamera

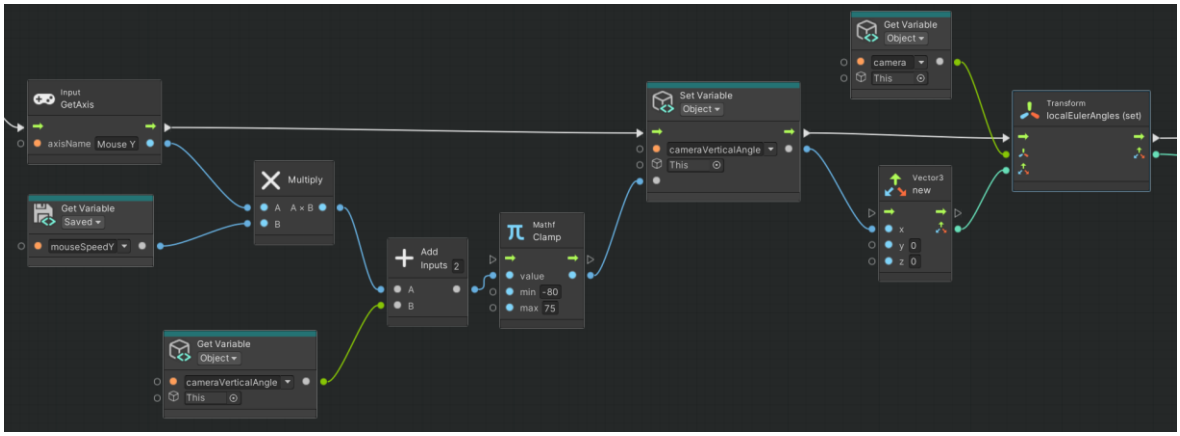
Pro tuto hru byl zvolen typ kamery z pohledu první osoby tzn. hráč prožívá hru jako z pohledu vlastních očí.

K ovládání kamery je očekáván pohyb myši. Kamera je rozdělena do dvou skupin pohledů – pohled nahoru a dolů a pohled doleva a doprava. Logiky pro pohled do stran je jednoduše docíleno pomocí funkce Rotate s omezením na osu Y, tedy pohled je otáčen okolo osy Y.



Obrázek 27 - Pohled do stran

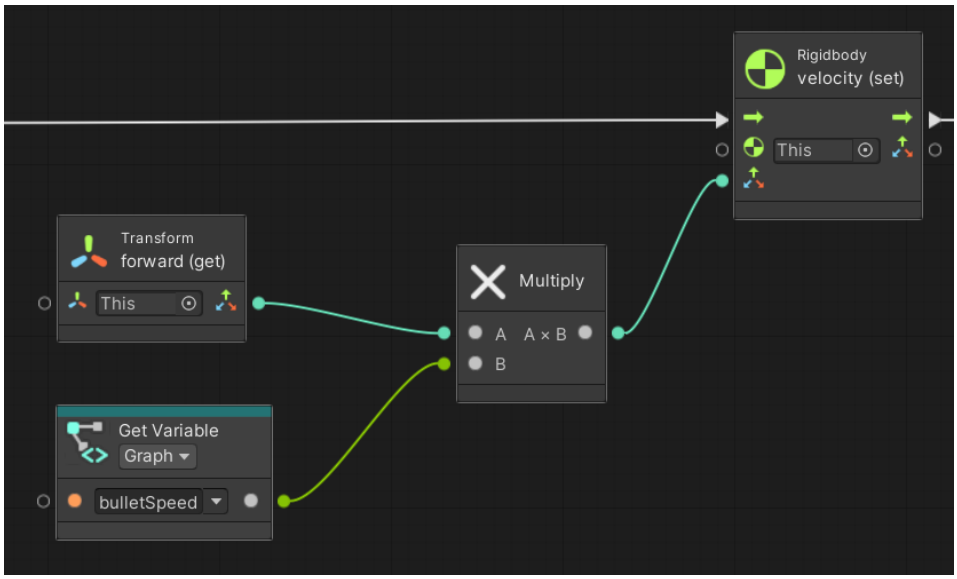
Docílení ovládání pohledu nahoru a dolů je již složitější z důvodu že nestačí jednoduše přidat funkci Rotate, neboť by se otáčel celý GameObject, a nejen kamera což by způsobilo, že při pohledu nahoru by se zvedli nohy hráče do nebe a naopak, jelikož kamera je v hierarchii jako potomek rodičovského GameObject Player. Je tedy nutné vytvořit proměnnou s odkazem na GameObject kamery a vypočítat její úhel pohledu pomocí funkce localEulerAngles. Tato funkce slouží k určení orientace objektu v 3D prostoru pomocí tří úhlů pitch (sklon), yaw (otočení) a roll (náklon).



Obrázek 28 - Pohled nahoru a dolů

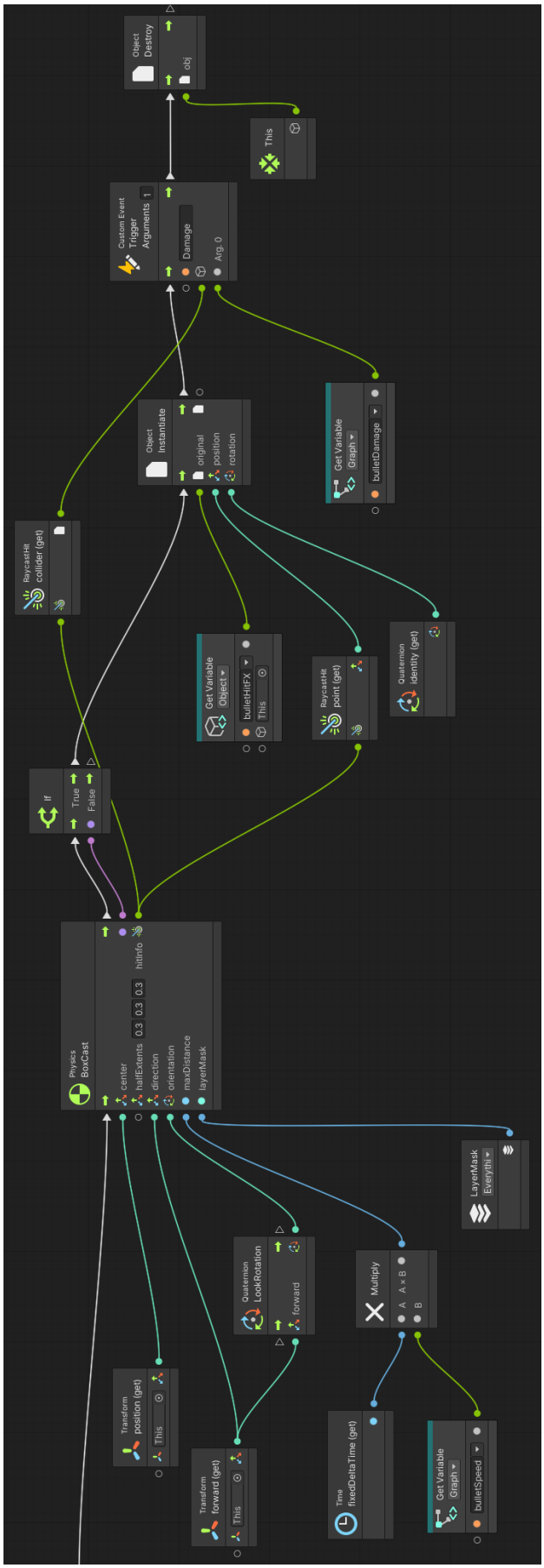
4.1.1.3 Střelba

Střelba je umožněna za pomocí dvou skriptů, jeden, který zajišťuje spuštění střelby a druhý který zajištuje logiku dopadu. Skript tedy čeká na spuštění, které je zapříčiněno stisknutím tlačítka (tlačítko levé myši) to spustí funkci, jenž předá data o směru náboje funkci, která vytvoří ve světě náboj s přednastavenými parametry pro rychlost a směr.



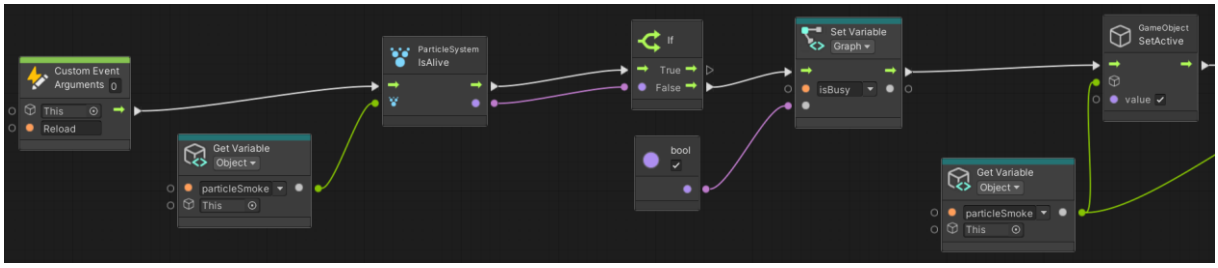
Obrázek 29 - Vytvoření střelného náboje

Po vystřelení a vytvoření náboje se spustí logika pro kolizi, která hlídá náraz a zároveň kontroluje objekt do kterého narazí. Při nárazu do nepřítele či hráče (logika pro střelbu je znovupoužita i pro střelbu nepřátel) se spustí funkce Damage, která odečte cílovému objektu zdraví.



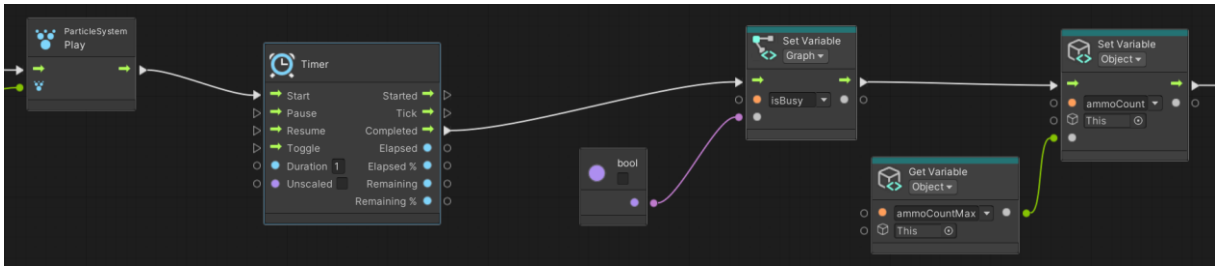
Obrázek 30 - Logika kolize vystřeleného náboje

Hráč má také omezený počet nábojů. Po vyčerpání zásobníku musí hráč přebít, aby mohl pokračovat ve střelbě. Po stisknutí klávesy pro přebití (R) se spustí logika, které spustí samotné přebití. Na místě musí být ověření, zda hráč již zrovna nepřebíjí, toto je vyřešeno pomocí funkce isAlive pro Particle Systém, neboť při přebití se vytvoří kouř nad zbraní indikující ochlazení zbraně ve smyslu, že zbraň je po ochlazení opět schopná střelby. Pokud kouř stále nad zbraní přebývá, hráč není schopen přebít. Toto zamezuje hráči opakovaného mačkání přebíjení, které by mohlo vytvořit z obyčejné pistole samopal.



Obrázek 31 - Spuštění přebití

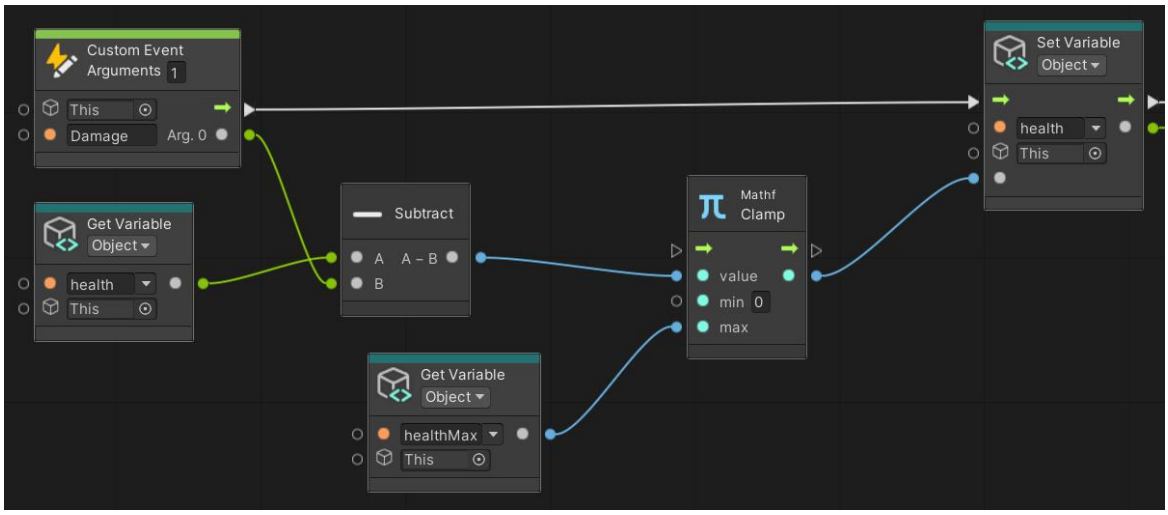
Pokud kouř již zmizel, funkce projde podmínkou a pokračuje. Po průchodu podmínkou se nastaví proměnná isBusy typu bool na true. Toto zamezí hráči střelbu při procesu přebíjení. Následně se vytvoří již zmíněný kouř a zároveň se spustí časovač jedné vteřiny, který znemožní střelbu krátce po přebití. Konečně se nastaví hodnota proměnné ammoCount na její maximální hodnotu a hráč má přebito, může tedy pokračovat ve střelbě.



Obrázek 32 - Dokončení logiky přebití

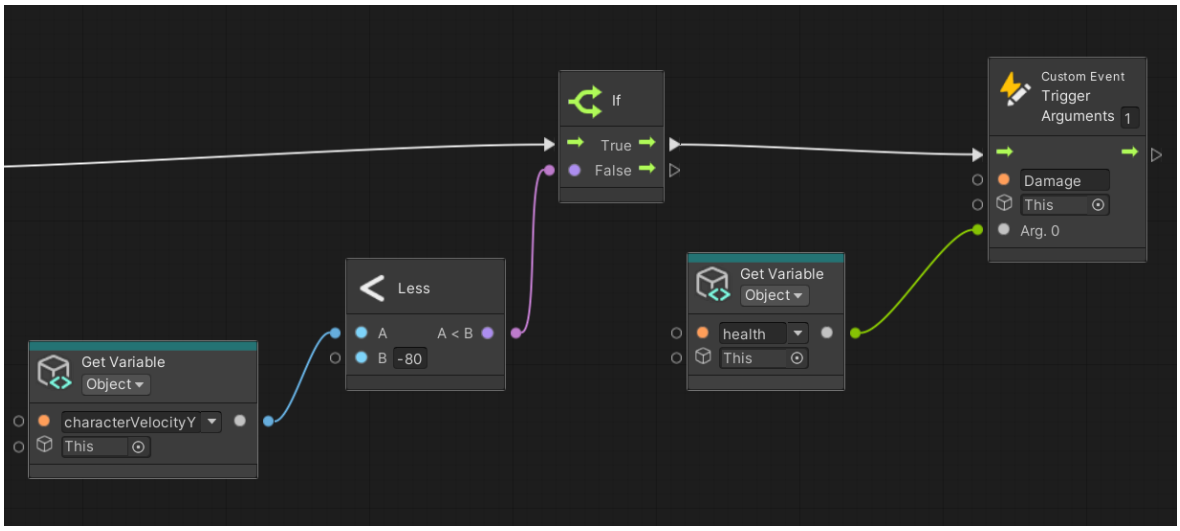
4.1.1.4 Zdraví

Jak postava hráče, tak i postavy nepřátel mají omezené množství zdraví. Je tedy nutné mít logiku, která bude zdraví spravovat. Logika pro odečítání zdraví je stejná pro hráče i pro nepřátelé. Po obdržení signálu při nárazu kulky se spustí funkce, která zkontroluje aktuální zdraví a od této hodnoty odečte příslušné množství. Zároveň je napojena logika pro kontrolu úmrtí, která nastane v případě, že aktuální zdraví klesne pod hodnotila nula.



Obrázek 33 - Odečtení zdraví

Pro hráče může nastat i jiný typ úmrtí, a to pádem z platformy, tedy další logika musí být namístě. Při přesáhnutí rychlosti změny hodnoty pohybu po ose Y se spustí funkce, která hráči udělí poškození v hodnotě jeho aktuálního zdraví.



Obrázek 34 - Kontrola rychlosti pádu

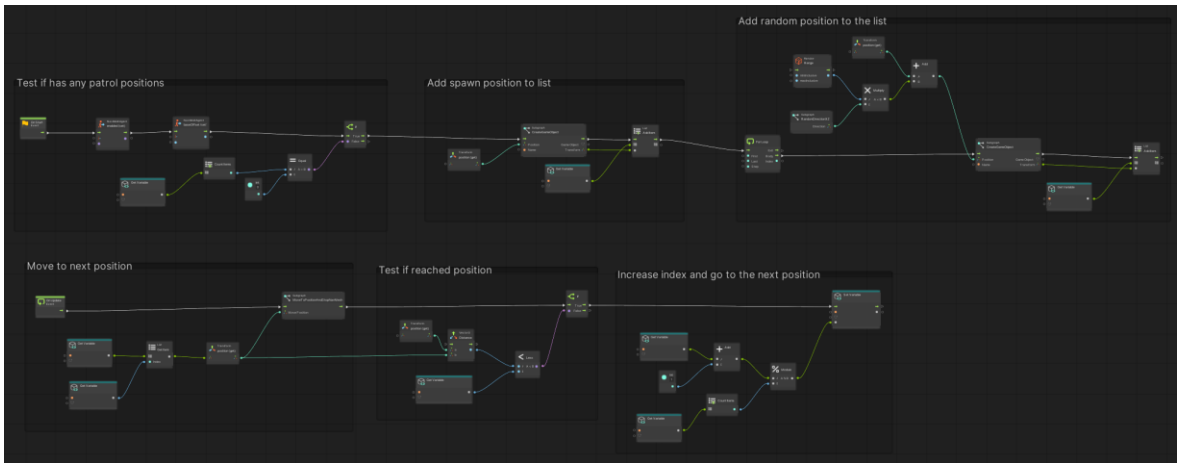
4.1.2 Nepřátelé

Hra obsahuje dva typy nepřátel – Sentinel a Defender. Oba typy se liší ve způsobu chování při spatření hráče. Oba roboti střílí dvě střely najednou. Sentinel má hodnotu zdravý osmdesát a Defender sto. Liší se také ve velikosti oblasti detekce hráče. Sentinel má oblast detekce dvanáct metrů, přičemž Defender má oblast o něco větší, a to devatenáct metrů.

4.1.2.1 Chování

Hlavní rozdíly jsou ve způsobu chování při detekování hráče. Defender je pevně umístěný robot který se nehýbe a pouze čeká až se hráč přiblíží do dané oblasti. Při vstoupení do oblasti se Defender nasměruje na hráče a začne střelbu.

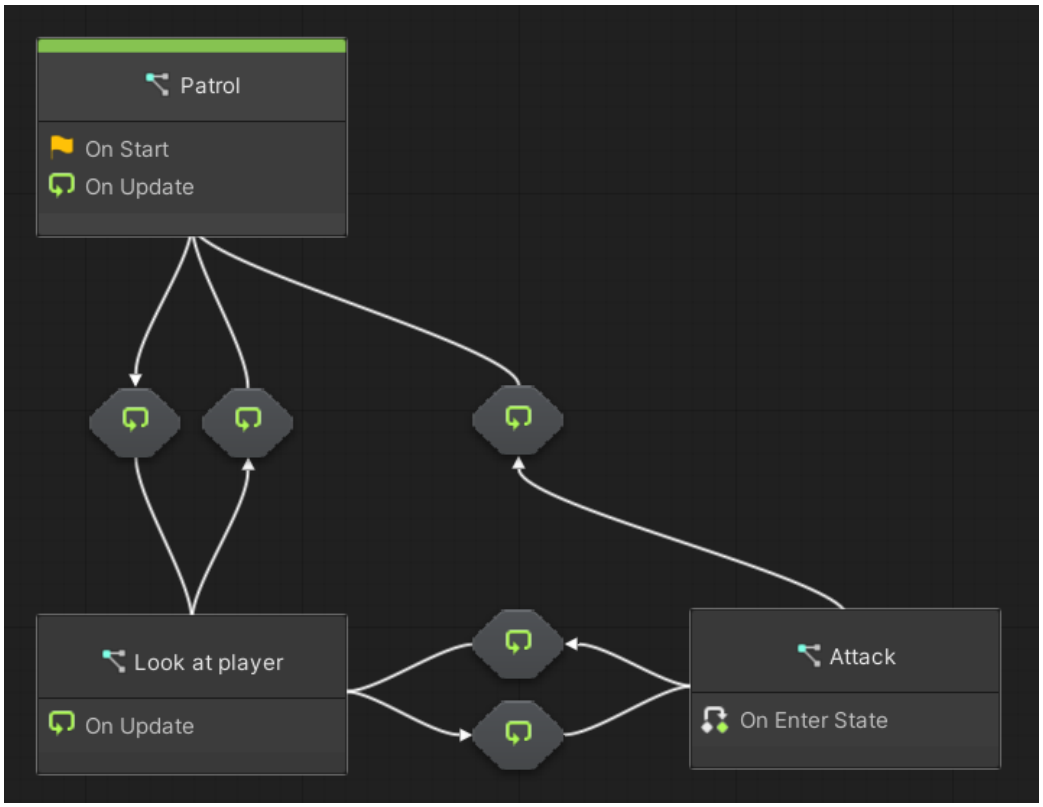
Sentinel hlídkuje v dané oblasti a přechází mezi různými body. Pohyb Sentinela je umožněn funkčností zvanou NavMesh.



Obrázek 35 - Přehled logiky pro hlídkování Sentinela

Pokud se hráč dostane do oblasti nepřítele typu Sentinel, tak Sentinel začne hráče pronásledovat a střílet. Tedy naopak od nepřítele typu Defender, Sentinel pouze nestojí, ale hráče pronásleduje, dokud hráče nezničí či hráč neunikne z jeho sledovací oblasti. Samotná logika střelby u obou typů nepřátel je stejná jako u hráče.

Skripty pro naprogramování logiky chování nepřátel bylo dosaženo pomocí State Graph, ve kterém byl vytvořena logika jednotlivých akcí.



Obrázek 36 - State Graph logika nepřátel

4.2 Vizuální zpracování

Všechny grafické prvky jako jsou textury, materiály a modely byly získány z volně dostupných zdrojů jako například Unity Asset Store, tedy nebyly vytvořeny, ale pouze staženy. Výhodou použití stažených modelů je, že usnadňují a zrychlují proces tvorby herního světa. Celý herní svět byl pojat ve futuristickém a minimalistickém stylu. Při vybírání grafických prvků je nutno vzít v potaz složitost modelu a počet polygonů. Pro lepší výkon je vhodnější zvolit modely s nižším počtem polygonů.

Pro vložení vybraných modelů do projektu je potřeba modely importovat pomocí Package Manager a jeho rozhraní. Po importování s vybraných modelů se následně objeví v sekci Assets. Poté lze jednoduše pomocí myši přetáhnout vybraný model z Assets do scény. Pro správné zobrazení modelu v herní scéně je vhodné model umístit na správnou pozici a nastavit správné měřítko, aby odpovídalo měřítku ostatních objektů v herní scéně.

Jak již bylo zmíněno, pro tuto práci bylo zvoleno stažení grafických modelů oproti vytvoření vlastních. Hlavním důvodem pro toto rozhodnutí bylo zaměření této bakalářské práce na Visual Scripting, a nikoli na tvorbu vlastních modelů postav. Stažením modelů postav lze efektivněji využít čas a zdroje na vývoj v samotném Visual Scripting.

4.2.1 Model postavy hráče

Postava hráče je robot zvaný „Y Bot“ [16]. Díky již připravenému modelu je práce zjednodušena. Postava má správně nastavenou kostru, díky které je přidání animací pro jednotlivé pohybové akce bezproblémové.

Obrázek 37 - Y Bot

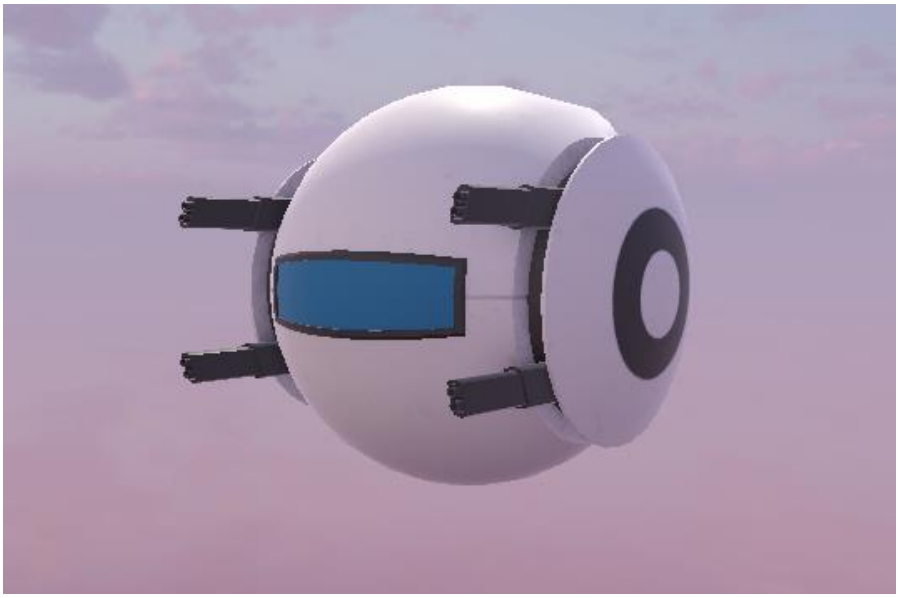
4.2.2 Model zbraně

Pro model zbraně byl stažen jednoduchý futuristický model, který má připomínat laserovou zbraň. Model je nazvaný „Sci Fi Futuristic Hand Gun“ [17].

Obrázek 38 - Model zbraně

4.2.3 Modely nepřátel

Ve hře jsou dva typy nepřátel, první, který hráče pronásleduje a druhý co pouze stojí. Pokud toto bude bráno v potaz, je nutné vybrat takové modely, které tomu logicky budou odpovídat. Model Sentinela je létající robot a model Defender je robot připomínající věž. Balíček modelů s roboty se nazývá „Sci fi Drones“ [18].

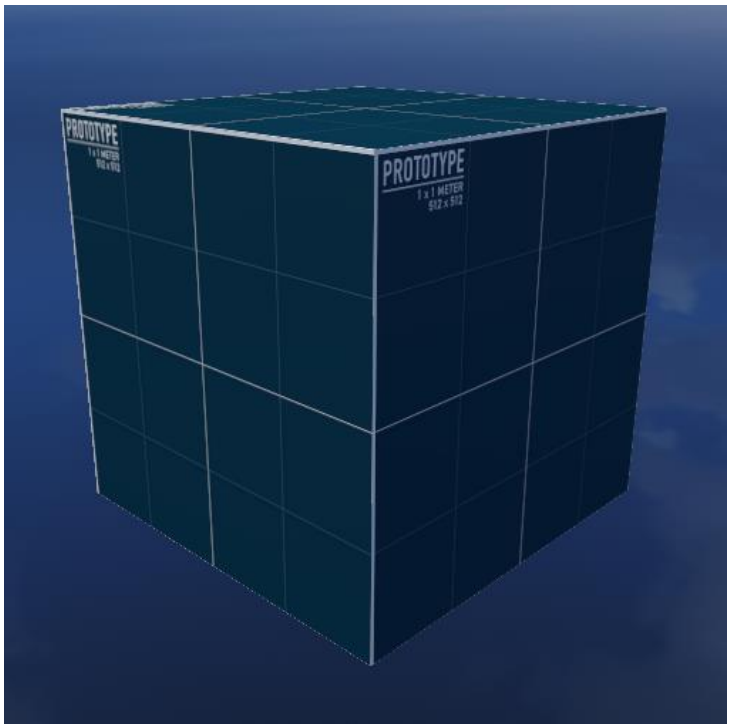


Obrázek 39 - Model Sentinel

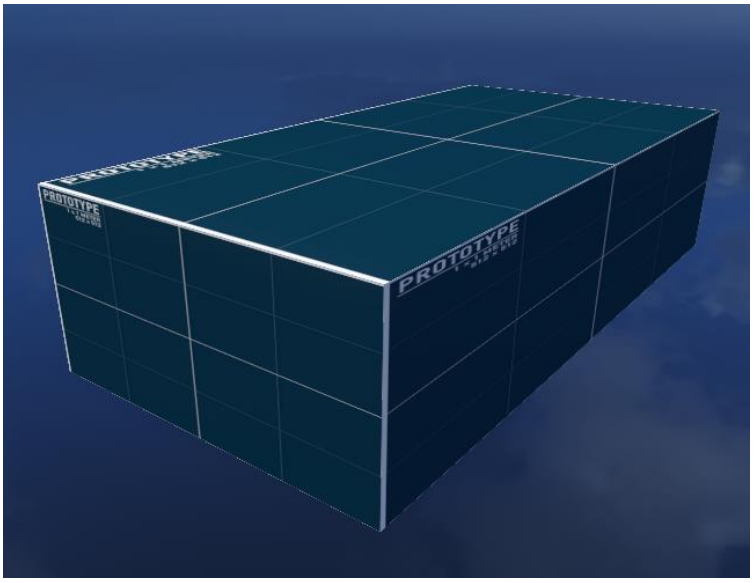
Obrázek 40 - Model Defender

4.2.4 Platformy

Platformy a obecně mapa je poskládaná z kvádrů a krychlí na kterých je použita textura zvaná Prototype. Jedná se o texturu, která na objektu vytváří čtverce. Následně jednotlivé platformy, na které hráč skáče jsou tvořené z různě velkých modelů, například 1x1x1 metrů nebo 2x1x2 metrů. Na jednotlivé platformy je pak aplikována komponenta Box Collider. Materiály se nazývají „Gridbox Prototype Materials“ [19].



Obrázek 41 - Platforma 1x1x1



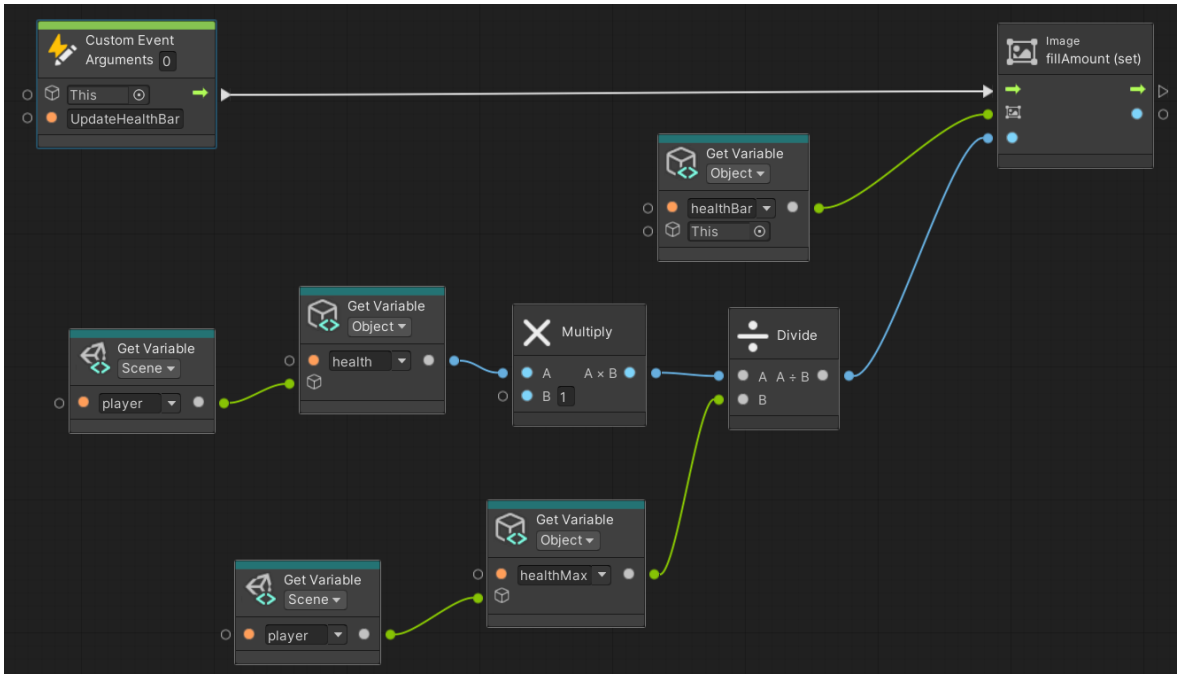
Obrázek 42 - Platforma 4x1x2

4.3 Uživatelské rozhraní a doplňkové scény

Uživatelské rozhraní hry je klíčový prvek pro správné ovládání a užívání hry hráčem.

4.3.1 Ukazatel zdraví

Hráč má omezené zdraví, a tudíž je nutné zobrazit ukazatel, který bude hráče informovat o jeho aktuálním stavu zdraví. Tohoto je docíleno pomocí jednoduché funkce, která vydělí aktuální zdraví se zdravím maximálním, výsledkem je desetinná hodnota, která je následně předána funkci fillAmount, jenž nastaví výplň lišty s ukazatelem zdraví. Tato celá funkce je spuštěna na základě signálu z funkce Damage po obdržení poškození.

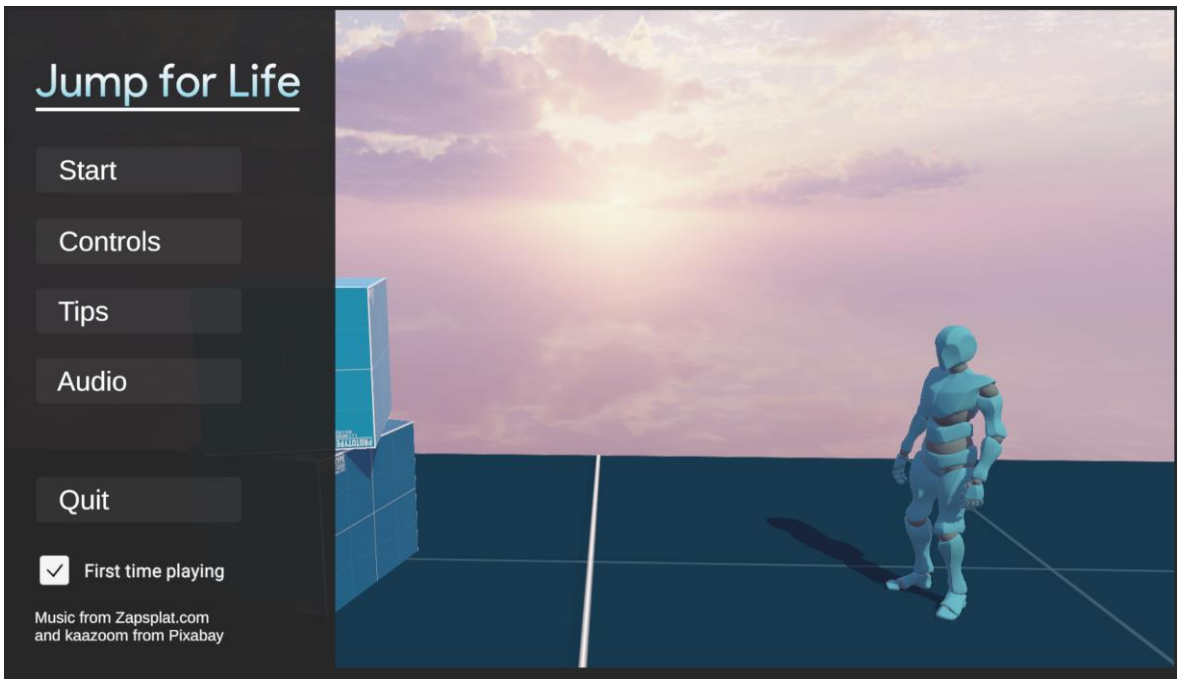


Obrázek 43 - Funkce pro zobrazení množství zdraví

Obrázek 44 - Ukazatel zdraví

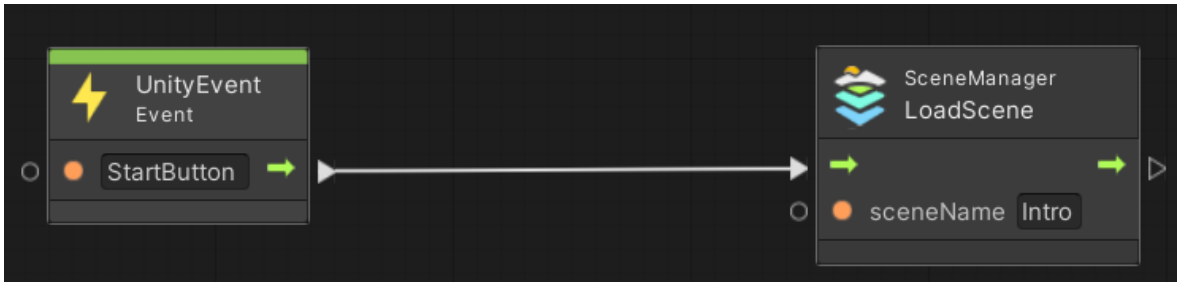
4.3.2 Hlavní menu

Hlavní menu, které je zobrazené před spuštěním samotné hry, obsahuje důležité funkčnosti, které hráč může využít. Hlavní menu obsahuje akce pro spuštění hry, zobrazení ovládání, tipy, nastavení zvuku, možnost ukončit program, zvolení možnosti zapnutí výuky a informace a autorech použité hudby.



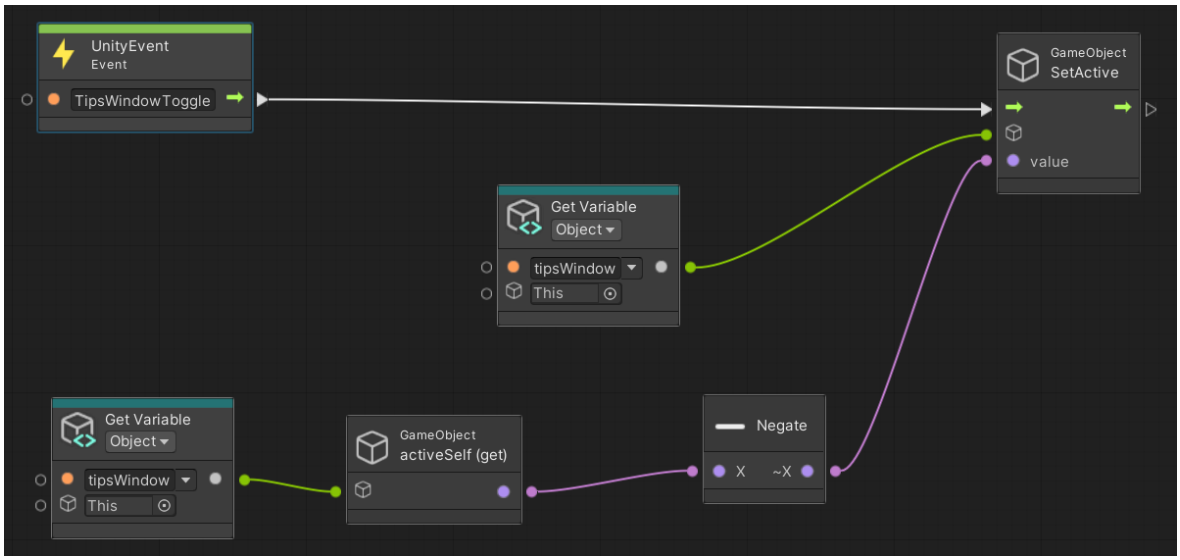
Obrázek 45 - Hlavní menu

Každé tlačítko spouští příslušnou akci. Možnost Start načte herní scénu, pomocí jednoduché funkce LoadScene.



Obrázek 46 - Spuštění herní scény

Controls, Tips a Audio zobrazí vyskakující okno, které překryje část obrazovky. Tohoto je docíleno pomocí přepínání stavu objektů. Funkce SetActive a negace aktuálního stavu umožní, že okno se zobrazí či schová.



Obrázek 47 - Přepínání aktuálního stavu okna

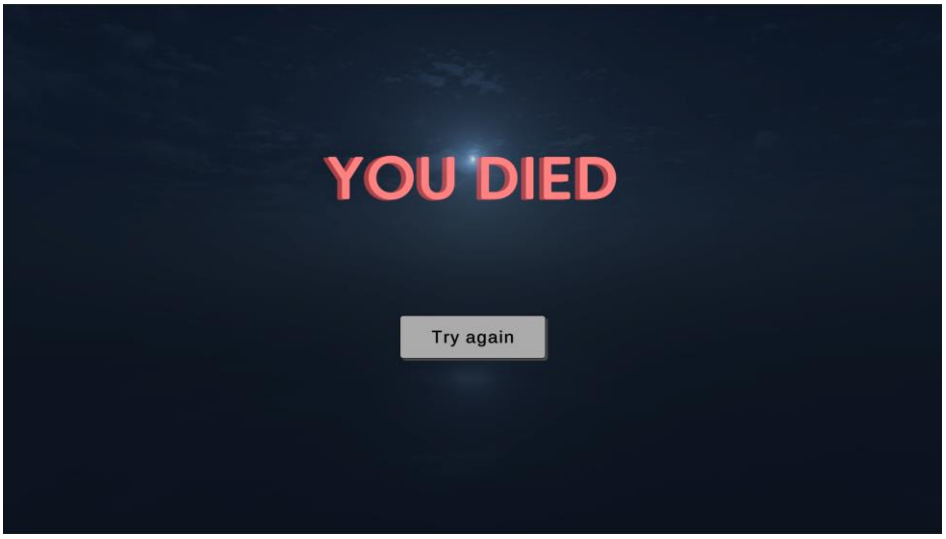
4.3.3 Menu pauzy

Menu pauzy je téměř stejné jako hlavní menu v pár rozdílech. Při spuštění pauzy se pozastaví celá hra, a tedy čas stojí. V hlavním menu takováto funkčnost není potřebná. Zároveň zde není zaškrtávací tlačítko, neboť to zde nemá smysl. Jeden z posledních rozdílů je tlačítko Resume, které na rozdíl od Start, pouze hru obnoví.

Obrázek 48 - Menu pauzy

4.3.4 Scéna smrti

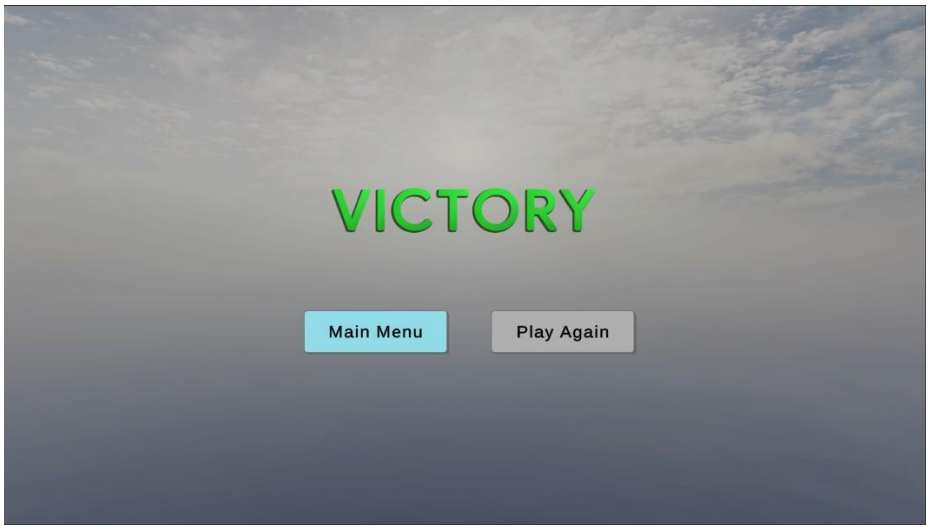
V případě že zdraví klesne na nebo pod nulu, hráč zemře a zobrazí se scéna, která hráče o dané situaci informuje. Hráč má možnost se znovu pokusit o zvládnutí výzvy. Tlačítko Try again načte znovu hlavní herní scénu pomocí funkce LoadScene. V daný moment je hráč začne od úplného začátku.



Obrázek 49 - Scéna smrti

4.3.5 Scéna výhry

Pokud hráč projde připravenou mapu a dosáhne zelené platformy, vyhrál a zobrazí se mu scéna výhry. Podobně jako u scény smrti, hráč má možnost začít znovu anebo se vrátit do hlavního menu. Obě možnosti jsou umožněné díky stejné funkci LoadScene.



Obrázek 50 - Scéna výhry

4.4 Uživatelské testování

Testování hry bylo provedeno na dvou skupinách uživatelů. První skupinou byli uživatelé spíše pokročilejší a druhou skupinou byli hráči spíše příležitostní. Jelikož vytvářená hra je mířená na obě tyto skupiny, ani jedna z testovacích skupin neměla s dohráním problém.

Při testování byly zjištěny dvě chyby. Jedna z chyb spočívala ve špatném systému přebíjení zbraně. Pokud uživatel střílel a zároveň mačkal opakovaně klávesu na přebití zbraně, mohl tak střílet neomezeně bez jakékoliv pauzy. Tento problém byl vyřešen jednoduše a to tak, že se zvýšila prodleva mezi přebitím a schopností střelby po přebití.

Druhá chyba spočívala spíše v lepší uživatelské zkušenosti. Při pádu hráče z platformy byla velice dlouhá doba, než hráč pádem zemřel a zobrazila se mu scéna úmrtí a mohl tak svůj pokus zopakovat. Tato chyba byla opravena snížením hodnoty rychlosti pádu potřebné ke spuštění funkce úmrtí a následného zobrazení scény úmrtí.

5 Výsledky a diskuse

Finální prototyp splňuje všechny plánované funkčnosti. Vytvořené prvky byly vytvořené tak, že umožnují snadné budoucí rozšíření. Hra je správně optimalizovaná, a tedy má výborný počet snímku za vteřinu. Do hry byl implementován krátký návod, aby každý hráč věděl, jak hru ovládat a jaký je její cíl.

Autor před tvorbou bakalářské práce a s ní související hrou, neměl žádné znalosti s vývojem her a s Visual Scripting. Po dokončení nabyl znalosti, které je možné využít dále.

Při tvorbě nastalo několik komplikací, které byly obtížné zvládnout, neboť programování ve Visual Scripting není rozšířené, a tedy pomocné návody nejsou dobře dostupné. Největší komplikací bylo docílení správného chování střelby, která byla vyřešena stylem pokus a omyl díky kterému se člověk naučí nejvíce.

5.1 Budoucí rozvoj a rozšíření

Jak bylo již zmíněno, hra je otevřena dalšímu rozšíření. Jedno z hlavních rozšíření by byla lepší AI (umělá inteligence) nepřátel, obzvláště situace, kdy hráč útočí na nepřítele, ale nepřítel na to nereaguje, pokud hráč není v jeho detekční oblasti. Hlavní chybou, kterou je potřeba v budoucím rozvoji vyřešit se také vztahuje k AI (umělá inteligence) nepřítele, jedná se o problém, při kterém nepřítel při pronásledování hráče dokáže procházet zdmi i přesto že by tomu tak být nemělo. Bohužel tato chyba je tak složitá, že se to začínajícímu vývojáři nepodařilo vyřešit. Další z rozšíření by bylo více úrovní, které by hráč musel dokončit, aby dosáhl vítězství. Zároveň by hra mohla obsahovat hlavního nepřítele, který by čekal na konci hry. Různorodost zbraní a schopností by přinesla hře značné dodala lepší hráčský zážitek.

6 Závěr

Cílem této práce bylo vytvořit hru v herním engine Unity za pomocí Unity Visual Scripting a seznámit čtenáře s potřebnými vědomostmi a pojmy k tvorbě her.

V teoretické části této bakalářské práce byly podrobně popsány základní prvky a principy Unity, které jsou nezbytné pro úspěšný vývoj hry. Dále byly popsány jednotlivé komponenty a nástroje, které byly použity v průběhu vývoje aplikace v prostředí Unity. V této části práce byly rovněž popsány různé platformy pro vývoj. Celkově tato část práce poskytuje čtenáři stručný přehled o videohrách, jejich vývoji a převážně o prostředí Unity.

V praktické části byl popsán nápad hry a následně implementace jednotlivých prvků. Hlavním cílem praktické části byl ale ovšem samotný vývoj, který probíhal v prostředí Unity za pomocí Visual Scripting.

Výsledkem práce je hra s plnohodnotnou hratelností, kterou je možné dokončit. Hra je pojmenována Jump for Life v překladu Skákej o život, neboť cílem hry je přeskákat všechny překážky k dosažení cíle. Hra obsahuje dva typy nepřátel, kteří se snaží hráči cestu za vítězstvím znemožnit.

7 Seznam použitých zdrojů

1. Salen, K., & Zimmerman, E. (2005). The game design reader: A rules of play anthology. [ebook] MIT Press. ISBN: 978-0262303170.

2. Hodent, C. (2020). The psychology of video games. [ebook] Routledge. ISBN: 978-1003045670

3. What is a Gaming Engine? – Arm®. Copyright © 2022. Defining the Future of Computing – Arm® [online]. [cit. 25.09.2022]. Dostupné z: https://www.arm.com/glossary/gaming-engines

4. Bradfield, C 2018, Godot Engine Game Development Projects: Build Five Cross-Platform 2D and 3D Games with Godot 3. 0, [ebook] Packt Publishing, Limited, Birmingham. ISBN 9781788831505

5. Satheesh PV. Unreal Engine 4 Game Development Essentials. [ebook] Packt Publishing, 2015. ISBN 978-1784391966.

6. Smith, M. (2018). Unity 2018 Cookbook. [ebook] Birmingham: Packt Publishing. ISBN 978-1788471909.

7. Simon Jackson. Mastering Unity Visual Scripting. [ebook] Packt Publishing, 2019. ISBN 978-1786463456

8. Unity – Manual: Unity User Manual 2021.3 (LTS). [online]. Copyright © 2021 Unity Technologies. [cit. 22.01.2023]. Dostupné z: https://docs.unity3d.com/Manual/index.html

9. Unity Visual Scripting | Visual Scripting | 1.8.0. [online]. Copyright © 2022 Unity Technologies [cit. 24.01.2023]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.visualscripting@1.8/manual/

10. Mega Man 11 on Steam. [online]. Copyright © 2023 Valve Corporation. All rights reserved. [cit. 20.02.2023].
Dostupné z: https://store.steampowered.com/app/742300/Mega_Man_11/

11. Call of Duty®: Black Ops III on Steam. [online]. Copyright © 2023 Valve Corporation. All rights reserved. [cit.20.02.2023].
Dostupné z: https://store.steampowered.com/app/311210/Call_of_Duty_Black_Ops_III/

12. Unity - Manual: Console Window. [online]. Copyright © 2021 Unity Technologies. [cit. 20.02.2023].
Dostupné z: https://docs.unity3d.com/Manual/Console.html

13. Graphs | Visual Scripting | 1.8.0. [online]. Copyright © 2022 Unity Technologies [cit. 20.02.2023].

Dostupné z: https://docs.unity3d.com/Packages/com.unity.visualscripting@1.8/manual/vs-graph-types.html

14. The interface | Visual Scripting | 1.7.8. [online]. Copyright © 2022 Unity Technologies [cit. 20.02.2023].
Dostupné z: https://docs.unity.cn/Packages/com.unity.visualscripting@1.7/manual/vs-interface-overview.html#the-blackboard

15. Variables | Visual Scripting | 1.7.8. [online]. Copyright © 2022 Unity Technologies [cit. 20.02.2023].
Dostupné z: https://docs.unity.cn/Packages/com.unity.visualscripting@1.7/manual/vs-variables.html

16. Mixamo. Mixamo [online]. Copyright © 2023 [cit. 10.03.2023]. Dostupné z: https://www.mixamo.com/

17. Sci Fi Futuristic Hand Gun – Mobility Arts | 3D Guns | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 10.03.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/props/guns/sci-fi-futuristic-hand-gun-90249

18. Sci fi Drones – Lukas Bobor | 3D Robots | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 10.03.2023]. Dostupné z: https://assetstore.unity.com/packages/3d/characters/robots/sci-fi-drones-90326

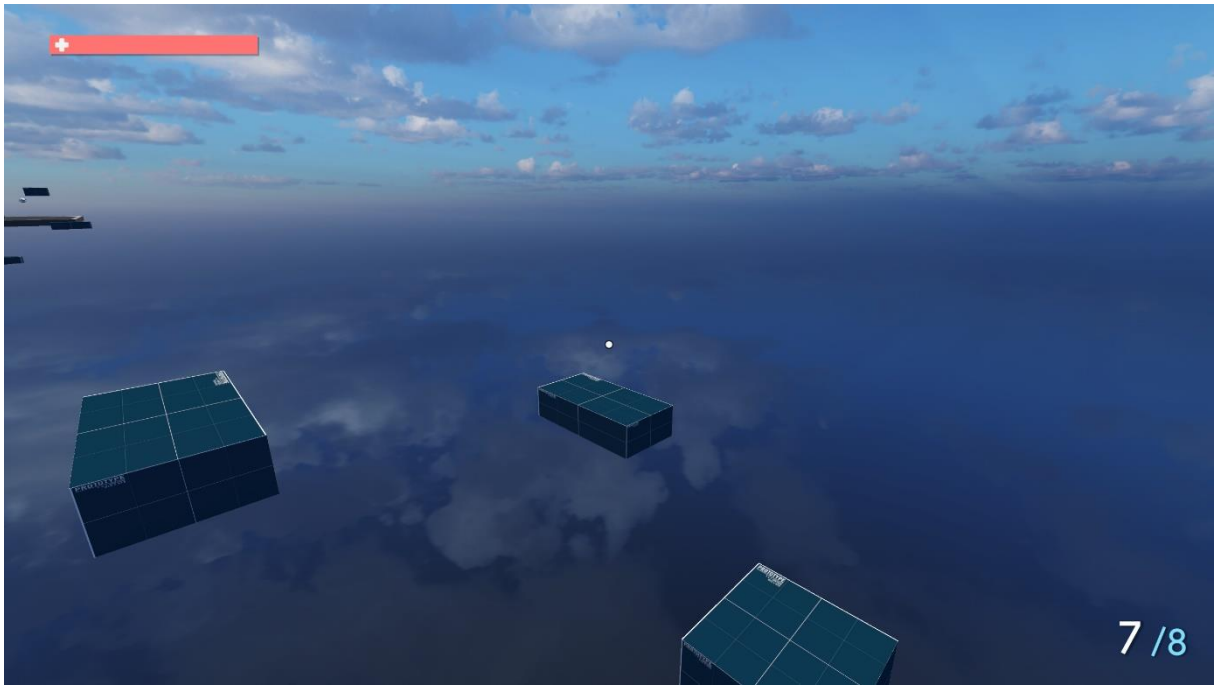
19. Gridbox Prototype Materials – Ciathyza | 2D Textures & Materials | Unity Asset Store. Unity Asset Store – The Best Assets for Game Making [online]. Copyright © 2023 Unity Technologies [cit. 10.03.2023]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/gridbox-prototype-materials-129127

8 Seznam obrázků, tabulek, grafů a zkratek

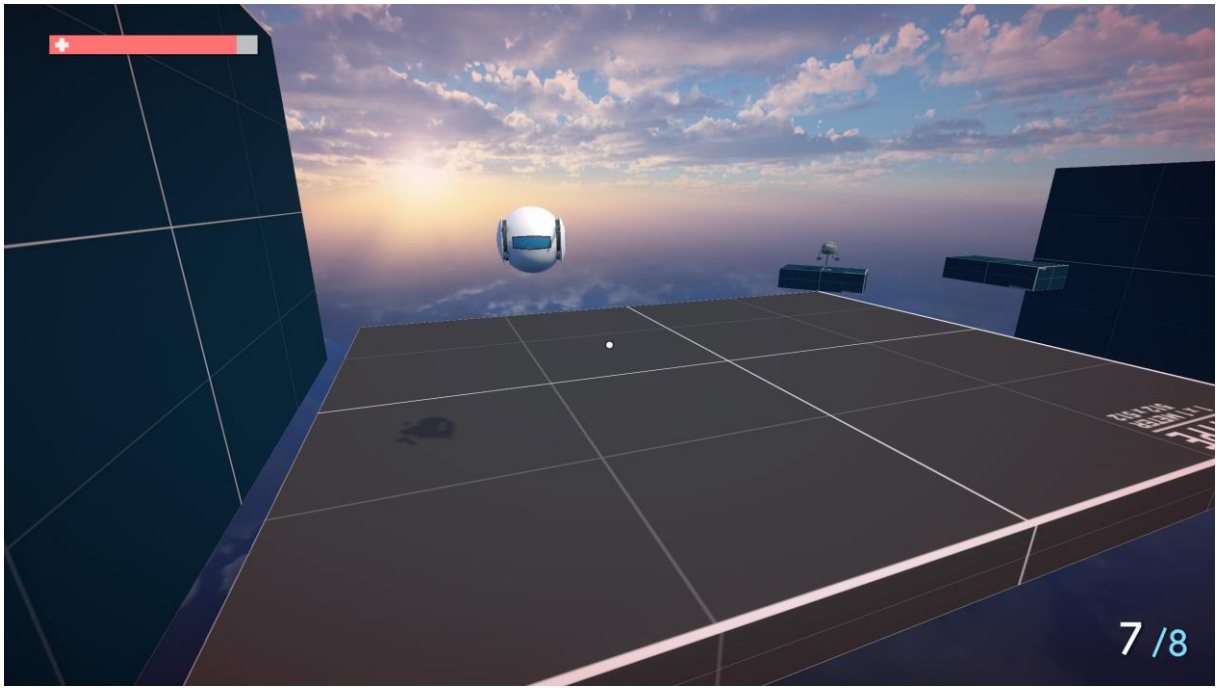
8.1 Seznam obrázků

9 Přílohy

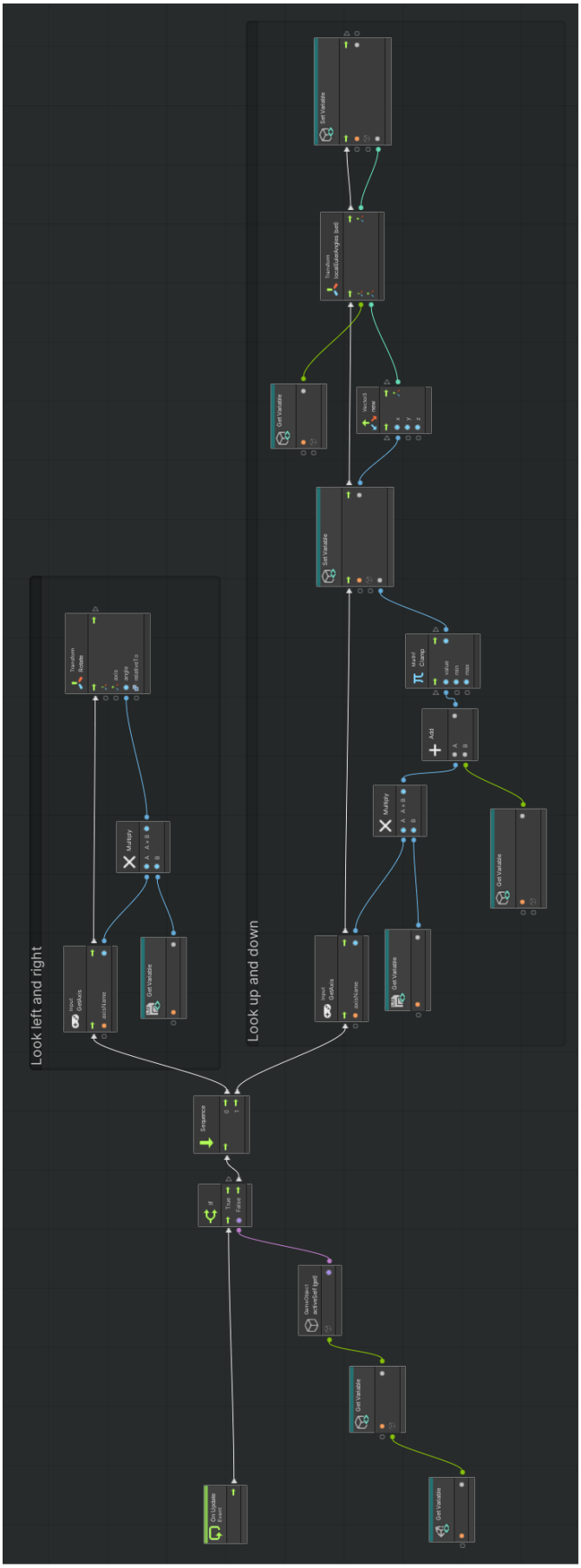
Odkaz ke stažení hry



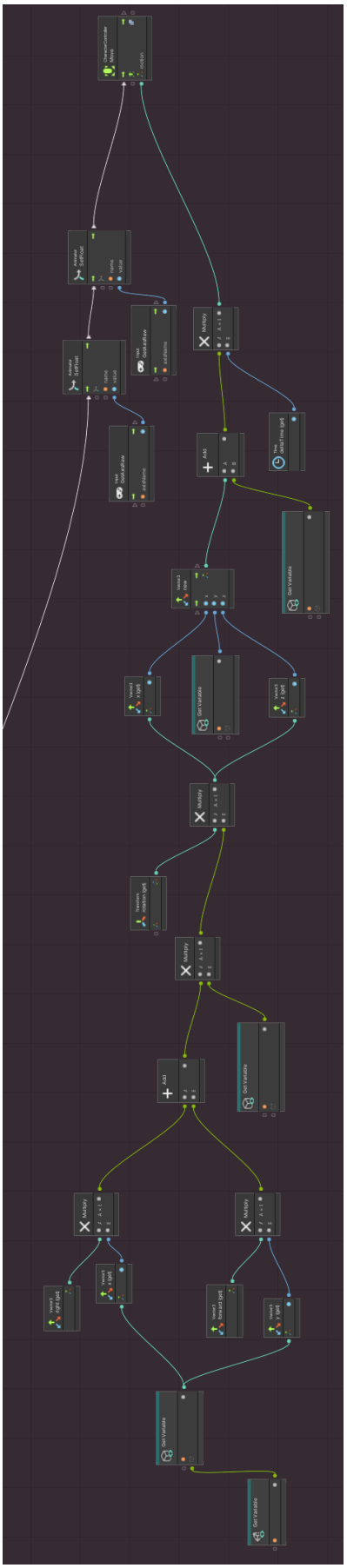
Obrázek 51 - Obrázek ze hry při skoku



Obrázek 52 - Obrázek ze hry při zranění od nepřátel



Obrázek 53 - Celková logika pohybu kamery



Obrázek 54 - Celková logika pohybu postavy