Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj 3D herní aplikace v Unreal Engine 4

Udovytskyi Maksym

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Maksym Udovytskyi

Informatika

Název práce

Vývoj 3D herní aplikace v Unreal Engine 4

Název anglicky

Development of a 3D game application in Unreal Engine 4

Cíle práce

Hlavním cílem je seznámit začínající vývojáře s procesem tvorby 3D akční herní aplikace v prostředí Unreal Engine a představit spravný postup při vývoji takové aplikace.
Při vývoji bude brán ohled na zkušenosti reálných uživatelů (UX), což může ovlivnit výsledný vzhled aplikace a zážitek z jejího používání.
Dílčím cílem práce je vytvoření samotné aplikace a implementace základních herních mechanik, fyziky a umělé inteligence pomocí základních nástrojů Unreal.

Metodika

Metodika zpracování teoretické části práce je založená na studiu odborných informačních zdrojů a na zvýraznění informaci, nezbytné v rámci vývoje počitačové hry v Unreal.

V praktické části všechno, prozkoumané v teoretické části, bude zaměřeno na vývoj počítačové hry v prostředí Unreal Engine 4 pomoci Blueprints (systému vizuálního skriptování), programovacího jazyku C++ a 3D softwaru Blender pro modelování herních lokaci a assetů.

Tato aplikace bude testována různými uživateli během vývoje, což může změnit směr vývoje aplikace.

Doporučený rozsah práce

35-40 stran

Klíčová slova

Herní aplikace, Unreal Engine, C++, Blueprints, 3D grafika

Doporučené zdroje informací

EPIC GAMES. Unreal Engine 4 Documentation [online]. [cit. 2023-06-12]. Dostupné z: https://docs.unrealengine.com/4.27/en-US/

MICROSOFT. C++ language documentation: Learn to use C++ and the C++ standard library. [online]. [cit. 2023-06-12]. Dostupné z: https://learn.microsoft.com/en-us/cpp/cpp/?view=msvc-170

NIXON, David. Beginning Unreal Game Development [online]. Berkeley, CA: Apress, 2020 [cit. 2023-04-30]. ISBN 978-1-4842-5638-1. Dostupné z: doi:10.1007/978-1-4842-5639-8

PLOWMAN, Justin. 3D Game Design with Unreal Engine 4 and Blender. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing, 2016. ISBN 978-1-78588-146-6.

SHERIF, William a Stephen WHITTLE. Unreal Engine 4 Scripting with C++ Cookbook. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing, 2016. ISBN 978-1-78588-554-9.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Tomáš Benda

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 2. 2024

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 13. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci " Vývoj 3D herní aplikace v Unreal Engine 4" 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 15.03.2024 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce Ing. Tomáši Bendovi za informativní konzultace a srozumitelný návod, jak tuto práci udělat lépe.

Vývoj 3D herní aplikace v Unreal Engine 4

Abstrakt

Tato bakalářská práce se zaměřuje na komplexní analýzu a implementace procesu vývoje herní aplikace v prostředí Unreal Engine 4. Práce se zabývá klíčovými aspekty vývoje her a poskytuje ucelený pohled na vývojový cyklus.

Teoretická část práce představuje Unreal Engine a jeho historii. Dále se popisují principy programování v Unreal Engine, agilní proces vývoje a související koncepty, jako je Technical Debt, User Experience a System Usability Scale. Podrobně jsou rozebrány jednotlivé fáze vývoje her, včetně zásadních Pre-production a Production, s důrazem na klíčové aspekty.

Praktická část práce popisuje postup tvorby konkrétní aplikaci s použitím teoretických poznatků v praxi. Pre-production fáze zahrnuje návrh hry a tvorbu Game Design Documentu. Production fáze se zabývá instalací a nastavením projektu, tvorbou a práci s herními assety, systémem verzování a dalšími aspekty vývoje. Podrobně jsou popsány jednotlivé kroky vývoje nezbytných herních funkci.

V průběhu vývoje bylo prováděno průběžné testování herních funkcí s cílem identifikovat a řešit potenciální problémy. Konečným krokem bylo provedení finálního hodnocení pomocí dotazníku System Usability Scale (SUS), který sloužil k celkovému vyhodnocení kvality a uživatelské přívětivosti vytvořené herní aplikace.

Celkově lze tato práce považovat za užitečný průvodce pro začínající vývojáře, kteří se chtějí seznámit s procesem vývoje herních aplikací na platformě Unreal Engine 4. Praktická implementace teoretických poznatků vytváří solidní základ pro úspěšný vývoj her a poskytuje užitečné poznatky a doporučení pro budoucí projekty v herním průmyslu.

Klíčová slova: Herní aplikace, Unreal Engine, C++, Blueprints, 3D grafika

Development of a 3D game application in Unreal Engine 4

Abstract

This bachelor thesis focuses on a comprehensive analysis and implementation of the game application development process in the Unreal Engine 4 environment. It deals with key aspects of game development and provides a comprehensive view of the development cycle.

The theoretical part of the thesis introduces Unreal Engine and its history. It describes the principles of programming in Unreal Engine, the agile development process, and related concepts such as Technical Debt, User Experience, and System Usability Scale. The individual phases of game development are thoroughly analyzed, including crucial Pre-production and Production phases, with an emphasis on key aspects.

The practical part of the thesis describes the process of creating a specific application using theoretical knowledge in practice. The Pre-production phase includes game design and the creation of a Game Design Document. The Production phase deals with project setup, creation and management of game assets, versioning system, and other development aspects. The individual steps of developing essential game functions are detailed.

Throughout the development, continuous testing of game features was conducted to identify and address potential issues. The final step involved conducting a final evaluation using the System Usability Scale (SUS) questionnaire, which served for an overall assessment of the quality and user-friendliness of the created game application.

Overall, this thesis can be considered as a useful guide for novice developers who want to familiarize themselves with the process of developing game applications on the Unreal Engine 4 platform. The practical implementation of theoretical knowledge creates a solid foundation for successful game development and provides useful insights and recommendations for future projects in the gaming industry.

Keywords: Game Application, Unreal Engine, C++, Blueprints, 3D graphics

Obsah

1 Úvod **11**

2 Cíl práce a metodika **12**

2.1 Cíl práce 12

2.2 Metodika 12

3 Teoretická východiska **13**

3.1 Unreal Engine 13

3.1.1 Historie Unreal Engine a základní informace 13

3.1.2 Programováni v Unreal Engine 14

3.2 Project Management 16

3.2.1 Agilní proces vývoje 17

3.3 Fáze procesu vývoje hry 17

3.4 Pre-production 18

3.4.1 Game Design Document 18

3.4.2 Prototyp 19

3.5 Production 19

3.5.1 Systém Kontroly Verzí 20

3.5.2 Game Level Design 22

3.5.3 Modelování a animace 24

3.5.3.1 Modelování 24

3.5.3.2 Animace 27

3.5.4 Texturování v Unreal Engine 29

3.5.5 Import 3D modelů a textur 30

3.5.6 Stavy hry/hráče 31

3.5.7 Audio 32

3.6 User Experience (UX) 33

3.7 System Usability Scale (SUS) 34

3.8 Post-production 35

4 Vlastní práce **36**

4.1 Pre-production fáze 36

4.1.1 Návrh hry 36

4.1.2 Game Design Dokument 36

4.2 Production-fáze 38

4.2.1 Instalace a nastavení projektu 38

4.2.2 Interface 40

4.2.3 Assety 41

4.2.4 Source Control 42
4.2.5 Práce s kamerou 43
4.2.6 Otáčení a pohyb 45
4.2.7 Stavy 46
4.2.8 Animace 48
4.2.9 Umělá inteligence 50
4.2.10 Bojová mechanika 52
4.2.11 Level 53
4.2.12 Audio 54

5 Výsledky a diskuse 55

6 Závěr 57

7 Seznam použitých zdrojů 58

8 Seznam obrázků, tabulek, grafů a zkratek 63
8.1 Seznam obrázků 63
8.2 Seznam tabulek 64
8.3 Seznam grafů 64

Přílohy 65
Příklady kódu 65
Snímky z prototypu hry 69

1 Úvod

V posledních letech si herní průmysl získává popularitu obrovským tempem. Před několika desetiletí byl obecný názor, že hry jsou aktivitou pro děti. Nyní se naopak fanoušky videoher stává stále více lidí všech věkových kategorií, od dětí až po seniory. Herní průmysl dosáhl vrcholu popularity během pandemie Covid a nezpomaluje. A je zřejmě, že takto populární odvětví může přinést obrovské výnosy vývojářským společnostem, které dosahují až miliard dolarů ročně a více.

Vývoj her je velmi obtížná práce, ale součástí týmu vývojářů her se nyní může stát doslova každý, protože k vytvoření hry již není nutné umět pouze programovat, jako tomu bylo před několika desetiletí. Vývoj her v dnešní době zahrnuje mnoho různých oblastí, jako je programování, 3D modelování, game design, práce se zvukem – například skladatele lze najmout na psaní hudby, a také i scénáristy, protože nyní je málokterá hra bez scénáře a syžetu.

A přesto ani tým profesionálů není podmínkou pro vytvoření hry. Značné procento vydaných her je produktem indie vývojářů (indie – independent), kteří své výtvory tvoří sami nebo v úzké skupině podobných nadšenců. V tom jim často, stejně jako velkým vývojářským skupinám, pomáhá herní engine (Unreal Engine, Unity atd.).

K těmto enginům existuje dokumentace a také obrovské množství různých výukových materiálů, ve kterých je možné se snadno splést, a proto vznikla tato práce, aby pomohla začátečníkům snadněji začít svou cestu do vývoje her. Tato práce je návodem k vytvoření jednoduché 3D akční hry, která pokryje všechny důležité fáze tvorby takové hry, aby čtenář pochopil, na co je nejlepší se na začátku cesty zaměřit, aby nezačal studovat vše za sebou, což může učení značně zkomplikovat.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem je seznámit začínající vývojáře s procesem tvorby 3D akční herní aplikace v prostředí Unreal Engine a představit spravný postup při vývoji takové aplikace. Při vývoji bude brán ohled na zkušenosti reálných uživatelů (UX), což může ovlivnit výsledný vzhled aplikace a zážitek z jejího používání.
Dílčím cílem práce je vytvoření samotné aplikace a implementace základních herních mechanik, fyziky a umělé inteligence pomocí základních nástrojů Unreal.

2.2 Metodika

Metodika zpracování teoretické části práce je založená na studiu odborných informačních zdrojů a na zvýraznění informaci, nezbytné v rámci vývoje počitačové hry v Unreal.
V praktické části všechno, prozkoumané v teoretické části, bude zaměřeno na vývoj počítačové hry v prostředí Unreal Engine 4 pomoci Blueprints (systému vizuálního skriptování), programovacího jazyku C++ a 3D softwaru Blender pro modelování herních lokaci a assetů.
Tato aplikace bude testována různými uživateli během vývoje, což může změnit směr vývoje aplikace.

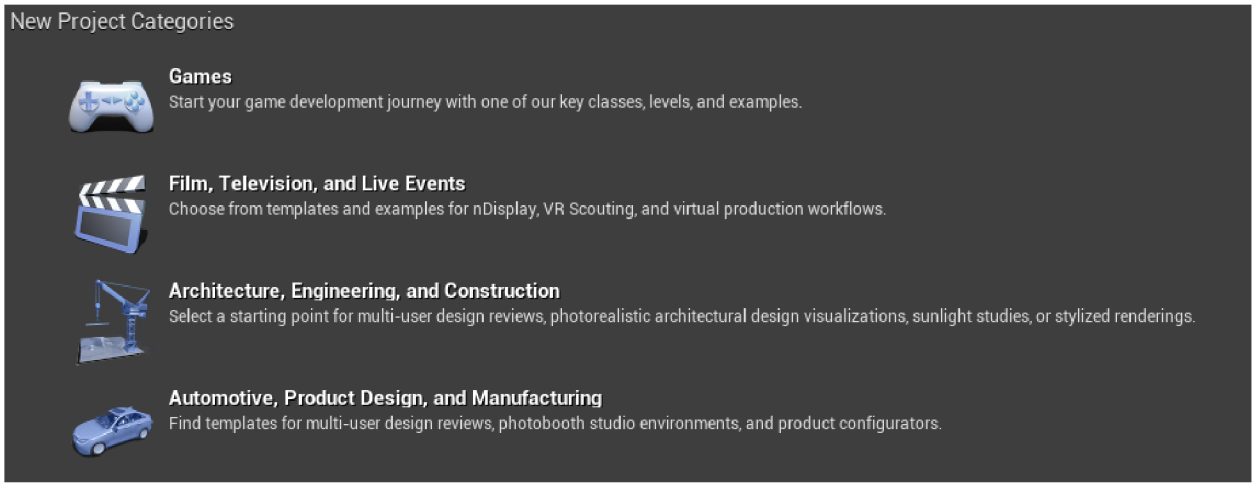
3 Teoretická východiska

3.1 Unreal Engine

3.1.1 Historie Unreal Engine a základní informace

Unreal Engine primárně byl enginem pro vývoj her v žánru shooter, vytvořený společnosti Epic Games. První hra, vytvořená pomoci tohoto enginu je "Unreal" – FPS (First Person Shooter), vydaný v roce 1998, odsud i pochází jméno enginu. Od té doby se engine mnoho vyvíjel, abych mohl podporovat jakýkoli žánr her. Engine se ale neomezuje jen na hry. Používá se také v architektuře, automobilovém průmyslu a mnoha dalších oborech, a navíc umožňuje vytvářet aplikace pro virtuální a rozšířenou realitu (Jensen, 2023).

Obrázek 1: Výběr kategorie při spuštění verze UE 4.27.2



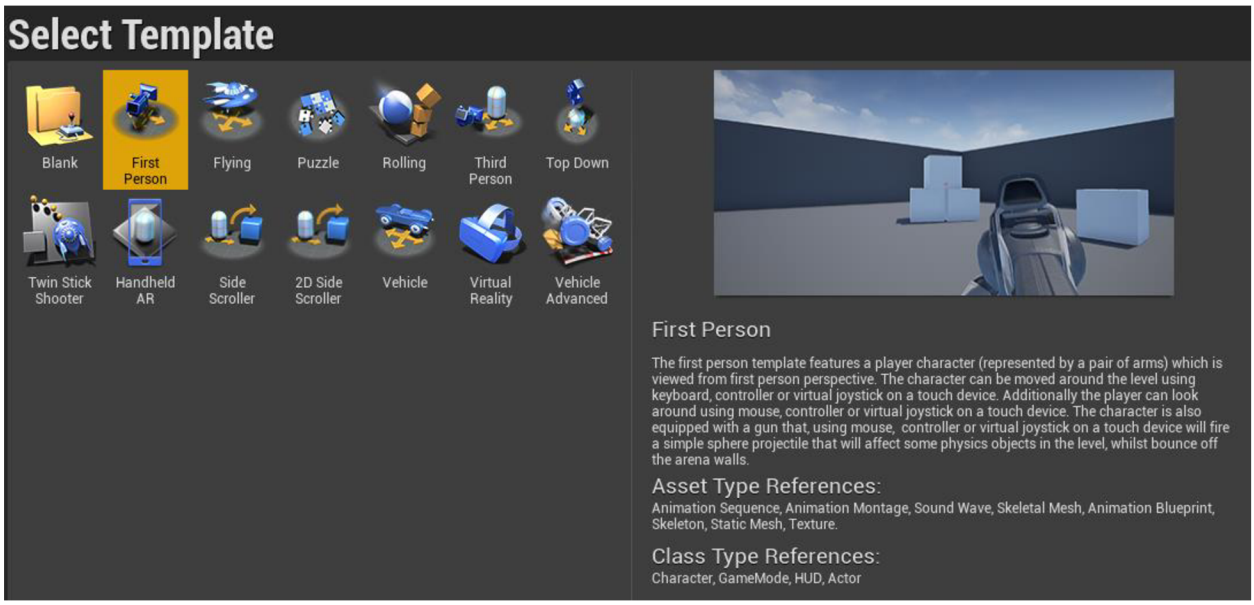
Zdroj: Autor

Odpověď Caleba Harwella (2020), vlastníka a vedoucího vývojáře Revolt Realities na otázku "Je možné použití Unreal Engine pro architekturu?":

„Absolutně, firmy to samozřejmě využívají neustále. Unreal má velmi výkonný editor v reálném čase, který opravdu dobře vykresluje i extrémně těžké objekty. Já bych navrhoval podívat se na assety UE4 na trhu, abyste přesně posoudili, jak lze ten engine použít. Má to velký potenciál, a vše, co vyžaduje, je znalost UE4 a vývojový pipeline.".

Bez ohledu na svou rozlehlost nemá tento engine tak vysokou bariéru vstupu a je přívětivý pro začátečníky. Unreal Engine je napsán na C++, což poskytuje stabilitu a výbornou alokaci paměti, kromě toho engine má velmi vysokou úroveň portability.
V Unreal Engine pro vývoj her se používá také C++, ale ačkoli se široce používá po celém světě, C++ se považuje za poměrně složitý programovací jazyk, což ztěžuje proces učení, ale také poskytuje více možností pro tvorbu hry. Unreal Engine je pro běžné uživatele zcela zdarma. Uživatel potřebuje pouze nainstalovaný pomoci oficiálního spouštěče engine, aby mohl začít pracovat. Díky obrovskému počtu vývojářů využívajících engine existuje stejně velká rozmanitost výukových materiálů o různých aspektech tvorby hry na tomto enginu. Unreal Engine poskytuje vývojářům her komplexní sadu nástrojů pro navrhování, modelování a vykreslování 3D obsahu. Obsahuje grafický engine pro rendering, fyzikální engine pro realistickou interakci, zpracování zvuku, a také jazyk programovaní C++ a skriptovací jazyk Blueprints pro herní logiku. Engine podporuje vysokou úroveň detailů a komplexních animací, což umožňuje vývojářům vytvářet pohlcující a dynamické herní zážitky (Erolin, 2022).

Obrázek 2: Nabídka šablon pro hru



Zdroj: Autor

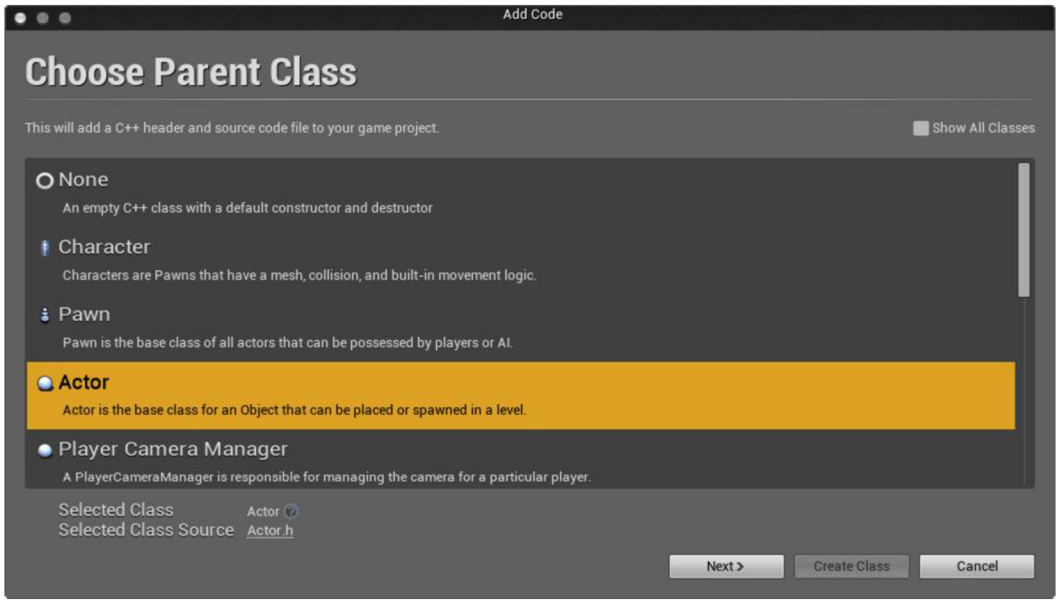
3.1.2 **Programováni v Unreal Engine**

Engine poskytuje výběr ze dvou možností programování hry, a to programovací jazyk C++ a systém vizuálního skriptování Blueprints.

Vývojáři mohou sami se rozhodnout, co přesně chtějí při vývoji použít, C++ nebo Blueprints. Obě metody mají klady i zápory, ale nejčastější názor je, že je třeba je vzájemně kombinovat, aby dosáhnout nejlepšího výsledku. Výhody Blueprints spočívají v tom, že oni umožňují velmi rychle vytvořit funkční prototyp něčeho, například nějaké herní mechaniky. Lze je také snadno ladit, a to je důležitá výhoda pro herní designéry, kteří mohou snadno a rychle měnit parametry objektu, než by se dostali do kódu. C++ na svou stranu má nepopiratelnou výhodu v rychlosti své práce a také se lépe hodí pro psaní herní logiky (Epic Games, 2024a).

Po vytvoření C++ třídy lze jí rozšířit pomocí Blueprints, což umožňuje programátorům vytvářet nové herní třídy v kódu, které mohou herní designéři snadno upravovat. Tento proces vždy začíná vytvořením C++ třídy pomoci Class Wizardu, která bude později rozšířena v Blueprints (Epic Games, 2024b).

Obrázek 3: Class Wizard



Zdroj: Epic Games (2024)

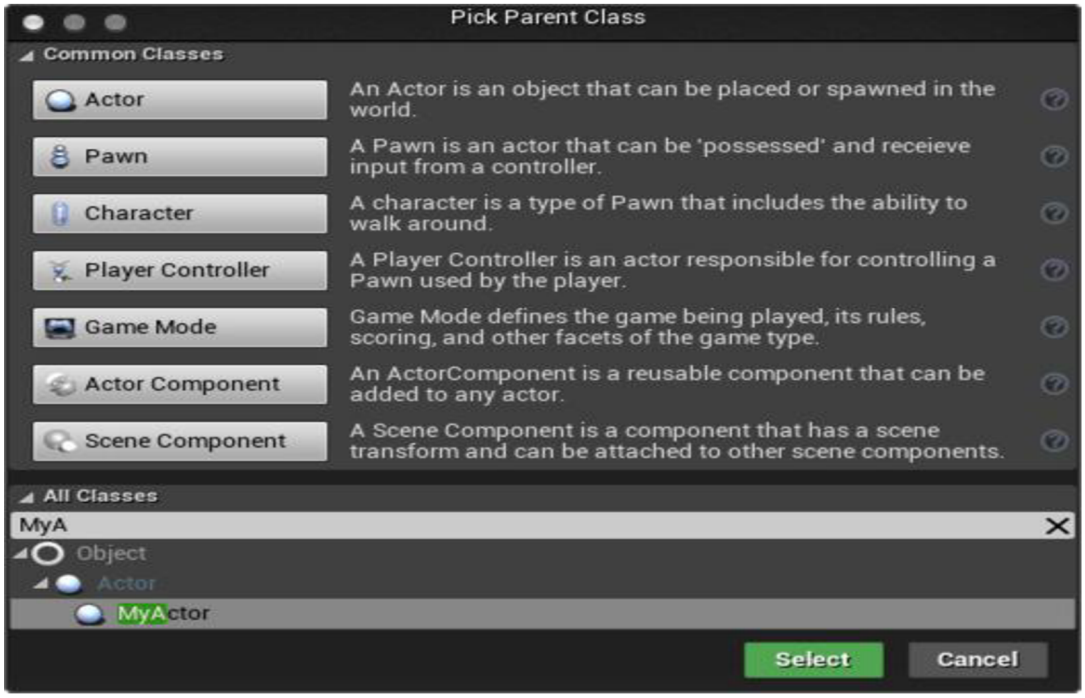
Při vytváření třídy Unreal Engine vygeneruje všechny potřebné soubory pro zahájení práce s novou třídou a také okamžitě otevře IDE (vývojové prostředí). Nová třída je generována s metodami BeginPlay a Tick, které lze změnit tak, aby vyhovovaly konkrétním potřebám vývojářů (Nepor, 2023).

BeginPlay: To je událost, která se spustí pouze tehdy, když je ve hře spuštěna instance třídy, což znamená, že logika zapsaná v BeginPlay nebude fungovat v případech, kdy hra nebo úroveň nebyla spuštěna nebo pokud instance třídy není ve hra (Packt, 2023).

Tick: Logika zapsaná v ticku se standardně volá každý snímek ve hře. Pokud již není potřeba, dobrým doporučením by bylo odstranit tuto funkci pro ušetření počítačových zdrojů (Packt, 2023).

Další logiku už píše sám programátor. Po implementaci herní logiky do třídy je dalším vývojovým krokem rozšíření třídy pomocí Blueprints. Pomocí Class Wizardu se vytvoří Blueprints třída, která dědí z požadované třídy C++ (Epic Games, 2024b).

Obrázek 4: Dědění z C++ třídy



Zdroj: Epic Games (2024)

3.2 Project Management

Vývoj her je složitý proces, který vyžaduje účast mnoha různých specialistů. Projektové řízení v tomto odvětví je zásadní pro zamezení promeškaných termínů a rozpočtů. Řízení projektu zahrnuje plánování, alokaci zdrojů a komunikaci. Schopnost přizpůsobit se měnícím se požadavkům a technologiím je také důležitá pro zajištění úspěchu hry (Teamhub, 2023).

3.2.1 Agilní proces vývoje

Agilní vývojový proces je metodika vývoje softwarů, která dělí vývojový proces do několika iteraci, což umožňuje flexibilně reagovat na změny požadavků v průběhu životního cyklu projektu. Týmy pracují v krátkých cyklech nazývaných sprinty, během nichž produkují funkční kód a výsledky této práce jsou testovány a ověřovány uživateli. Klíčové faktory úspěchu zahrnují jasně definovaný backlog(nahromadění) úkolů, častou integraci a minimalizaci technického dluhu (Microsoft, 2023).

Klíčovým rysem procesu agilního vývoje je myšlenka technického dluhu (TD) a myšlenka uživatelské zkušenosti (UX), která se za tím vyvinula jako jedna z metod agilního vývoje (Rodriguez et al., 2023).

Technical Debt (TD): V agilním vývoji se technický dluh často hromadí jako krátké vývojové cykly, aby se zjednodušil konečný návrh produktu, aby mohl být co nejrychleji výroben. V průměru se technický dluh odhaduje na 23 % dodatečného času, který vývojáři ztratí na prací. Technický dluh navíc zahrnuje i úroky a ukazuje se, že jistina je definována jako efektivní náklady na renovaci technického dluhu a úroky jsou dodatečné úsilí potřebné v budoucnu k údržbě softwaru s nahromaděným technickým dluhem (Rodriguez et al., 2023).

3.3 Fáze procesu vývoje hry

Každý, kdo má praktické zkušenosti s videoherním průmyslem, zná jednoduchý fakt: proces vývoje her může být chaotický. Ačkoliv žádné plánování nemůže úplně eliminovat produkční překážky či náročné termíny, pustit se do projektu vývoje hry bez jasně stanoveného plánu je spolehlivý způsob, jak se neúspěchu vyhnout. Ať už je vývojářská společnost herní studio AAA nebo nezávislý vývojář, je nezbytný strukturovaný proces vývoje hry (Nuclino, 2023a).

Proces vývoje hry je rozdělen do fází, od vzniku konceptu hry až po jeho samotné dokončení. Bez ohledu na rozsah vyvíjeného projektu, ať už jde o nezávislou hru nebo AAA projekt, je proces vždy velmi podobný a je rozdělen do tří fází: předprodukce (pre-production), produkce (production) a postprodukce (post-production). Takové důležité fáze vývoje hry by měly být prozkoumány podrobněji (Stefyn, 2022).

3.4 Pre-production

Vývoj jakéhokoli projektu vždy začíná plánováním projektu. V této fázi jsou stanoveny základní koncepty projektu, vytvořeny prototypy projektu, sepsány představy o tom, o jaký projekt by se mělo jednat a sestaven akční plán. V této fázi musí vývojáři odpovědět na otázky jako (Stefyn, 2022):

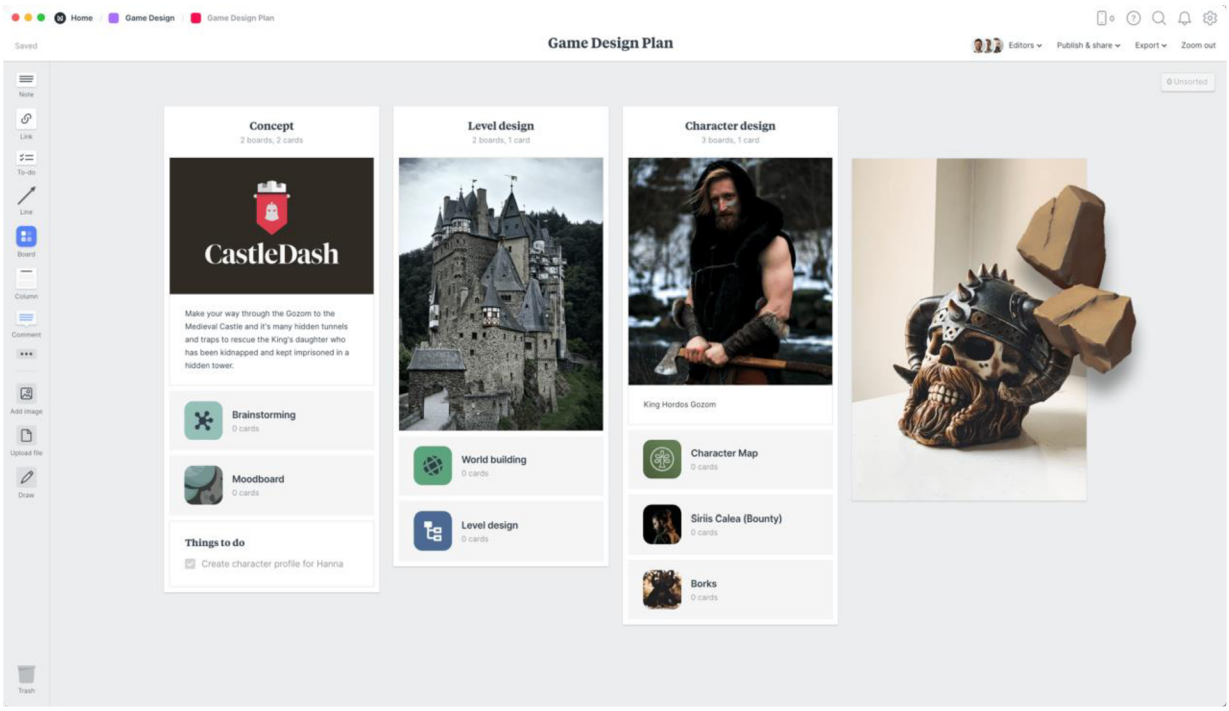
- Jaká je hlavní myšlenka hry?
- Jaký je žánr hry?
- Kdo je cílová skupina?
- Kdy a kde se hra odehrává?
- Kdo jsou postavy?
- Jaké jsou naše odhadované náklady na vývoj této hry?
- Budeme muset najmout herní vývojáře nebo další členy týmu?
- Jak můžeme hru zpeněžit?
- Jaký je náš odhadovaný časový rámec pro spuštění hry?

Odpovědi na tyto otázky pomohou rozhodnout o nápadu na hru, ale dobrý nápad a schopnost napsat kód k vytvoření hry nestačí. Vývoj hry bez dokumentu o herním designu může v budoucnu vést k nepříjemným následkům, takže před zahájením práce by si každý vývojář měl zmapovat plán, co chce vyvinout (Nuclino, 2023b).

3.4.1 Game Design Document

Game Design Document (GDD) je plánem, jak vytvořit svou hru. Nastiňuje plány vývojářů pro postavy, prostředí, hratelnost, úrovně, vizuální styl a mnoho dalšího. Dříve byli psány jako dlouhé textové dokumenty. Moderní GDD jsou vizuální a snadno použitelné. Jsou plné inspirativních snímků, video ukázek hry, a dokonce i zvuku, které pomohou týmu vývojářů vizualizovat konečný produkt (Milanote, 2024).

Obrázek 5: Moderní GDD



Zdroj: Milanote (2024)

3.4.2 Prototyp

Během Pre-production fáze je také běžné prototypovat prostředí, postavy, ovládací schémata a další prvky ve hře. Do budování světu je investováno velké úsilí. Nápady jsou rozvedeny ve formě scénářů, konceptu a maketu rozhraní, aby možné bylo vidět, jak vypadají, jak se cítí a jak se vzájemně ovlivňují (Nuclino, 2023a).

3.5 Production

Produkční fáze je nejdelší fází vývoje, zabírá většinu času na vytvoření hry a obvykle trvá 1 až 4 roky. V této fázi se hra začíná formovat, jsou realizovány všechny nápady nastolené v předprodukční fázi. Do hry je vetkán děj, staví se celé světy, vytvářejí se aktiva, která tyto světy naplňují, jsou stanovena pravidla hry a je napsán kód. V této fázi se rozlišují hlavní fáze vývoje (Stefyn, 2022):

- **První hratelné:** V této fázi prototyp začíná nabývat vzhledu skutečné hry. Náčrty jsou nahrazeny hotovými assety. Taková podoba projektu podněcuje vývojáře, aby přicházeli s novými nápady, co by mohlo vypadat lépe a jak tomu lze přizpůsobit gameplay.

- **Vertical slice:** Tato fáze obsahuje plně hratelný vzorek, který se obvykle používá k předvedení hry investorům. Tento vzorek již může poskytnout velmi jasnou představu o tom, čím tento projekt je, jak bude vypadat a co se v něm dá dělat.
- **Pre-alfa:** Většina práce leží v této fázi. Většina obsahu je připravena a je čas na důležitá rozhodnutí. Jaká část obsahu by měla být změněna nebo zcela odstraněna. Co ještě dodat do hry pro zpestření, případně pro vyplnění nedostatečné hratelnosti.
- **Alpha:** hra je kompletně naplněna všemi potřebnými mechanikami a funkcemi a hru lze hrát od začátku až do konce. Možná bude nutné vyměnit některé detaily, jako jsou assety, ale ovládací prvky a funkce hry by měly fungovat správně. V této fázi se do práce aktivně pouštějí testeři, kteří kontrolují, zda vše funguje správně a hladce, a v případě potřeby hlásí chyby vývojovému týmu.
- **Beta:** V tuto chvíli je veškerý content připraven a integrován, tým se nyní musí více zaměřit na optimalizaci produktu než na nové funkce.
- **Release:** Vývoj produktu byl dokončen a hra je připravena k vydání k prodeji.

3.5.1 Systém Kontroly Verzí

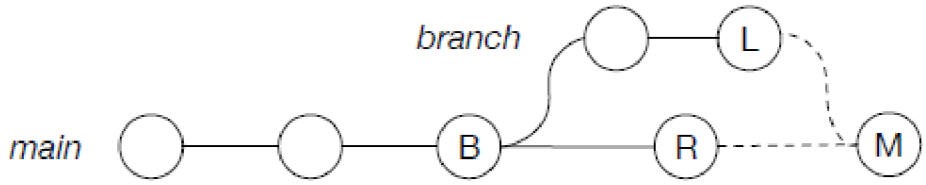
Úspěšný vývoj jakéhokoli produktu závisí na schopnosti týmu koordinovat své akce. Systém správy verzí umožňuje vývojovým týmům paralelizovat úkoly a rychle identifikovat nekonzistence v práci. Součástí funkčnosti systému správy verzí je schopnost slučovat větve. Když se slučují dvě větve s jejich vlastními změnami v každé, získává se nova verze, která kombinuje změny z každé větve. Slučování větví je založeno na různých algoritmech, což vede k různým výsledkům. Jednou z takových slučovacích technik je three-way merge pattern (Santos a Teles, 2023).

Three-way merge pattern:

Česky třícestné slučování je také známe jako pull-based model a merge scenario. Tento model spočívá ve větvení hlavního úložiště do brančů. Dále vývojáři, nezávisle na jiných vývojářích, provádějí své změny, jako například zavedení nových funkcí do produktu. Po dokončení práce potvrzuje se sloučení větev a konečný výsledek skončí v hlavním úložišti. Existují i jiné způsoby, jak integrovat kód do projektu, ale mohou bránit

ve sledování historie změn projektu v budoucnu. Pro usnadnění se tedy obvykle používá třícestný model (Vale, 2023, s. 4).

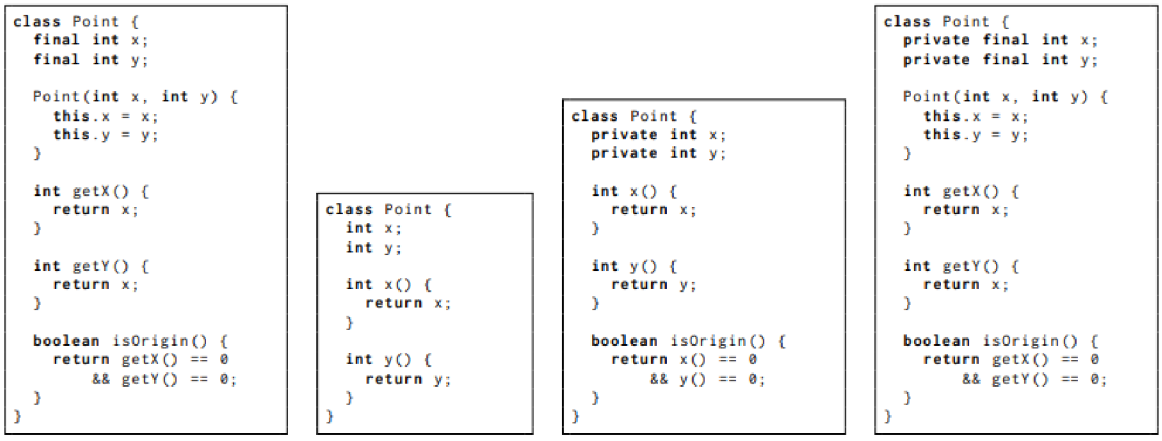
Obrázek 6: Sloučení větev v Git



Zdroj: Santos a Teles (2023)

Bohužel ne vždy sloučení větví proběhne bez problémů. Pokud jeden nebo více vývojářů změní stejnou část kódu v různých větvích, vznikne konflikt sloučení. K nápravě takového konfliktu je potřeba jej vyřešit například výběrem změn pouze z jedné větve, nebo provedením nových změn (Vale, 2023, s. 71).

Obrázek 7: Ilustrace sloučení dvou různých větví do jedné konečné



```
class Point {
  final int x;
  final int y;

  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  int getX() {
    return x;
  }

  int getY() {
    return x;
  }

  boolean isOrigin() {
    return getX() == 0
        && getY() == 0;
  }
}
```

(a) Left branch version.

```
class Point {
  int x;
  int y;

  int x() {
    return x;
  }

  int y() {
    return y;
  }
}
```

(b) Base version.

```
class Point {
  private int x;
  private int y;

  int x() {
    return x;
  }

  int y() {
    return y;
  }

  boolean isOrigin() {
    return x() == 0
        && y() == 0;
  }
}
```

(c) Right branch version.

```
class Point {
  private final int x;
  private final int y;

  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  int getX() {
    return x;
  }

  int getY() {
    return x;
  }

  boolean isOrigin() {
    return getX() == 0
        && getY() == 0;
  }
}
```

(d) Ideal merged version.

Zdoj: Santos a Teles (2023)

Binary Assets:

Hry obvykle obsahují kombinaci textových a binárních souborů. Například textové soubory jsou kódové a konfigurační soubory. Assety, textury a herní úrovně mohou být binární. Velkým problémem při práci se systémy kontroly verzí je to, že všechny konflikty sloučení jsou řešeny v textovém formátu, zatímco u binárních souborů to nelze provést, což nutí pouze jednoho vývojáře pracovat na konkrétním binárním souboru, takže to nevede ke konfliktu, který nelze vyřešit. Situace, když více než jedna osoba, která se snaží

pracovat se stejném binárním souborem, často vede ke ztrátě práce jednoho vývojáře v případě konfliktu. Některé systémy správy verzí mohou tento problém částečně vyřešit uzamčením přístupu k binárnímu souboru, takže na něm může pracovat pouze jedna osoba (Code Capers, 2016).

3.5.2 Game Level Design

Level neboli úroveň ve hrách je určité místo, ve kterém se odehrávají hlavní herní akce. Level design je jednou z vývojových fází, během které se vytváří samotná lokace pro hráče, mise a cíle pro hráče na této mapě. Celkovým cílem level designu je vytvářet interaktivní situace a události, které by hráč mohl zažít, a to vzbudí v něm zájem do hry. Někdy je level design zaměňován s designem prostředí a jsou považovány za synonyma, i když v mnoha malých vývojářských studiích je jeden designér zodpovědný za oba typy designu, jedná se o dva různé koncepty (Nuclino, 2023c):

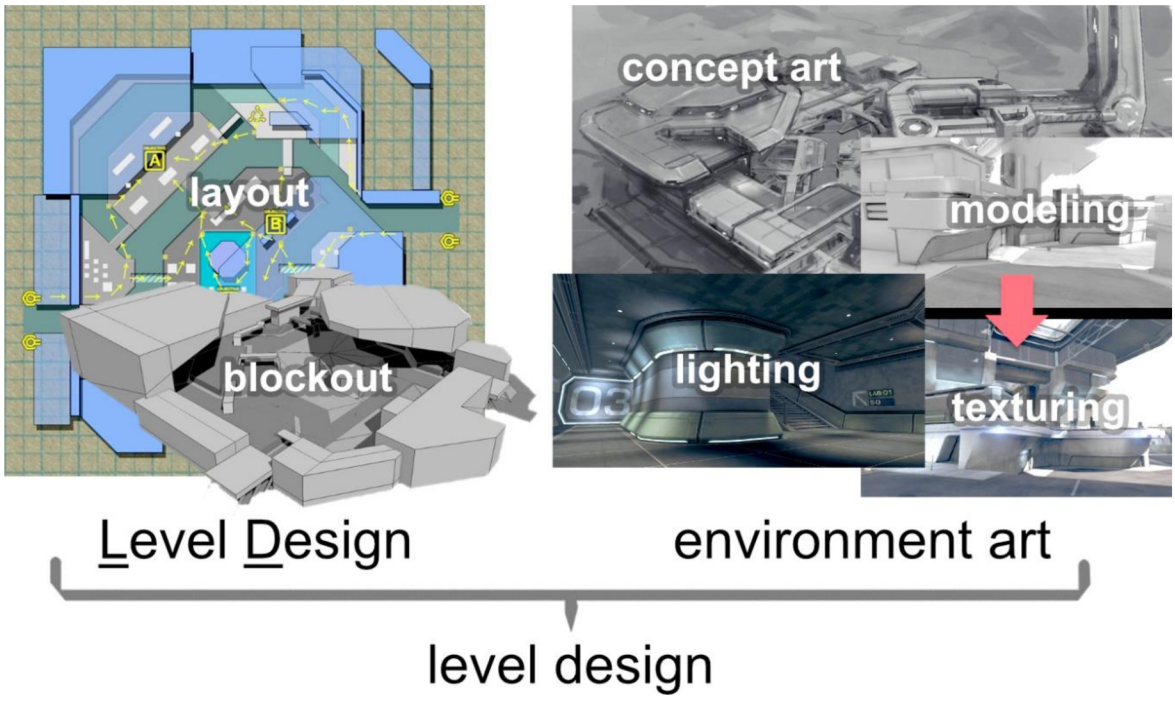
- **Design prostředí** je zaměřen pouze na kompozici jako celek
- **Level design** propojuje všechny prvky úrovně, přímo ovlivňuje uživatelský zážitek, určuje herní mechaniky, gameplay, umisťuje překážky a pomáhá odhalit historii herního světa.

Většina 3D level designu v projektech je rozdělena do těchto procesů (The Level Design Book, 2023):

- **Pre-Production:** Je vytvořen plán, přicházejí nápady pro úroveň.
- **Combat:** Volitelná fáze pro hry s bojovým systémem. Zde se rozhoduje o tom, jak a kde bude hráč interaktovat s nepřítelem, ať už je to umělá inteligence nebo člověk.
- **Layout:** Vytvoří se náčrty, 2D plán úrovně s pohledem shora.
- **Blocking out:** Konstrukce základního hrubého 3D tvaru úrovně a další testování této úrovně.
- **Scripting:** Představení událostí a interaktivních objektů (mise, triggery, dveře, AI atd.).
- **Lighting:** Nastavení vhodného osvětlení.

- **Environment Art:** Zobrazení všech připravených assetů a dekorací a dokončení úrovně s nimi.
- **Release:** Zveřejnění projektu.

Obrázek 8: Fáze Level Designu



Zdroj: The Level Design Book (2023)

Blocking out v Unreal:

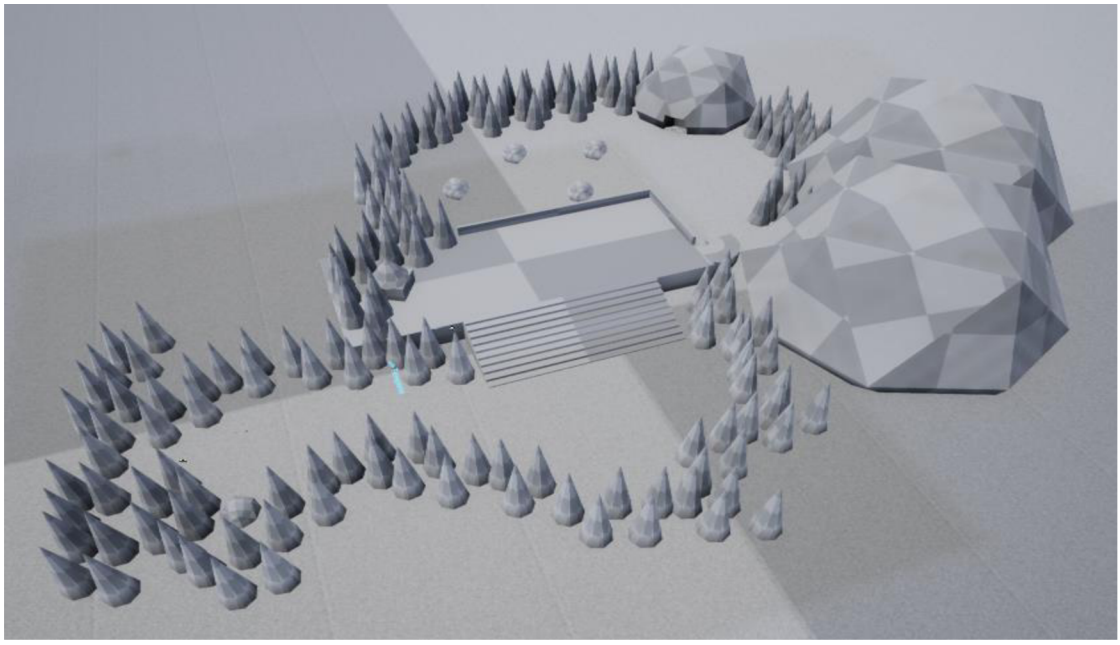
V Unreal Engine se Blocking out obvykle provádí pomocí nástroje Brush (štětec). V kontextu 3D modelování znamená Brush téměř totéž jako Mesh(3D reprezentace objektu sestávající z kolekce vrcholů a polygonů), ale s některými rozdíly:

- Brushem jsou nejzákladnější a nejprimitivnější tvary.
- Specifičnosti štětců je to, že zabírají více paměti počítače, ale jejich výhodou oproti Meshům je snadná úprava takových štětců.

Důležitým nastavením pro štětec je jeho typ, aditivní nebo subtraktivní. Aditivní přidává geometrii do úrovně a subtraktivní ji z úrovně odstraňuje, což znamená, že vyprázdní prostor v sobě. Nastavení Hollow způsobí, že štětec je uvnitř prázdný, což je skvělý nástroj pro rychlé vytvoření místnosti nebo budovy (Nixon, 2020, s. 92).

Je důležité si uvědomit, že ačkoliv byly štětce v minulosti používány jako hlavní nástroj pro budování úrovní, nyní byly z velké části nahrazeny statickými meshemi. Nyní je však stále doporučeno používat štětce v raných fázích vývoje k rychlému prototypování úrovní a objektů a v budoucnu je nahradit hotovými meshemi (Epic Games, 2024c).

Obrázek 9: Prototyp první úrovně



Zdroj: Autor

3.5.3 **Modelování a animace**

3D modelování a animace jsou nedílnou součástí vývoje téměř každé moderní hry. Kvalitní modely a animace významně přispívají k estetice, funkčnosti a použitelnosti hry. Pro vývoj her je rozhodující používání spolehlivého 3D softwaru. Patří mezi ně například Blender, Maya a 3ds Max, které jsou schopny produkovat kvalitní výsledky odpovídající potřebám vývojářů a vidění hry. V konečném důsledku může správný vývojový software pomoct výrazně zlepšit celkovou kvalitu a úspěch hry, ale jak modelování, tak animace jsou složité procesy s mnoha detaily, které je třeba si uvědomit (Taylor, 2023).

3.5.3.1 Modelování

3D modelování pro hry spočívá ve vytváření assetů, které se přidávají na herní scénu. Assetem může být cokoliv od textur po postavy a urovně. Proces vytváření 3D modelu často začíná náčrtem, který určuje samotný koncept modelovaného objektu. Následuje proces tvorby 3D modelu v některém ze speciálních 3D editorů. Poté následuje

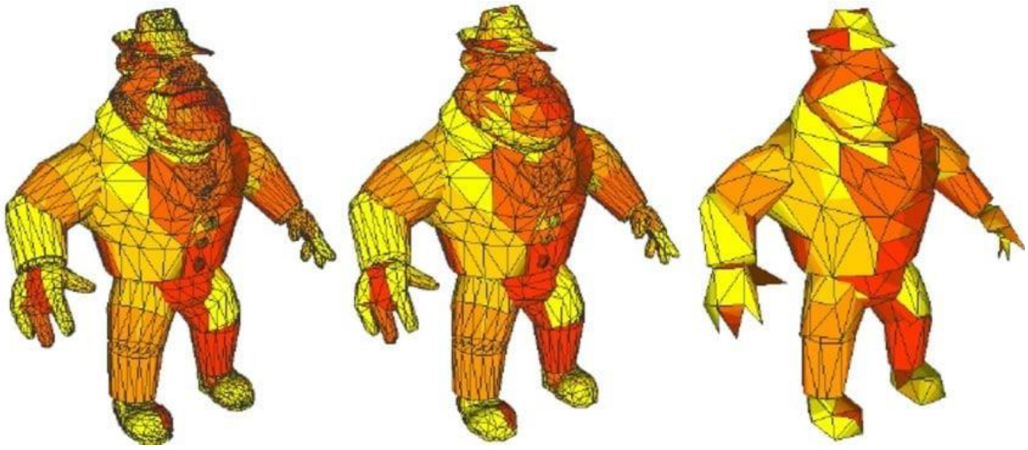
texturování vytvořeného objektu. V případě statických objektů tím práce končí, ale u dynamických objektů, jako například postava, musí být modely riggované a animované. Tyto funkce obvykle zajišťuje herní engine (Fuentes, 2022).

Typy a techniky:

Profese 3D umělce se dělí na různé specializace a využívá se v různých oblastech. Existují umělci, kteří vytvářejí prostředí, vytvářejí postavy a vozidla. Každá z těchto oblastí také používá různé techniky, jako je organické modelování, modelování tvrdých povrchů nebo procedurální modelování (Fuentes, 2022):

- **Level of Detail (LOD):** Česky úroveň detailů – je to speciální technika, která umožňuje optimalizovat hru a usnadňuje tak práci počítače. V zásadě tato technika zahrnuje snížení počtu polygonů, které má model, čím dále ten model je od hráče. Po přiblížení je model nahrazen verzí s vysokým počtem polygonů. To usnadňuje proces vykreslování objektů, protože na vzdálených objektech a objektech v pozadí již není úroveň detailů tak důležitá a patrná.

Obrázek 10: Low-poly modelování



Zdroj: Ripolles (2006)

- **Procedurní Modelování:** Jedná se o způsob vytváření dat spíše algoritmicky než ručně, s použitím připravené základny assetů. Skvělým příkladem je přidání trávy do hry, protože není praktické vytvářet každé stéblo trávy ručně. Správným řešením by zde bylo použít několik připravených modelů a rozmístit je po celém místě.

Obrázek 11: Procedural modelling

Zdroj: Schwarz a Müller (2015)

- **Sculpting:** Sochařství je preferovanou metodou pro organické modelování. Používá se hlavně k vytváření postav s vysokým počtem polygonů, které poskytují dobré podrobnosti o jejich anatomii.

Obrázek 12: 3D Sculptures

Zdroj: sotn1ks (2020)

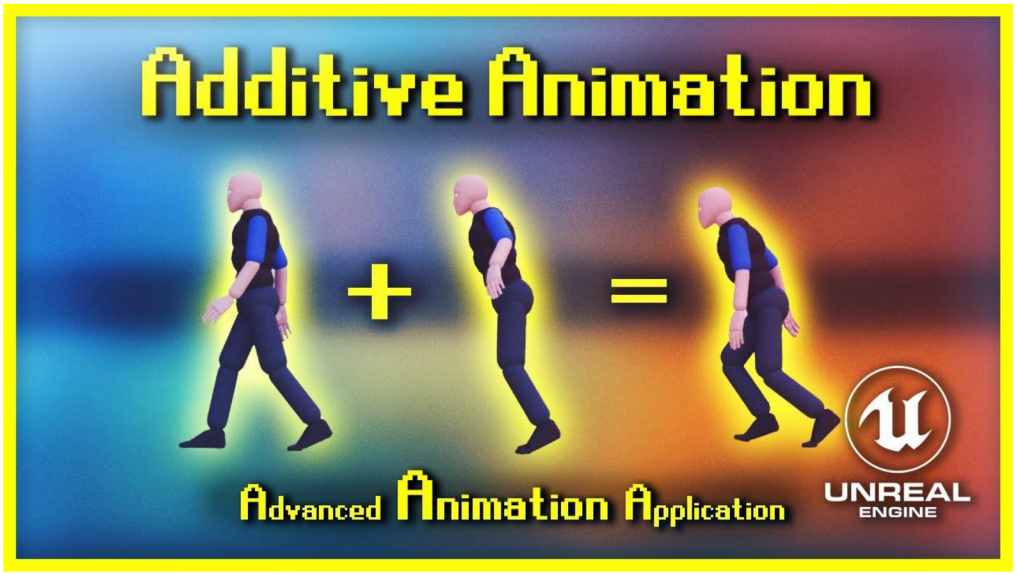
- **Hard-Surface Modelling:** Tato technika pro vytváření hladkých povrchů, a je populární pro sci-fi hry. Používá se hlavně pro modelování vozidel nebo zbraní.

- **Voxel Modelling:** Voxel modelování je fakticky 3D pixel art. Pro modelování se používají voxely (bloky).

3.5.3.2 Animace

Animace ve videohrách je pohyb postav, předmětů a prostředí. Herní animátoři používají k realizaci těchto animací různé techniky a programy. Tento proces začíná skicováním a končí modelováním v programech jako Autodesk a Blender. Obecně řečeno, animace, jako u 2D nebo 3D modelu, je sekvence snímků nakreslených nebo modelovaných objektů, které jsou smíchány, propojeny a divák nebo hráč vidí animovaný, zdánlivě „živý“ objekt. Tomu se říká aditivní animace a ve videohrách animace se provádí pomocí herních enginů, které určují, jak bude postava nebo objekt komunikovat se světem a jaké animace by měly hrát (Toronto Film School, 2023).

Obrázek 13: Additivní Animace



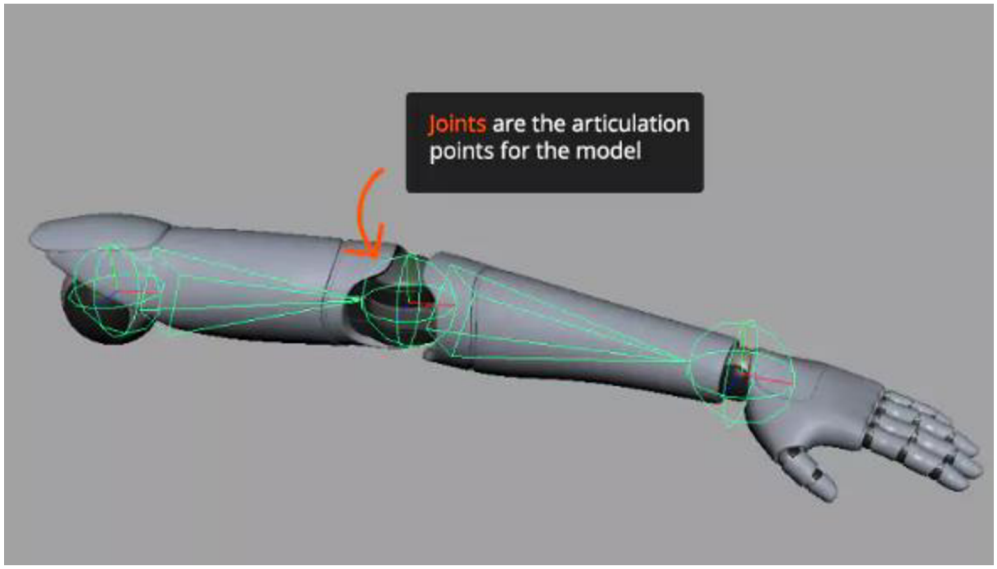
Zdroj: PrismaticaDev (2022)

Před začátkem animace jakékoli postavy, je nutné nejprve začít s riggingem. 3D rigging je proces vytváření kostry pro postavu, v důsledku čehož lze postavu již posouvat a animovat, protože jinak se postava bude problematicky deformovat a pohybovat v prostoru (PluralSight, 2023).

Rigging používá následující prvky:

- **Joint:** Kloub, také známý jako kost. Funguje to stejně jako lidské klouby. V místech, kde se 3D model potřebuje pohybovat a ohýbat, je umístěna tato kost, která nastavuje osu rotace pro končetinu.

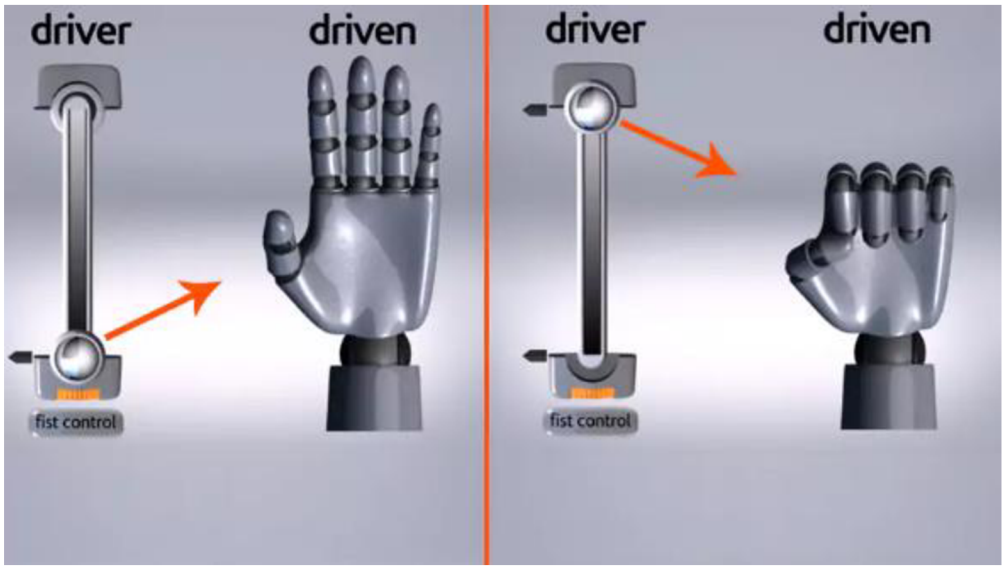
Obrázek 14: Joint illustration



Zdroj: Pluralsight (2023)

- **Driven Keys:** Během riggingu se pro urychlení procesu vytváření animace používají klíče k ovládání různého počtu objektů, například ke změně polohy ruky.

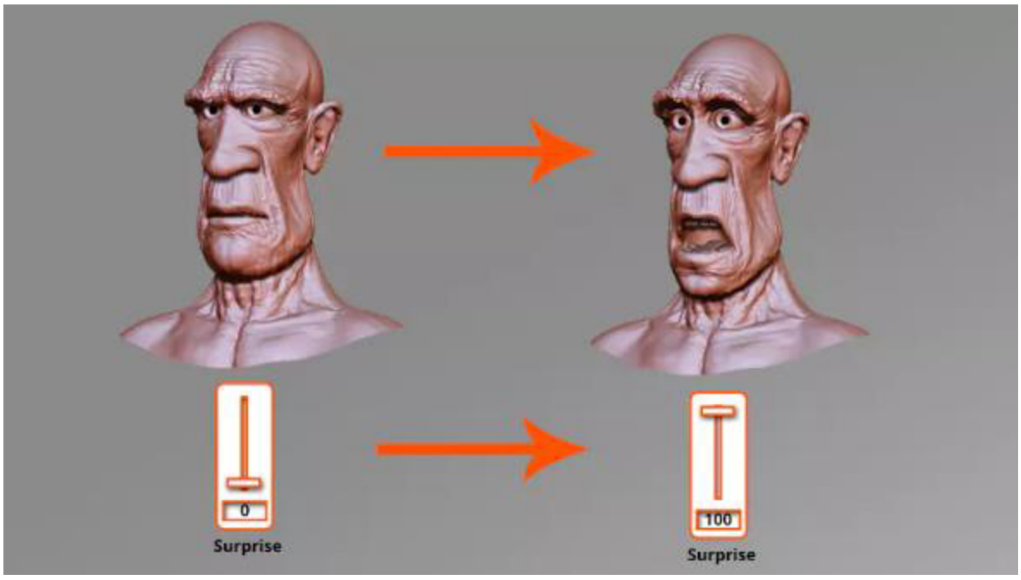
Obrázek 15: Driven Keys



Zdroj: Pluralsight (2023)

- **Blend Shape:** Tato technika umožňuje změnit tvar jednoho předmětu na tvar jiného. Obvykle se používá v animaci obličeje.

Obrázek 16: Blend Shape

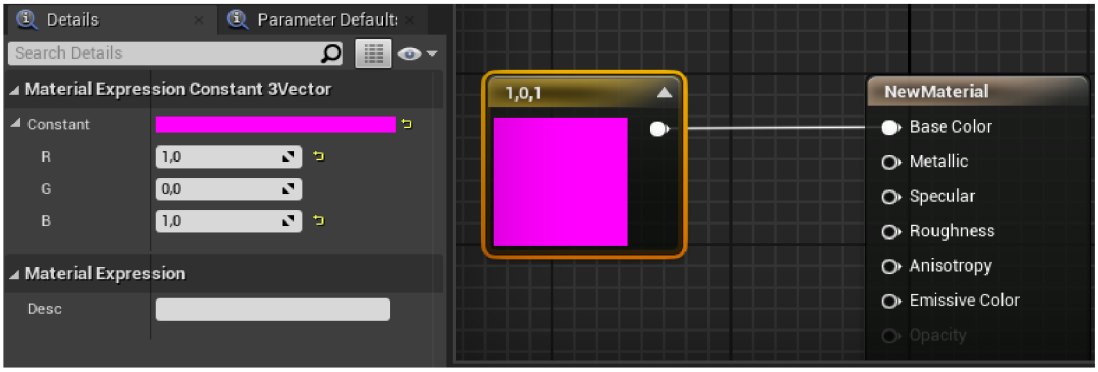


Zdroj: Pluralsight (2023)

3.5.4 Texturování v Unreal Engine

Textury jsou jakékoli obrázky použité v assetu Materiál a jsou aplikovány na povrch objektu, na který je tento materiál aplikován. Unreal umožňuje vytvořit primitivní texturu nastavením určitých hodnot pixelů pro základní barvu materiálu, ale v zásadě se textury vytvářejí v jiných aplikacích pro úpravu obrázků a poté se importují do editoru prostřednictvím Content Browseru (Prohlížeče obsahu) (Epic Games, 2024d).

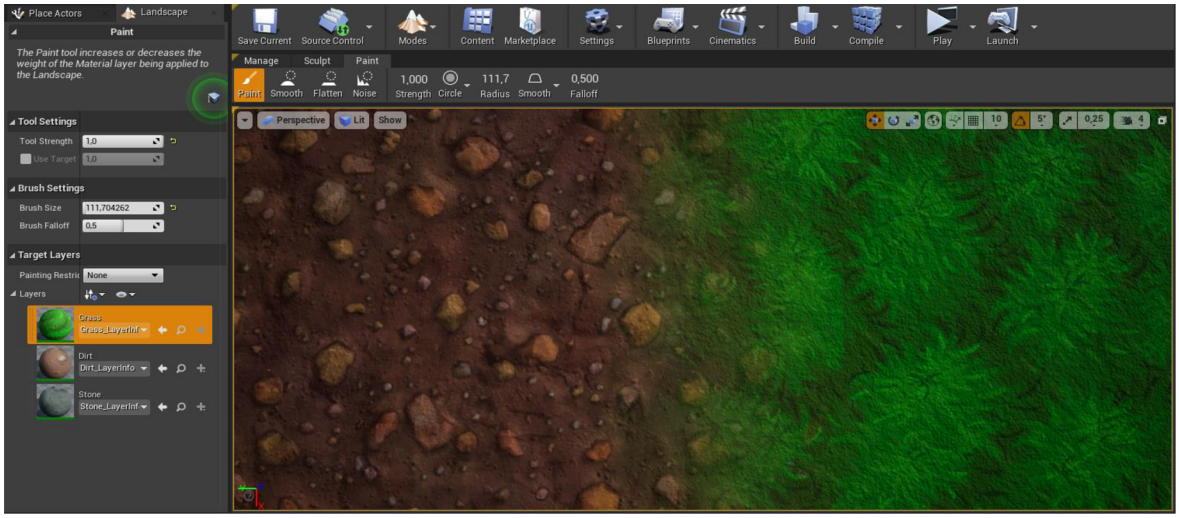
Obrázek 17: Tvorba primitivní textury



Zdroj: Autor

Landscape texturování: texturování krajiny se liší od aplikace textury na mesh. Hlavní rys texturování krajiny je spojen s mícháním textur pomocí Landscape Layer Blend a dalším rozmalováním lokaci pomocí nástroje malování (Paint) (World of Level Design, 2020).

Obrázek 18: Míchaní dvou textur nástrojem pro malování

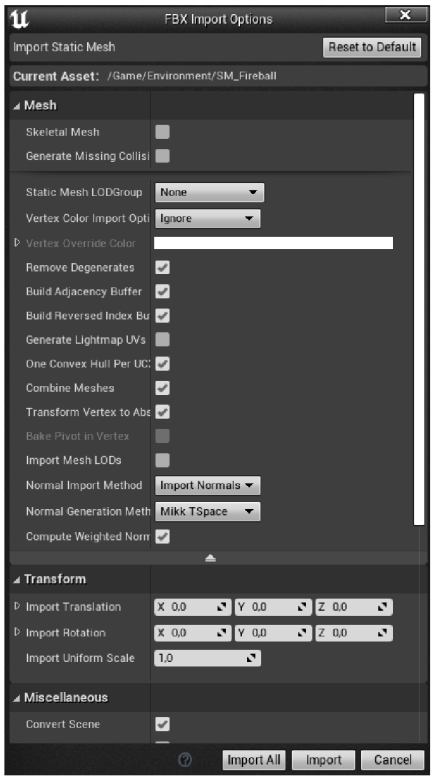


Zdroj: Autor

3.5.5 Import 3D modelů a textur

Import 3D modelů do Unreal Engine je důležitý, ale jednoduchý proces, který umožní přenést dříve vytvořené modely a textury ve speciálních programech do svého projektu pro další manipulaci v editoru. Níže jsou uvedeny nejdůležitější kroky pro import 3D modelů a jejich odpovídajících textur (Warfighter digital, 2023):

Obrázek 19: Opci importu do UE



Zdroj: Autor

1. Vytvoření 3D modelu v programu jako 3DS Max, Maya nebo Blender.

2. Do UE se exportují pouze kompatibilní soubory, jako například FBX, OBJ a DAE.

3. V otevřeném projektu je nutné vybrat “File>Import”. V okně, které se otevře, je nutné uvést cestu k vytvořeným 3D objektům a importovat. Je možné to také provést přetažením souboru do editoru(Drag and drop).

4. Na panelu opci importu je nutné vybrat příslušná nastavení a importovat pro další použiti nových assetů ve svém projektu.

3.5.6 **Stavy hry/hráče**

Třídy stavu hry/hráče jsou třídy, které obsahují data o hráči nebo hře. V singleplayeru existuje jenom jeden stav hráče, v multiplayeru má každý hráč svou vlastní instanci stavu hráče. Tyto třídy jsou také velmi důležité pro hry pro více hráčů (Satheesh, 2021, s. 59):

- **Gamemod:** Tato třída určuje pravidla hry. V závislosti na stavu hry (například počtu zabitých protivníků) určuje různé herní prvky, jako je změna režimů v online hře.
- **Game state:** tato třída sleduje stav hry, například v multiplayeru se může jednat o přechod herního režimu z předzápasového stavu (příprava na začátek zápasu) do stavu zahájení hry, o kterých mohou být hráči následně informováni.
- **Player state:** Tyto třídy obvykle obsahují informace, jako je počet nepřátel, které hráč zabil, jak velké poškození hráč udělil a v jakém režimu se aktuálně nachází. To znamená, že jde v podstatě o třídu se statistikami pro každého hráče.

3.5.7 **Audio**

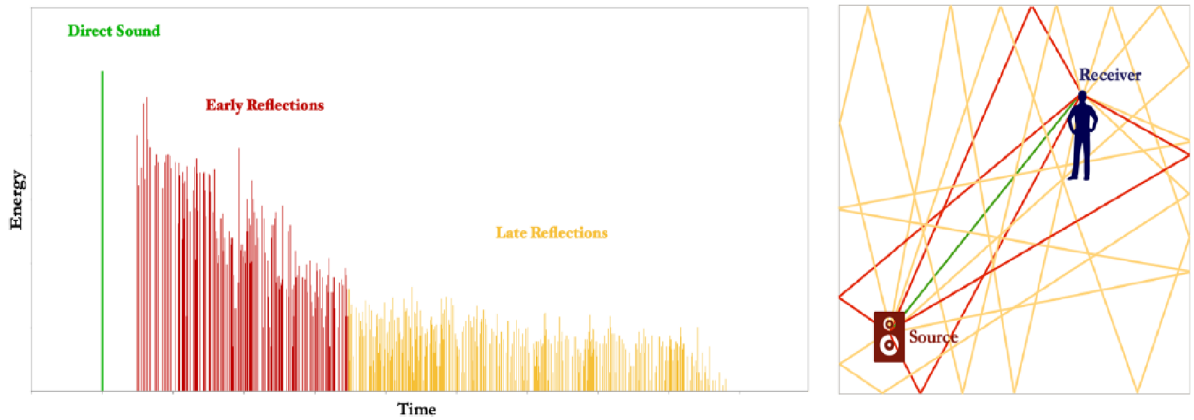
Zvukový engine v Unreal poskytuje širokou sádu nástrojů pro práci se zvukem. Díky kombinaci schopností Blueprints a multiplatformního zvukového mixeru je engine schopen produkovat vysoce kvalitní zvuk pomocí digitálního zpracování signálu (Epic Games, 2024e).

Jak funguje šíření zvuku v enginu? Analýza šíření zvuku je rozdělena do tří hlavních částí: zdroj zvuku, přenos a přijímač zvuku. Běžnou praxí pro modelování šíření zvuku je rozdělení odrazů zvuku na přímý zvuk, ranní odrazy a pozdní reverberaci (Firat et al., 2022, s. 539 - 556):

- **Přímý zvuk (Direct Sound):** Je to zvuk přenášený přímo od zdroje k příjemci bez jakýchkoli odrazů.
- **Ranní odrazy (Early Reflections):** Toto je zvuk, který dorazí k přijímači během prvních 50-80 ms po příjmu přímého zvuku. Jsou důležité pro přenos informací o umístění zdroje zvuku. Ovlivňují šířku, témbr a prostorovost zvuku.
- **Pozdní reverberace (Late Reverberations):** Je to zbývající odražený zvuk, který dorazí do přijímače po ranním odraženém zvuku. Pozdně odražený zvuk postrádá informace o okolí a vytváří difúzní zvukové pole.

Obrázek níže znázorňuje časový a prostorový vývoj reakce prostředí po přehrávání zvuku:

Obrázek 20: Přímý zvuk, ranní odrazy a pozdní reverberace



Zdroj: Firat et al. (2022)

Reverberace: Reverberaci je fenomén setrvání zvuku po jeho zastavení v důsledku mnohonásobných odrazů od povrchů uvnitř uzavřeného prostoru. Tyto odrazy mají tendenci se hromadit a postupně slábnout, když je pohltí předměty v uzavřeném prostoru. V podstatě se jedná o ozvěnu, ale vzdálenost mezi zdrojem zvuku a odraznou plochou je menší (BYJU'S, 2018).

Difúzní zvukové pole (Diffuse Field): V místnostech, ve kterých se přímý zvuk opakovaně odráží od povrchů, se objevuje difúzní pole. V difuzním zvukovém poli je zvuková energie téměř stejná na všech stranách pole (Schoeps Mikrofone, 2024).

3.6 User Experience (UX)

Uživatelskou zkušenosti je to, co uživatel očekává, že zažije při používání jakéhokoli produktu, fyzického nebo digitálního. UX design je důsledkem procesu UCD (User-centered design), který zahrnuje takové iterace čtyřfázového cyklu (Medium, 2018):

- Pochopení a specifikace použitého obsahu
- Specifikace požadavků na interakci
- Vytváření designových řešení
- Vyhodnocování rozhodovacích dat vůči požadavkům

Role expertů UX probíhají během tohoto cyklu a jsou rozděleny do (Coursera, 2024):

- **UX designer:** Designér UX je zodpovědný za pochopení uživatelů produktu. Pracují na zlepšení interakce mezi uživatelem a aplikací, pohodlí a potěšení z používání konečného produktu. Interakce uživatele s produktem může být tak jednoduchá, jako použití aplikace, webové stránky nebo hraní počítačové hry.
- **Interaction Designer:** Je odpovědný za sledování produktu, ujištění se, že funguje správně, a za hodnocení interakce uživatele s ním.
- **UI developer:** Je zodpovědný za vývoj pokynů pro vývoj uživatelských rozhraní. Slouží jako spojovací článek mezi týmy UX vývojářů a vývojářů softwarů.

Celkovým cílem těchto vývojářů je vytvářet technické artefakty různé úrovně abstrakce a složitosti, jako jsou scénáře a drátové modely. Jak postupuje sestavení aplikace, tak drátěné modely se nahrazují funkčními prototypy, objevují se indikátory

systematického používání a jsou odesílány k vyhodnocení odborníkům. Po zpracování je artefakt vrácen do zpracování a pro zjištění přesnější informace o interakci po testování uživateli, stejně jako po A/B testech (Rodriguez et al., 2023).

User testy: Uživatelské testy, v nichž uživatelé testují aplikace/produkt provedením určité sekvence speciálních úloh. Jsou nákladné, ale poskytují vynikající posouzení problémů uživatelské zkušenosti s daným produktem, jako je úroveň dokončení těchto úkolů, čas strávený na nich, míra radosti z používání produktu a výsledky dotazníků (Rodriguez et al., 2023).

A/B testy: Jsou populární metodou pro porovnávání verzí webové stránky nebo aplikace proti sobě, aby se zjistilo, která z nich funguje lépe. Tento test používá statistickou analýzu k určení, která verze nejlépe vyhovuje požadavkům projektu (Optimizely, 2021).

3.7 System Usability Scale (SUS)

SUS je dotazník, umožňující rychle posouzení použitelnosti produktu a je kvantitativní metodou hodnocení. Tento dotazník se skládá z deseti otázek s pěti možnostmi odpovědí, od „Rozhodně souhlasím“ po „Rozhodně nesouhlasím”. Tento dotazník umožňuje hodnocení široké škály různých produktů a služeb (QuestionPro, 2018).

Dotazník SUS se stal standardem pro hodnocení produktů, a to vše díky velmi snadnému způsobu hodnocení produktu pro respondenty, který umožňuje získat poměrně přesný názor na použitelnost jejich produktu. Dotazník se skládá z 10 otázek, které dokážou poměrně přesně zhodnotit jakýkoli produkt a vypadá takto (usability.gov, 2023):

Tabulka 1: SUS dotazník

1.	Myslím si, že bych chtěl(a) tento systém často používat.
2.	Myslím si, že systém zbytečně složitý.
3.	Myslím si, že systém je snadno použitelný.
4.	Myslím si, že bych potřeboval(a) podporu technické osoby, abych tento systém mohl(a) používat.
5.	Myslím si, že různé funkce v tomto systému byly dobře integrovány.
6.	Myslím si, že v tomto systému je příliš mnoho nekonzistence.

7.	Myslím si, že většina lidí se naučí tento systém používat velmi rychle.
8.	Myslím si, že systém je velmi neohrabaný na používání.
9.	Cítil(a) jsem se velmi sebejistě při používání systému.
10.	Musel(a) jsem se naučit hodně věcí, než jsem se mohl(a) pustit do práce s tímto systémem.

Zdroj: usability.gov (2023)

Jak se tento průzkum provádí? Obecně se SUS používá až poté, jak získal respondent možnost produkt vyzkoušet, ale před konečnými diskusemi a výsledky. Respondent by neměl nad položenými otázkami přemýšlet, ale rychle odpovídat. Dále zbývá jen udělat závěry (Brooke, 2023).

Po správném výpočtu výsledků posouzení zbývá pouze dát jim správné charakteristiky. Výsledek hodnocení 68 bodů nebo více je dobrý výsledek, ale výsledek pod touto hranicí znamená, že produkt potřebuje zlepšení (Alathas, 2018).

Tabulka 2: Tabulka pro interpretace SUS hodnocení

SUS skóre	**Hodnocení**
>80.3	Výborně
68-80.3	Skvělé
68	Dobře
51-68	Špatně
<51	Hrozně

Zdroj: Alathas (2018)

3.8 Post-production

V závislosti na obchodním modelu se může fáze postprodukce lišit. Hry se mohou dále vyvíjet, získávat nový kontent, jako jsou úrovně, nové mechaniky, příslušenství nebo plnohodnotná DLC, neboli mohou jít do dokončení a připravovat se na finální vydání (Johan Karlsson, 2020).

4 Vlastní práce

4.1 Pre-production fáze

4.1.1 Návrh hry

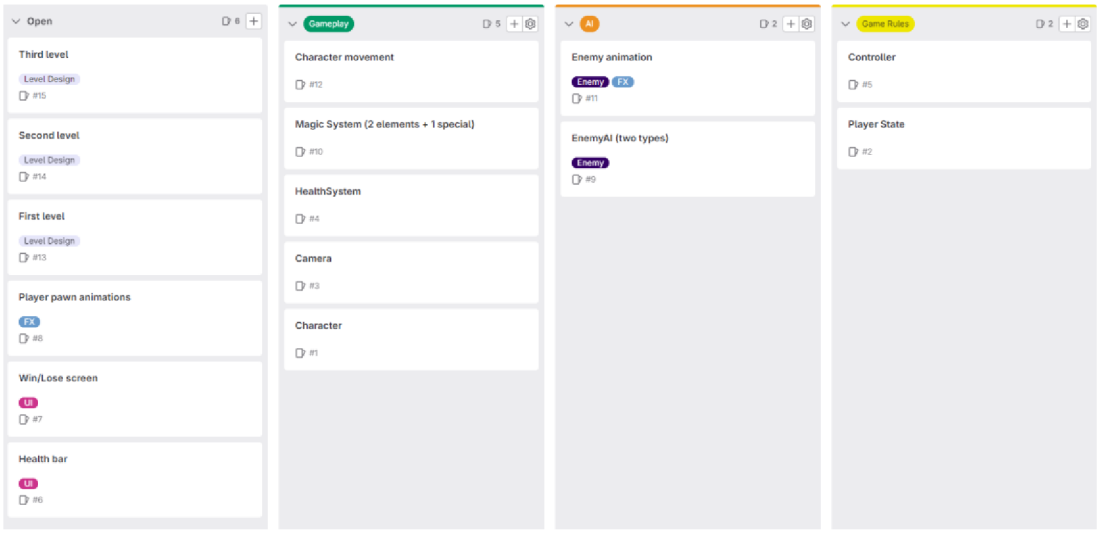
Proces tvorby hry vždy začíná přemýšlením o tom, co to bude za hra. Například jaký žánr, setting (místo, kde se odehrávají události hry), styl (pixel art, voxel grafika, běžné 2D nebo 3D atd.), jak to bude vypadat (živý barevný svět nebo temný a děsivý atd.), jaké kvality modely v ní budou použity, pro jaké uživatele je určena z hlediska věku a výkonu hardwaru, co je ve hře potřeba udělat, jaké herní mechaniky budou implementovány, jak v ní vyhrát a desítky dalších bodů, o kterých je nutné mít představu před začátkem vývoje. Všechny tyto informace jsou zaznamenány v Game Design Dokumentu, ke kterému je pak možné přistupovat kdykoli během vývoje.

Při výběru herního žánru byly zvažovány různé varianty, jako je Shooter, Strategie, Akční adventura a Top-Down. Po pečlivém zvážení padla autorská volba na kříženec mezi Akční adventurou a Top Down žánrem. Důležité je také rozhodnout o settingu hry. Pro hru byl zvolen Fantasy setting. Zvolený styl byl jasný a kreslený. Všechny podrobnosti a další nápady o projektu jsou zaznamenány v herním design dokumentu.

4.1.2 Game Design Dokument

Pro vytvoření designového dokumentu existují aplikace pro pohodlné zaznamenávání všech nápadů na hru a jejich třídění. Vizuální složka takových programů hraje důležitou roli v pohodlí práce s dokumentem návrhu. Populárními možnostmi jsou Trello a desky na Gitlab. Pro tento projekt byl jako úložiště vybrán GitLab a tam byl vytvořen GDD pro tento projekt a vypadá takhle:

Obrázek 21: GDD projektu



Zdroj: Autor

Tento GDD na Gitlab byl postaven na základě všech nápadů pro hru, které nebyly odmítnuty. Během vývoje projektu, je možné na desce GitLabu přidávat nové úkoly a hotové úkoly lze odesílat do sekce dokončených úkolů. Kompletní seznam nápadů pro tento projekt vypadá takto:

- Hra je křížencem mezi akční adventurou a žánrem Top Down s prvky střílečky a plošinovky.
- Hra bude pro jednoho hráče.
- Setting hry je fantasy.
- Většina assetů pro tuto hru budou vytvořeny ručně v programu Blender v kresleném low-poly stylu, což také sníží zátěž počítače.
- Kamera se na hráče dívá z oblohy s možností přiblížení.
- Pohyb a střelba jsou možné pomocí klávesnice.
- Hráč má schopnost běhat a skákat, aby překonal určité překážky.
- Hra bude rozdělena do několika úrovní a každá bude mít jedinečný mechanismus pro dokončení dané úrovně.
- Hráč bude mít možnost použít několik kouzel k dokončení hry.

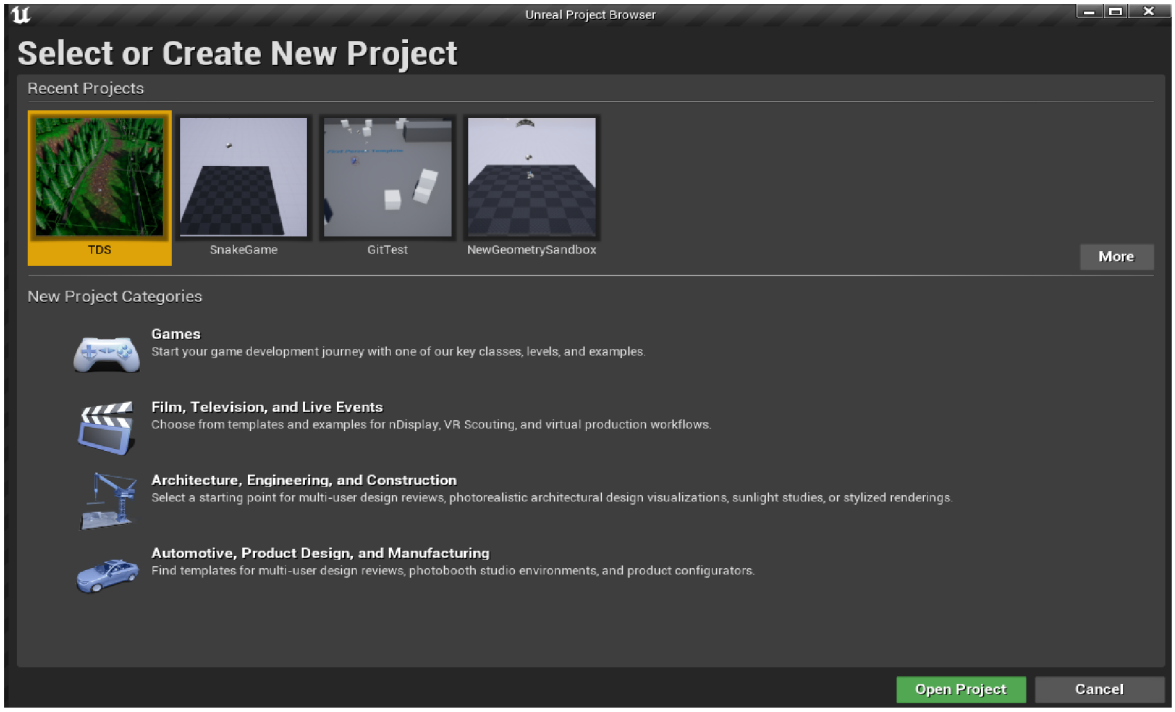
- Hra bude mít AI, která je hlavní překážkou pro dokončení jedné z úrovní, a bude mít několik typů.
- Aby se dostat ke každé další úrovně, je nutné se dostat na konec předchozí mapy.
- Klíčovou mechanikou pro další úroveň bude překonání plovoucích plošin.
- Vytvoří se speciální nebojové kouzlo, které je nutné k dokončení jedné z úrovní.

4.2 Production-fáze

4.2.1 Instalace a nastavení projektu

Nejpve je nutno nainstalovat Unreal Engine požadované verze. V práci je použitá verze UE 4.27.2.

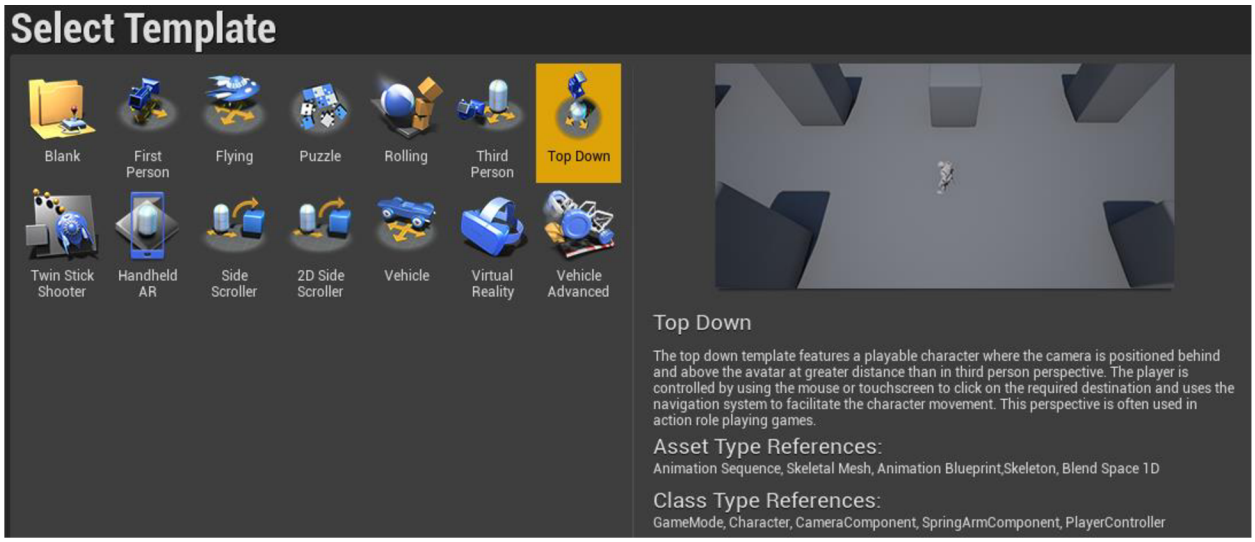
Obrázek 22: UE4 Browser



Zdroj: Autor

Po otevření kterékoli z kategorií, zobrazí se seznam všech dostupných šablon, což usnadňuje zahájení vývoje. Kategorie her má dostatečnou sadu šablon pro všechny populární žánry. Pro tento projekt je vybrána šablona Top Down.

Obrázek 23: Šablony UE4



Zdroj: Autor

Projekt je hybridní, to znamená napsaný v C++ spolu s Blueprints; k tomu je třeba vybrat projekt C++ v nastavení projektu. V projektu bude zahrnut i Starter Content, který obsahuje standardní assety Unrealu, materiály a textury. Po výběru adresáře je možné vytvořit projekt.

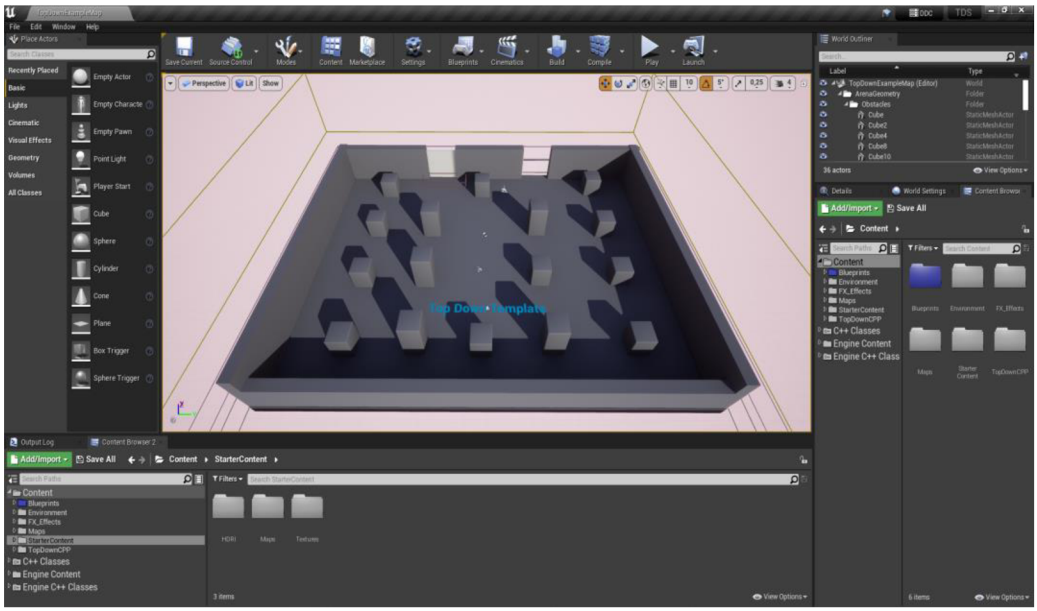
Obrázek 24: Nastavení projektu

Zdroj: Autor

4.2.2 **Interface**

Nejprve je nutné porozumět rozhraní Unreal Editoru, protože zde se začíná práce s jakýmkoli programem. Níže je rychlý pohled na rozhraní:

Obrázek 25: UE4 Interface



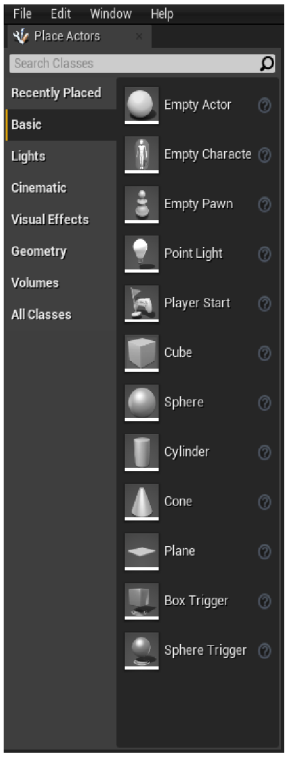
Zdroj: Autor

Obrázek 26: Horní panel

Zdroj: Autor

V horní části je viditelná panel s velmi důležitými instrumenty, jako je Uložení projektu, Source Control, Modes, Build, Compile a Play pro spuštění simulace.

Obrázek 27: Place Actors



Zdroj: Autor

Zleva se nachází panel s nejčastěji používanými Aktory (objekty, které lze umístit na scéně).

Vpravo a dole jsou panely standardně používané pro zobrazení logů, zobrazení detailů o objektu vybraném ve scéně, seznam všech aktuálně existujících objektů na scéně a také prohlížeč obsahu – složka projektu.

Jakýkoli panel lze přizpůsobit tak, aby vyhovoval potřebám vývojáře; k tomu je možné jednoduše přetáhnout jiné okno na požadované místo a také zobrazit požadované okno v nabídce Windows.

4.2.3 **Assety**

Obrázek 28: Low-poly strom

Zdroj: Autor

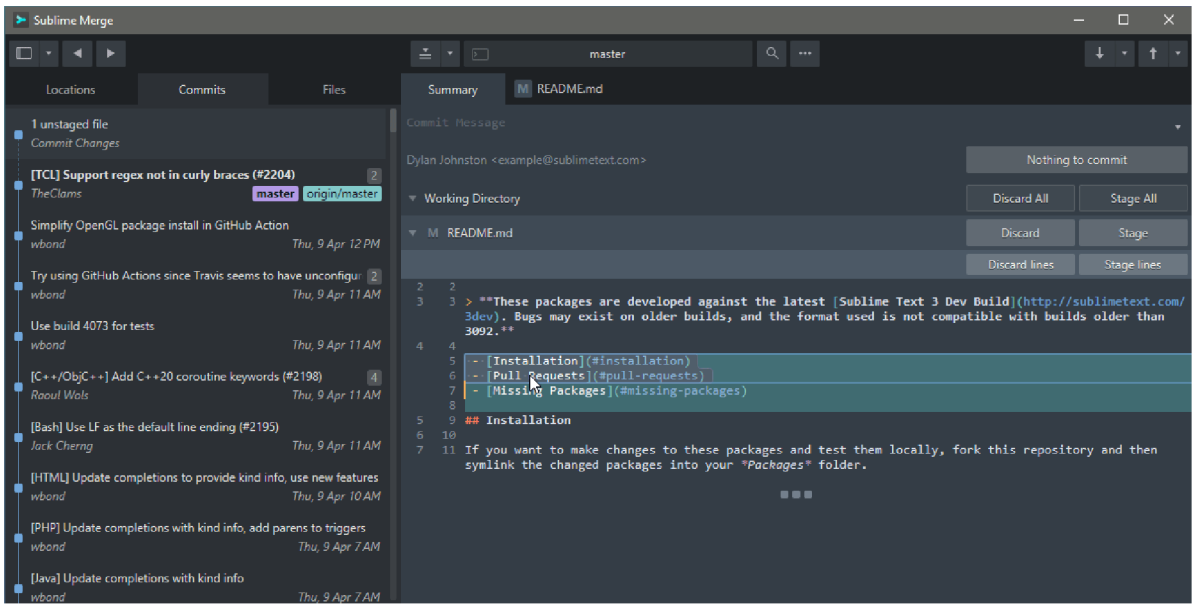
Jakákoli hra potřebuje assety, které naplní hru. Assetem je nějaká entita, která představuje část herního obsahu a má určité vlastnosti. Například viditelné objekty, částice, materiály a textury objektů. V tomto projektu budou použity standardní assety poskytnutí vývojáři Unreal, volné assety z internetu i ti, které vytvořil autor v programu Blender.

4.2.4 Source Control

Nejdůležitější částí při zahájení práce na jakémkoli projektu je připojení projektu k Systému kontroly verzí neboli Git. Git umožňuje uložit jakoukoli změnu v projektu a v budoucnu se v případě potřeby vrátit ke staré verzi projektu. Docela podobné ukládání jako ve hrách. Bez systému kontroly verzí může každá chyba stát téměř všechnu práci.

Pro tvorbu tohoto projektu byl použít grafický klient Git Sublime Merge, ale aby fungoval na Windows, je nutné si nejprve stáhnout Git Bash.

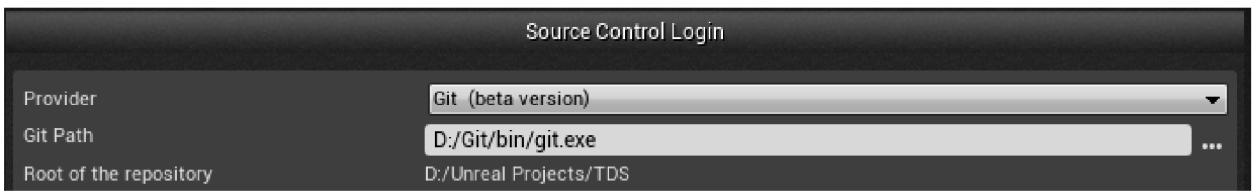
Obrázek 29: Sublime Merge



Zdroj: Sublime Merge official site

Dalším krokem je připojení k samotnému git. Unreal to dělá rychle a snadno. Kliknutím na Source Control je potřeba vybrat git, cestu k souboru git bash, adresu předem vytvořeného úložiště na webu, například GitHub nebo GitLab, a uložit nastavení. Nyní by měla být každá důležitá změna uložena v git a v případě potřeby existuje přístup ke staré verzi projektu.

Obrázek 30: Nastavení Source Control



Zdroj: Autor

Je důležité si pamatovat, že při psaní herní logiky v C++ lze případné konflikty při slučování snadno opravit v textovém formátu a při práci s Blueprints bude jakýkoli asset nebo psaná logika binárním souborem a bude možné jej opravit pouze pro jednoho vývojáře přímo v Unreal Editoru.

4.2.5 **Práce s kamerou**

První funkcí implementovanou ve hře bylo ovládání kamery, protože hru nelze hrát, pokud kamera není správně nakonfigurována. Šablona Top Down již nabízí kameru, standardní pro tento žánr, umístěnou nad hráčovým pěšcem, ale pro tento projekt kamera musela být modifikovaná nejen pro pohodli koncového uživatele, ale i pro testera, a proto bylo rozhodnuto o rozšíření základní funkcionality kamery.

Byly vytvořeny funkce pro rolování a pro otáčení kamery pro lepší výhled. Funkce rolování byla zpočátku prototypována v Blueprints, ale stala se značně těžkopádnou a byla místo přenesení do C++ ponechaná v Blueprints a později upravená. Podstatou rolování je pohyb kamery po ose Z v závislosti na vstupu z kolečka myši. V Unrealu jsou jakékoli vstupy přidány prostřednictvím nastavení vstupů v sekci Project Settings->Engine->Input a to přidáním vstupu do Axis mappings nebo Action Mappings.

Klíčové rozdíly mezi těmito mapováními jsou následující:

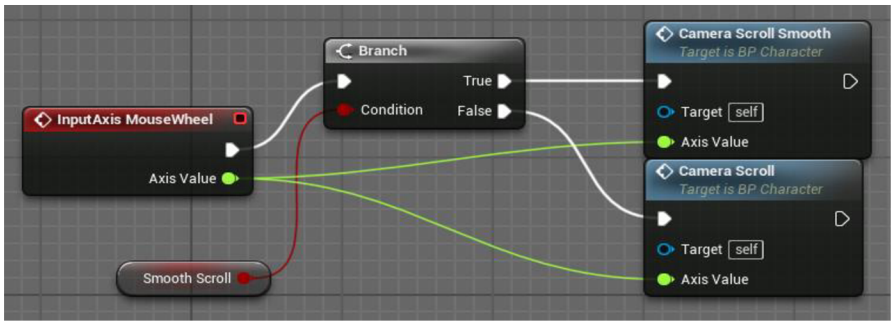
Axis mapping: Osové mapování je navrženo tak, aby zpracovávalo spojité vstupy, které mohou nabývat různých hodnot v určitém rozsahu.

Příklad: Pokud je potřeba zpracovat vstup pro řízení směru pohybu (vpřed/vzad nebo vlevo/vpravo), pak je pro tento typ vstupu vhodné Axis Mapping.

Action mapping: Mapování akcí se používá ke zpracování diskrétních událostí, jako jsou stisknutí kláves, které mají dva stavy: stisknuto nebo nestisknuto.

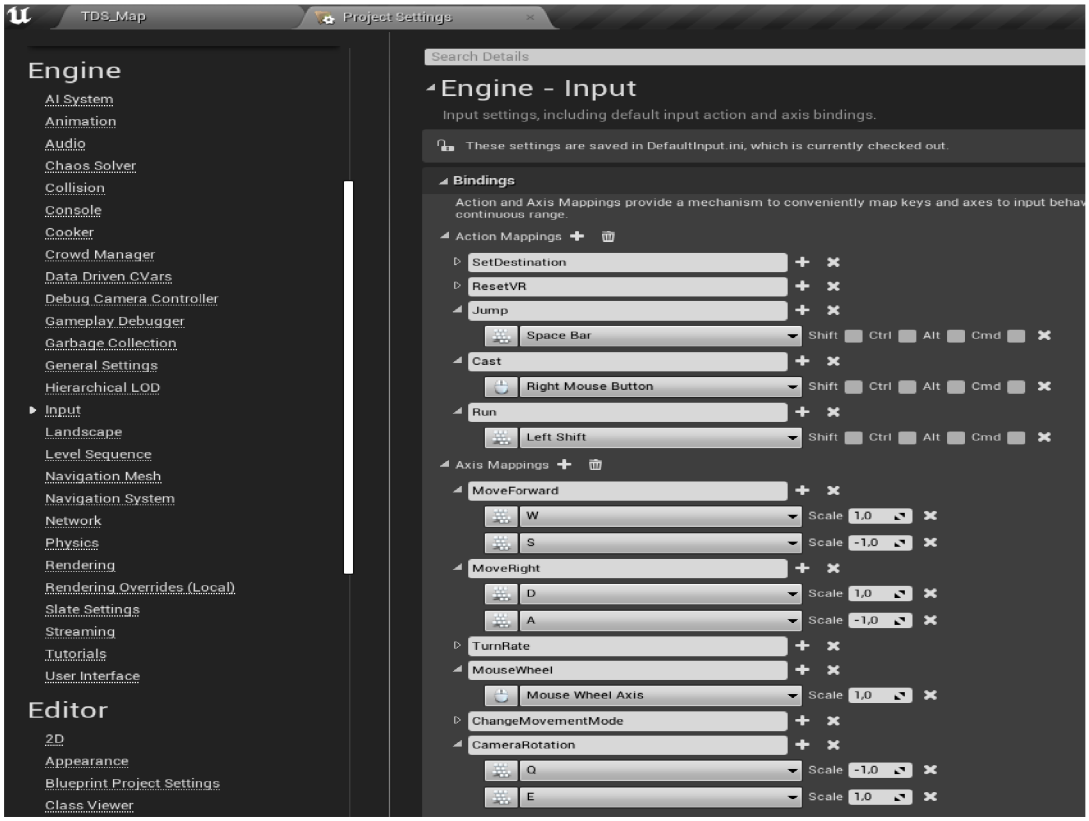
Příklad: Pokud je potřeba zpracovat události jako „Výstřel“, „Skok“ nebo „Otevření dveří“, pak bude mapování akcí pohodlnější, protože tyto události jsou ve své podstatě diskrétní.

Obrázek 31: Volání funkce rolování při rolování kolečkem myši



Zdroj: Autor

Obrázek 32: Adding inputs



Zdroj: Autor

Další funkcí je rotace kamery, již implementovaná v C++. Camera Boom je zde komponenta ze šablony, zodpovědná za propojení kamery s postavou, kterou lze již pohodlně konfigurovat.

```
void ATDSCharacter::HorizontalCameraRotation(float Side)
{
	FRotator Rotator = CameraBoom->GetRelativeRotation();
	if (Side == 1.0f)
	{
		Rotator.Yaw += 1.5;
		CameraBoom->SetRelativeRotation(Rotator);
	}
	else if (Side == -1.0f)
	{
		Rotator.Yaw -= 1.5;
		CameraBoom->SetRelativeRotation(Rotator);
	}
}
```

4.2.6 Otáčení a pohyb

Dalším krokem byla implementace vhodného pohybu a rotace postavy, protože ve výchozím nastavení na této šabloně mohla postava přejít do určené lokaci pouze kliknutím na něj levým tlačítkem myši. Pro tento účel byla v Blueprints prototypována funkce otáčení postavy. Poté byla v C++ implementována funkce MovementTick, která zodpovídá za pohyb pěšky hráče a její rotaci. Kód je v příloze [1].

Stojí za to zvážit tuto funkci podrobněji. Níže uvedená část je opakováním funkce, prototypované v Blueprints a je zodpovědná za otočení pěšky ke kurzoru. Připojením knihovny „Kismet/GameplayStatics.h“ získává se přístup k ovladači libovolného hráče. Pokud je hra vyvíjena jako hra pro jednoho hráče, pak je nalezení hráčova ovladače docela snadné, protože bude mít vždy index 0.

```
APlayerController* MyController = UGameplayStatics::GetPlayerController(GetWorld(), 0);
	if (MyController)
	{
		FHitResult ResultHit;
		MyController->GetHitResultUnderCursorByChannel(ETraceTypeQuery::TraceTypeQuery6, false,
ResultHit);

		float FindRotatorResultYaw = UKismetMathLibrary::FindLookAtRotation(GetActorLocation(),
ResultHit.Location).Yaw;
		SetActorRotation(FQuat(FRotator(0.0f, FindRotatorResultYaw, 0.0f)));
	}
```

Další místo, kam ukazuje kurzor, zaznamenáno v objektu třídy FHitResult, což je standardní třída Unrealu, obsahující informace například o bodu kontaktu/průsečíku různých povrchů. Dále se zjistí úhel otočení k nalezenému bodu a postava se o tento úhel otočí. Další zvažovanou částí je tato část, která je zodpovědná za pohyb postavy:

```
	FVector FVec = CameraBoom->GetForwardVector();
	FVector RVec = CameraBoom->GetRightVector();
```

```
FRotator Rot = FVec.Rotation();
Rot.Pitch = 0.0f;
FVec = Rot.Vector();

AddMovementInput(FVec, AxisX);
AddMovementInput(RVec, AxisY);
```

Pokud je kamera nehybná a je ve standardní poloze na světových koordinátách, budou poslední dvě čáry stačit k pohybu, ale aby fungovaly, je nutné nejprve přidat pohybovou komponentu do standardní funkce pěšce SetupPlayerInputComponent, která je zpočátku umístěna v souboru třídy pěšce, stejně jako metody Tick a BeginPlay. Kód je v příloze [2].

Pro použití pohyblivé a otočné kamery, která byla přidána v pozdějších fázích vývoje, byl přidán zbývající kód, který umožňuje pohybovat postavou nikoli podle světových koordinát, jak tomu bylo dříve, ale relativně ke kameře.

```
FVector FVec = CameraBoom->GetForwardVector();
FVector RVec = CameraBoom->GetRightVector();
FRotator Rot = FVec.Rotation();
Rot.Pitch = 0.0f;
FVec = Rot.Vector();
```

V důsledku toho se tato funkce začala volat na standardním Ticku.

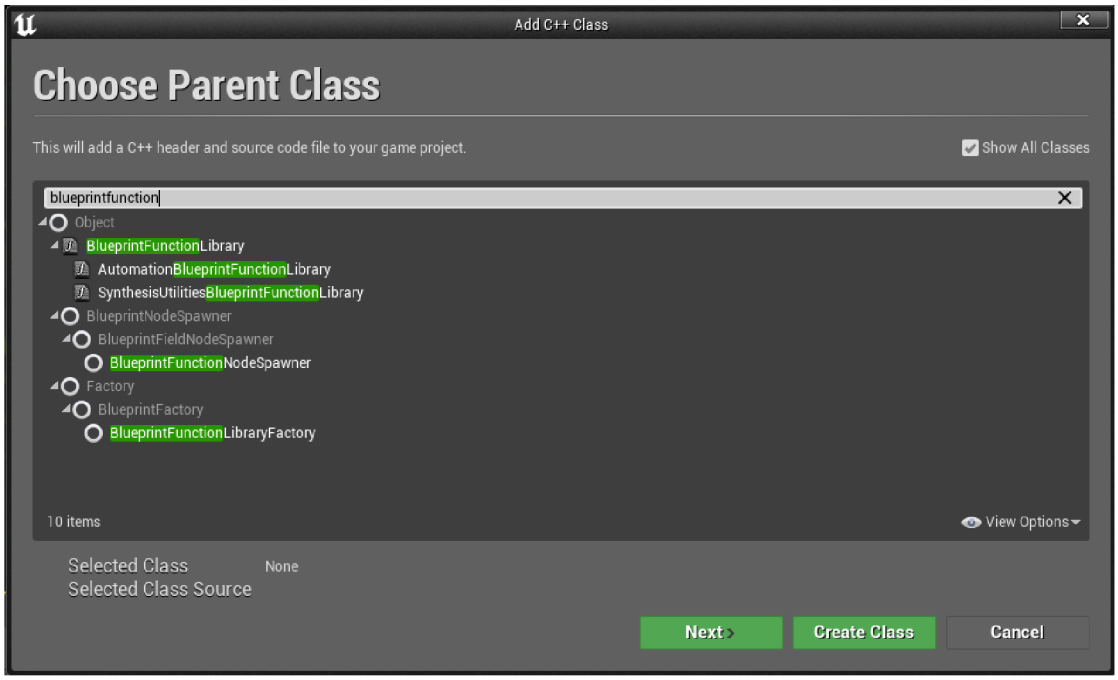
4.2.7 **Stavy**

Jelikož pohyb a kamera byly připraveny, dalším řešením bylo vytvořit třídu obsahující stavy postavy, aby se v budoucnu v závislosti na konkrétním stavu postavy měnily její vlastnosti nebo animace.

Jednoduchým příkladem, proč jsou stavy hráčů potřeba, by byl klidový stav, kdy se postava nehýbe, nehraje bojová hudba, nespawnují se protivníci a jenom se přehrává animace stání.

Vzhledem k tomu, že se neplánuje vytvoření žádné instance třídy se stavy, ale to bude pouze něco jako knihovna funkcí, ze které se budou brát nějaká data, bude tato třída zděděna z třídy knihovny funkcí.

Obrázek 33: Vytvoření funkční knihovny



Zdroj: Autor

Tato knihovna umožňuje vytvářet statické funkce (funkce tříd) pro použití v Blueprints. Takové funkce mohou provádět různé operace a poskytovat pohodlné metody pro Blueprints, a nevyžadují vytvoření nové instance objektu. Příklad kodu v příloze [3].

Enum obsahuje všechny stavy postavy použité ve hře. V závislosti na konkrétním stavu se může změnit rychlost postavy nebo animace.

Stavy se během hry po nějaké akci mění. Nejjednodušší způsob, jak to udělat, je pomocí Blueprints, po přidání vstupu z klávesnici v nastavení projektu, například Běh postavy na Left Shift.

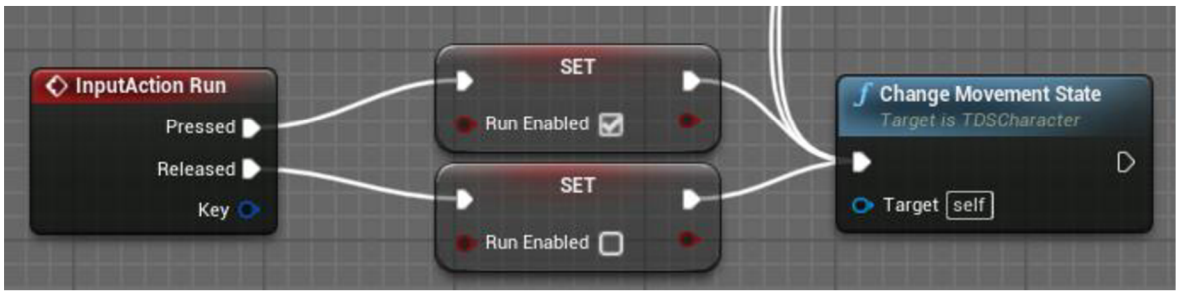
Obrázek 34: Nastavení akce na vstupu

Zdroj: Autor

Změna stavu postavy a následně i její rychlosti se provádí v samotné třídě postavy pomocí funkcí ChangeMovementState a CharacterUpdate. Příklad kodu v příloze [4].

Je docela snadné zavolat funkci změny stavu s požadovaným vstupem pomocí Blueprints. Ve Blueprints postavy událost vstupu z klávesnice změní stav připravených booleovských proměnných, a následně zavolá funkci změny stavu postavy:

Obrázek 35: Změna stavu postavy



Zdroj: Autor

Je důležité pochopit, že ve výchozím nastavení se funkce a proměnné popsané v C++ v Blueprints neobjeví. Pro jejich zobrazení v Blueprints je nutné použít macrosy UPROPERTY pro proměnné a UFUNCTION pro funkce podobným způsobem:

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
        bool CastEnabled = false;
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
        bool RunEnabled = false;
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
bool JumpEnabled = false;

UFUNCTION(BlueprintCallable)
        void CharacterUpdate();
UFUNCTION(BlueprintCallable)
void ChangeMovementState();
```

4.2.8 Animace

V Unreal Engine animace hrají důležitou roli a používají se k vytvoření živých a zajímavých postav a objektů ve hře. Bez animací se neobejde téměř žádná hra, protože například bez vhodné animace bude běžící postava prostě statický objekt pohybující se v prostoru.

Hlavní animační komponenty používané v jakékoli hře jsou Animation Blueprints, Blend Spaces a Animation Sequences. Nejprve je nutné pochopit, z čeho se skládá jakákoli animace. Každá složitá animace se skládá z animačních sekvencí.

Animation Sequences (Animační sekvence) jsou jednoduché animace zapsané v souboru. Může to být jedna konkrétní animace, například animace skoku, útoku nebo pohybu. Tyto animace lze použít v animačních Blueprints a na dalších místech ve hře.

Blend Spaces (Prostory míchaní) se používají k hladkému přechodu mezi různými animacemi v závislosti na hodnotách parametrů. Jako příklad je možné použíti prostoru míchaní k hladkému pohybu mezi animacemi chůze vpřed, vzad, doleva a doprava v závislosti na hodnotách os pohybu.

V tomto projektu blend space míchá animace běhu, chůze a skoku, v důsledku čehož čím vyšší bude rychlost postavy, tím výraznější bude její sprint, a v případě zpomalení postava postupně začne jen chodit a úplně se zastaví. a animace nečinnosti se spustí v případě nulové rychlosti. Hodnota výšky samozřejmě ovlivňuje animace skoku.

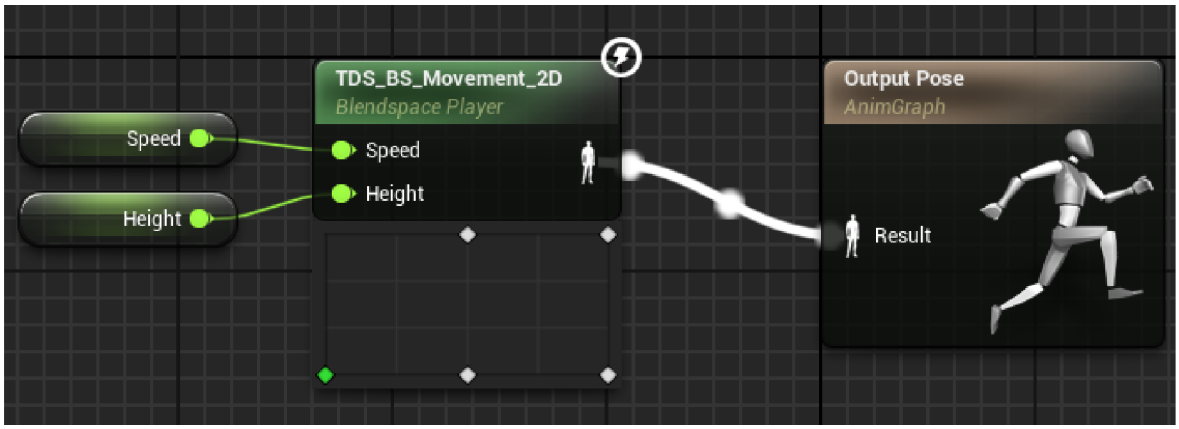
Obrázek 36: Prostor míchaní v UE

Zdroj: Autor

Animation Blueprints (Animační Blueprints) jsou grafickým znázorněním animační logiky pro postavu nebo objekt. Animace se programuje v animačním grafu. Běžným příkladem použití animačního Blueprints je přepínání mezi animacemi chůze a běhu nebo skoku v závislosti na hodnotách rychlosti a výšky postavy.

V tomto projektu je v animačním grafu již vytvořený a nakonfigurovaný blend space spojen s výstupní pózou postavy. Zde je možné také spustit jednoduchou animaci v závislosti na jakýchkoli podmínkách.

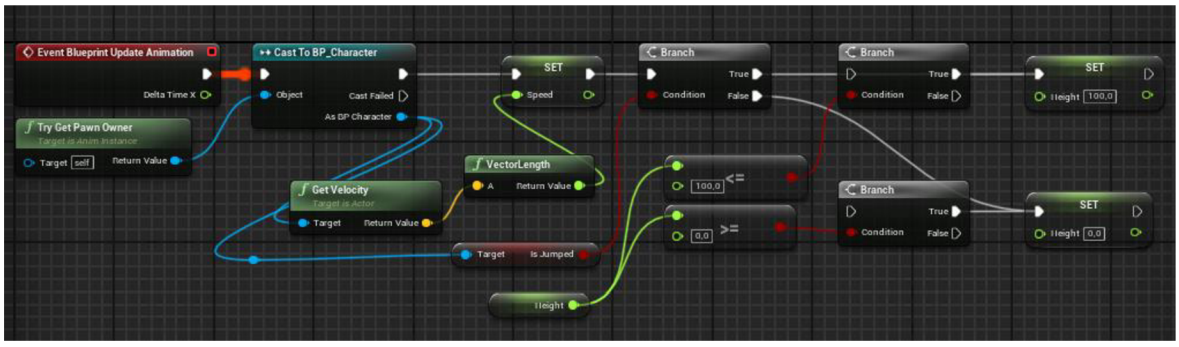
Obrázek 37: Nastavení konečné pózy v animačním Blueprint



Zdroj: Autor

Ve grafu události je možné v závislosti na jakýchkoli podmínkách změnit hodnoty nahrazené v prostoru mícháni, což odpovídajícím způsobem změní konečnou animaci. Níže uvedený příklad kódu převezme hodnotu rychlosti postavy a přisune jí do proměnné rychlosti prostoru mícháni, změní hodnoty výšky, když postava skočí, a tak se nakonec spustí požadovaná animace.

Obrázek 38: Programování animačního Blueprint



Zdroj: Autor

4.2.9 **Umělá inteligence**

Umělá inteligence je důležitou součástí každé hry, protože je téměř vždy zodpovědná za zábavu hráče, ať už se jedná o přátelskou AI hrající důležitou postavu v zápletce, nebo nepřátelskou AI, která je pro hráče testem, kterým musí projít.

Umělou inteligenci lze implementovat různými způsoby. Je možné to napsat v C++, implementovat jí v Blueprints nebo k tomu je možné použít pohodlné nástroje enginu. Behavior Tree (Strom chováni) a Blackboard (Deska) jsou klíčovými součástmi systému

umělé inteligence (AI) v Unreal Engine. Poskytují flexibilní a rozšiřitelný způsob, jak definovat a ovládat chování postav ve hře.

Strom chování je struktura sloužící k určování logiky rozhodování a kontroly chování postavy. Skládá se z uzlů, které představují různé akce, podmínky a kombinace těchto prvků. Příklady typů uzlů ve stromu chování:

Composite Nodes (Složené uzly):

Sekvence: Spouští podřízené uzly postupně, dokud jeden z nich neselže.

Selektor: Spouští podřízené uzly postupně, dokud jeden z nich neuspěje.

Decorator Nodes (Dekorační uzly):

Blackboard Decorator: Umožňuje použít data z Blackboard k rozhodnutí, zda spustit uzel.

Task Nodes (Uzly úkolu):

Move To: Úkol přesunout postavu do určitého bodu.

Útok: Úkol zaútočit na nepřítele.

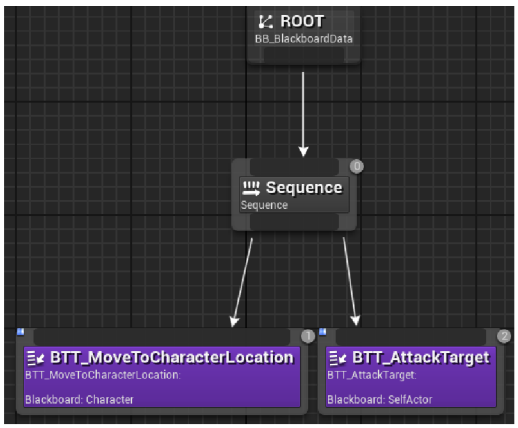
Blackboard je datová struktura, která se používá k ukládání a výměně informací mezi stromem chování a aktuálním stavem postavy. Obsahuje proměnné zvané klíče, které představují data používaná k rozhodování ve stromu chování.

Jak již bylo zmíněno, kromě standardních úkolů je možné implementovat své vlastní, což bylo provedeno pro tento projekt. K úkolu je vytvořen Blueprint třídy BTTask_BlueprintBase a v něm je již zapsána logika určitého úkolu a data jsou přenesena do proměnných desky.

V tomto projektu byly implementovány úkoly k získání náhodného bodu v okruhu od postavy, útoku na cíl a pohybu směrem k hráči. Zbytek požadované funkcionality byl již ve standardní sadě úloh.

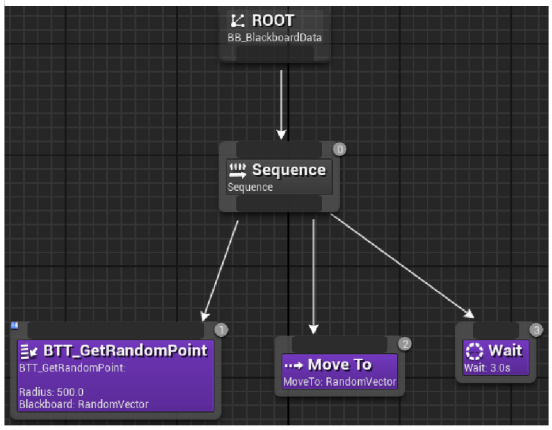
Nakonec byly implementovány dva stromy chování pro útok a patrolu oblasti.

Obrázek 40: Dřevo chování pro útok



Zdroj: Autor

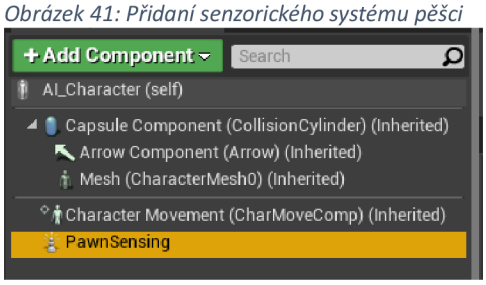
Obrázek 39: Dřevo chování pro patrolu



Zdroj: Autor

Důležitou součástí téměř každé umělé inteligence je její senzorový systém. Tento systém je důležitý zejména ve hrách se stealth mechanikou, tedy s možností tajného chování a průchodu. Funguje ten systém jako uši a oči.

Obrázek 41: Přidaní senzorického systému pěšci



Zdroj: Autor

V tomto projektu byl tento systém přidán pouze proto, aby umělá inteligence nemohla vždy předem znát pozici hráče a bylo nutné ji nejprve vidět. Tuto komponentu lze přidat do AI Blueprint a nakonfigurovat všechny její pocity.

4.2.10 **Bojová mechanika**

Umělá inteligence je připravena, dalším krokem je dát hráči možnost bránit se nepříteli. Pro tento projekt bylo rozhodnuto dát hráči dvě různá bojová kouzla, která fungují docela jednoduchým způsobem – v jakékoli hře jsou kulky, šípy, praky nebo kouzla jsou projektily, pro které engine poskytuje standardní, pohodlně konfigurovatelnou třídu.

Pro vytvoření magického projektilu byla v tomto projektu vytvořena třída C++, zděděná od Actoru s připojeným souborem "GameFramework/ProjectileMovementComponent.h". Proces tvorby projektilu je dost podobný jako vytváření meshu na sceně, rozdíl je pouze v dodatečném nastavení nového objektu, například nastavení rychlosti střely.

Jedinou funkcí této třídy v tomto projektu je její let a zničení při dosažení určité vzdálenosti, aby nedošlo k zaplnění počítačových zdrojů. Veškerá funkčnost je implementována pomocí funkce Tick.

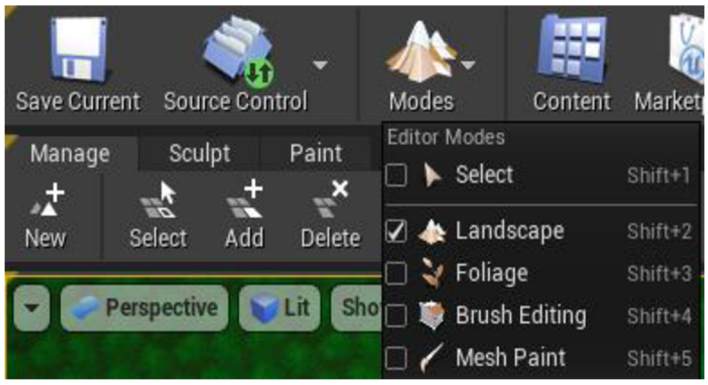
```
void AMagicProjectile::Tick(float DeltaTime)
{
	Super::Tick(DeltaTime);

	float Distance = FVector::Dist(this->GetActorLocation(), PawnPointer->GetActorLocation());
	if (Distance >= 6000.0f)
	{
		this->Destroy();
	}
}
```

Dalším řešením bylo zdědit třídy Blueprints z nové třídy projektilů a změnit tyto projektily v editoru přidáním 3D modelů a efektů. V budoucnu je možné interakci projektilů se světem realizovat jakýmkoli způsobem. Aby došlo k poškození nepřítele, byla do Blueprint AI přidána reakce na event dotyku projektilu, která vede ke snížení zdraví a následné smrti.

4.2.11 **Level**

Pro vytvoření úrovně v enginu existuje řada nástrojů. Vytvoření úrovně začíná vytvořením krajiny, kterou lze později změnit.

Obrázek 42: Nastroje pro vytváření úrovně



Pro vytvoření úrovně v enginu existuje řada nástrojů. Vytvoření úrovně začíná vytvořením krajiny pomoci nástroje Landscape. Tento nástroj umožňuje přidat sektory do krajiny, změnit jejich velikost a poté formovat úroveň.

Zdroj: Autor

Dalším důležitým aspektem při vytváření úrovně je přidání vegetace. Existuje na to nástroj Foliage, které umožní rychle zaplnit úroveň meshemi vegetace nebo čímkoli jiným. Vznikly tak dvě úrovně hry.

4.2.12 **Audio**

Přidání zvuku do Unreal Engine je docela jednoduché. Zvukový soubor je možné přidat pomocí metody Drag and drop, ale typ souboru musí být WAV. Dále lze tento zvukový soubor přidat na scénu, takže když se hra spustí, začne se hrát, ale je možné také propojit přehrávání zvukového souboru s nějakou akcí nebo událostí, ale k tomu je třeba přidat zvukový soubor do speciálního zvukového assetu Sound Cue a zavolat ho pomocí nody Play Sound 2D.

Obrázek 43: Volaní funkce přehrávání zvuku

Zdroj: Autor

5 Výsledky a diskuse

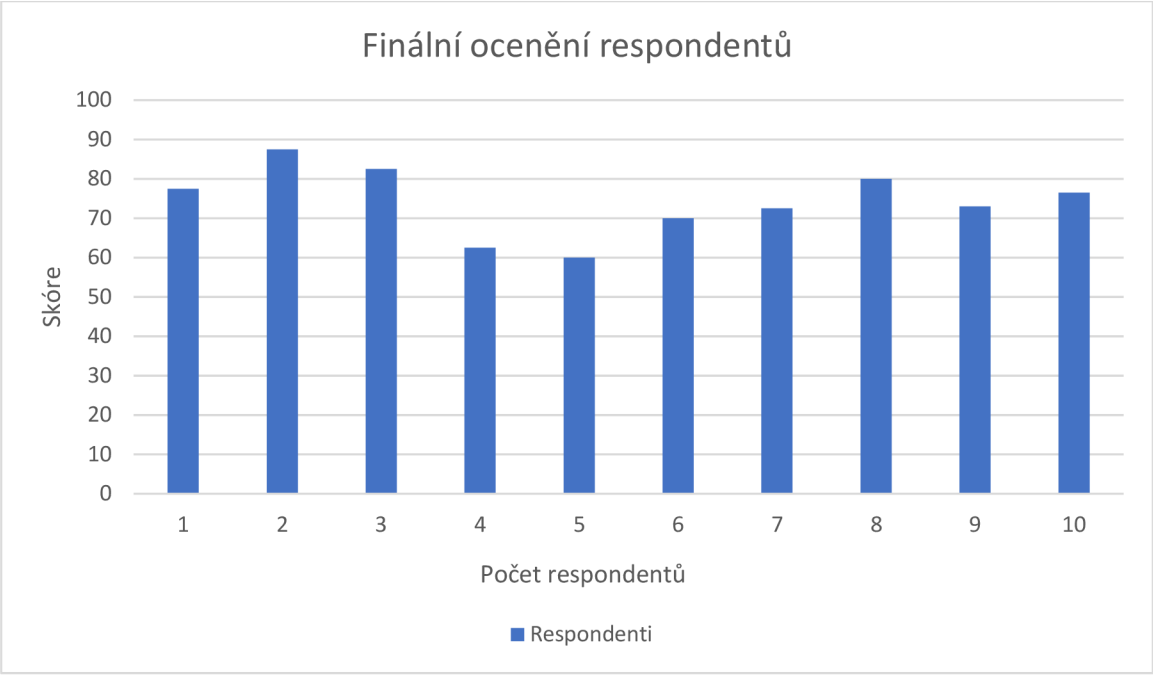
Během vývoje, každá významná inovace byla otestována a následně představena několika účastníkům závěrečného testování prototypu hry, a tím pádem každá inovace se stávala předmětem diskuse různých lidí a na základě všeobecného hodnocení bylo rozhodnuto, co by se s inovací mělo dělat. Tento přístup umožnil přivést funkčnost stávajících herních funkcí do vhodného stavu pro finální verze hry a v některých případech zcela změnil přístup k vývoji některých herních aspektů, což vynucovalo modifikovat zavedenou funkci, zcela ji změnit, použít jinou technologii nebo ji zcela odstranit. Tento přístup se autorovi zdál vhodný pro vývoj malých aplikací nebo her.

Přestože byl projekt testován během vývoje, plnohodnotné posouzení je možné pouze s hotovým produktem, a proto byl na základě všech možností, které hra poskytuje, sestaven seznam otázek, které by měly poskytnout poměrně jasnou představu o tom, jak lidé vidí tuto hru.

Po dokončení hry dostal každý tester seznam otázek SUS, ale standardní otázky byly přizpůsobeny tak, aby lépe vyhovovaly realitě tohoto projektu a které měly zhodnotit technologie použité ve hře, kvalitu vykonání konkrétních herních aspektů a obecné dojmy ze hry, a tento dotazník vypadá takto:

1. Myslím si, že bych rád hrál tuto hru opakovaně.
2. Myslím si, že je prototyp hry zbytečně složitý.
3. Myslím si, že se prototyp hry snadno používá.
4. Myslím si, že bych potřeboval/a pomoc člověka z technické podpory, abych mohl/a ten prototyp hrát.
5. Myslím si, že jsou funkce do prototypu hry dobře začleněny.
6. Myslím si, že je prototyp hry nekonzistentní.
7. Myslím si, že se s prototypem hry většina lidí naučí rychle pracovat.
8. Myslím si, že se prototyp hry těžko ovládá.
9. Při hraní prototypu se cítím jistě.
10. Musím se naučit hodně věcí, abych mohl/a začít prototyp hrát.

Graf 1: Finální ocenění respondentů



Zdroj: Autor

Podle výzkumu provedeného odborníky na UX je skóre 68 dobrým průměrem. Skóre pod 68 bodů znamená, že produkt je třeba zlepšit, a pokud je skóre vyšší, znamená to, že uživatelé jsou s produktem spokojeni. Průměrné skóre v tomto průzkumu je 74, což je docela dobrý výsledek podle tabulky pro interpretaci hodnot hodnocení.

6 Závěr

Tato bakalářská práce představuje syntézu teoretických a praktických poznatků z oblasti vývoje počítačových her na platformě Unreal Engine 4. Cílem této práce bylo seznámit čtenáře s klíčovými principy a postupy při tvorbě herního prototypu na této platformě a poskytnout ucelený pohled na celý proces vývoje her.

Teoretická část práce se detailně věnovala různým aspektům vývoje her, zahrnujícím programování v UE4, tvorbu herního design dokumentu, hodnocení uživatelského zážitku (UX) a použití System Usability Scale. Dále byly rozebrány klíčové fáze produkce her, jako jsou předprodukce a produkce, a zkoumány nástroje a postupy, jako jsou Git pro verzování kódu, design herních úrovní, modelování a animace, texturování, řízení stavů hráče a hry a aspekty zvukového designu.

Praktická část práce podrobně popisuje proces vytváření herního prototypu na Unreal Engine 4. Zahrnuje ukázky kódu v programovacím jazyce C++ pro různé aspekty hry, přičemž důraz je kladen na implementaci klíčových herních funkcí. Průběžné testování prototypu bylo prováděno v průběhu vývoje s různými osobami, což umožnilo identifikaci a řešení potenciálních problémů a zdokonalení herního zážitku.

Výsledky průběžného testování a konečného vyhodnocení pomocí SUS dotazníku poskytují užitečné informace o kvalitě uživatelského rozhraní a celkové uživatelské spokojenosti s vyvinutým herním prototypem. Tyto poznatky jsou klíčové pro další zdokonalování a optimalizaci vytvořeného herního prostředí.

Celkově lze konstatovat, že bakalářská práce splnila svůj cíl a poskytla komplexní pohled na proces vývoje počítačových her na Unreal Engine 4. Získané poznatky by mohly sloužit jako užitečný základ pro budoucí výzkum a vývoj v oblasti herního průmyslu.

7 Seznam použitých zdrojů

ALATHAS, Hadi. How to Measure Product Usability with the System Usability Scale (SUS) Score. Online. MEDIUM. 2018. Dostupné z: https://uxplanet.org/how-to-measure-product-usability-with-the-system-usability-scale-sus-score-69f3875b858f. [cit. 2024-03-01].

BROOKE, John. SUS: A quick and dirty usability scale. Online. Dostupné z: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale. [cit. 2024-02-06].

BYJU'S. Reverberation. Online. 2018. Dostupné z: https://byjus.com/physics/reverberation/#:~:text=Reverberation%20is%20the%20phenomenon%20of,.%2C%20within%20a%20closed%20surface.. [cit. 2024-02-13].

COURSERA. What Does a UX Designer Do? Online. 2024. Dostupné z: https://www.coursera.org/articles/what-does-a-ux-designer-do. [cit. 2024-02-25].

CODE CAPERS. Version control: Effective use, issues and thoughts, from a gamedev perspective. Online. 2016. Dostupné z: https://www.what-could-possibly-go-wrong.com/version-control/#version-control-and-game-development. [cit. 2024-02-07].

EPIC GAMES. Balancing Blueprint and C++. Online. 2024. Unreal Engine 4 Documentation. 2024a. Dostupné z: https://docs.unrealengine.com/4.27/en-US/Resources/SampleGames/ARPG/BalancingBlueprintAndCPP/. [cit. 2024-01-25].

EPIC GAMES. C++ and Blueprints. Online. 2024b. Unreal Engine 4 Documentation. 2024. Dostupné z: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ClassCreation/CodeAndBlueprints/. [cit. 2024-03-11].

EPIC GAMES. Geometry Brush Actors. Guide to using Brushes to create level geometry in Unreal Editor. Online. 2024c. Unreal Engine 4 Documentation. Dostupné z: https://docs.unrealengine.com/4.27/en-US/Basics/Actors/Brushes/. [cit. 2024-02-19].

EPIC GAMES. Textures. Image assets used in Materials to apply to surfaces or drawn on-screen by the HUD. Online. 2024d. Unreal Engine 4 Documentation. Dostupné z: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Textures/. [cit. 2024-02-08].

EPIC GAMES. Audio Engine Overview. Online. 2024e. Unreal Engine 4 Documentation. Dostupné z: https://docs.unrealengine.com/4.27/en-US/WorkingWithAudio/AudioEngineOverview/. [cit. 2024-02-13].

EROLIN, Justice. What Is Unreal Engine? Online. BairesDevBlog. 2022. Dostupné z: https://www.bairesdev.com/blog/what-is-unreal-engine/. [cit. 2024-03-13].

FIRAT, Hasan Baran; MAFFEI, Luigi a MASULLO, Massimiliano. 3D sound spatialization with game engines: the virtual acoustics performance of a game engine and a middleware for interactive audio design. Online. 2021, roč. 2022, s. 539 - 556. Dostupné z: https://doi.org/10.1007/s10055-021-00589-0. [cit. 2024-02-13].

FUENTES, Lauren. 3D Modeling for Games: The Best Software Tools. Online. All3DP. 2022. Dostupné z: https://all3dp.com/2/3d-modeling-for-games-software/#google_vignette. [cit. 2024-02-09].

HARWELL, Caleb. Is it possible to use the unreal engine for architecture? Online. 2020. Dostupné z: Quora, https://www.quora.com/Is-it-possible-to-use-the-unreal-engine-for-architecture. [cit. 2024-03-15].

JENSEN, K. Thor. 25 Years Later: The History of Unreal and an Epic Dynasty. Online. PCMag. 2023. Dostupné z: https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty. [cit. 2024-03-11].

KARLSSON, Johan. Modern Game Design Process: Pre to Post Production. Online. 2020. Dostupné z: https://www.perforce.com/blog/hns/modern-game-design-process. [cit. 2024-02-08].

MEDIUM. UCD vs UX: What's the difference? Online. Medium. 2018. Dostupné z: https://uxplanet.org/ucd-vs-ux-whats-the-difference-255443efa5f#:~:text=While%20user%2Dcentered%20design%20refers,concept%20rather%20than%20a%20process.. [cit. 2024-03-11].

MICROSOFT. Co je agilní vývoj? Online. Microsoft Learn. 2023. Dostupné z: https://learn.microsoft.com/cs-cz/devops/plan/what-is-agile-development. [cit. 2024-03-11].

MILANOTE. Create a modern, lightweight Game Design Document in Milanote. Online. Dostupné z: https://milanote.com/guide/game-design-document. [cit. 2024-02-06].

NEPOR, Vladimír. Začíname s C++ v Unreal Engine. Online. VRapps. 2023. Dostupné z: https://www.vrapps.cz/blog/zaciname-s-cpp-v-unreal-engine. [cit. 2024-03-11].

NIXON, David. Beginning Unreal Game Development. Foundation for Simple to Complex Games Using Unreal Engine 4. Online. Apress, 2020. ISBN 978-1-4842-5638-1. Dostupné z: https://doi.org/10.1007/978-1-4842-5639-8. [cit. 2024-02-19].

NUCLINO. How to Write a Game Design Document (GDD). Online. Nuclino. 2023b. Dostupné z: https://www.nuclino.com/articles/write-game-design-document. [cit. 2024-03-11].

NUCLINO. Video Game Development Process. How successful games are made, from start to finish. Online. 2024a. Dostupné z: https://www.nuclino.com/articles/video-game-development-process. [cit. 2024-02-06].

NUCLINO. Video Game Level Design. Learn how to design challenging, immersive, and engaging levels. Online. 2024c. Dostupné z: https://www.nuclino.com/articles/level-design. [cit. 2024-02-06].

OPTIMIZELY. A/B testing. Online. 2021, 2024. Dostupné z: https://www.optimizely.com/optimization-glossary/ab-testing/. [cit. 2024-03-14].

PACKT. BeginPlay and Tick. Online. 2023. Dostupné z: https://subscription.packtpub.com/book/game-development/9781800209220/1/ch01lvl1sec11/beginplay-and-tick. [cit. 2024-03-11].

PLURALSIGHT. Key 3D Rigging Terms to Get You Moving. Online. 2023. Dostupné z: https://www.pluralsight.com/blog/film-games/key-rigging-terms-get-moving#:~:text=What%20is%203D%20rigging%3F,be%20deformed%20and%20moved%20around.. [cit. 2024-02-09].

QUESTIONPRO. System Usability Scale: What it is, Calculation + Usage. Online. 2018. Dostupné z: https://www.questionpro.com/blog/system-usability-scale/#:~:text=The%20System%20Usability%20Scale%20(SUS)%20is%20a%20questionn

aire%20that%20is,systems%2C%20whether%20software%20or%20hardware.. [cit. 2024-03-13].

RODRIGUEZ, Andres; CRUZ GARDEY, Juan; GRIGERA, Julian; ROSSI, Gustavo a GARRIDO, Alejandra. UX debt in an agile development process: evidence and characterization. Online. 2023. Dostupné z: https://doi.org/10.1007/s11219-023-09652-2. [cit. 2024-02-06].

SATHEESH, Pv. Beginning Unreal Engine 4 Blueprints Visual Scripting. Online. 2021. ISBN 978-1-4842-6395-2. Dostupné z: https://doi.org/10.1007/978-1-4842-6396-9. [cit. 2024-02-08].

SCHOEPS MIKROFONE. DIFFUSE FIELD. Online. 2024. Dostupné z: https://schoeps.de/en/knowledge/knowledge-base/technical-basics/diffuse-field-1.html. [cit. 2024-02-14].

STEFYN, Nadia. How video games are made: the game development process. Online. CG Spectrum. 2022. Dostupné z: https://www.cgspectrum.com/blog/game-development-process. [cit. 2024-02-06].

TAYLOR, Noah. 3D Animation and 3D Modelling services in game development. Online. 2023. Medium. Dostupné z: https://medium.com/@nareshjuego/3d-animation-and-3d-modelling-services-in-game-development-a2158af0322e. [cit. 2024-02-09].

TEAMHUB. Project Management Tools for Game Development: A Comprehensive Guide. Online. Dostupné z: https://teamhub.com/blog/project-management-tools-for-game-development/. [cit. 2024-03-14].

TELES, André R. a SANTOS, André L. Code Merging using Transformations and Member Identity. Online. Roč. 2023. Dostupné z: https://doi.org/10.1145/3622758.3622891. [cit. 2024-02-06].

THE LEVEL DESIGN BOOK. How to make a level. Overview of general workflow and concepts for video game level design. Online. 2023. Dostupné z: https://book.leveldesignbook.com/process/overview. [cit. 2024-02-06].

TORONTO FILM SCHOOL. What is Video Game Animation and How Does it Work? Online. 2023, 2023. Dostupné z: https://www.torontofilmschool.ca/blog/what-is-video-game-animation-and-how-does-it-work/. [cit. 2024-02-09].

USABILITY.GOV. System Usability Scale (SUS). Online. 2024. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [cit. 2024-02-06].

VALE, Gustavo; COSTA, Heitor a APEL, Sven. Predicting merge conflicts considering social and technical assets. Online. Roč. 2024. Dostupné z: https://doi.org/10.1007/s10664-023-10395-8. [cit. 2024-02-06].

WARFIGHTER DIGITAL. How to import 3D models into the Unreal Engine. Online. 2023. Dostupné z: https://www.warfighterpodcast.com/blog/how-to-import-3d-models-into-the-unreal-engine/#:~:text=Open%20the%20Unreal%20Engine%20and,ones%20you%20want%20to%20import.. [cit. 2024-02-08].

WORLD OF LEVEL DESIGN. UE4: Step-by-Step to Your First Landscape Material (2 Texture Paint Layers). Online. 2020. Dostupné z: https://www.worldofleveldesign.com/categories/ue4/first-beginner-landscape-material.php. [cit. 2024-02-07].

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

Obrázek 1: Výběr kategorie při spuštění verze UE 4.27.2 13
Obrázek 2: Nabídka šablon pro hru 14
Obrázek 3: Class Wizard 15
Obrázek 4: Dědění z C++ třídy 16
Obrázek 5: Moderní GDD 19
Obrázek 6: Sloučení větev v Git 21
Obrázek 7: Ilustrace sloučení dvou různých větví do jedné konečné 21
Obrázek 8: Fáze Level Designu 23
Obrázek 9: Prototyp první úrovně 24
Obrázek 10: Low-poly modelování 25
Obrázek 11: Procedural modelling 26
Obrázek 12: 3D Sculptures 26
Obrázek 13: Additivní Animace 27
Obrázek 14: Joint illustration 28
Obrázek 15: Driven Keys 28
Obrázek 16: Blend Shape 29
Obrázek 17: Tvorba primitivní textury 29
Obrázek 18: Míchaní dvou textur nástrojem pro malování 30
Obrázek 19: Opci importu do UE 31
Obrázek 20: Přímý zvuk, ranní odrazy a pozdní reverberace 32
Obrázek 21: GDD projektu 37
Obrázek 22: UE4 Browser 38
Obrázek 23: Šablony UE4 39
Obrázek 24: Nastavení projektu 39
Obrázek 25: UE4 Interface 40
Obrázek 26: Horní panel 40
Obrázek 27: Place Actors 41
Obrázek 28: Low-poly strom 41
Obrázek 29: Sublime Merge 42
Obrázek 30: Nastavení Source Control 42
Obrázek 31: Volání funkce rolování při rolování kolečkem myši 44
Obrázek 32: Adding inputs 44
Obrázek 33: Vytvoření funkční knihovny 47
Obrázek 34: Nastavení akce na vstupu 47
Obrázek 35: Změna stavu postavy 48
Obrázek 36: Prostor míchaní v UE 49
Obrázek 37: Nastavení konečné pózy v animačním Blueprint 50
Obrázek 38: Programování animačního Blueprint 50
Obrázek 39: Dřevo chování pro patrolu 52
Obrázek 40: Dřevo chování pro útok 52
Obrázek 41: Přidaní senzorického systému pěšci 52
Obrázek 42: Nastroje pro vytváření úrovně 53
Obrázek 43: Volaní funkce přehrávání zvuku 54
Obrázek 44: První úroveň 69

Obrázek 45: Druhá úroveň 69
Obrázek 46: Třetí úroveň 69

8.2 Seznam tabulek

Tabulka 1: SUS dotazník 34
Tabulka 2: Tabulka pro interpretace SUS hodnocení 35

8.3 Seznam grafů

Graf 1: Finální ocenění respondentů 56

Přílohy

Příklady kódu

[1]

```
void ATDSCharacter::MovementTick(float DeltaTime)
{
	FVector FVec = CameraBoom->GetForwardVector();
	FVector RVec = CameraBoom->GetRightVector();
	FRotator Rot = FVec.Rotation();
	Rot.Pitch = 0.0f;
	FVec = Rot.Vector();

	AddMovementInput(FVec, AxisX);
	AddMovementInput(RVec, AxisY);

	APlayerController* MyController = UGameplayStatics::GetPlayerController(GetWorld(), 0);
	if (MyController)
	{
		FHitResult ResultHit;
		MyController->GetHitResultUnderCursorByChannel(ETraceTypeQuery::TraceTypeQuery6, false,
ResultHit);

		float FindRotatorResultYaw = UKismetMathLibrary::FindLookAtRotation(GetActorLocation(),
ResultHit.Location).Yaw;
		SetActorRotation(FQuat(FRotator(0.0f, FindRotatorResultYaw, 0.0f)));
	}
}
```

[2]

```
void ATDSCharacter::SetupPlayerInputComponent(UInputComponent* NewInputComponent)
{
	Super::SetupPlayerInputComponent(NewInputComponent);

	NewInputComponent->BindAxis(TEXT("MoveForward"), this, &ATDSCharacter::InputAxisX);
	NewInputComponent->BindAxis(TEXT("MoveRight"), this, &ATDSCharacter::InputAxisY);
}

void ATDSCharacter::InputAxisX(float Value)
{
	AxisX = Value;
}

void ATDSCharacter::InputAxisY(float Value)
{
	AxisY = Value;
}
```

[3]

```
UENUM(BlueprintType)
enum class EMovementState : uint8
{
	Walk_state UMETA(DisplayName = "Walk state"),
	CastWalk_state UMETA(DisplayName = "Cast Walkd state"),
	Run_state UMETA(DisplayName = "Run state"),
	CastRun_state UMETA(DisplayName = "Cast Run state"),

	//For animations
	Jump_state UMETA(DisplayName = "Falling state")

};

USTRUCT(BlueprintType)
struct FCharacterSpeed
{
	GENERATED_BODY()

public:

	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
		float WalkSpeed = 600.0f;
	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
		float CastWalkSpeed = 300.0f;

	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
		float RunSpeed = 800.0f;
	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
		float CastRunSpeed = 500.0f;

	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Movement")
		float JumpSpeed = 500.0f;
};
```

[4]

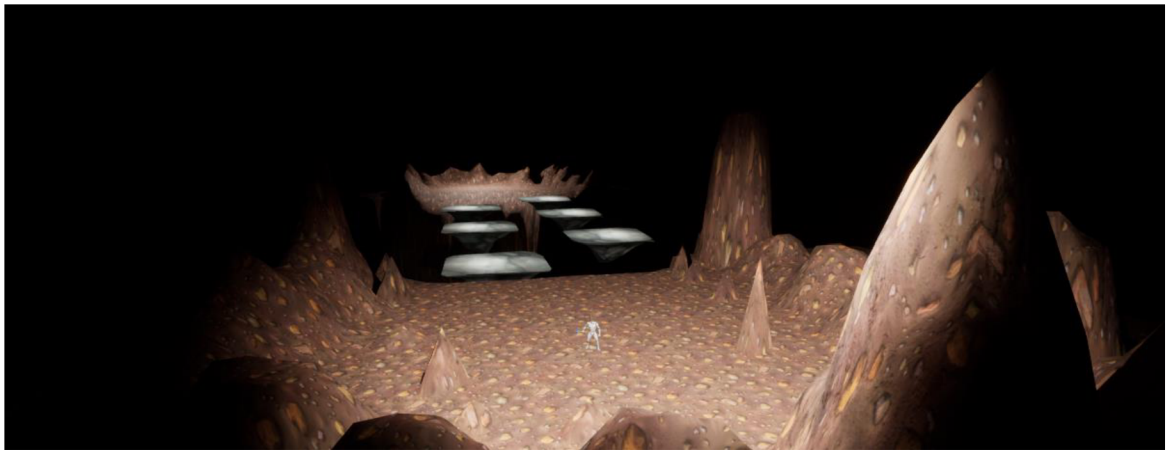
```
void ATDSCharacter::ChangeMovementState()
{
	if (JumpEnabled)
	{
		MovementState = EMovementState::Jump_state;
	}
	else
	{
		if (!RunEnabled && !CastEnabled)
		{
			MovementState = EMovementState::Walk_state;
		}
		if (!RunEnabled && CastEnabled)
		{
			MovementState = EMovementState::CastWalk_state;
		}
		if (RunEnabled && CastEnabled)
		{
			MovementState = EMovementState::CastRun_state;
		}
		if (RunEnabled && !CastEnabled)
		{
			MovementState = EMovementState::Run_state;
		}
	}

	CharacterUpdate();
}



void ATDSCharacter::CharacterUpdate()
{
	float ResSpeed = 600.0f;
	switch (MovementState)
	{
	case EMovementState::Walk_state:
		ResSpeed = MovementInfo.WalkSpeed;
		break;
	case EMovementState::CastWalk_state:
		ResSpeed = MovementInfo.CastWalkSpeed;
		break;
	case EMovementState::Run_state:
		ResSpeed = MovementInfo.RunSpeed;
		break;
	case EMovementState::CastRun_state:
		ResSpeed = MovementInfo.CastRunSpeed;
		break;
	case EMovementState::Jump_state:
		ResSpeed = MovementInfo.JumpSpeed;
		break;
	default:
		break;
	}
	GetCharacterMovement()->MaxWalkSpeed = ResSpeed;
}
```

Snímky z prototypu hry

Obrázek 44: První úroveň

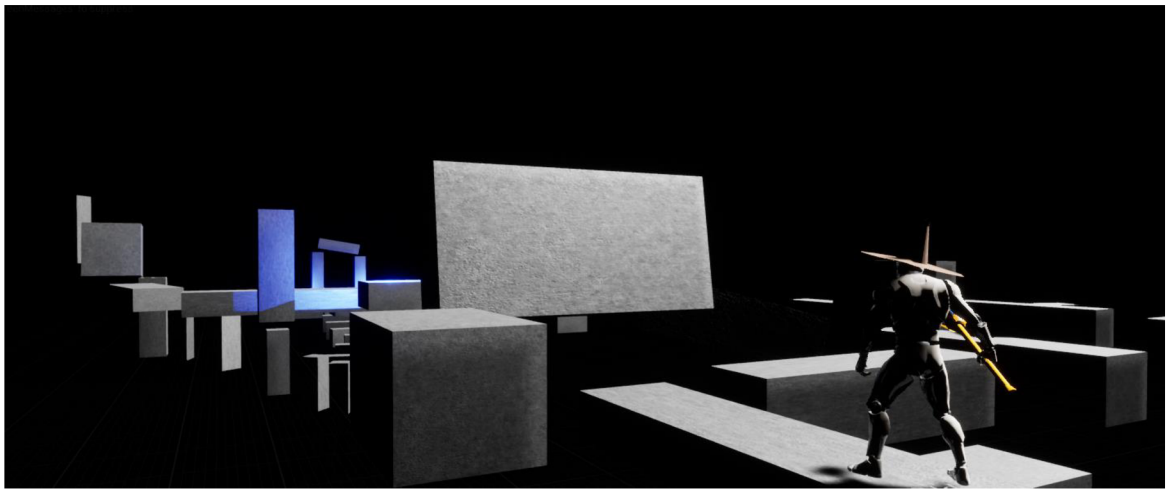
Zdroj: Autor

Obrázek 45: Druhá úroveň



Zdroj: Autor

Obrázek 46: Třetí úroveň



Zdroj: Autor