Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj 3D herní aplikace pro virtuální realitu

Štěpán Braun

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Štěpán Braun

Informatika

Název práce

Vývoj 3D herní aplikace pro virtuální realitu

Název anglicky

Development of a 3D game application for virtual reality

Cíle práce

Cílem bakalářské práce je vytvořit 3D akční herní aplikaci pro virtuální realitu pomocí herního enginu Unity. Dílčím cílem práce je představení správných postupů při vytváření aplikace pro virtuální realitu v tomto enginu, které může být užitečné nejen začínajícím vývojářům.

Metodika

Teoretická část práce se bude zabývat tématem vývoje aplikací. V této části budou popsány zejména obecné, ale i praktické postupy pro vývoj aplikací a her. Okrajově budou představeny herní enginy pro vývoj aplikací, virtuální realita a metody distribuce vytvořené aplikace. Praktická část bude zaměřena na implementování zjištěných poznatků z teoretické části do reálného návrhu a vývoje aplikace. Výsledkem bude vytvoření herní aplikace pro virtuální realitu pro platformu Windows. Součástí praktické části bude také realizace uživatelského testování. Na základě poznatků z testování bude aplikace upravena a následně bude definován její budoucí vývoj.

Doporučený rozsah práce

35-40 stran

Klíčová slova

Vývoj aplikace, Virtuální realita, Unity, herní engine, asset, programování

Doporučené zdroje informací

COWAN, Brent a Bill KAPRALOS, 2014. A Survey of Frameworks and Game Engines for Serious Game Development [online]. Dostupné z: doi:10.1109/ICALT.2014.194

DOOLEY, John, 2017. Software Development, Design and Coding. APress. ISBN 148423152X

LINOWES, Jonathan, 2020. Unity 2020 Virtual Reality Projects. Packt. ISBN 9781839217333

SCHELL, Jesse, 2008. The Art of Game Design: A Book of Lenses. CRC Press. ISBN 0123694965

ZIMMERMAN, Eric a Katie SALEN, 2003. Rules of Play. MIT Press. ISBN 978-0-262-24045-1

Předběžný termín obhajoby

2022/23 ZS – PEF

Vedoucí práce

Ing. Tomáš Benda

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 26. 11. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj 3D herní aplikace pro virtuální realitu" 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 29.11.2022 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Tomášovi Bendovi za odborné vedení této práce, pomoc a cenné rady.

Vývoj 3D herní aplikace pro virtuální realitu

Abstrakt

Tato bakalářská práce se zabývá vývojem herní aplikace pro virtuální realitu v herním enginu Unity.

V teoretické části se práce zaměřuje na obecné postupy při vývoji softwaru. Následně je čtenář seznámen s technikami pro tvorbu návrhu hry v rámci herního designu. Další pasáž práce je věnována virtuální realitě, herním enginům a distribuci hotové hry. V neposlední řadě jsou představeny metody uživatelského testování.

V praktické části práce byl na základě literární rešerše vytvořen návrh hry. Poté byla implementována virtuální realita v herním enginu Unity. Následovala demonstrace použití návrhových vzorů na části kódu.

V závěru práce bylo realizováno uživatelské testování, které se skládalo z plnění zadaných scénářů, vyplnění dotazníku a interview uživatelů s autorem hry. Testování bylo zaměřeno zejména na přehlednost ovládání a uživatelského rozhraní. Zkoumán byl také vliv hry na vznik kybernevolnosti. Z dotazníku vyplynulo, že uživatelské rozhraní i ovládání je přehledné. Kybernevolnost se objevila u 30 % uživatelů, kteří zmínili lehkou nevolnost. Výsledkem následného interview bylo odhalení třech chyb, které byly následně opraveny. Hra Fumus byla hodnocena pozitivně. Nejčastěji zmiňované pozitivum hry byl design levelů.

Celkovým výsledkem praktické části je funkční herní aplikace pro virtuální realitu pro platformu Windows, která se skládá ze tří herních úrovní.

Klíčová slova: Vývoj aplikace, Virtuální realita, Unity, herní engine, asset, programování

Development of a 3D game application for virtual reality

Abstract

This bachelor thesis deals with the development of a virtual reality game application in the Unity game engine.

In the theoretical part, the thesis focuses on general software development practices. Subsequently, the reader is introduced to the techniques of game design. Another part of the thesis is about virtual reality, game engines and distribution of the finished game. Finally, user testing methods are introduced.

In the practical part of the thesis, a game design was created based on knowledge acquired from the theoretical part. After that, virtual reality was implemented in the Unity game engine. This was followed by a demonstration of the application of the design patterns with code examples.

At the end of the thesis, user testing was carried out. It consisted of users performing the given scenarios, filling in a questionnaire and being interviewed by the author of the thesis. The testing focused mainly on the clarity of the controls and user interface. The influence of the game on the emergence of cybersickness was examined as well. The questionnaire showed that the user interface and controls were clear. Cybersickness appeared in 30% of users who mentioned slight nausea. A follow-up interview resulted in the discovery of three bugs, which were subsequently fixed. The game Fumus was rated positively. The level design was the most frequently mentioned as the positive of the game.

The overall result of the practical part is a functional virtual reality game application for the Windows platform, which consists of three game levels.

Keywords: Application development, Virtual reality, Unity, game engine, asset, programming

Obsah

1 Úvod

Virtuální realita je perspektivní obor, do kterého velké firmy v posledních letech investují značné finanční prostředky. Využití nalézá nejen v zábavním sektoru, ale i například v oblastech zdravotnictví, vzdělávání, průmyslu či architektury.

Vytváření aplikací pro virtuální realitu, vzhledem ke složitosti dané technologie, není zcela jednoduché. Z tohoto důvodu existují na trhu herní enginy, které usnadňují práci při vývoji dané aplikace. Mezi nejvíce využívané herní enginy patří Unity a Unreal Engine. V rámci teoretické části této práce budou představeny nástroje v herním enginu Unity, které vývojáři usnadní implementaci virtuální reality v aplikaci.

Hlavním cílem této bakalářské práce je vytvoření herní aplikace pro virtuální realitu v herním enginu Unity. Práce se však zaměřuje i na popis obecných postupů při tvorbě softwaru, které mohou být užitečné pro začínající vývojáře. Předmětem práce je také představení postupů pro vytvoření zábavné hry, ke které se budou hráči vracet. V neposlední řadě budou zmíněny postupy pro otestování vytvořené hry.

Vytvořená hra bude otestována v rámci uživatelského testování. Zkoumán bude zejména vznik kybernevolnosti a přehlednost hry. Dalším cílem testování bude odhalení potencionálních vzniklých chyb při tvorbě hry. Díky poznatkům respondentů testování bude možné upravit hru za účelem minimalizování vzniku kybernevolnosti, existence chyb a zlepšení přehlednosti hry. V závěrečné části práce bude definován budoucí vývoj hry.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je vytvořit 3D akční herní aplikaci pro virtuální realitu pomocí herního enginu Unity. Dílčím cílem práce je představení správných postupů při vytváření aplikace pro virtuální realitu v tomto enginu, které může být užitečné nejen začínajícím vývojářům. Dalším dílčím cílem práce je představení správných postupů při tvorbě návrhu herní aplikace, tak aby maximalizovala zájem hráče vracet se k hraní dané hry. Mezi další dílčí cíle taktéž patří minimalizování vzniku kybernevolnosti, která se často u VR aplikací vyskytuje. V neposlední řadě je dílčím cílem práce také minimalizovat počet chyb ve vytvořené videohře, které ovlivňují herní zážitek ze hry.

2.2 Metodika

Teoretická část práce se bude zabývat tématem vývoje aplikací. V této části budou popsány zejména obecné, ale i praktické postupy pro vývoj aplikací a her. Okrajově budou představeny herní enginy pro vývoj aplikací, virtuální realita a metody distribuce vytvořené aplikace. Praktická část bude zaměřena na implementování zjištěných poznatků z teoretické části do reálného návrhu a vývoje aplikace. Výsledkem bude vytvoření herní aplikace pro virtuální realitu pro platformu Windows. Součástí praktické části bude také realizace uživatelského testování. Testování bude probíhat plněním dvou předem definovaných testovacích scénářů, včetně protokolu think aloud, následovaný dotazníkem a interview s autorem práce. Na základě poznatků z testování bude aplikace upravena a následně bude definován její budoucí vývoj.

3 Teoretická východiska

3.1 Vývoj softwaru

Tato kapitola se zabývá obecnými principy při vývoji softwaru. Vývoj softwaru je komplexní činnost zahrnující několik fází. Mezi tyto fáze patří koncepce, analýza požadavků, návrh, implementace (programování a ladění), testování, vydání, údržba a vyřazení softwaru. Některé fáze vývoje mohou být vynechány, obecně ale vývoj softwaru zahrnuje většinu zmíněných činností. (Dooley, 2017)

3.1.1 Metodiky vývoje softwaru

Metodiky vývoje softwaru jsou sady doporučených praktik a postupů, které pokrývají celý životní cyklus softwaru. (Kodytek, 2022) Metodiky vývoje softwaru umožňují efektivní řízení projektu. Každá metodika má svoje výhody a nevýhody. (Dooley, 2017) V následující kapitole jsou zmíněné často využívané metodiky.

3.1.1.1 Vodopádový vývoj

Vodopádový vývoj je tradiční model vytvořený roku 1970 Winstonem Roycem. (Dooley, 2017) Název dostal podle schématu, jelikož jednotlivé fáze probíhají chronologicky a připomínají tak vodopád. Tento model se zabývá všemi standardními fázemi životního cyklu. Mezi dané fáze patří analýza požadavků, návrh softwaru, implementace, testování softwaru, vydání a údržba. Každá část pak vyžaduje detailní dokumentaci. Před začátkem další fáze by měla být ideálně ukončená fáze předchozí. (Dooley, 2017) Tato podmínka může být problematická například tehdy, pokud se během vývoje změní požadavky. (Koďousková, 2022)

3.1.1.2 Agilní vývoj

Agilní vývoj je zaměřen na vývoj malých a středních projektů. Vyžaduje méně dokumentace než vodopádový vývoj, neboť podstatou této metodiky je to, že základem každého softwaru je fungující kód. Daná metodika umožňuje vývojářům rychle reagovat na měnící se požadavky, jelikož klade důraz na časté vydávání průběžných verzí, zapojení zákazníka do vývoje a přepisování kódu tak, aby byl jednodušší a snazší na údržbu. (Dooley, 2017) Vývoj aplikace je rozdělen do několika cyklů. Každý cyklus obsahuje všechny fáze vývoje softwaru. Na konci cyklu je prototyp aplikace představen klientovi. Na základě jeho připomínek je software upraven v následujícím cyklu. Klient má tak možnost průběžně reagovat na proběhlé změny a měnit požadavky. (Koďousková, 2022)

3.1.1.3 Code and Fix

Code and Fix není metodika v pravém slova smyslu. Tento způsob vývoje softwaru je využíván zejména začínajícími programátory a obsahuje minimum zmíněných fází procesu vývoje. Podstatou tohoto modelu je strávit co nejméně času přípravami a co nejdříve začít psát kód. Vývojář se tedy snaží pouze obecně pochopit daný problém a následně začíná se psaním softwaru. Pokud se objeví po zkompilování kódu nějaký problém, snaží se jej ihned vyřešit. Tento proces opakuje do té doby, než se mu podaří software dokončit. Tato metodika není vhodná pro práci v týmu. Využije jí pouze jedinec, který se snaží o vytvoření rychlého prototypu. (Dooley, 2017)

3.1.2 Životní cyklus softwaru

Následující podkapitola se zabývá představením jednotlivých fází vývoje softwaru. Proces, který rozděluje vývoj softwaru na jednotlivé fáze, se nazývá životní cyklus softwaru. (Dooley, 2017)

Analýza požadavků
První fází vývoje softwaru je analýza požadavků. Jejím cílem je definovat požadavky softwaru a skládá se ze tří základních částí. (Dooley, 2017)

1. Rozdělení požadavků do kategorií a uspořádání do souvisejících oblastí.
2. Udělení priority jednotlivým požadavkům.
3. Prozkoumání vztahu jednotlivých požadavků (Dooley, 2017)

Analýza požadavků je klíčová ke splnění veškerých požadavků klienta. (Desyatnikov, 2022)

Návrh
Další fází vývoje softwaru je návrh. Při této fázi se na základě analýzy požadavků modeluje způsob, jakým bude software fungovat. Při navrhování softwaru se řeší například uživatelské rozhraní, architektura softwaru nebo bezpečnost. (JEVTIC, 2019) Pro vizualizaci návrhu softwaru jsou často klíčové UML diagramy, kterým je věnována kapitola UML (3.1.3). (MŮČKA, 2020)

Implementace
Implementování softwaru začíná v návaznosti na vytvoření návrhu softwaru. Jedná se o přepisování návrhu do zdrojového kódu dle požadavků programovacího jazyka. (JEVTIC, 2019)

Verifikace (testování)
Cílem verifikace softwaru je zjistit, zda funguje dle požadavků a neobsahuje chyby. V případě existence chyby je aplikace vrácena vývojáři k opravení. (Desyatnikov, 2022) Testování aplikace bude dále řešeno v kapitole Testování (3.8).

Spuštění a údržba
Při fázi spuštění je aplikace zpřístupněna běžným uživatelům. (JEVTIC, 2019) Údržba aplikace slouží k opravě chyb, které nebyly odhaleny při fázi verifikace, nebo pro rozšiřování funkcí aplikace. (Desyatnikov, 2022)

3.1.3 UML

Unified modelling language je klíčový nástroj pro analýzu a návrh softwaru. Jelikož se stal standardem, je pro programátora znalost UML nezbytná. (Čápka, 2022a) Z tohoto důvodu byla v této práci UML věnována tato kapitola.

UML je grafický jazyk, který se obecně používá buď jako náčrt, plán nebo programovací jazyk. Jako náčrt je používán pro snazší zobrazení problému a tím usnadňuje komunikaci. Jako plán slouží k implementaci pro vývojáře. Zároveň také poslouží k dokumentaci pro vytvoření softwaru. Jako programovací jazyk je možné UML diagram vygenerovat na kód, který slouží jako základní kámen pro další upravování. (Čápka, 2022a) K vizualizaci modelu v UML se používají diagramy. Rozdělujeme je do 3 skupin na strukturní diagramy, diagramy chování a diagramy interakce (viz tabulka č.1). (Nishadha, 2022)

V následujících podkapitolách budou zmíněny diagramy, které jsou nejčastěji používané při vývoji softwaru.

Strukturní diagramy	**Diagramy chování**	**Diagramy interakce**
Diagram tříd	Diagram aktivit	Sekvenční diagram
Diagram komponent	Diagram případů užití	Diagram komunikace
Diagram složených struktur	Stavový diagram	Diagram interakcí
Diagram nasazení		Diagram časování
Diagram balíčků		
Diagram objektů (diagram instancí)		
Diagram profilů		

Tabulka 1 Diagramy

3.1.3.1 Diagram tříd

Diagram tříd je základní stavební kámen každého objektově orientovaného modelu. Používá se k zobrazení struktury softwaru. (Pitman, 2005) Popisuje atributy, operace a vztahy mezi jednotlivými třídami. (Nishadha, 2022) Třídu je možné definovat jako šablonu pro vytváření objektů v objektově orientovaném programování, která reprezentuje skupinu objektů se

stejným stavem a chováním. Příkladem je třída Auto. Instance této třídy (objekty) jsou například jednotlivé druhy automobilů (Škoda, Ford, Volkswagen). Třída se v UML znázorňuje pomocí boxu rozděleného na 3 části. (Pitman, 2005) Název třídy se nachází v horní části boxu (obrázek č.1). V prostřední části jsou atributy. Spodní část obsahuje metody. Vztah mezi třídami je definován různými druhy šipek. (Nishadha, 2022) Diagram tříd by měl obsahovat všechny třídy, které bude aplikace obsahovat a v případě, že jej programátor přepíše do kódu, musí být funkční. (Čápka, 2022b)

Obrázek 1 Příklad diagramu tříd (Zdroj: Autor)

Viditelnost

Pomocí viditelnosti je možné definovat, zda jednotlivé atributy a metody konkrétní třídy budou viditelné a zda mohou být použité jinými třídami (viditelnost a přístupnost). Jedná se o důležitou součást diagramu, která je znázorněna pomocí symbolů. (IBM, 2021)

Viditelnost typu private se značí symbolem –. Třída je viditelná a přístupná pouze v rámci třídy.

Viditelnost typu public se značí symbolem +. Třída je viditelná a přístupná pro všechny třídy.

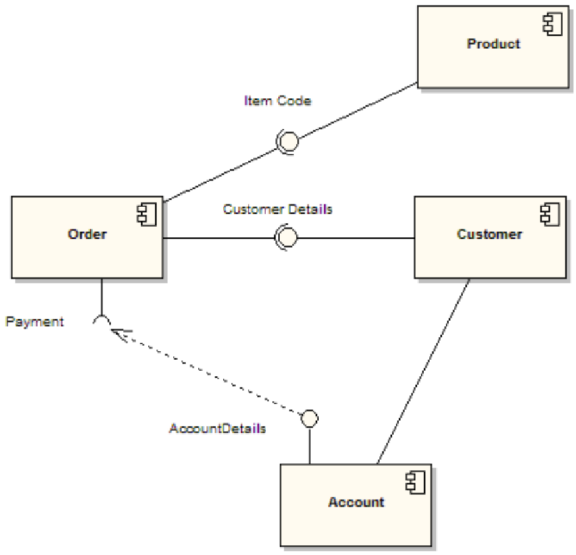
Viditelnost typu protected se značí symbolem #. Třída je viditelná a přístupná v rámci třídy a podtříd.

Viditelnost typu package se značí symbolem ~. Třída je viditelná a přístupná pouze v rámci třídy ze stejného balíku. (IBM, 2021)

3.1.3.2 Diagram komponent

Diagram komponent zobrazuje strukturální vztahy mezi komponenty softwarového systému. (Nishadha, 2022) Při modelování velkých systémů je běžné, že se software rozděluje do

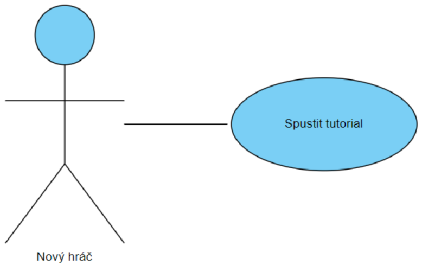
několika subsystémů. K tomu je určena právě komponenta. Jedná se o nahraditelnou část většího systému se skrytým obsahem. Komponenta by měla být navržena tak, aby mohla být opětovně použita. (Pitman, 2005) Komponenty mezi sebou navzájem komunikují pomocí rozhraní a mohou obsahovat atributy a metody. (Nishadha, 2022) Diagram komponent se od diagramu tříd liší vyšší mírou abstrakce. Komponenta je běžně implementovaná jednou nebo více třídami (či objekty). V uml je komponenta reprezentovaná pomocí obdélníku s názvem komponenty a klíčového slova <<component>>, které může být nahrazeno i patřičným symbolem ikony komponentu v pravém horním rohu. (Sparx Systems, 2022) Příklad diagramu komponent je demonstrován na obrázku č.2, který reprezentuje systém na objednávání produktu. Komponenta Customer a Product dodají komponentě Order požadované rozhraní o detailech zákazníka a kódu produktu. Komponenta Account je propojená s komponentou Order, která má požadované rozhraní Payment a poskytne ji rozhraní o účtu zákazníka. (Sparx Systems, 2022)



Obrázek 2 Příklad diagramu komponent (Sparx Systems, 2022)

3.1.3.3 Diagram případů užití

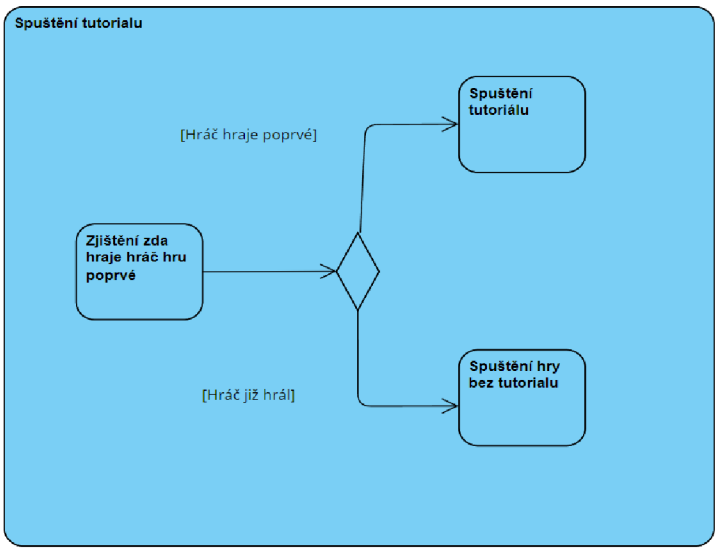
Diagram případů užití zobrazuje chování systému z pohledu uživatele. Diagram popisuje funkcionalitu systému. Nepopisuje však, jakým způsobem bude daná funkcionalita vykonávaná. Jedná se o soubor událostí, které vedou k dosažení cíle. Diagram případů užití bývá často první diagram, který se při návrhu informačního systému vytváří. Skládá se z případu užití (definovaná funkcionalita) a aktéra. Případ užití vychází z požadavků klienta. V diagramu je zpravidla značen pomocí elipsy s názvem dané funkcionality. Aktér je buď inicátor případu užití (aktivní aktér) nebo může být sám iniciován případem užití (pasivní aktér). Může se jednat o reálného uživatele či jakýkoliv vnější systém. Aktérem tak může být například uživatel, administrátor či server. (Čápka, 2022c) Velmi často bývá aktérem i čas, který případ užití aktivuje v rámci daného času či intervalu. (Nishadha, 2022) V diagramu je znázorněn jako postava s odpovídajícím názvem. (Čápka, 2022c) Příklad diagramu užití je demonstrován na obrázku č.3. Případ užití je spuštění tutoriálu, aktivním aktérem je nový hráč.



Obrázek 3 Příklad diagramu užití (Zdroj: Autor)

3.1.3.4 Diagram aktivit

Diagram aktivit popisuje jednotlivé kroky procesu, které je nutné vykonat od začátku k dosažení cíle. Zobrazuje propojení jednotlivých aktivit. (Microsoft 365 Team, 2019) Diagram aktivit zachycuje aktivity, které se skládají z jedné nebo více akcí. Akce reprezentují jeden krok aktivity a jsou dále nedělitelné. Aktivita se v UML značí jako obdélník se zaoblenými rohy. (Pitman, 2005) Diagram aktivit je možné využít i mimo modelování softwaru v libovolném podnikovém prostředí. (Microsoft 365 Team, 2019) Příklad diagramu aktivit je demonstrován na obrázku č.4. Tento diagram reprezentuje aktivitu spuštění tutoriálu při zapnutí hry. Mimo aktivity obsahuje diagram uzel rozhodnutí, který na základě podmínky určí odpovídající následující aktivitu. (Pitman, 2005) V diagramu je zakreslen jako kosočtverec.



Obrázek 4 Příklad diagramu aktivit (Zdroj: Autor)

3.2 Objektově orientované programování

Objektově orientované programování (OOP) je v současné době jedním z nejčastěji používaným programovacím paradigmatem, jež spočívá v organizování návrhu softwaru kolem objektů. Objekt lze definovat jako datové pole, obsahující jedinečné atributy a chování. Mezi největší výhody objektově orientovaného programování patří možnost opakovaného použití kódu, rozdělení vývoje mezi více vývojářů, bezpečnost a možnost snadného rozšiřování kódu. Základními konstrukčními prvky objektově orientovaného programování jsou třídy, objekty, metody a atributy. Mezi nejvýznamnější objektově orientované programovací jazyky patří C#, Java, Python nebo Visual Basic .Net. (Gillis, 2021) Objektově orientované programování bude využito v praktické části této bakalářské práce, jelikož psaní kódu při vývoji herní aplikace bude probíhat v jazyce C#. Tento programovací jazyk bude využit v návaznosti na volbu herního enginu Unity pro vypracování praktické části. Herní enginy a jejich programovací jazyky jsou dále popsány v kapitole Game Engine (3.3).

3.2.1 Návrhové vzory

Při navrhování struktury kódu objektově orientovaného programování lze použít návrhové vzory. Jedná se o navržená znovu použitelná řešení často se vyskytujících problémů. (Ampatzoglou, 2007) Existuje 23 základních návrhových vzorů, které popisují obecné řešení nejčastějších problémů. V současné době, ale existuje mnoho návrhových vzorů, které se zabývají řešením specifických problémů. (Pecinovský, 2007)

V následujících podkapitolách jsou popsány návrhové vzory, které budou použity při psaní kódu v praktické části této bakalářské práce, bude-li k tomu vhodná příležitost.

3.2.1.1 Jedináček (singleton)

Jedináček najde uplatnění v případě, kdy je potřeba, aby v programu existovala pouze jedna instance třídy. Této třídě je pak přidělen globální přístup. Instance dané třídy může být použita kdykoliv je potřeba. Díky tomu je možné zamezení vytváření nadbytečných objektů. Samotná třída je zodpovědná za to, aby nebyla vytvořená další instance. Návrhový vzor jedináček je možné využít například v případě, že je nutné mít pouze jednu databázi, kterou bude možné sdílet se zbytkem softwaru. (Sarcar, 2018)

3.2.1.2 Pozorovatel (observer)

Návrhový vzor pozorovatel řeší problém závislostí mezi objekty. V tomto návrhovém vzoru je mnoho pozorovatelů, kteří sledují určitý objekt. Pokud dojde ke změně uvnitř objektu, pozorovatelé jsou o této změně informováni. Pozorovatelé se mohou libovolně přihlašovat a odhlašovat k objektu v závislosti na tom, zda chtějí být o změnách informováni. Příkladem návrhového vzoru pozorovatel je propojení uživatelského rozhraní (pozorovatel) a databáze. Pokud dojde ke změně uvnitř databáze, je uživatelské rozhraní o této změně informováno a zobrazí aktuální informace. (Sarcar, 2018)

3.2.1.3 Stav (state)

Tento návrhový vzor umožňuje objektu změnit chování dojde-li ke změně jeho vnitřního stavu. Objekt se pak chová, jako by se stal instancí jiné třídy. (Sarcar, 2018) Tento návrhový vzor eliminuje nutnost velkého počtu podmínek v kódu a dělá jej jednodušším a přehlednějším. Umožňuje taktéž snazší hledání chyb v chování objektu, neboť je stav přímo v názvu třídy objektu. Příkladem návrhového vzoru stav je třída Hráč v akční videohře, která má stavy HráčVMenu, HráčVBoji a HráčMimoBoj. Při změně stavu se hráč chová jako instance třídy daného stavu. Jednotlivé stavy mají definované rozdílné chování po stlačení levého tlačítka myši. Stav HráčVMenu umožní hráči interagovat s tlačítky menu, HráčVBoji útočit a HráčMimoBoj interagovat s předměty. (Haney, 2011)

3.2.1.4 Šablonová metoda (template method)

Šablonová metoda definuje kostru algoritmu a umožňuje rozdělení jednotlivých kroků do podtříd. Struktura algoritmu se však nemění. (Sarcar, 2018) Použití tohoto návrhového vzoru je vhodné, pokud máme například v aplikaci algoritmy se stejnou strukturou, ale odlišným chováním.
Šablonová metoda dokáže zpřehlednit kód a umožnit snazší rozšiřování systému. (Čápka, 2022d)

3.3 Game engine

Game engine je nástroj usnadňující vývojáři vývoj softwaru. Jedná se o sadu nástrojů a rozhraní umožňující skriptování, renderování, animace a další užitečné funkce. Dané funkce může vývojář využít a díky tomu ušetřit čas i peníze, jelikož nemusí vše vytvářet od úplného začátku. Obecně herní enginy disponují grafickým rozhraním. Každý engine má svoje specifika. Například programovací jazyk, který je využíván při psaní skriptů. Velká vývojářská studia si při vývoji vytváří vlastní enginy, které pak používají při další tvorbě. V současné době je na trhu k dispozici několik herních enginu, které může kdokoliv využívat. Mezi nejvíce používané patří Unity a Unreal Engine. (Cowan, 2014)

3.3.1 Unreal Engine

Unreal engine od firmy Epic games je jeden z nejvíce využívaných herních enginů. Nevyužívá se pouze pro tvorbu her, ale najde uplatnění i při tvorbě filmu a videí, jelikož zkušený uživatel tohoto enginu je schopen vytvořit fotorealistické prostředí, a dokonce je využít v reálném čase ve spojení s kamerou. Skriptování v tomto enginu probíhá pomocí jazyka C++. (Andrade, 2015) Unreal engine nabízí placené i bezplatné licence. V případě zvolení bezplatné licence a překonání příjmu 1 milionu dolarů si Epic games účtuje 5 % v licenčních poplatcích. (Epic Games, 2022)

3.3.2 Unity

Unity je herní engine, který byl poprvé představen v roce 2005. Stejně jako Unreal engine umožňuje Unity vývoj her pro počítače, konzole, smartphony i webové prohlížeče. Pokud roční obrat vývojáře nepřekročí 100 000 dolarů, může Unity využívat bezplatně. Skriptování probíhá v jazyce C# nebo UnityScript. (Andrade, 2015) Pro tvorbu hry v této bakalářské práci byl využit herní engine Unity, zejména kvůli předchozí zkušenosti autora této práce s programovacím jazykem C#.

3.4 Hra

Jedním z cílů této bakalářské práce je vytvoření herní aplikace pro virtuální realitu. Tato kapitola definuje pojem videohry a zabývá se principy návrhu videohry, které budou následně využívány v praktické části této práce.

Francouzský spisovatel a filozof Roger Cailloise definuje hru jako "dobrovolnou interaktivní činnost, při níž se jeden nebo více hráčů řídí pravidly, která omezují jejich chování, a rozehrává umělý konflikt, který končí kvantifikovatelným výsledkem.“ (Esposito, 2005) Videohra dle definice označuje hru, kterou můžeme hrát díky audiovizuálním zařízení. Mezi tyto zařízení patří vstupní zařízení, jako například myš, klávesnice, gamepad a taktéž i zařízení výstupní – například monitor nebo reproduktory. Videohry mají často příběh, avšak existují i hry bez příběhu – například hra Tetris. (Esposito, 2005) Podle zařízení, na kterém se daná videohra může hrát, je lze rozdělit na počítačové, konzolové a mobilní. V současné době se často hry vyvíjí jak pro počítač, tak pro konzole. V řadě případů existují i mobilní verze her, které byly původně vyvíjeny pro ostatní platformy. Videohry jsou určeny primárně pro zábavu, jsou ale i využívány ke vzdělávacím účelům. Příkladem je videohra Operation Flashpoint od českého studia Bohemia interactive studio. Tento vojenský simulátor byl v minulosti využíván při výcviku vojáků. (Robinson, 2012) Hry najdou využití i ve školách, kupříkladu historická série Age of Empires pomáhá v hodinách dějepisu a dokáže oživit vyučovací hodiny, díky poměrně přesnému popisu historických událostí. Existují i případy, kdy vznikne speciální vyučovací edice již populárních her. Jeden z příkladů je Minecraft. I přes to, že vyučující samotnou hru do výuky již aktivně zařazovali, vznikla nová edukativní verze Minecraft Education. Tato verze je vhodná mimo jiné při hodinách chemie, neboť umožňuje provádění řady chemických pokusů. (Panja, 2021)

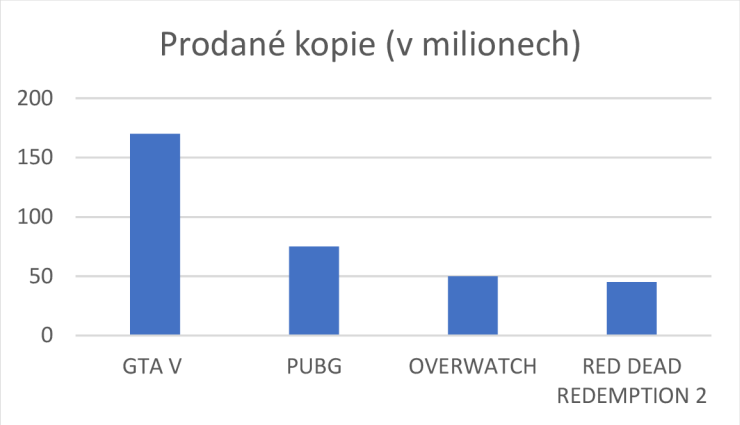
3.4.1 Žánry

Podle herního stylu můžeme videohry rozdělit do několika kategorií – žánrů. Není neobvyklé, že je videohra mixem několika žánrů. Mezi nejčastější žánry patří akční hry, strategie, sportovní hry, simulátory, závodní hry či adventury. (Doherty, 2018)

Pro vývoj hry v této bakalářské práci byl zvolen žánr akční hra, jelikož s tímto žánrem měl autor práce nejvíce zkušeností.

3.4.1.1 Akční hra

Cílem akční hry je pomocí přidělených zbraní zneškodnit nepřátele. Tento herní žánr obvykle klade důraz na rychlé reakce hráče. Akční videohry mohou nabídnout hráči pohled z první osoby (first person shooter), kdy hráč vidí pouze ruce se zbraní, nebo z pohledu třetí osoby, kdy vidí celou herní postavu. (Doherty, 2018) V žebříčku nejprodávanějších videoher historie se nachází 4 akční videohry. Jedná se o hru GTA V, PUBG, Overwatch a Red Dead Redemption 2. Dohromady se prodalo více než 340 milionu kopii těchto her. (Sirani, 2022)



Graf 1 Prodané kopie v milionech (Sirani, 2022)

3.4.2 Herní design

"Herní design je rozhodování o tom, jaká by hra měla být." (Schell, 2019) V širším slova smyslu se jedná o rozhodování o pravidlech, odměnách, trestech, rychlosti hry, vzhledu a všem ostatním co hráč během hraní zažije. Cílem herního designéra je vytvořit zábavnou hru, kterou si hráč bude užívat a vracet se k ní. Jelikož je průběh vytváření herního designu prakticky vymýšlení nápadů, není k němu třeba žádné speciální vybavení. Začátečníkovi stačí papír a tužka. Herní design není aplikovatelný pouze na počítačové hry, ale lze jej využít i pro deskové hry, karetní hry nebo třeba pohybové hry. Tato bakalářská práce se však bude zabývat herním designem počítačové hry. Herní designer musí promyslet velkou řadu aspektů, které ovlivňují zážitek z hraní dané hry. (Schell, 2019)

3.4.2.1 Pravidla

Jeden z nejdůležitějších aspektů hry, které musí herní designer řešit, jsou pravidla a cíl hry. Pravidla určují, jakým stylem bude daná hra hrána, následky akcí a cíle. Pravidla by měla být lehká na pochopení. Pravidla obvykle vznikají a zdokonalují se průběžně s vývojem hry. Také je třeba si určit, jaký je hlavní cíl hry a zda je jasný všem hráčům. (Schell, 2019)

3.4.2.2 Aspekt překvapení

Další důležitý aspekt je aspekt překvapení. Lidský mozek je nastaven tak, že se dokonce i při nepříjemným překvapení aktivuje centrum potěšení. Designer by si tak měl klást otázku, co hráče překvapí při hraní hry. Může se jednat o příběh, vzhled hry, pravidla. Záleží na

kreativitě designera, překvapení hráče se dá docílit spousty kreativními způsoby, ale je to jedna z věcí, která by neměla ve hře chybět. (Schell, 2019)

3.4.2.3 Zábava

Jeden z nejdůležitějších aspektů je aspekt zábavy. Herní designer musí řešit otázku, která část hry je zábavná, která naopak není a z jakého důvodu tomu tak je. Zpravidla platí že úkoly, které je člověk povinen plnit, nejsou vnímané jako zábavné. Je třeba si při navrhování hry uvědomit, aby hráč neplnil úkoly pouze z povinnosti. Jednou z možností, jak oživit stereotypní úkoly, je překonávání vlastních rekordů. Je třeba ale myslet na to, že každý hráč je jiný a činnosti, které baví jednoho hráče, mohou jinému hráči přijít nudné. Další ze způsobů, jak neztratit zájem hráče o hru, je adekvátní odměňování za dobré výkony. Odměny by však neměly být příliš časté a jednoduché k získání, jelikož by mohla klesnout hráčova motivace k jejich získání. Designer tak musí řešit několik otázek současně – jak často hráče odměňovat, jak docílit, aby byly odměny dostatečně atraktivní nebo například zda spolu odměny souvisí. Zábavnost hry se dá paradoxně zvýšit i pomocí trestů. Pomocí trestů je možné vytvářet větší výzvy nebo například přinutit hráče více riskovat. V případě, že je balanc mezi potenciální odměnou a trestem při riskování adekvátně vybalancován a hráči se podaří uspět, je získaná odměna vnímaná mnohonásobně uspokojivěji než stejná odměna získaná bez riskování. Je nutné vybrat správný balanc tak, aby hráče tresty dostatečně motivovaly hrát a zároveň jej nefrustrovaly natolik, že hru přestane hrát. Designer řeší otázky, jako například z jakého důvodu je hráč trestán a čeho se tím má dosáhnout, nebo zda jsou tresty hráčem vnímané fér. (Schell, 2019) Existují však i hry, které si zakládají na vysoké obtížnosti a častém trestání hráčů. Příkladem je herní série Dark Souls, která je považována za jednu z nejtěžších her, a i přes to je velmi úspěšná. Překonávání jednotlivých levelů je pro hráče velkou výzvou, a díky tomu jsou u hry schopni strávit hodiny. (Stanton, 2016) Je třeba dodat, že každý hráč preferuje jinou herní obtížnost a z tohoto důvodu je dobrý nápad do hry implementovat možnost volby obtížnosti. Hráči, kteří mají rádi větší výzvy, si zvolí těžší obtížnost. Naopak hráči preferující lehčí hry, zvolí obtížnost snazší. Zpravidla platí, že odměny jsou lepším motivačním prvkem než tresty. (Schell, 2019)

3.4.2.4 Příběh

Dalším významným aspektem hry je příběh. Designer by se měl nejdříve rozmyslet, zda a z jakého důvodu by měla hra mít příběh. Jak již bylo zmíněno v předchozích kapitolách, existují i hry bez příběhu, které dosáhly velkého úspěchu. Příběh dokáže oživit stereotypní úkoly a dodat hráči motivaci se ve hře posouvat a odpovídat si tím na dosud nezodpovězené otázky. Jeden z problémů, na které je si třeba dát pozor, je klišé. Pokud hráč od začátku tuší, jak daný příběh skončí, může ztratit zájem hru dohrát. Jedním z řešení jsou nečekané zvraty, které zároveň podpoří zmiňovaný aspekt překvapení. (Schell, 2019)

3.4.2.5 Dovednost

Další aspekt, který designer řeší, je dovednost. Je třeba si rozmyslet, které dovednosti budou od hráčů vyžadovány. Zda jsou někteří hráči v těchto dovednostech přirozeně lepší než ostatní hráči a zda se mohou hráči v daných dovednostech trénováním zlepšovat. Možnost zlepšovat se v dovednosti, která za lepší provedení hráče odmění, dokáže hráče donutit se ke hře vracet. (Schell, 2019)

3.4.2.6 Délka hry

K dalším aspektům patří čas. Hra by neměla být příliš krátká, ale ani tak dlouhá, aby hráče nezačala nudit. Přidání časového limitu může vytvořit větší výzvy a více vzrušující zážitek. Časový limit však není vhodný pro každou hru a je třeba si rozmyslet, zda se do dané hry hodí. (Schell, 2019)

3.4.2.7 Estetika

V neposlední řadě je třeba vyřešit aspekt estetiky. Volbou správného estetického stylu je možné zaujmout pozornost hráče, protože vizuální stránka hry je jedna z prvních částí hry, kterých si hráč všimne. Jedná se o grafické zpracování, prostředí, vzhled hlavní postavy nebo třeba zvuky. Není třeba volit fotorealistické grafické zpracování, je možné zaujmout i jednoduchým originálním zpracováním. Je nutné však podotknout, že pouze skvělé grafické zpracování nezaručí úspěch hry. Hráči častěji upřednostní zábavné hry s horší grafikou před nudnou hrou s dokonale propracovaným grafickým zpracováním. (Schell, 2019)

Mimo zmíněné aspekty herního designu existuje spousta dalších aspektů ovlivňující zážitek ze hry. Zmíněny však byly pouze ty, které mohou být řešeny v rámci praktické části této bakalářské práce.

3.5 Virtuální realita, rozšířená realita

Virtuální realita je „počítačově generovaná simulace 3D prostředí“. (Linowes, 2020) Pomocí speciálního headsetu se může uživatel rozhlížet po okolí a pomocí ovladačů interagovat s prostředím nebo se pohybovat. Virtuální realita najde uplatnění zejména ve zdravotnictví a zábavě. Její použití je však možné ve všech odvětvích, které používají 3D modely a počítačově generovanou grafiku. Správně použití virtuální reality dokáže zlepšit velké množství současných online služeb od marketingu, sociálních sítí či online obchodu. Rozšířená realita využívá počítačově generované objekty a snímky reálného světa. Nejčastější využití rozšířené reality je na chytrých telefonech s využitím vestavěného fotoaparátu. Existují však i headsety, jako například Hololens od firmy Microsoft. Tyto headsety jsou v podstatě průhledné brýle, na které se vykresluje počítačově generovaná grafika. (Linowes, 2020) Rozšířená realita najde uplatnění ve velkým množství sfér. Lze použít například při designování interiéru, kdy uživatel může do prostoru vkládat nábytek či jiné objekty. Uplatnění najde také například ve zdravotnictví, prezentování produktů, při opravování a údržbě komplexní zařízení, cestovním ruchu či jiných odvětvích. (PAINE, 2018)

3.5.1 Zařízení pro virtuální realitu

Zařízení pro přehrání virtuální reality můžeme rozdělit do dvou kategorií. Počítačové (na stejném principu funguje i konzolová virtuální realita) a mobilní. Mobilní zařízení se začala používat s vydáním Google Cardboard. Jedná se o jednoduché zařízení se dvěma čočkami a

slotem na mobilní zařízení. Obrazovka telefonu po vložení do Cardboardu přehraje obraz, který se pomocí čoček rozdělí na dva obrazy, každý určený pro jedno oko uživatele. Mobilní zařízení sleduje pohyb hlavy a podle toho mění obraz. Pokud se tedy uživatel podívá doprava, i obraz na zařízení se pohne stejným směrem. Mobilní zařízení k vytváření obrazu využívá vlastní procesor. Jelikož mají mobilní zařízení zpravidla menší výkon než počítače, je tak komplexita obrazu omezená. Počítačová virtuální realita využívá propojení počítače a headsetu. Zatímco počítač pomocí procesoru a grafické karty vykresluje výsledný obraz, headset se stará o snímání pohybu a doručení obrazu k uživateli pomocí displeje. Headset se k počítači připojuje pomocí drátového připojení či bezdrátového připojení s nízkou latencí. (Linowes, 2020) V současné době existují i headsety s vlastním procesorem, které je sice možné připojit k počítači pro potřebu vyššího výkonu, avšak jsou schopny fungovat i samostatně. Příkladem headsetů pro virtuální realitu je Oculus Quest 2, HTC Vive Pro 2, Sony Playstation VR nebo Valve Index. Liší se mezi sebou cenou, funkcemi a zařízením, pro které jsou určeny. (Greenwald, 2022)

3.5.2 Kybernevolnost

Jeden z velkých problémů používání virtuální reality je nevolnost spojená s jejím používáním. Tento druh nevolnosti bývá označen jako kybernevolnost. (Marinho, 2021) Dle proběhlých studií 20 až 80 % respondentů zažilo nějakou formu kybernevolnosti při používání virtuální reality. (Chandra, 2022) Mezi příznaky kybernevolnosti patří pocení, nevolnost, dezorientace nebo závratě. Kybernevolnost může vznikat během i po použití virtuální reality. Faktory, které nejspíše zapříčiňují kybernevolnost, jsou rozlišení obrazovky, grafické zpracování prostředí, zorné pole, přesnost a rychlost odezvy snímání pohybu, věk uživatele, jeho předchozí zkušenosti s virtuální realitou a náchylnost ke kinetóze (nevolnost při cestování v dopravním prostředku). Jeden z důvodů vzniku kybernevolnosti je rozpor mezi vizuálními vjemy a rovnovážným ústrojím. Uživatel virtuální reality má díky headsetu pocit, že se pohybuje, v realitě ale sedí na místě. Dalším důvodem je nízký počet snímků za vteřinu. (Marinho, 2021) Doporučený počet snímků za vteřinu pro virtuální realitu je 90. Nižší hodnoty mohou zapříčinit vznik kybernevolnosti. (IrisVR, 2021) Počet snímků může být však limitován výkonem zařízení, náročností aplikace či obnovovací frekvencí headsetu. Obnovovací frekvence udává, kolikrát za vteřinu se obnoví obraz na obrazovce. Pokud tedy výkon zařízení i náročnost aplikace umožní vyšší počet snímků za vteřinu, ale headset má nízkou obnovovací frekvenci, vysoké snímky za vteřinu se dostatečně nevyužijí. (Intel Corporation, 2022) K minimalizování rizika vzniku kybernevolnosti jsou doporučeny časté přestávky. V případě, že chce vývojář minimalizovat rizika vzniku kybernevolnosti u uživatelů jeho hry, může do hry implementovat upozornění na přestávky každých 30 minut nebo navrhnout herní úkoly tak, aby trvaly právě 30 minut nebo méně. Taktéž by se měl snažit o optimalizování hry, aby byla schopná vykreslit vysoký počet snímků za vteřinu. (Marinho, 2021)

Součástí praktické části této bakalářské práce bude uživatelské testování, které se bude mimo jiné věnovat i kybernevolnosti a minimalizování rizik jejího vzniku.

3.5.3 Unity XR

Pro vývoj aplikace pro virtuální realitu lze stejně jako při vývoji obyčejné aplikace využít herní engine. Pro vývoj aplikace pro virtuální realitu v Unity je možné využít integrovaný XR Interaction toolkit a XR Plugin management. Jedná se o sady nástrojů, které umožňují například propojení headsetů od různých výrobců a virtuální kamery, snímání jeho pohybu nebo možnost interagovat s objekty pomocí ovladačů. V minulosti si musel vývojář tyto funkce buď sám naskriptovat nebo stáhnout z externích knihoven. První komponentou XR toolkitu je XR rig camera. Tato komponenta vytvoří virtuální kameru, která kopíruje pohyb headsetu. Vývojář má možnost volby ze 2 módů – roomscale xr rig a stacionary xr rig. Roomscale xr rig je určený pro scény, kde se uživatel volně pohybuje v určené zóně. Stacionary xr rig je určený pro scény, ve kterých uživatel pouze stojí nebo sedí na místě. Tyto módy lze libovolně měnit i v průběhu vývoje. Po přidání této komponenty do scény, připojení podporovaného headsetu k počítači a zmáčknutí tlačítka play v Unity se v headsetu ukáže náhled scény včetně možnosti rozhlížet se po okolí. Xr rig kromě kamery přidá objekty pro levý a pravý ovladač. Oba objekty se skládají z několika komponent. XR Controler se stará o sledování pozice, rotace ovladače a detekci zmáčknutých tlačítek. XR Ray Interactor se používá k interakci s objekty. XR Interactor Line Visual a Line Renderer vytvoří a vykreslí pomocnou linii, která udává uživateli informace, na jaké místo míří ovladač. (Linowes, 2020) V nové verzi XR toolkitu je XR Rig nahrazen komponentou XR Origin, fungují ale na podobných principech.

3.6 Herní asset

Herní asset je libovolný prvek, který je součástí hry. Jedná se o grafické prvky (3D modely, 2D modely, loga, ikony,textury), zvuky (hudba, zvukové efekty) a dokonce i samotný kód. Vývojář má zpravidla dvě možnosti, jak dané assety získat. Může si je vytvořit sám, nebo si je pořídit z obchodu. Pro vytvoření assetů potřebuje vývojář patřičný software. V případě 2D grafiky jsou nejvyužívanější programy, například bezplatný program Gimp, placená alternativa Adobe Photoshop, Adobe Illustrator či jiný grafický editor. Pro 3D grafiku je třeba využít modelovací program. Vývojář si může vybrat mezi bezplatným programem jako například Blender. Využít však může i placené programy, jedním z nich je Cinema 4D. (Vionix, 2022)

3.6.1 Unity Asset Store

Pokud chce vývojář ušetřit čas, může si assety obstarat v internetovém obchodě. Herní engine Unity má vlastní obchod Unity Asset Store, kde se nachází velké množství assetů, ze kterých si může kdokoliv vybrat. Je zde řada 3D i 2D modelů, zvukové efekty, šablony, vizuální efekty či řada užitečných nástrojů. Unity pak nabízí možnost vybrané assety rovnou importovat do enginu. Na webové stránce obchodu si uživatel vybere konkrétní asset, klikne na možnost Add to my assets, přijme podmínky užívání a poté zvolí možnost Open in unity. Importování pak probíhá pomocí nástroje Package Manager v Unity. Uživatel si označí

zvolený asset, stáhne jej a tlačítkem import importuje do projektu. Zároveň je možné na Unity asset store vkládat vlastní assety a případně je i prodávat. (Unity Technologies, 2022)

3.7 Distribuce hotové hry

Herní aplikaci lze distribuovat dvěma způsoby. Pomocí digitálních a fyzických kopií. V posledních letech se staly digitální kopie dominantním způsobem distribuce her. V roce 2021 téměř 90 % her ze všech konzolových her, které byly vydány ve Spojených státech amerických, bylo distribuováno výhradně v digitální podobě. 10,3 % her bylo distribuováno kombinovaně pomocí digitálních i fyzických kopií a pouze 0,6 % her výhradně pomocí fyzických kopií. (Pledge Times, 2022)

Digitální distribuce
Existuje řada digitálních obchodů, na kterých je možné videohry distribuovat. Výběr vhodného obchodu může být však limitován platformou, pro kterou je daná hra určena. Herní konzole Sony Playstation a Xbox nebo například chytré telefony od firmy Apple nabízí možnost distribuce digitálních kopií pouze pomocí jejich vlastních obchodů. Vývojáři her pro počítače mohou volit z více alternativ. Největší digitální platforma pro distribuci her je Steam. (MICHAUD, 2012) K distribuci hry na této platformě je třeba si založit účet, zaplatit jednorázový poplatek, vyplnit platební a daňové informace, naplánovat vydání, včetně nahrání buildu a vyčkat čekací lhůtu, během které proběhne otestování produktu zaměstnanci Steamu. Před vydáním první hry je také nutné vyčkat čekací lhůtu 30 dní, během které se ověřuje totožnost vydavatele. (Valve Corporation, 2022) Jedna z nevýhod této platformy je vysoký podíl Steamu na tržbách. Vývojář obdrží pouze 70 % z tržby a zbylých 30 % zůstane Steamu. Alternativou je Epic games store. Epic games si účtuje pouze 12 % z tržby hry a zbylých 88 % zůstane vývojáři. Tato platforma však není otevřená pro kohokoliv, jelikož Epic games si sami vybírají, kterou hru na platformě vydají. (Konvoy, 2021) Tato možnost tedy není vhodná pro začínající vývojáře, kteří nejsou zastoupeni vydavatelem nebo pokud se jedná o jeden z jejich prvních herních projektů.

3.8 Testování

Každá aplikace či hra by měla být před vydáním řádně otestována. Existuje několik druhů testů, kterými by si měla aplikace projít. Řadu z nich zvládne udělat samotný vývojář buď sám nebo s pomocí testovacích nástrojů. Jedná se například o testování výkonu, kdy se testují aspekty, jako například počet snímků za vteřinu, stabilita a náročnost aplikace na zařízení. (Game-ace, 2021)

3.8.1 Uživatelské testování

Dalším důležitým testem je test použitelnosti. Tento test se zaměřuje na hodnocení toho, jak je hraní hry příjemné a pohodlné. Vývojáři se často samotnému nepodaří odhalit veškeré problémy, které hra obsahuje. Proto je vhodné provádět toto testování na nezávislém uživateli. V ideálním případě na několika uživatelích. (Game-ace, 2021) Obvykle se testuje

na 6 až 12 uživatelích a je vhodné vybírat jednotlivé uživatele tak, aby korespondovali s cílovou skupinou dané hry. (Long, 2021) Před zahájením testování je třeba si vyjasnit konkrétní cíle testování. Mezi tyto cíle patří například informace o prvních pocitech uživatele při hraní hry, pochopení ovládání hry, zábavnost konkrétních prvků či otestování určitých herních mechanismů. Od tohoto se odvíjí průběh testování. (Bryant, 2020)

3.8.1.1 Průběh testování

Uživatel může být ponechán volnému hraní bez instrukcí, častěji ale následuje pokyny dle předem připraveného scénáře. Scénář se skládá z úkolů, jež má uživatel během testování vykonat. Úkoly jsou vybírány účelově dle konkrétních cílů testování. (Soukup, 2020) Uživatel může být před testováním požádán, aby nahlas popisoval svoje myšlenky a pocity. Jedná se o testovací metodu Think Aloud, která může pomoct objasnit uživatelova očekávání a které prvky jsou pro něj matoucí. (USABILITY BOK, 2012) Moderátor testování si na základě pozorování a popisování myšlenek uživatelem zapisuje poznámky o konkrétní problémech, které uživatel během testování zaznamenal. (Soukup, 2020)

3.8.1.2 Vyhodnocení testování

Po dokončení testování vyplní uživatel připravený dotazník. Následuje interview, které je tvořeno otevřenými otázkami. Na základě daného dotazníku, interview a poznámek pořízených během pozorování testované osoby při hraní, dojde k vyhodnocení testování. Pokud z výsledků testování vyplyne, že hra obsahuje chybu nebo problém, který ovlivňuje herní zážitek, měla by být hra upravena tak, aby tyto problémy byly eliminovány. (Long, 2021)

4 Vlastní práce

Cílem této části práce je vytvořit herní aplikaci pro virtuální realitu na základě poznatků získaných z teoretické části práce. Součástí vývoje herní aplikace je návrh hry, implementování virtuální reality v herním enginu, využití návrhových vzorů v rámci implementování kódu, uživatelské testování, úprava hry na základě poznatků získaných z uživatelského testování a diskuze o budoucím vývoji hry.

Při vývoji herní aplikace FUMUS byl využit způsob vývoje Code and Fix. Tento způsob vývoje softwaru byl představen v podkapitole Code and Fix (3.1.3.3).

4.1 Návrh hry

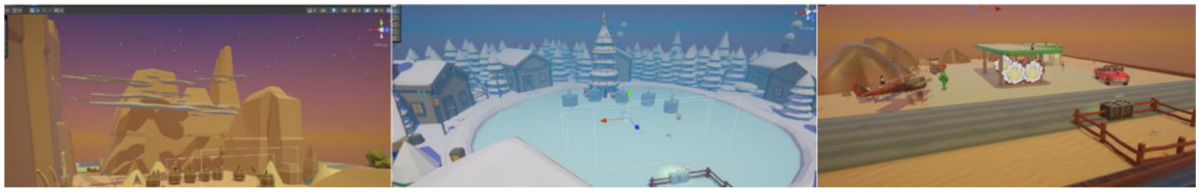
Na začátku vývoje herní aplikace pro virtuální realitu byl vytvořen návrh hry. Návrh hry byl vytvořen na základě poznatků z kapitoly Hra (3.4). Hra FUMUS je akční hra pro virtuální realitu z pohledu první osoby. Hra obsahuje 4 herní úrovně včetně tutoriálu. Cílem hráče je překonat jednotlivé úrovně dosažením daného počtu bodů. Body lze získat trefováním cílů pomocí zbraně. Dosažení daného počtu bodů umožní hráči postup do další úrovně. Hráči je umožněno pohybovat se v rámci omezeného prostoru. Hra taktéž obsahuje bonusové prvky, které hráči přidají výhody. Bonusové prvky byly použity na základě poznatků z teoretické části z kapitoly herní design k vytvoření aspektů překvapení a zábavy. Zároveň byl do hry přidán časový limit na splnění jednotlivých úrovní za účelem vytvoření větší výzvy a vzrušujícího zažitku z hraní hry. Časový limit byl přidán v návaznosti na podkapitolu Délka hry z teoretické části práce. Dále byla do hry implementovaná funkce nejvyššího skóre, která motivuje hráče k překonávání vlastních rekordů. Dovednosti, které bude hráč při hraní využívat, jsou rychlost a přesnost. V případě zlepšení dané dovednosti bude hráč odměněn vyšším skórem. Funkce nejvyššího skóre a dovednosti přímo reflektují poznatky z teoretické části z kapitoly Zábava (3.4.2.3) a Dovednost (3.4.2.5). Jelikož se jedná o hru zaměřenou zejména na překonávání vlastních rekordů, byla zvolena možnost vytvoření hry bez příběhu, jelikož by se mohlo jednat o rušivý element.

4.1.1 Bonusové prvky

Ve hře se nacházejí tři prvky, které hráči nabízejí unikátní výhody. Prvním z nich je bonusová zbraň, která má více nábojů. K jejímu získání je třeba trefit 4 lahve během 15 vteřin. Dalším bonusovým prvkem je pohybující se terč, který po trefení přidá hráči 2 body a 5 vteřin navíc. Terč se objeví pokaždé, když hráč dosáhne skóre, jež je dělitelné 4. Posledním bonusovým prvkem je létající pták. V případě, že jej hráč trefí, získá zaměřovač na zbraň na 10 vteřin. Dále hra obsahuje dvě skryté funkce, které však uživateli nepřidají žádné výhody. Jednou z nich je možnost zasažení modelu letadla v herní úrovní „shell“. Po opakovaném zasažení vrtule letadlo odletí a hráč obdrží zbraň střílející bubliny. Tato zbraň však není určena k plnění úkolu a slouží pouze k pobavení hráče. Dále se ve stejném levelu nacházejí benzínové pumpy, které po zásahu explodují.

4.1.2 Audiovizuální zpracování

Pro vzhled hry byl zvolen jednoduchý kreslený styl (lowpoly). Jednotlivé herní úrovně jsou zasazeny do unikátních prostředí. První úroveň je zasazena do prostředí pouště. Další úroveň byla inspirovaná vánoční tématikou. Třetí úroveň se odehrává na čerpací stanici inspirované 60. léty minulého století. Každá úroveň byla zároveň doplněna unikátní tematickou hudbou a hlasovým komentářem, který hráči pomáhá vysvětlit nastalé herní situace. Herní prostředí bylo vytvořeno přímo v herním enginu Unity kombinováním volně dostupných 3D objektů.



Obrázek 5 Grafické zpracování herních úrovní (Zdroj: Autor)

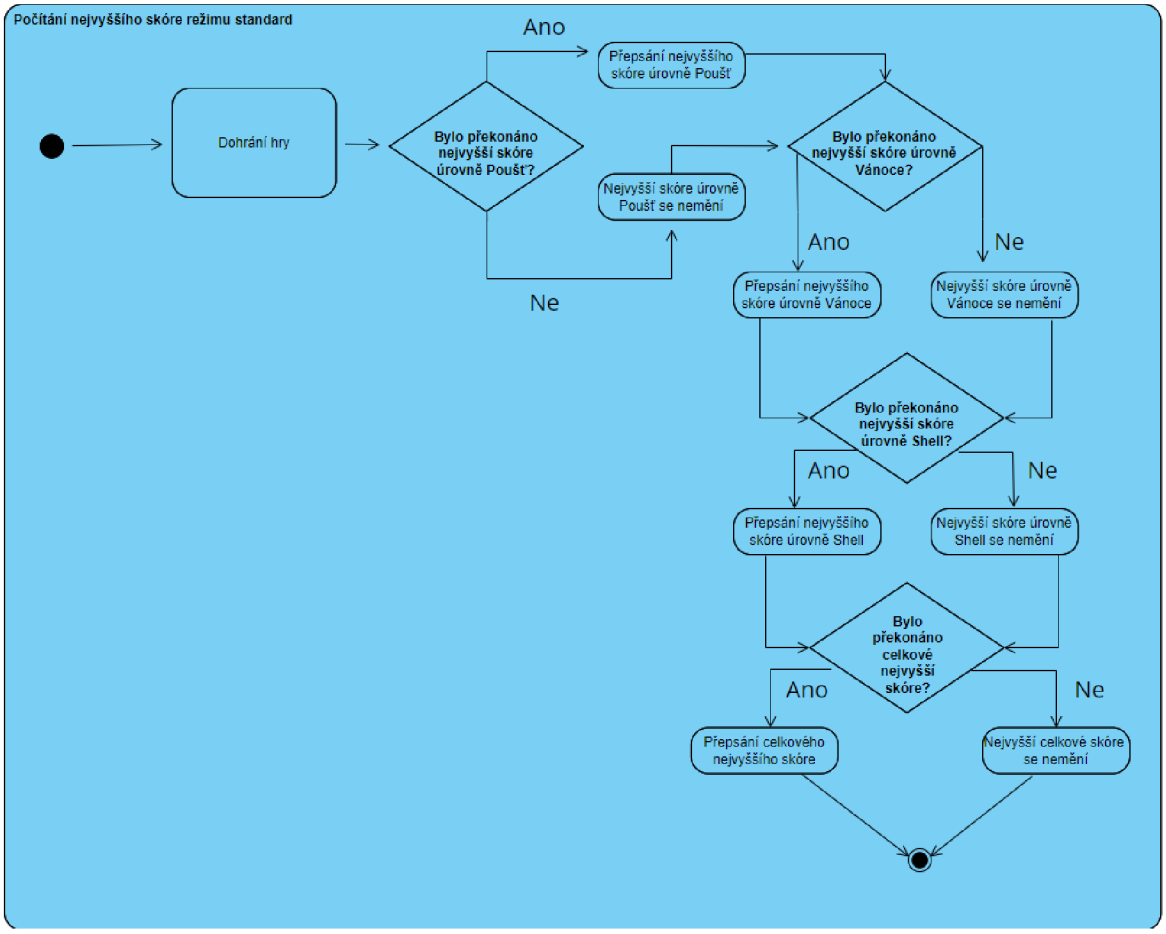
4.1.3 UML

Pro přehlednost a zefektivnění práce při tvorbě hry FUMUS byly vytvořeny UML diagramy. Využit byl diagram aktivit a diagram tříd.

Diagram aktivit

Diagram aktivit byl využit pro návrh počítání nejvyššího skóre ve standardním režimu (obrázek č.6). Standardní režim se skládá ze tří herních úrovní a je třeba nejdříve zjistit, zda v jednotlivých herních úrovní došlo k překonání nejvyššího skóre. Aktivita začíná dohráním hry. Dále byl použit uzel rozhodnutí, který na základě toho, zda bylo skóre dané úrovně překonáno či nikoliv, zvolí odpovídající akci. Po zanalyzování výsledků jednotlivých herních úrovní je třeba zjistit, zda bylo překonáno celkové nejvyšší skóre. Po vybrání odpovídající akce aktivita končí.

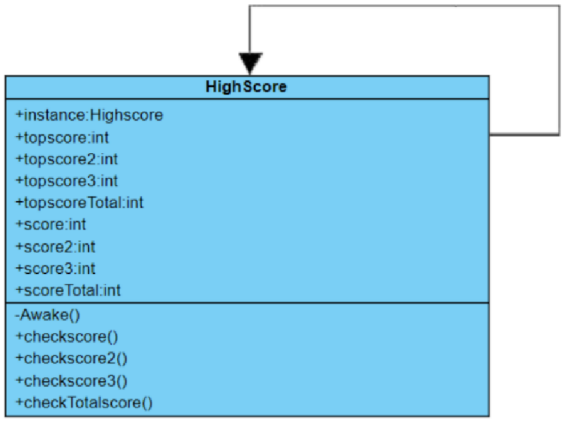
Tento systém počítání nejvyššího skóre je výhodný například v případě, že hráč nepřekoná celkové nejvyšší skóre, ale pouze nejvyšší skóre jednotlivé úrovně.



Obrázek 6 Diagram aktivit (Zdroj: Autor)

Diagram tříd

Před začátkem implementace hry byl vytvořen diagram tříd. Vzhledem k autorově minimálním zkušenostem s vývojem softwaru a zvolenou metodikou code and fix, musela být struktura aplikace několikrát předělávána. Pro demonstrování diagramu tříd, byl tedy vybrán jiný diagram, a to diagram návrhového vzoru jedináček z kapitoly Jedináček (4.2.1.1).



Obrázek 7 Diagram tříd (Zdroj: Autor)

4.2 Implementace hry

4.2.1 Xr Implementace

Pro usnadnění práce při implementování virtuální reality ve hře FUMUS byl použit XR Interaction toolkit a XR Plugin Management. XR Plugin Management se aktivuje v Project Settings. Následně se zvolí cílová platforma pro hru, v případě hry Fumus operační systém Windows. Po aktivaci tohoto pluginu je možné zobrazit náhled hry ve VR headsetu. Herní kamera však nereaguje na pohyb headsetu. Pro využití XR Interaction toolkitu je třeba jej nainstalovat pomocí nástroje Package Manager a vyhledání toolkitu dle jména "com.unity.xr.interaction.toolkit". Při instalování toolkitu byl zároveň autorem naimportován asset "Starter Assets", jež je součástí toolkitu, avšak při instalaci se sám nenaimportuje. Tento asset bude využit v pozdější části práce. Po instalaci XR Interaction toolkitu byl do scény autorem přidán objekt XR Origin, který obsahuje objekt Main Camera, s komponentou Tracked Pose Driver kopírující pohyb VR headsetu. Následně je třeba do objektu XR Origin přidat komponentu Input Action Manager, která určí, na které vstupy bude XR Origin reagovat. V případě hry FUMUS byla zvolena sada XR Default Input Action, jež je součástí "Starter Assetu". Dále je třeba do objektu XR Origin přidat komponentu Locomotion system, jež spravuje přístup komponent umožňující pohyb hráče k objektu XR Origin.

4.2.1.1 Pohyb

Ve hře FUMUS se hráč pohybuje v rámci malého omezeného prostoru. Z tohoto důvodu byl zvolen kontinuální pohyb hráče pomocí joysticku na ovladači. Alternativní možností pohybu byla teleportace, která by však vzhledem k velikosti hracího pole mohla být kontraproduktivní. Do objektu XR Origin byla přidána komponenta Continuous Move Provider, jenž umožňuje kontinuální pohyb hráče. Zároveň umožní vývojáři, mimo jiné, upravit rychlost pohybu a přidat referenci na vstup. V případě hry FUMUS byl zvolen levý ovladač pro pohyb hráče po okolí a pravý ovladač pro ovládání rotace kamery. Vývojář může volit ze dvou způsobů rotace kamer. Snap Turn okamžitě otočí kameru v autorem definovaném úhlu na základě směru vstupu. Continuous Turn otáčí kameru plynule v závislosti na délce vstupu. Oba způsoby je možné integrovat pomocí stejnojmenné komponenty, která se vloží do XR Origin objektu. V rámci této práce byl zvolen způsob Snap Turn. Následně je třeba přidat komponentu Character Controller, která umožní kolize hráče s objekty, jež mají vlastní collider. Dále je žádoucí, aby se výška hráče ve hře aktualizovala vzhledem k pozici headsetu v realitě. O to se stará komponenta Character Controller Driver, která po přidání umožní například skrčení hráče.

4.2.1.2 Interakce s předměty

K vytvoření ruky, která bude kopírovat pohyb ovladačů, byl použit volně dostupný 3D model ruky. Na tento model byla následně přidána komponenta Xr Controller, která slouží k interpretaci hodnot vstupů a pozice ovladače. Dále byla přidána komponenta Xr Direct Interactor, která umožní přímou interakci s předměty, kterých se ruka dotýká. K tomu je však nutné přidat na objekt ruky collider a na cílový předmět komponentu XR Grab Interactable. Ta na cílový objekt automaticky aplikuje komponentu Rigidbody, která se stará

o simulování fyziky objektu. Díky ní lze například zapnout gravitaci objektu. Zároveň je třeba na objekt aplikovat Box Collider (či jiný collider podporovaný komponentou Rigidbody). Collider definuje fyzický tvar objektu a je využíván k detekci objektů.

4.2.1 Psaní kódu

Kód pro hru Fumus byl psán v objektově orientovaném programovacím jazyce C#. Autor využil návrhové vzory jedináček, pozorovatel a stav, které byly představeny v teoretické části práce.

4.2.1.1 Jedináček

Návrhový vzor jedináček byl použit při vytváření třídy pro spravování nejvyššího skóre. Podmínka uvnitř metody Awake se stará o to, aby vždy existovala pouze jedna instance jedináčka. Jelikož instance jedináčka je nastavena jako statická, může k němu přistupovat libovolný skript a lze tak mezi sebou snadno propojovat objekty. Metoda DontDestroyOnLoad se stará o to, aby byl jedináček aktivní i při změně scény.

```
public static highscore Instance = null;
	private void Awake()
	{
		if (Instance == null)
		{
			Instance = this;
		}
		else if (Instance != this)
		{
			Destroy(gameObject);
		}
		DontDestroyOnLoad(gameObject);
	}
```

Ukládání nejvyššího skóre bylo dále vyřešeno pomocí třídy PlayerPrefs. Jedná se o třídu, která dokáže ukládat hodnoty do registrů operačního systému. Díky tomu je možné uchovat hodnoty i při vypnutí hry. Pokud uživatel dosáhne nového nejvyššího skóre, přepíše se původní hodnota v registru pomocí metody SetInt. Při zapnutí hry se hodnota nejvyššího skóre načte z registru pomocí metody GetInt.

4.2.1.2 Pozorovatel

Návrhový vzor pozorovatel byl využit při tvorbě skriptu pro zobrazení skóre v rámci uživatelského rozhraní. Jelikož se skóre mění s každým zásahem lahve, je žádoucí aby objekt starající se o detekování zásahu informoval objekt starající se o změnu skóre. V třídě, která se stará o detekování zásahu lahve, byla vytvořena instance delegáta OnHit.

```
public static event Action OnHit;
```

Pomocí metody Invoke lze vyvolat všechny registrované metody. Je třeba však zkontrolovat, zda existuje registrovaná metoda. Toho lze docílit přidáním otazníku za název instance delegáta. Předejde se tak vzniku chyby v případě, že žádná metoda registrovaná není.

```
OnHit?.Invoke();
```

Registrování metody probíhá zpravidla uvnitř metody OnEnable. V případě, že je v plánu registrovat více než jednu metodu, je nutné použit operátor „+“. Nedojde tak k přepsání původní registrované metody. Odstranění registrace konkrétní metody probíhá uvnitř metody OnDisable. Metoda givePoints přidává bod skóre a aktualizuje text na uživatelském rozhraní.

```
private void OnEnable()
	{
		states.OnHit += givePoints;
	}
	private void OnDisable()
	{
		states.OnHit -= givePoints;
	}
	public void givePoints()
	{
		score++;
		textScore.text = "Skóre: " + score.ToString();}
```

V případě, kdyby chtěl autor přidat nové funkcionality po zasažení lahve, stačí registrovat novou metodu k instanci delegáta OnHit.

4.2.1.3 Stav

Návrhový vzor stav je demonstrován na části kódu ze třetí herní úrovně. Jelikož cílem třetí úrovně je trefit každou lahev alespoň jednou, musí lahev po prvním zásahu přidat bod. Opětovným trefením lahve by však nemělo dojít k dalšímu navýšení skóre. Díky využití návrhového vzoru stav, lze vytvořit dva stavy reprezentující lahev, která ještě nebyla trefena a lahev, která již zasažena byla. Výhoda tohoto postupu je snadné rozšiřování kódu v případě, že by autor v budoucnu chtěl přidat další interakce se zásahy lahví. Návrhový vzor byl v tomto případě implementován pomocí typu výčtu Stav a příkazu switch. Díky použití návrhové vzoru bylo možné přidat rozdílné zvukové efekty a změnu materiálu po jednotlivých zásazích. Zároveň bylo možné zničit objekt až po druhém zásahu. První zásah lze iniciovat pouze kulkou, druhý zásah lze iniciovat pomocí kulky nebo dopadem lahve na jiný objekt.

```
void OnCollisionEnter(Collision col)
	{
		switch (stav)
		{
			case Stav.first:
				givePoint();
				break;
			case Stav.second:
				noPoint();
				break;
		}
}
public enum Stav
	{
		first,
		second,
	}
```

5 Výsledky a diskuse

5.1 Uživatelské testování

Výsledkem vývoje je funkční herní aplikace pro virtuální realitu, která se skládá ze třech herních úrovní a s definovaným koncem hry. Po dokončení první verze hry bylo provedeno uživatelské testování. Cílem testování bylo zjistit přehlednost uživatelského rozhraní a vysvětlení ovládání, vliv hry na vznik kybernevolnosti a odhalení chyb při hraní hry. Testování bylo provedeno na 10 uživatelích, kteří korespondovali s cílovou skupinou hry. Jednalo se o 10 mužů ve věku od 21 do 24 let. Po dokončení testování uživatel vyplnil dotazník a poté následovalo interview s tvůrcem hry.

5.1.1 Testování

Před začátkem testování byl uživatel požádán o popisování myšlenek nahlas (think aloud protokol). Uživatelské testování bylo tvořeno 2 scénáři. V rámci prvního scénáře měl uživatel za úkol z hlavního menu spustit standardní režim, dokončit tutoriál a alespoň první herní úroveň. V druhém scénáři měl uživatel za úkol z hlavního menu spustit konkrétní level „Shell“ a následovně se opět vrátit do hlavního menu. Cílem tohoto scénáře bylo poukázat na případné nedostatky uživatelského rozhraní. Při testování byl uživatel pozorován a jeho poznámky byly zapisovány.

5.1.2 Dotazník

Otázka 1
První otázka měla za cíl zjistit, jak často respondent hraje videohry. Polovina respondentů jsou aktivní hráči, kteří hrají 5x až 7x týdně. Dva respondenti uvedli, že videohry hrají minimálně jednou a maximálně čtyřikrát týdně. Tři respondenti jsou příležitostní hráči, kteří videohry hrají alespoň několikrát v měsíci. Žádný respondent neuvedl, že se hraní videoher nevěnuje vůbec.

Otázka 2
Druhá otázka se zabývala zkušeností respondenta s virtuální realitou. Dva respondenti uvedli, že jsou pravidelní uživatelé virtuální reality. Šest respondentů virtuální realitu alespoň jednou vyzkoušelo a dva ji v den testování použitelnosti vyzkoušeli poprvé. Účast uživatelů, kteří nemají s virtuální realitou zkušenost, byla klíčová zejména proto, že při vývoji hry byl zvolen standardní způsob ovládání hry pro virtuální realitu. S tímto způsobem ovládání může mít zkušený uživatel virtuální reality zkušenost a výsledky zkoumání přehlednosti ovládání by tak mohly být zkresleny.

Otázka 3
Třetí otázka se týkala vzniku kybernevolnosti při hraní hry FUMUS. Sedm respondentů uvedlo, že při hraní žádnou formu kybernevolnosti nepocítili. Tři respondenti uvedli, že se jim při hraní hry udělalo špatně. Všichni tři uvedli, že se jednalo o lehkou nevolnost.

Otázka 4
Čtvrtá otázka se týkala přehlednosti uživatelského rozhraní. Devět respondentů uvedlo, že uživatelské rozhraní bylo přehledné a jasné. Jeden uživatel uvedl, že uživatelské rozhraní bylo spíše nepřehledné.

Otázka 5
Pátá otázka měla za cíl zjistit, jak dobře bylo vysvětlené ovládání hry. Všech deset respondentů uvedlo, že ovládání hry jim bylo jasné. Jelikož součástí testování byli i uživatelé, kteří virtuální realitu zkoušeli poprvé, a s ovládáním podobných her před tím neměli zkušenosti, můžeme předpokládat, že k pochopení ovládání hry byl klíčový tutoriál, ve kterém je ovládání podrobně popsané.

Otázka 6
Poslední otázka se týkala pocitu z hraní hry FUMUS. Devět respondentů uvedlo, že z hraní mělo pozitivní pocit. Jeden uživatel uvedl, že měl neutrální pocit.

5.1.3 Interview

Cílem interview bylo zjistit uživatelův pocit ze hraní hry a konkrétní problémy při hraní hry. Interview se skládalo ze čtyř otevřených otázek.

První otázka měla za cíl zjistit, co se uživateli na hře líbilo. Z odpovědí vyplývá, že se uživatelům nejvíce líbil design levelů. Dále se jim líbila fyzika střílení, možnost měnit zbraně, rozdílné zpracování zbraní v jednotlivých levelech, ovládání, bonusy a voiceover.

Uživatel	Co se Vám líbilo na hře?
1	Design levelů
2	Voiceover
3	Zpracování 3. levelu
4	Grafické zpracování, fyzika zbraně ve třetím levelu
5	Design levelů, bonusy
6	Unikátní zpracování zbraní dle levelu
7	Fyzika kulky, možnost měnit zbraně
8	Design levelů
9	Dobře vysvětlené ovládání, design levelů
10	Design levelů

Tabulka 2 Odpovědi na otázku: Co se vám líbilo na hře?

Druhá otázka měla za cíl zjistit, které konkrétní věci se uživateli na hře nelíbily. Uživatelům se nelíbilo, že hra byla příliš krátká, mohli se pohybovat pouze v omezeném prostoru, dále nepřesnost zbraně a animace kulky.

Uživatel	Co se Vám nelíbilo na hře?
1	Hra byla příliš krátká
2	Nic
3	Možnost pohybu v omezeném prostoru
4	Nic
5	Nic
6	Přesnost zbraně
7	Nic
8	Animace kulky
9	Nic
10	Hra byla příliš krátká

Tabulka 3 Odpovědi na otázku: Co se vám nelíbilo na hře?

Třetí otázka měla za cíl zjistit, co by uživatelé na hře změnili nebo co by přidali. Nejvíce se odpovědi týkaly přidávání obsahu. Jedná se o přidání více levelů, zbraní, bonusů, či možnost pohybovat se ve větším prostoru. Zmíněné změny se týkaly přesnosti míření.

Uživatel	Co byste na hře změnili?
1	Více levelů
2	Více unikátních bonusů
3	Možnost pohybovat se ve větším prostoru
4	Více zbraní
5	Nic
6	Větší přesnost zbraně, možnost mířit
7	Nic
8	Větší přesnost zbraně
9	Jiné zpracování menu ve hře
10	Více levelů

Tabulka 4 Odpovědi na otázku: Co byste změnili na hře?

Poslední otázka měla za cíl zjistit, zda se při hraní objevila nějaká chyba. Odpovědi na danou otázku pomohly autorovi identifikovat vzniklé chyby, které nemusely být objeveny při testování samotným autorem.

Uživatel	Jakou chybu jste během hraní objevili?
1	Nic
2	Nabíjení zbraně se opakovaně spouští při zmáčknutí tlačítka
3	Nic
4	Zbraň po upuštění na zem v prvním levelu zmizela
5	Nabíjení zbraně se opakovaně spouští při zmáčknutí tlačítka
6	Nic
7	Nabíjení zbraně se opakovaně spouští při zmáčknutí tlačítka
8	Nic
9	Nic
10	Poslední level napíše vítězství i v případě, že hráč nesestřelí jednu lahev

Tabulka 5 Odpovědi na otázku: Jakou chybu jste při hraní objevili?

5.1.4 Shrnutí výsledků testování

Na základě uživatelského testování bylo zjištěno, že hra je pro nového uživatele přehledná a snadno se v ní dokáže orientovat. Zároveň však byly objeveny tři chyby, které ovlivňují herní zážitek a byly opraveny v následující podkapitole.

5.1.5 Upravení hry dle poznatků z interview

Na základě poznatků z uživatelského testování došlo k úpravě hry. Úprava se týkala chyb, které byly popsány uživateli v interview.

5.1.5.1 Chyba 1

Pokud hráči došly ve zbrani náboje a stiskl tlačítko pro střelbu, zbraň se začala nabíjet. Pokud však uživatel stiskl tlačítko pro střelbu před dokončením nabíjení, začala se zbraň nabíjet znovu. Tento problém byl vyřešen přidáním podmínky a proměnné typu bool „reloading“.

```
if (currentAmmo == 0)
        {
                if (reloading == false)
                {
                        reloading = true;
                        horejsek.GetComponent<Animator>().Play("nabijeni");
                        zasobnik.Play();
                        StartCoroutine(reload());
                }
        }



IEnumerator reload()
        {

                yield return new WaitForSeconds(2);
                currentAmmo=maxAmmo;
                reloading = false;
        }
```

Díky úpravě kódu není možné spustit animaci nabíjení, jestliže nabíjení již probíhá.

5.1.5.2 Chyba 2

Pokud hráč upustí zbraň v prvním levelu na zem, zbraň zmizí. Tato chyba vznikla, jelikož autor nechal ve scéně přebytečný objekt reprezentující podlahu. Pod tímto objektem se nacházel objekt skutečné podlahy a zbraň byla schovaná mezi těmito dvěma objekty. K odstranění chyby tak došlo při odstranění přebytečného objektu.

5.1.5.3 Chyba 3

V posledním levelu hry má hráč za úkol najít a sestřelit 9 lahví. K úspěšnému dokončení levelu však stačilo sestřelit pouze 8 lahví. Chyba vznikla při úpravě kódu a špatném napsání podmínky. K odstranění chyby došlo po upravení hodnoty v podmínce.

5.2 Diskuze a budoucí vývoj hry

Na základě poznatků z uživatelského testování došlo ke zjištění, že řada hráčů by ocenila více herního obsahu. Hráčům trvalo v průměru 5 až 15 minut k dohrání celé hry. Obtížnost jednotlivých levelů však byla záměrně nastavena tak, aby bylo možné hru během testování bez potíží odehrát celou. V případě vydání plnohodnotné hry by tak měla být obtížnost hry vyšší, aby její hraní trvalo delší dobu. Dále by měly být přidány nové úrovně, nové zbraně a bonusy, aby se předešlo brzké ztrátě zájmu hráče o hru. Vznik kybernevolnosti se objevil u 30 % uživatelů, minimalizování rizik vzniku kybernevolnosti bylo docíleno pomocí dobré optimalizace hry, nutnosti minimálního virtuálního pohybu hráče a taktéž díky délce herních úrovní, jelikož jednotlivé herní úrovně byly časově omezené jednou minutou. Díky proběhlému uživatelskému testování byly odhaleny 3 chyby, které by nemusely být objeveny samotným autorem. Následně byly chyby opraveny. V současném stavu by hra FUMUS mohla sloužit jako demonstrace virtuální reality pro nové hráče. V době psaní této bakalářské práce neměl autor plány k rozšiřování hry a budoucímu vývoji, avšak poznatky nabyté z tvorby hry FUMUS může využít v budoucnu při tvorby komplexnější videohry.

6 Závěr

Tato bakalářská práce se zabývala tvorbou 3D herní aplikace pro virtuální realitu. V teoretické části se autor věnoval popisu obecných postupů pro tvorbu aplikace. Představeny byly metodiky vývoje softwaru, jednotlivé fáze vývoje softwaru, často využívané diagramy UML, objektově orientované programování, návrhové vzory a využití herních enginů pro tvorbu aplikace.

Část práce je také věnována tvorbě videohry a popisu žánrů a technik využívaných při tvorbě návrhu hry v rámci herního designu. Byly zde popsány jednotlivé aspekty, které mají vliv na hráčovu motivaci vracet se ke hře a prožívání pocitu zábavy během hraní hry.

Následně se teoretická část práce zabývá virtuální realitou. Dále je zde rozvinuta problematika vzniku kybernevolnosti a minimalizování rizik jejího vzniku. Zároveň byla představena metoda implementování virtuální reality v herním enginu Unity. V neposlední řadě byly představeny metody distribuování hotové hry, testování aplikace včetně uživatelského testování a herní assety.

Praktická část práce se v první řadě zabývala tvorbou návrhu hry. Při tvorbě návrhu hry byly využity poznatky z teoretické části z podkapitoly Herní design. Návrh hry byl vytvořen tak, aby obsahoval prvky, které budou motivovat hráče vracet se ke hře. Následně byl vytvořen vzhled jednotlivých herních úrovní pomocí kombinování volně dostupných 3D modelů přímo v editoru herního enginu Unity. Zároveň byly vybrány vhodné zvukové efekty a hudba, která podtrhuje atmosféru jednotlivých herních úrovní. Dále byla implementována virtuální realita pomocí XR Interaction toolkitu. Následovalo demonstrování použití návrhových vzorů na části kódu. V práci byly využity návrhové vzory jedináček, pozorovatel a stav.

Na závěr bylo provedeno uživatelské testování s 10 uživateli. Pro tyto uživatele byly vytvořeny dva scénáře, které měli během testování následovat. Cílem testování bylo zjistit přehlednost uživatelského rozhraní a vysvětlení ovládání, vliv hry na vznik kybernevolnosti a odhalení chyb při hraní hry. Po dokončení testování uživatelé vyplnili dotazník a následovalo interview s autorem hry.

První dvě otázky z dotazníku se zabývaly zkušenostmi respondentů s hraním her a virtuální reality. Z dat dotazníku vyplynulo, že respondenti jsou aktivní hráči videoher a 20 % respondentů nemělo zkušenost s virtuální realitou. Účast uživatelů, kteří nemají s virtuální realitou zkušenost, byla klíčová zejména proto, že při vývoji hry byl zvolen standardní způsob ovládání hry pro virtuální realitu. S tímto způsobem ovládání by mohl mít zkušený uživatel virtuální reality zkušenost a výsledky zkoumání přehlednosti ovládání by tak mohly být zkresleny. Následovaly otázky na vznik kybernevolnosti, přehlednost uživatelského rozhraní a ovládání hry. Z dat dotazníku vyplynulo, že lehká míra kybernevolnosti vznikla u 30 % uživatelů. Přehlednost uživatelského rozhraní potvrdilo 90 % respondentů. Přehlednost vysvětlení ovládání hry bylo potvrzeno všemi uživateli.

Následující interview mělo za cíl zjistit konkrétní prvky hry, které se uživatelům líbily, které se naopak nelíbily a co by na hře změnili. Dalším cílem interview bylo odhalení chyb, které se objevily při hraní hry. Z odpovědí interview se nejvíce uživatelům líbil design jednotlivých herních úrovní, nejčastěji zmiňované negativum hry byla její délka. Respondenti uvedli, že by do hry přidali více obsahu. Zároveň byly odhaleny tři chyby, které byly v další části práce opraveny.

Cílem práce bylo vytvořit 3D akční herní aplikaci pro virtuální realitu pomocí herního enginu Unity. Dílčími cíli bylo představení správných postupů při vytváření aplikace pro virtuální realitu v tomto enginu, představení správných postupů při tvorbě návrhu videohry, aby maximalizovala zájem hráče vracet se k hraní dané videohry, minimalizování vzniku kybernevolnosti a minimalizování počtu vzniklých chyb. Všechny tyto cíle byly v rámci teoretické a praktické části naplněny v plném rozsahu. I přes autorovy minimální zkušenosti s tvorbou herní aplikace před začátkem vypracování této bakalářské práce byla úspěšně vytvořena 3D akční videohra pro virtuální realitu na platformu Windows.

7 Seznam použitých zdrojů

AMPATZOGLOU, Apostolos a Alexander CHATZIGEORGIOU, 2007. *Evaluation of object-oriented design patterns in game development* [online]. Dostupné z: doi:10.1016/j.infsof.2006.07.003

ANDRADE, A., 2015. *Game engines: a survey* [online]. In: . s. 6 [cit. 2022-11-22]. Dostupné z: doi:10.4108/eai.5-11-2015.150615

BRYANT, Aidan, 2020. Everything you need to know about user testing your game. In: *Usertesting* [online]. [cit. 2022-11-22]. Dostupné z: https://www.usertesting.com/blog/user-testing-games

COWAN, Brent a Bill KAPRALOS, 2014. *A Survey of Frameworks and Game Engines for Serious Game Development* [online]. Dostupné z: doi:10.1109/ICALT.2014.194

ČÁPKA, David, 2022a. *Lekce 1 - Úvod do UML* [online]. In: . [cit. 2022-05-29]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

ČÁPKA, David, 2022b. Lekce 5 - UML - Class diagram. In: *Itnetwork* [online]. [cit. 2022-11-22]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

ČÁPKA, David, 2022c. Lekce 2 - UML - Use Case Diagram. In: *Itnetwork* [online]. [cit. 2022-11-22]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

ČÁPKA, David, 2022d. Lekce 14 - Template Method (šablonová metoda). In: *Itnetwork* [online]. [cit. 2022-11-22]. Dostupné z: https://www.itnetwork.cz/navrh/navrhove-vzory/gof/template-method-navrhovy-vzor

DESYATNIKOV, Ruslan, 2022. SDLC (Software Development Life Cycle) Phases, Process, Models. In: *Software Testing Help* [online]. [cit. 2022-11-22]. Dostupné z: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/#1_Requirement_Gathering_and_Analysis

DOHERTY, Shawn, Joseph KEEBLER, Shayn DAVIDSON, Evan PALMER a Christina

FREDERICK, 2018. *Recategorization of Video Game Genres*. Dostupné z: doi:10.1177/1541931218621473

DOOLEY, John, 2017. *Software Development, Design and Coding*. APress. ISBN 148423152X.

EPIC GAMES, 2022. UNREAL ENGINE Licensing options. In: *Unrealengine* [online]. [cit. 2022-11-22]. Dostupné z: https://www.unrealengine.com/en-US/license

ESPOSITO, Nicolas, 2005. *A Short and Simple Definition of What a Videogame Is*. Compiègne Cedex.

GAME-ACE, 2021. Types of game testing. In: *Game-ace* [online]. [cit. 2022-11-22]. Dostupné z: https://game-ace.com/blog/types-of-game-testing/
GILLIS, Alexander S., 2021. Object-oriented programming (OOP). In: *Techtarget* [online]. [cit. 2022-11-22]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

GREENWALD, Will, 2022. The Best VR Headsets for 2022. In: *Pcmag* [online]. [cit. 2022-11-22]. Dostupné z: https://www.pcmag.com/picks/the-best-vr-headsets

HANEY, Michael, 2011. Design Patterns in Game Programming. In: *Gamedeveloper* [online]. [cit. 2022-11-22]. Dostupné z: https://www.gamedeveloper.com/programming/design-patterns-in-game-programming

CHANDRA, Ananth, Fatima JAMIY a Hassan REZA, 2022. *A Systematic Survey on Cybersickness in Virtual Environments*. Dostupné z: doi:10.3390/computers11040051

IBM, 2021. Visibility in domain modeling class diagrams. In: *Ibm* [online]. [cit. 2022-11-22]. Dostupné z: https://www.ibm.com/docs/en/radfws/9.6.1?topic=classifiers-visibility

INTEL CORPORATION, 2022. What Is Refresh Rate and Why Is It Important?. In: *Intel* [online]. [cit. 2022-11-22]. Dostupné z: https://www.intel.com/content/www/us/en/gaming/resources/highest-refresh-rate-gaming.html

IRISVR, 2021. The Importance of Frame Rates. In: *Irisvr* [online]. [cit. 2022-11-22]. Dostupné z: https://help.irisvr.com/hc/en-us/articles/215884547-The-Importance-of-Frame-Rates

JEVTIC, GORAN, 2019. What is SDLC? Phases of Software Development, Models, & Best Practices. In: *PhoenixNAP* [online]. [cit. 2022-11-22]. Dostupné z: https://phoenixnap.com/blog/software-development-life-cycle

KOĎOUSKOVÁ, Barbora, 2022. CO JE AGILNÍ VÝVOJ APLIKACÍ A KDY HO VYUŽÍVAT. In: *Rascasone* [online]. [cit. 2022-11-22]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-agilni-vyvoj

KODYTEK, Samuel, 2022. Lekce 1 - Úvod do metodologie vývoje softwaru. In: *Itnetwork* [online]. [cit. 2022-11-22]. Dostupné z: https://www.itnetwork.cz/navrh/metodiky/uvod-do-metodologie-vyvoje-softwaru

KONVOY, 2021. PC Distribution Platforms. In: *Konvoy* [online]. [cit. 2022-11-22]. Dostupné z: https://www.konvoy.vc/newsletter/deep-dive-pc-distribution-platforms

LINOWES, Jonathan, 2020. *Unity 2020 Virtual Reality Projects*. Packt Publishing Limited. ISBN 1839217332.

LONG, Sebastian, 2021. Getting More From Usability Testing Your Game. In: *Playerresearch* [online]. [cit. 2022-11-22]. Dostupné z: Getting More From Usability Testing Your Game

MARINHO, Adrianoda, Uwe TERTON a Christian JONES, 2021. *Cybersickness and postural stability of first time VR users playing VR videogames*. Dostupné z: doi:https://doi.org/10.1016/j.apergo.2022.103698

MICROSOFT 365 TEAM, 2019. Jednoduchý návod k UML diagramům a modelování databází. In: *Microsoft* [online]. [cit. 2022-11-22]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

MICHAUD, Laurent, 2012. *World Video Game Market*. ISBN 978-2-84822-231-8.

MŮČKA, JAN, 2020. Vývoj software: jaké jsou základní bezpečnostní principy?. In: *Master* [online]. [cit. 2022-11-22]. Dostupné z: https://www.master.cz/blog/vyvoj-software-zakladni-bezpecnostni-principy/

NISHADHA, 2022. UML Diagram Types Guide: Learn About All Types of UML Diagrams with Examples. In: *Creately* [online]. [cit. 2022-11-22]. Dostupné z: https://creately.com/blog/diagrams/uml-diagram-types-examples/

PAINE, JAMES, 2018. 10 Real Use Cases for Augmented Reality. In: *Inc* [online]. [cit. 2022-11-22]. Dostupné z: https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html

PANJA, Vennela a Jill BERGE, 2021. *Minecraft Education Edition's Ability to Create an Effective and Engaging Learning Experience*. Dostupné z: doi:https://doi.org/10.47611/jsrhs.v10i2.1697

PECINOVSKÝ, Rudolf, 2007. *Návrhové vzory: [33 vzorových postupů pro objektové programování]*. Brno: Computer Press. ISBN 97880-251-1582-4.

PITMAN, Neil, 2005. *UML 2.0 in a nutshell*. Sebastopol: O'Reilly. ISBN 05-960-0795-7.

PLEDGE TIMES, 2022. Are physical video games on their way to disappearing? A study reflects the evolution of the market. In: *Pledgetimes* [online]. [cit. 2022-11-22]. Dostupné z: https://pledgetimes.com/are-physical-video-games-on-their-way-to-disappearing-a-study-reflects-the-evolution-of-the-market/

ROBINSON, Nick, 2012. *Videogames, Persuasion and the War on Terror: Escaping or Embedding the Military—Entertainment Complex?*. Dostupné z: doi:10.1111/j.1467-9248.2011.00923.x

SARCAR, Vaskaran, 2018. *Design Patterns in C#*. APRESS L.P. ISBN 978-1-4842-3640-6.

SCHELL, Jesse, 2019. *The Art of Game Design*. 3. Taylor & Francis Ltd. ISBN 9781138632059.

SIRANI, Jordan, 2022. The 10 Best-Selling Video Games of All Time. In: *Ign* [online]. [cit. 2022-11-22]. Dostupné z: https://www.ign.com/articles/best-selling-video-games-of-all-time-grand-theft-auto-minecraft-tetris

SOUKUP, Jan, 2020. Jak udělat uživatelské testování svépomocí. In: *Honzasoukup* [online]. [cit. 2022-11-22]. Dostupné z: https://www.honzasoukup.cz/blog/jak-udelat-uzivatelske-testovani-svepomoci

SPARX SYSTEMS, 2022. UML 2 Tutorial - Component Diagram. In: *Sparxsystems* [online]. [cit. 2022-11-22]. Dostupné z: https://sparxsystems.com/resources/tutorials/uml2/component-diagram.html

STANTON, Rich a Will FREEMAN, 2016. The 25 hardest video games of all time. In: *Theguardian* [online]. [cit. 2022-11-22]. Dostupné z: https://www.theguardian.com/technology/2016/mar/18/the-25-hardest-video-games-of-all-time

UNITY TECHNOLOGIES, 2022. Quick guide to the Unity Asset Store. In: *Unity3d* [online]. [cit. 2022-11-22]. Dostupné z: https://unity3d.com/quick-guide-to-unity-asset-store

USABILITY BOK, 2012. Think Aloud Testing. In: *Usabilitybok* [online]. [cit. 2022-11-22]. Dostupné z: https://www.usabilitybok.org/think-aloud-testing

VALVE CORPORATION, 2022. Distribuční program systému Steamworks. In: *Steamgames* [online]. [cit. 2022-11-22]. Dostupné z: https://partner.steamgames.com/steamdirect

VIONIX, 2022. What is an asset in game development?. In: *Vionixstudio* [online]. [cit. 2022-11-22]. Dostupné z: https://vionixstudio.com/2021/12/04/what-is-an-asset-in-game-development/

7 Seznam obrázků, tabulek, grafů a zkratek

7.1 Seznam obrázků

7.2 Seznam tabulek

7.3 Seznam grafů

8 Přílohy

8.1 Hra Fumus

Funkční verze hry Fumus je dostupná na následujícím odkaze:
https://drive.google.com/drive/folders/124YuEzGrtxwLct1n0Og56Ez4ehXnKatN?usp=sharing