ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra Počítačů

Bakalářská práce

Návrh a implementace RPG hry žánru „fantasy“ pro více hráčů prostřednictvím moderního herního engine

Ondřej Pejša
pejsaond@fel.cvut.cz

Květen 2021
Vedoucí práce: RNDr. Ladislav Seredi

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pejša** Jméno: **Ondřej** Osobní číslo: **483836**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh a implementace RPG hry žánru „fantasy“ pro více hráčů prostřednictvím moderního herního engine

Název bakalářské práce anglicky:

Design and implementation of a multiplayer fantasy RPG using a modern game engine

Pokyny pro vypracování:

Porovnejte volně dostupné herní enginy z hlediska jejich použitelnosti k vytvoření 2D RPG hry v žánru Fantasy. Navrhněte vlastní RPG ve zvoleném žánru (vytvořte vhodnou dějovou linii, lokace, interakce, předměty, nepřátele a další postavy) Dokumentujte mapu lokací, možné stavy hry, a pokud to bude vhodné, vytvořte i vlastní mediální prvky (grafika, zvuky). Při implementaci mějte na zřetel maximální možnou konfigurovatelnost hry s cílem umožnit i neprogramátorovi její rozšíření či drobnější modifikace: zvažte přidání editoru předmětů, zbraní, lokací, apod. Hru plánujte a implementujte jako typ on-line multiplayer, umožňující hrát hru v síťovém prostředí pro více hráčů, optimálním využitím technických možností game engine. V reálném prostředí otestuje vlastní hru i uživatelské editory. Na základě uživatelských testů vylaďte hratelnost. Na závěr diskutujte přínos hry a jeho inovace oproti stávajícím RPG.

Seznam doporučené literatury:

1. Ernest Adams. Fundamentals of Role-Playing Game Design [online]. ©2014 New Riders [15.12.2020]. Available at: https://books.google.cz/books
2. Chris Bradfield . Godot Engine Game Development Projects [online]. ©2018 Packt Publishing [15.12.2020]. Available at: https://ebookcentral.proquest.com/lib/cvut/reader.action?docID=5446052
3. Unity vs Godot: Game Engine Show Down [online]. libsdl.org [15.12.2020]. Available at: https://www.gamedesigning.org/engines/unity-vs-godot/
4. Godot Docs – 3.2 branch [online]. ©2020, Juan Linietsky, Ariel Manzur and the Godot community [15.12.2020]. Available at: https://docs.godotengine.org/en/stable/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ladislav Serédi, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

RNDr. Ladislav Serédi podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování / Prohlášení

Děkuji panu Ladislavu Seredi za přijmutí mé bakalářské práce, průběžnou kontrolu, vstřícnost a neocenitelné informace které mi velmi pomohli ve vypracování. I v této nelehké distanční době jsme společně zvládli dotáhnout toto netypické téma do zdárného konce. Ještě jednou děkuji a těším se na další spolupráci.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

.....

Abstrakt / Abstract

Tento dokument je zaměřen na vývoj 2D RPG hry v žánru fantasy. Aplikace je navržena podle standardních prvků pro typ RPG her jako jsou různé předměty, lokace, postavy a podobně. Implementace probíhá v moderním herním enginu kde součástí vývoje je analýza repetitivní činnosti a s ní vybudování rozhraní pro možnost přispívání prvků do hry bez znalosti programování. Jejím obsahem je úvod do problematiky, analýza, volba vhodného nástroje, podrobný popis implementace samotné aplikace a následný souhrn činnosti.

Klíčová slova: Hra na hrdiny, Godot, Příběh, Multiplayer

This document is focused on developing 2D RPG fantasy video game. Application is designed with standard elements known in the common RPG titles, these elements could be items, locations, characters etc. Implementation of the game takes place in modern game engine. Developing include analysis of repetitive activity and creation of interface for contributing game objects into the game without knowledge of programming. The content consists of introduction, analysis, choice of the appropriate tool, detailed description of the implementation and conclusion.

Keywords: Role-playing game, Godot, Story, Multiplayer

Title translation: Design and implementation of a multiplayer fantasy RPG using a modern game engine

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

1.1 Myšlenka tématu

Role-playing game nebo také v překladu hra na hrdiny je žánr který staví na tvorbě fiktivního světa s herními charaktery ovládanými samotnými hráči, charaktery které jsou ovládány systémem či pravidly a poté Game-Masterem jako role která dohlíží nad děním celé hry. Účastníci si vybírají na začátku hry jednotlivé role, těch může být několik zde záleží na konkrétní hře, poté se postupuje podle příběhu, který je měněn na základě činností jednotlivých postav nebo je příběh přímo vytvářen. Pravidla a jakýsi zformulovaný svět může být velmi složitý a proto je potřeba již zmiňovaný Game-Master, ten do dění hry přímo nezasahuje, ale kontroluje pravidla, řeší spory, čte dialogy a stará se o celkový chod hry. [1]

Tento populární žánr se v dnešní době používá v několika podobách a tím je například klasické stolní čili Pen and Paper game, LARP nebo-li Live Action Role Play v překladu role na živo kde jednotlivé charaktery tvoří sami hráči, a ve videohrách zde se původní podstata RPG může velmi lišit a záleží na konkrétní hře. [1]

1.2 Představení historie

První náznaky jakési hry s předstíranými charaktery a improvizovaným dialogem sahají do 16. století. Okolo 19. a 20. století se začaly objevovat první stolní hry s prvky RPG ale také tzv. parlour game v překladu salónní hry jako je Jury Box který je považován za jakého si předchůdce her na hrdiny. Stolní RPG hry získaly velkou popularitu okolo roku 1950 až 1970 díky několika známým příběhům jako je například Pán Prstenů od R.R.Tolkiena. V roce 1970 válečné fantasy hry známe jako Wargames daly vznik dnešnímu modernějšímu typu her na hrdiny a tím je například velmi známá hra Dungeons and Dragons která byla publikována roku 1974 společností Tactical Studies Rules a bylo několik vydání z nichž poslední je z roku 2014 jako 5 edice. [2][3]

Mnoho počítačových her na hrdiny mají kořeny právě z klasických stolních her, zde se používá obdobná terminologie, nastavení a herní mechanismy. Velmi důležitými předky se považují tituly jako Dungeons & Dragons, WarGames ale také strategické stolní hry jako jsou klasické šachy nebo dobrodružné hry například Colossal Cave Adventure což je textově založená hra pro mainframe. Prvotní hry které byly schopny běžet na mainframu začínají okolo roku 1970, tyto hry byly kompletně textové, dobrými příklady jsou například Dungeon nebo pedit5. Je dobré podotknout že se zde NPC charaktery a itemy reprezentovali pomocí ASCII znaků a to dalo vzniknout další podmnožině video her nazývající se “roguelikes”. [4]

Pokud bychom měli říct jaké RPG se považuje jako prvotní pro mikropočítač byl by to zřejmě již tolikrát zmíněný Dungeon & Dragons napsaný Peterem Trefonasem. Tato na svou dobu nadčasová hra byla publikována pro osobní model počítače TRS-80 řady 1. Neměli bychom také zapomenout na jména jako Temple of Apshai nebo Odyssey:

The Compleat Apventure jež byla vydána pro platformu Apple II. [5] Po roce 1990 se začínají objevovat již typy RPG her založené na sprite-based formě, což znamená, že již nejsou pouze textové, ale takové které známe dnes - animované. V této době byly známými vydavateli například Interplay Entertainment nebo Blizzard North. Za zmínku stojí v tuto dobu hry Baldur's Gate, Icewind Dale nebo Diablo série. Začíná doba kde se do pozadí dostávají 3D enginy a po roce 2000 se stávají dominantou her. [6]

Pro kompletnost našich prvopočátků se ještě zmíníme o pár prvotních a známých titulech pro herní konzole. Do našeho seznamu bychom měli zařadit Dragonstomper na model Atari 2600, Dragon Quest pro platformu NES a Final Fantasy pro konzoli NES. Final Fantasy představil boční pohled bojů, mimochodem tato myšlenka se poté stala normou pro většinu titulů. Okolo roku 1988 RPG titul Dragon Warrior přišel s myšlenkou představení systému tzv. charakter progression system který umožňoval změnit třídy postavy během hry a to nebyla jediná inovace, jsou zde i další jako denní cykly, různé itemy, a úkoly přístupné jen v určitou herní dobu. [7]

1.3 Stolní hry

Ve většině stolních her tohoto typu se přebírá myšlenka z první publikované RPG hry Dungeons & Dragons kde účastníci jsou v menší skupině a vyberou si nebo vytvoří svůj charakter se kterým poté budou hrát v průběhu hry - určí mu například jeho historii, ale také různé numerické hodnoty které jsou poté použity v různých situacích. Game-master začíná úvodem a popsáním charakterů postav. Hráči poté popisují různé akce svých postav a GM odpovídá na tyto akce a říká jejich výsledek. Například pokud hráč má postavu v nějaké lokaci, GM tuto lokaci popíše. Ve stejné víře funguje například i bojový systém - ten je založený na numerických statistikách postavy a čím vyšší je například síla charakteru tím větší je šance na úspěch pro vykonání akce. [8]

1.4 Reálné / kostýmové hry

Kostýmová hra naživo přezdívaná LARP vznikla právě ze stolních RPG her. Zde hráči jsou fyzicky zainteresovaní do hry a vykonávají akce či plní cíle s různými okolnostmi na reálný svět což je veliký rozdíl od stolních her kde jsou charaktery popisovány pouze verbálně. Jejich přímá interakce může být manipulována skupinou lidí, kteří tento svět řídí a aranžují ho podle pravidel. První LARP byl zahájen okolo roku 1970 a dále se rozšiřoval do různých stylů her. Tyto hry mohou být od několika hodin až po několik dnů s jednotkami až stovkami zainteresovanými lidmi s tím že zde nejsou žádní diváci. [9]

Obrázek 1.1. Kostýmová hra, LARP

1.5 Video hry

Počítačové RPG hry obsahují často bohatý a komplexní příběh s otevřeným světem, zde záleží na konkrétní hře, protože mechanismy jednotlivých her se v tomto často liší. Obvykle hráč ovládá jednu postavu u které si vybírá rasu (charakter) a přiděluje jí atributy. Během průběhu hraní se hlavní postava střetává s NPC (non-Player Character) kde chování těchto postav je v roli počítače. Hráč svou postavu vylepšuje za různě provedené činnosti, sbírá a obchoduje s předměty a zlepšuje svoje zkušenosti tzv. levlovacím systémem. Počítačová hra v tomto ohledu je rozdělena na single-player což je klasický režim kde hráč hraje sám bez zásahu dalších hráčů a poté multi-player (MMORPG) kde je v daném světě několik hráčů kteří plní různé úkoly a mohou hrát spolu. [10]

Kapitola 2
Představení konceptu

2.1 Vlastnosti a popis hry

Vývoj hry se zaměří na fantasy tématiku s konkrétním příběhem ve 2D perspektivě. Ve hře je možnost pohybu v jakémsi otevřeném světě a přecházet mezi lokacemi které na sebe navazují přes různé přístupové body. Avšak lokace mají vždy svojí určitou velikost, a hráč či NPC nemohou za tyto vymezené hranice. Herní svět je viditelný z ptačí perspektivy to znamená, že kamera je umístěna nad všemi objekty. Funkcionalita hry je v zásadě standardní a převzatá od ostatních titulů her, to znamená že hráč se může pohybovat po lokaci s tím že některé objekty jsou průchodné, neprůchodné nebo postava bude pod těmito texturami. Po lokaci budou umístěny jak normální NPC od kterých je možno získat úkol, tak agresivní postavy které bude třeba nějakým způsobem přemoci. Itemy (tj. předměty) ve hře je možné najít na zemi, získat ze zabitých nepřátel, z úkolů nebo je koupit či prodat na tržišti. Hráč získané itemy dostává do svého inventáře. Tyto itemy může použít a jejich použití záleží na konkrétním účelu tohoto předmětu, to znamená že pokud je nalezený předmět například zbraň postava si tento předmět může nasadit, naopak pokud je item například nějaký neznámý lektvar tak ho postava může pozřít a získat tím určitý bonus ve hře.

Hra obsahuje systém úrovní, to znamená že hráč získává zkušenosti, díky kterým dosahuje jakési úrovně vyspělosti. Po dosažení úrovně má hráč možnost zlepšovat svojí postavu respektive její vlastnosti a postava se tím stává v nějakých ohledech (například v boji nebo pohybu postavy) silnější a lepší. Hráč prochází rozvětvený herní příběh během kterého hráč následuje instrukce, které mu byly zadány. Úkoly jsou doplněny dialogy postav které v textové formě sdělují své potřeby, příběh a ostatní informace. Po přijetí úkolu má možnost hráč zobrazit tyto úkoly v knize questů. Jakmile je úkol splněn tak hráč získá odměnu, která závisí na obtížnosti.

Aplikace poskytuje uživateli informace přes Heads-Up Display zkráceně HUD což jsou grafické prvky zobrazující například životy, výdrž, úrovně postavy, vlastnosti a podobně. Hráč má možnost otevřít několik oken ve hře při hraní. Jako nejpodstatnější okna jsou inventář, výbava postavy, kniha questů, obchod s předměty, okno pro zadávání úkolů od NPC nebo dialog pro menu hry.

2.2 Inovace hry

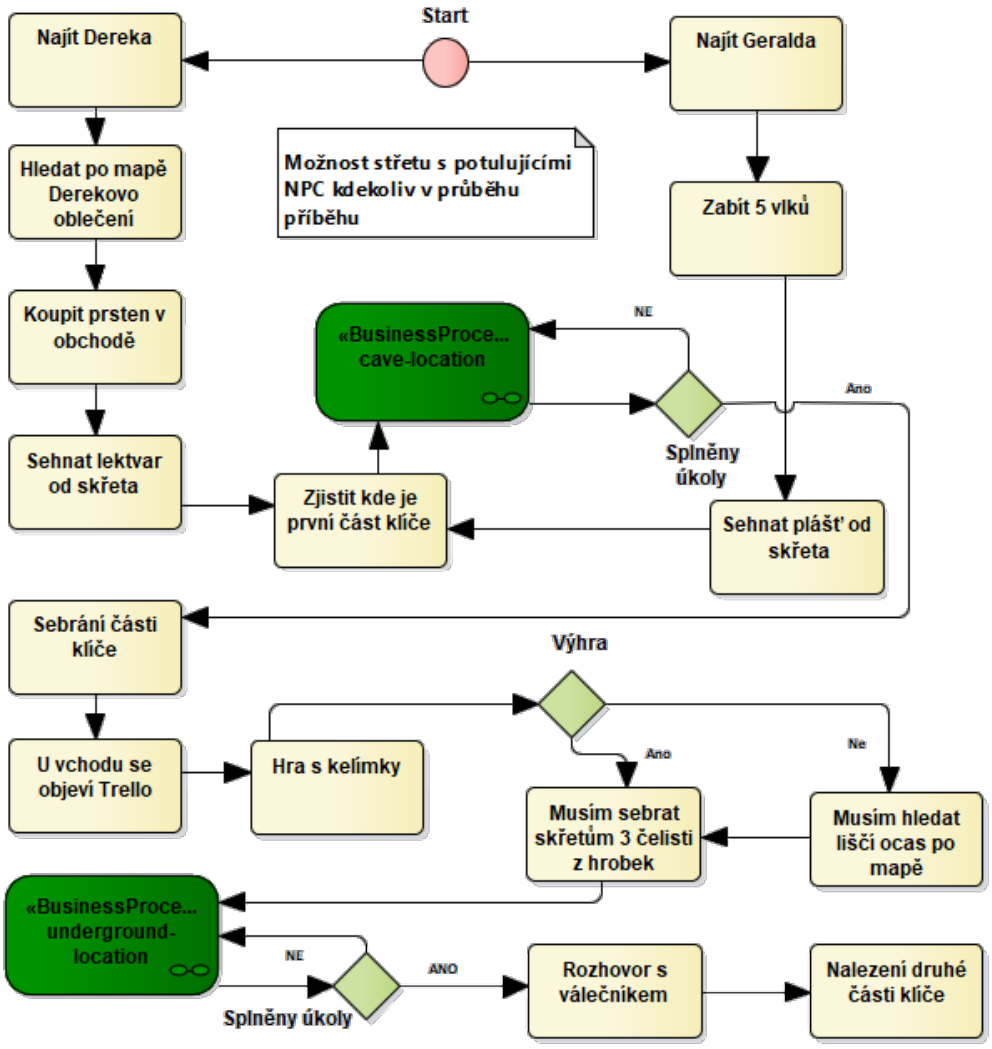
Jako inovace hry se považuje zajímavá možnost zpřístupnit hráčům rozšiřování základní hry bez znalosti programování, a zároveň umožnit vývojáři rychlejší a méně náročnější tvorbu předmětů do hry. Programátor tedy nebude muset psát přímo zdrojový kod pro každý předmět ve hře, ale generovat soubory s kódem přímo do adresáře se hrou. Rozšíření tedy bude možné přes uživatelsky přívětivé rozhraní. Možností jak tohoto docílit je zde více. Buďto externím programem, který přímo nesouvisí se hrou nebo zabudovaný designerem v samotné hře.

V našem případě je vybrána možnost zabudování designéru přímo do hry, která se zdá být výhodnější nejen z pohledu soudržnosti aplikace, ale také většího pohodlí. V designéru je konkrétně možnost vytvářet několik typů předmětů, které jsou po rekompilaci aplikace následně instancovány do tržiště. Je samozřejmostí úprava různých parametrů, nahrání vlastních textur ze souborového systému a podobně.

2.3 Příběh hry

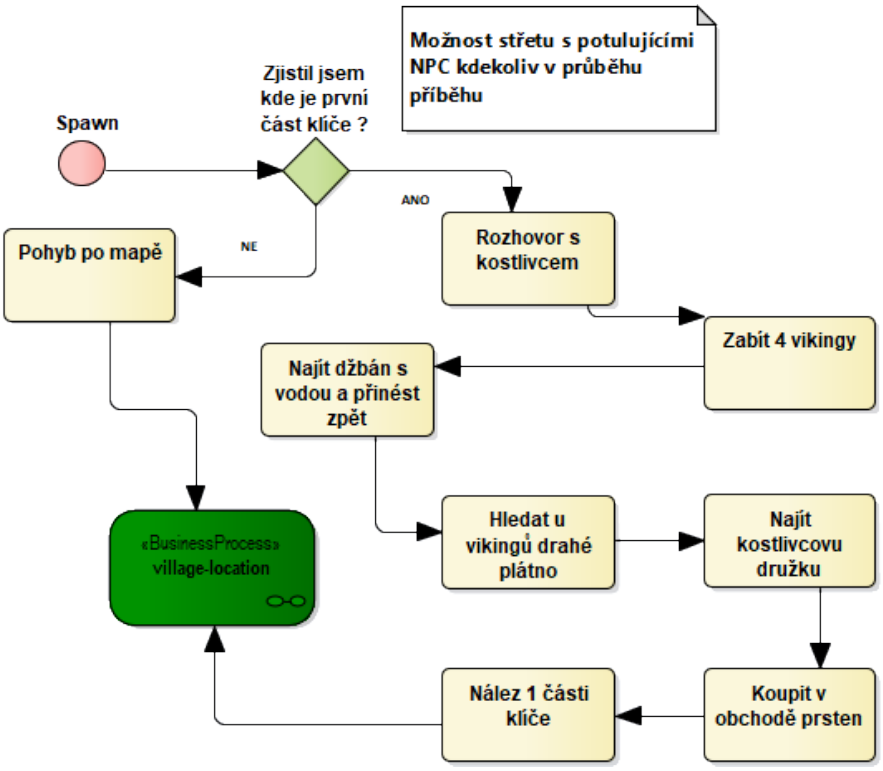
Příběh hry je popsán pomocí vývojových diagramů, které zobrazují dějový tok a zároveň rozvětvení příběhu. Diagramy se dělí do jednotlivých lokací - vesnice, jeskyně, podzemí a tržiště které ale nemá žádnou dějovou linii. Tyto diagramy na sebe navazují, to znamená že například z diagramu vesnice se mohu dostat do příběhového diagramu pro jeskyni.

2.4 Lokace vesnice



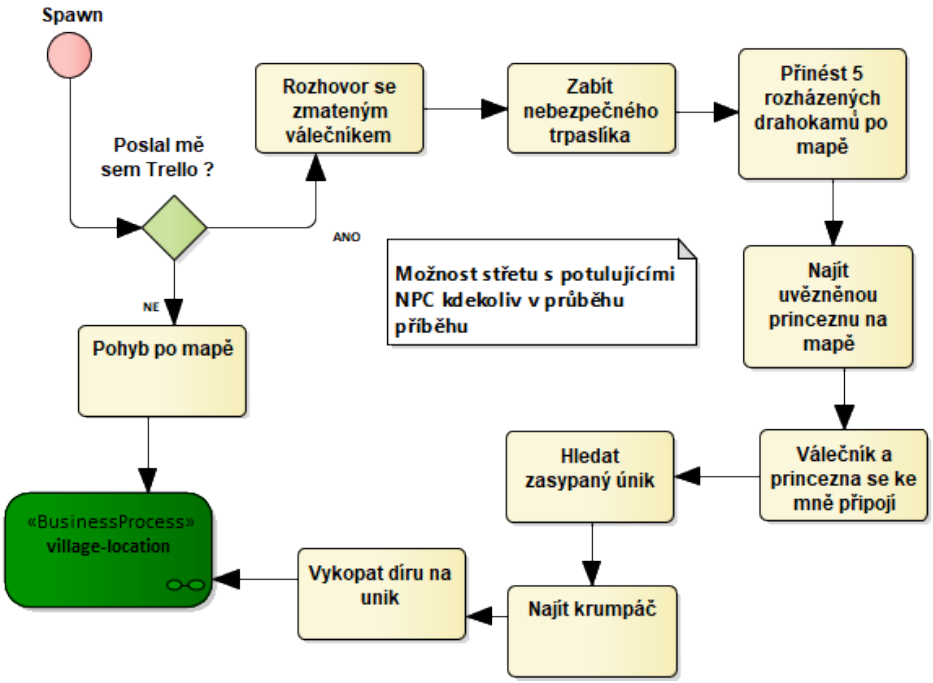
Obrázek 2.1. Lokace vesnice, vývojový diagram

2.5 Lokace jeskyně



Obrázek 2.2. Lokace jeskyně, vývojový diagram

2.6 Lokace podzemí



Obrázek 2.3. Lokace podzemí, vývojový diagram

Kapitola 3
Herní engine

3.1 Volba herního enginu

V první řadě musíme mít jasno a předem určeno jak chceme hru vytvořit. Zda použijeme nástroj, který by nám urychlil vývoj nebo budeme celou aplikaci psát od samotného začátku což může mít za následek razantní zpomalení vývoje ale bude absolutní kontrola nad kódem. K této otázce je také potřeba si uvědomit co konkrétně vlastně děláme. Je potřeba tedy udělat průzkum a uvědomit si naše cíle a potřeby.

Náš cíl je vybudovat sprite-based 2D hru z ptačí perspektivy na blíže nespecifikovanou platformu. Dále by bylo žádoucí nějakým způsobem umožnit rozšiřitelnost hry uživatelům bez hlubší znalosti programování. Vzhledem k tomu že nechceme znovu vynalézat kolo a zároveň je na trhu velké množství různých nástrojů, které pomáhají s vývojem, je potřeba si tyto nástroje projít a vybrat nějaký specificky k našim potřebám. Nástrojů, tj. herních enginů existuje celá řada například Unity, Unreal Engine, Cry Engine nebo Godot, přičemž má každý své výhody a nevýhody.

Nejdřív je potřeba zjistit zda se s game enginem vůbec vyplatí pracovat. Zkusme se podívat na jazyk C/C++ s tím že nebude cílem použít žádný high-level engine pouze je potřeba v tomto jazyku zprostředkovat grafické okno s canvasem aby byla někde možnost vykreslovat hru, to bude znamenat že nejsou k dispozici žádné věci jako je kolizní systém a podobně. Pokud je potřeba taková funkcionalita, bude nutností nalézt nějakou grafickou knihovnu která nám toto umožní. Těch zase existuje více, například SMFL nebo SDL. Podívejme se například na SDL tedy Simple DirectMedia Layer, ta umožňuje podle dokumentace pouze low-level přístup na audio, klávesnici, myš, joystick a grafický hardware. V tomto případě opravdu jediné co knihovna umí je načíst texturu ze souboru a vykreslení na plátno to znamená že všechny věci jako různé animace, kolize či nějaké časování a podobně je potřeba udělat vlastnoručně nemluvě o tom že se vše bude muset psát velmi genericky protože textury mohou být různé jak velikostí, tak formátem atd. [11] Už to zní velmi složitě při uvědomění kolik práce to vlastně zabere celým společnostem které tyto potřeby vyvíjí několik let, často ve velmi početných týmech. Pokud bychom chtěli tedy začít od takto low-level knihovny je evidentní že budeme muset udělat několik podsystémů hry a na tom poté hru budovat - to je ale zbytečné protože tyto podsystémy již dostaneme, pokud začneme s nějakým modernějším a vyspělejším herním enginem.

Na otázku zda se vyplatí engine pro naše potřeby je evidentní že dokážeme s přesvědčením odpovědět ano. Je dobré vybrat nějaké známější enginy, které už jsou prověřené komunitou a tudíž je předpoklad že nebudou dělat větší a nepředvídatelné problémy. Avšak to nebude jediná výhoda známějších enginů. Další bonus je v případném supportu nebo dokumentaci bez které se rozhodně větší projekt neobejde s tím že je potřeba nezapomenout taky na dobrou komunitu a její pomoc. Měli bychom rozhodně vzít nějaký TOP seznam herních enginů a ty vzájemně porovnat a také brát v úvahu rok 2020. Podle serveru indiegamedev.net jsou kandidáti Unity Game Engine, Unreal Engine, RPG Maker, OGRE, Godot a GameMakerStudio. [12] Je samozřejmé že enginů

je mnohem větší množství, avšak všechny se porovnat nemohou a jak už bylo zmíněno je lepší vybírat právě ty známější. Volba nástroje také samozřejmě závisí na osobních preferencí jedince.

3.2 Unity Game Engine

Unity bylo poprvé vydáno v roce 2005 a podle Unity Website PR je nejčastější volbou pro herní vývojáře a to pravděpodobně okolo 45% což z něho činí nejpoužívanější herní engine aktuálně ve světě. Díky obrovské komunitě má engine velké množství velmi přívětivých tutoriálů a v podstatě vše co je potřeba řešit je velmi jednoduché najít. Unity Asset Store nabízí pro Unity Engine obdivuhodné množství vlastních assetů které je možné použít, bohužel tyto assety jsou často zpoplatněné, avšak často je možnost vyzkoušet free demo pro určitou sadu. [13]

Unity je založené na komponentách, například pro vytvoření nějaké postavy je nutné připravit model, poté renderer a následně přidat `rigidBody` pro kolize. Pokud chceme postavu dále vylepšovat, přidáme například `AudioSource` do kterého můžeme vložit zvuky postavy atd. K vytvořeným komponentám je možno napsat scripty v jazyku C# nebo využít vizuální scriptování. [13]

Tento známý game engine má implementováno několik toolů přímo pro tvorbu 2D her jako je například tilemap který výrazně usnadňuje budování lokací. Pro náš účel je tento engine celkem vhodný a myslím si že v něm můžeme relativně bez problému pracovat, avšak vzhledem k rozsahu připraveného projektu představuje zbytečně komplikované řešení. [14]

Výhody	Nevýhody
Uživatelsky přívětivý.	Rezervuje velké množství místa na disku.
Multi-platformní.	Při build operaci velké binární soubory.
Zdarma.	Pokud je vaše hra výdělečná musíte přejít
Mnoho možností.	na placenou verzi.
Obsáhlý asset store.	

Tabulka 3.1. Unity Game Engine

3.3 Unreal Engine

Unreal Engine byl vyvíjen společností Epic Games, kde první použití tohoto enginu bylo představeno ve hře z první osoby z roku 1998. Jeho slávu asi není nutno tolik představovat, v roce 2014 byl zapsán do Guinessovy knihy světových rekordů jako nejúspěšnější herní engine. UE je velmi kvalitní nástroj pro tvorbu především 3D her, je známý tím že se v něm tvoří zejména AAA hry například PlayerUnknown's Battlegrounds, Hellblade: Senua's Sacrifice nebo Bordelands 3. [15] [16] Jeho komunita rozhodně není malá a dokumentace je také pěkně zpracovaná. Avšak díky jeho velikosti tento engine spíše míří na teamy lidí se specializacemi na různé oblasti hry, dalo by se říct že pro jednoho vývojáře je to obrovská komplexní sada nástrojů která v řadě případů ani nebude využita, a vzhledem k tomu že cílí na větší tituly není vhodný ani pro jednoduché hry. To nemění nic na tom že je tento engine na špičkové úrovni a jeho renderovací schopnosti jsou lepší než v případě jiných enginů.

Výhody	Nevýhody
Tvorba AAA her.	Není vhodný pro single vývojáře.
Velké množství nástrojů.	Zabere 70GB místa na disku.
Asset store.	Potřeba znalosti C/C++.
Nabízí granty.	

Tabulka 3.2. Unreal Engine

3.4 RPG Maker

RPG Maker je engine zaměřující se přímo na 2D RPG hry. Je to velmi dobrá volba pro začátečníka. Avšak není zadarmo, což je škoda a zároveň ostatní enginy dokážou to co tento s tím že RPG Maker je soustředěn pouze na RPG hry to znamená že práce v něm na RPG začíná a končí, kdežto práce s jiným herním enginem znamená znalosti i do jiných projektů z důvodu obecnosti.

Výhody	Nevýhody
Přívětivý pro začátečníka.	Není zdarma.
Soustředí se na RPG přímo.	Nelze v něm dělat nic jiného než RPG.

Tabulka 3.3. RPG Maker

3.5 OGRE

Flexibilní 3D engine napsaný v C++, podle tvůrců napsaný tak aby byl jednoduchý a intuitivní. Má velmi aktivní komunitu a v roce 2005 byl vyhlášen projektem měsíce. V tomto enginu bylo vybudováno několik komerčních her jako Ankh, Pacific Storm nebo Torchlight. Avšak pro naši potřebu je nevhodný jak už bylo uvedeno je výhradně soustředěn na 3D projekty.

Výhody	Nevýhody
Jednoduchost a přívětivost prostředí.	Pouze pro 3D hry.
MIT licence.	Komunita není tak rozsáhlá.

Tabulka 3.4. OGRE

3.6 Godot

Godot engine je relativně nový nástroj pro tvorbu 2D / 3D her. Je open-source a velmi dobrou zprávou je, že hru vytvořenou v tomto enginu vývojář kompletně vlastní a nemusí platit žádné licenční poplatky. Podle komunity je velmi dobrou volbou pro nezávislé (Indie) vývojáře. Jeho slabinou je aktuálně malá komunita, například oproti Unity je tato komunita opravdu zanedbatelná. V některých recenzích pro tento engine je zmiňováno že na dokumentaci bude potřebovat ještě hodně práce, avšak já to jako problém nevidím a současnou dokumentaci považuji za velmi kvalitní. [12] [17]

Godot využívá poměrně inovativního způsobu vývoje her, pomocí uzlů které se na sebe mohou připojovat a odpojovat - tento způsob přichází s řadou výhod ale i omezení. Pro scriptování je možnost vybrat buď nativní jazyk což je Godot Script nebo C++ a C#. Je zde také možnost vizuálního scriptování.

Výhody	Nevýhody
Inovativní způsob vývoje.	Malá komunita vývojářů.
Open source.	Pro začátečníka může být matoucí.
Velmi aktivně vyvíjen.	Problémy a pomoc se obtížněji hledají.
Lightweight pro starší počítače.	Neexistuje velmi známá hra v tomto enginu.

Tabulka 3.5. Godot

3.7 GameMaker Studio

GameMaker Studio bylo poprvé vydáno v roce 1999. Jeho hlavní účel je vývoj 2D počítačových her. Komunita tohoto nástroje kladně hodnotí rychlou křivku učení. GameMaker využívá "drag and drop" vizuálního programovacího jazyka nebo klasického scriptovacího jazyka nazívaného Game Maker Language. Avšak je spíše brán jako nástroj pro začínající vývojáře a nováčky. Nedávné verze softwaru se ale zaměřují také na pokročilé vývojáře. Přesto v tomto enginu byly napsány relativně známé a úspěšné hry například Undertale. [12] [18]

Výhody	Nevýhody
Zaměření na 2D hry.	Není zadarmo.
Jednoduché pro začátečníky.	Malá komunita.

Tabulka 3.6. GameMaker Studio

3.8 Komunitní volba

Pro náš výběr by bylo dobré zveřejnit také nějaké statistiky herních enginů které vybrala komunita. Pro tuto statistiku sem vybral server slant.co který je poměrně známý pro porovnávání ekvivalentních věcí. Je asi dobré zdůraznit že je potřeba tuto statistiku brát s nadhledem. První enginy byly doporučovány od většího množství lidí, zbylé již hodnotili jednotlivci. Celkově se hodnocení ke dni 30.9.2020 zúčastnilo 155 uživatelů. [19]

Nejlepší herní enginy podle slant.co komunity
Godot
Unreal Engine 4
Unity Engine
Orx
LibGDX
CryEngine
LOVE
Duality

Tabulka 3.7. Nejlepší herní enginy podle slant.co.

3.9 Finální výběr

Po určitém váhání sem nakonec zvolil Godot Engine, jak z hlediska zvážení výhod a nevýhod tak personální preference. Myšlenka připojování uzlů do stromů kterou tento

engine aplikuje se zdá jako dobrý nápad a neměl by být pro vývoj naší hry na škodu, naopak. Godot Engine má očividně velmi dobré hodnocení v recenzích a je porovnáván s enginama jako je Unity nebo Unreal Engine s tím že se hodí pro 2D Indie hry menšího rozsahu a je doporučován individuálním developerům. Další výhodu cítím v potenciálu enginu, který se aktuálně rapidně rychle vyvíjí a získává na popularitě. Pokud bych vybral nějaký více user-friendly engine zaměřený přímo na RPG tak by získané zkušenosti na tomto softwaru nebyly tak cenné jako v Godotu který je určen pro všechny typy her a dá se využít kdekoliv. Jako další velká výhoda je absolutní svoboda jak v použití herního enginu tak nakládání s finální hrou. Na rozdíl od Godota v mnoha případech je totiž potřeba platit určité poplatky.

Vzhledem k tomu že na hře nepracuji jenom na jednom silnějším počítači, ale také na slabším notebooku je potřeba vybrat více light-weight prostředí což Godot splňuje. Po instalaci na Linux zabral pouze 100MB a je bez problému spustitelný na notebooku s 256 MB grafickou kartou, 8GB Ram a Intel I5 z roku 2015 a vzhledem k tomu že disk je veliký 128GB, tak je nepředstavitelné spustit instalaci například Unreal Enginu který by zabral 70 GB na disku.

Kapitola 4
Godot

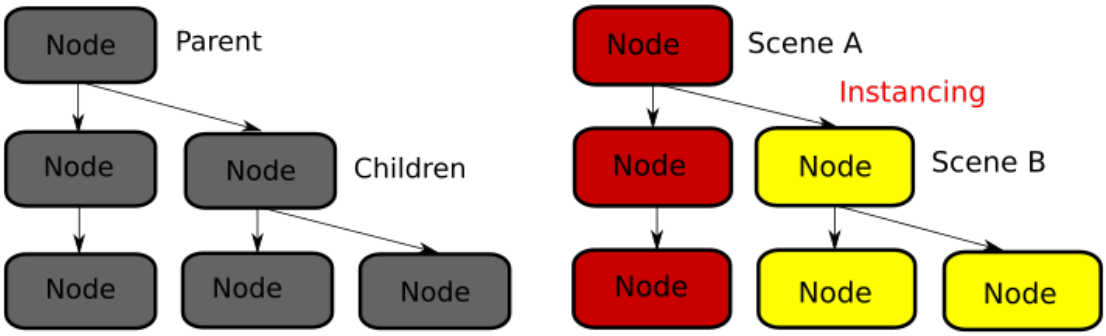
4.1 Jak vyvíjet v Godotu

Vývoj v Godotu je poněkud specifický, je založen na stromové struktuře ve scénách. Stromová struktura je složena z uzlů s tím že uzel může být cokoliv ve hře, tedy jak viditelný prvek tak neviditelný například časovaný Timer pro nějakou událost ve hře, nebo herní charakter chodící po vybrané lokaci. Představme si že máme herní charakter ve hře ten je tedy reprezentován nějakým stromem z uzlů, tyto uzly jsou nějakým způsobem uspořádané. Jako první uzel je root na tento uzel se připojují další uzly neboli děti. Root by v případě herního charakteru měl být zděděn od `KinematicBody2D` dále budeme potřebovat reprezentovat nějakou oblast která bude detekovat kolize, takže připojíme `CollisionShape` do našeho roota a v neposlední řadě bychom chtěli aby byl charakter viditelný, takže připojíme další uzel a tím je `AnimatedSprite`. Uzly mají svoje vlastnosti se kterými je po připojení do stromu možno manipulovat - to znamená že v uzlu `AnimatedSprite` mohu nahrát sadu textur které budou viditelné a zároveň danou boolean hodnotou spustit nějakou rychlostí přepínání textur tak aby charakter vypadal animovaně. Tuto manipulaci mohu vykonat jak v Godot interface tak pomocí vytvořeného scriptu.

Pokud bychom se zaměřili na tom jak je možné v game enginu scriptovat máme v první řadě na výběr několik jazyků. Godot nabízí svůj nativní jazyk Godot Script který je syntaxí velmi obdobný jazyku python a zároveň je vývojáři doporučovaný, avšak máme také na výběr jazyk C++ nebo C#.

Soubor s koncovkou .gd je v godotu brán jako třída, Godot je tedy objektově založen. Je důležité si uvědomit, že zde není žádná syntaxe pro deklaraci třídy, ale soubor jako takový představuje jednu určitou třídu. Takový soubor se scriptem je možno připojovat na libovolný uzel. V tomto scriptu je možnost jak manipulovat s daným uzlem tak se všemi jeho potomky, ale možností je více.

Představme si, že náš charakter má střelnou zbraň, například kuš, která vystřeluje šípy. Pro naši stromovou strukturu to bude znamenat že herní charakter bude mít připojen uzel zbraň tak aby jí mohl držet v rukou, tato zbraň má viditelný šíp a po vystřelení šípu je potřeba node šípu odpojit od zbraně a připojit do lokace abychom dosáhli vystřelení a střela se mohla pohybovat po lokaci. Tato myšlenka je velmi důležitá a znamená to, že jednotlivé podstromy mohu odpojovat a připojovat jinam. Tím pádem můžu předpřipravovat různé větší objekty a scény a dávat je do ještě větších scén a tím celou strukturu do určité míry dekomponovat. Tento způsob myšlení nám dovoluje separátně testovat stromy bez ohledu na dalších strukturách hry.



Obrázek 4.1. Vlevo vztah rodiče a potomka, vpravo instancování scén.

Obrázek s červeno žlutými uzly představuje scénu A (červená barva) která obsahuje již interně nějaké ulzy, do této scény se připojuje další scéna B (žlutá barva).

4.2 Signály

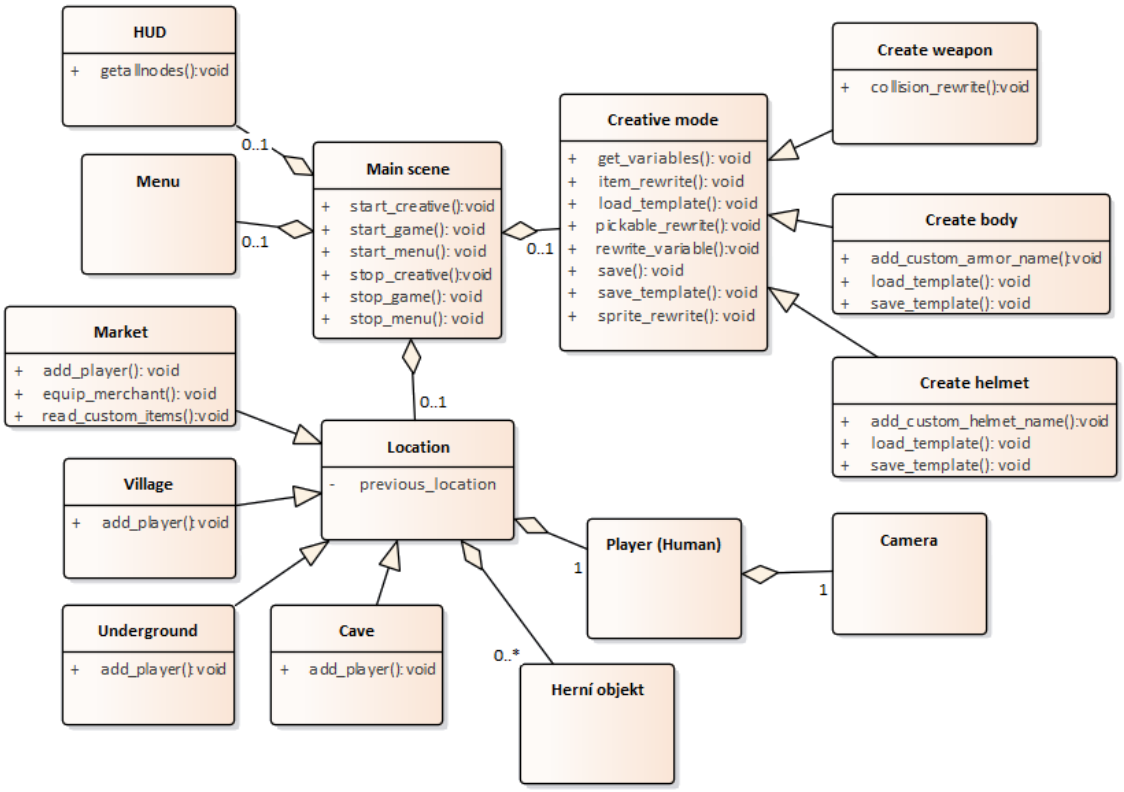
Za velmi důležitou funkcionalitu, o které je dobré se zmínit a která umožňuje jakousi komunikaci mezi uzly se považují signály. Godot signály jsou do jisté míry obdobné jako velmi běžný programovací pattern zvaný Observer / Listener. Jakýkoliv uzel může emitovat signál. Signály mohou být buď vlastní, nebo již předem připravené. Tyto signály pak mohou ostatní uzly zachytávat a vykonávat nějaké akce podle toho jaký signál byl emitován.

Příklad předpřipraveného signálu je vstup nějakého neznámého uzlu do zóny jiného uzlu, který informuje o tom, že do určité vymezené oblasti někdo vstoupil - ten poté informuje všechny kteří si tento signál zaregistrovaly a vykonají nějakou akci podle potřeby. Vlastní signály jsou takové že programátor může signál pojmenovat a emitovat ho v určitých situacích pomocí scriptu kdy zrovna potřebuje. [20]

Kapitola 5
Vývoj a herní mechanismy

5.1 Schématický model aplikace

Hra je rozdělena do několika částí. Pro lepší pochopení a přehlednost je vyobrazen tento UML diagram který představuje hrubý pohled na celkovou strukturu aplikace.



Obrázek 5.1. Class diagram aplikace

5.2 Hierarchie projektu

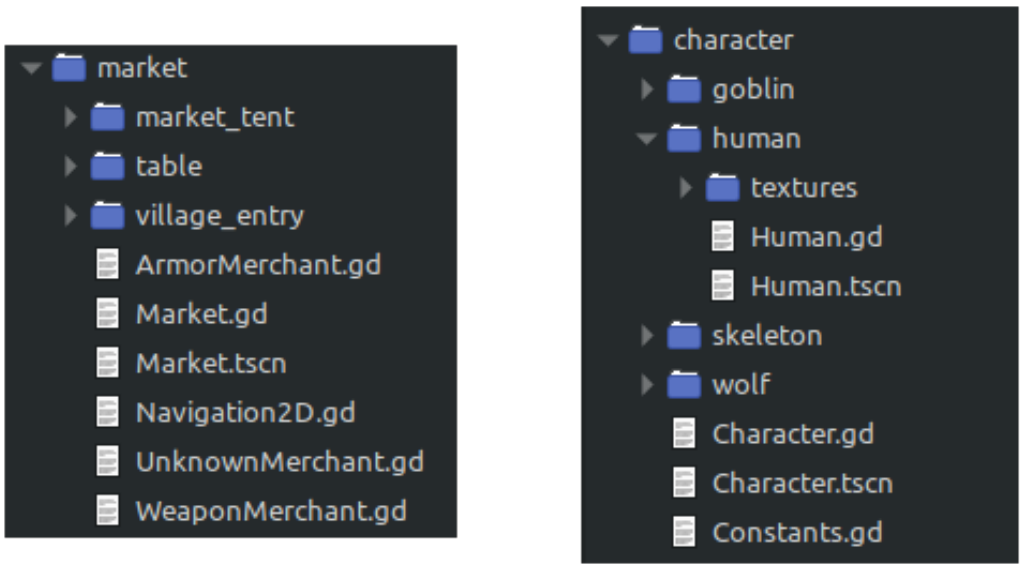
Hierarchii projektu na disku bychom mohli rozdělit na dva styly které se běžně používají, a to buď uspořádat projekt podle typu souborů, nebo podle použití - jakési dědičnosti. Pokud bychom rozdělili projekt podle typu souborů, znamenalo by to vytvořit adresáře pouze pro scripty, scény, textury a ostatní soubory který by měli vždy předpřipravený adresář do kterého by se vkládali. Naproti tomu, uspořádaní podle použití by znamenalo rozdělit adresářovou strukturu do jednotlivých komponent, které se vyskytují ve hře, tyto komponenty mají vždy vše potřebné v daném adresáři. To má za následek, že scripty, scény, textury a ostatní soubory jsou vždy uložené v komponentě - tedy na jednom místě. Tento fakt nám dá velmi dobrou schopnost tvořit dědičnost, tak jak se

projevuje ve hře, protože né všechny komponenty jsou viditelné, tudíž mohu udělat abstraktní komponentu a v ní tvořit další zděděné.

V našem konkrétním případě byl aplikován styl podle použití, důvodů bylo několik, jednak tento styl je velmi vhodný s Godotem a jeho myšlenkou budování stromové hierarchie uzlů, to znamená že je tato struktura ve hře dobře viditelná i v souborovém systému a následně se jednoduše přidávají komponenty které dědí od jiných nebo jednoduše tuto komponentu mají obsahovat. Další důvod pro aplikaci byla lepší přehlednost v projektu, tento fakt se začne projevovat významně při větších aplikacích obsahujících velké množství scén a uzlů. Pokud bychom potřebovali vidět například scénu vesnice znamenalo by to z rootu projektu se dostat do komponenty `Location`. Z této komponenty se dědí další již viditelné takže komponenta `Village`, pokud bych se dostal do této konkrétní komponenty zobrazili by se všechny soubory co komponenta `Village` obsahuje, to znamená její vlastní scénu, scripty, textury a další komponenty které jsou napojené a tudíž je obsahuje ve stromové struktuře hry. Toto by při jiném uspořádání adresářů a souborů buď nešlo namodelovat vůbec, nebo velmi obtížně a chaoticky.

5.3 Skladba komponenty

Komponenta je v podstatě adresář na disku znázorňující herní objekt a to buď viditelný, nebo abstraktní. V komponentě jsou všechny potřebné soubory, které objekt ve hře používá a zároveň zde mohou být další vnořené komponenty a to ze dvou důvodů. První důvod je kvůli dědičnosti, druhý důvod je že nadřazená komponenta má ve hře ostatní vnořené komponenty připojené jako děti. Ilustrace těchto případů může být ukázána na dvojici obrázků. V prvním je komponenta `Character` která je abstraktní, avšak jsou v ní zanořené komponenty které od ní dědí tedy například `Human`. Ve druhém případě je zobrazena struktura komponenty `Market` kde vnořené komponenty `table`, `market_tent` a `village_entry` nic nedědí, ale jsou připojené do této konkrétní lokace ve hře.



Obrázek 5.2. Komponenta vlevo ukazuje vnoření ostatních komponent, komponenta vpravo dědičnost.

5.4 Lokace

Hra je rozdělena do konkrétních lokací, v jeden okamžik může hráč mít zobrazenou pouze jednu lokaci, ve které se pohybuje, pokud se chce přesunout do jiné je potřeba najít vždy nějaký východ z aktuální do požadované. Tento problém je řešen v hlavní scéně se scriptem `main.gd` který se stará o více funkčnosti, jedním z nich jsou změny lokací. Při spuštění hry se lokace instancují a aktuální se připojí jako potomek do hlavní scény. Při změně se odpojí aktuální lokace a připojí se ta která bude následovat. Při této činnosti je potřeba specifikovat kam umístit hráče v závislosti z jaké oblasti přišel. Pokud hráč přišel z tržiště do vesnice bude u vchodu do tržiště, naopak pokud přišel z jeskyně bude se vyskytovat u vchodu do jeskyně. Tato variabilita se vyřeší tím že objekt `Lokace.gd` obsahuje informaci o tom jaká oblast byla předchozí to nám dá možnost pozicovat hráče při přepínání.

Další potřebný prvek ke kontrole tohoto přepínacího systému jsou vchody, v aplikaci nazvané jako `EntryPoint`. Pokud do tohoto vchodu hráč přijde, spustí se proces přepnutí. `EntryPoint` v lokaci má tedy kolizní oblast z Godot Enginu která sleduje určitou plochu a pošle signál do `EntryPoint` objektu, zde se odbaví a přepošle se dál do rodičovského uzlu tedy objektu lokace která emituje další vlastní signál `change_location`, ten je sledován v hlavním skriptu a po jeho emitaci se lokace přepne. Informace o tom jaká lokace se má připojit do hlavní scény je schovaná v argumentu jako string konstanta která se posílá se signálem.

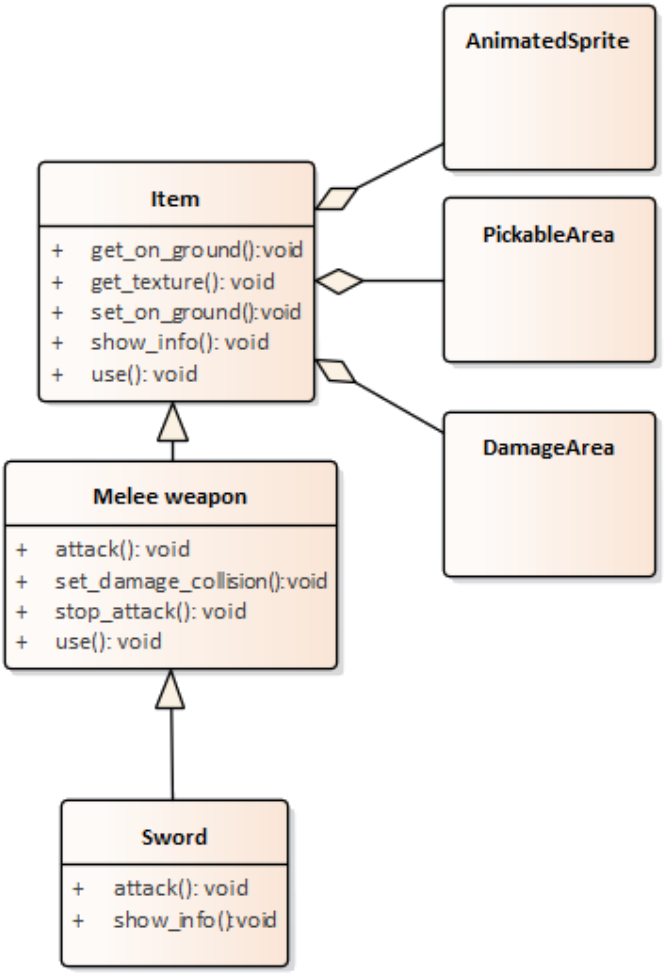
Obrázek 5.3. Grafická ukázka lokace tržiště.

5.5 Předměty

Předměty se rozdělují na několik typů, jako jsou zbraně, brnění, spotřební věci a questové předměty. Všechny tyto předměty dědí od abstraktní scény `Item.tscn` s přidruženým scriptem `Item.gd`. Tato scéna obsahuje potřebné prvky, které musí kterýkoliv předmět zaručovat, jako je animační Node pro texturu, oblast pro možné detekování předmětů na zemi, schopnost zobrazit informace na informačním okně používaném při

obchodu nebo v inventáři. Ikonky které se zobrazují v GUI, jako je právě inventář, jsou vráceny z `Item` objektu a při jejich požádání se vezme textura která se používá běžně ve hře, například při držení předmětu v ruce, avšak ta se nadále upravuje, škáluje a následně se smíchá s pozadím tak aby výsledně byl předmět v jakémsi rámečku a nebyl příliš malý nebo velký. Použité metody pro úpravu jsou tedy `get_rect()`, `resize()` a `blend_rect()` kde první dvě metody se starají o ořezávání a škálování zatímco třetí metoda dokáže smíchat dva obrázky do sebe. V případě předmětu kopí by byla jeho textura v ikonce velmi malá z důvodu výšky, v konstruktoru se tedy nastaví oblast na textuře a tím se vyznačí s jakou částí se má pracovat takže v tomto konkrétním případě bude vidět ostří a horní polovina dřevěné tyče. Předměty které mají určitou funkcionalitu (možnost nasazení či konzumace) přepisují metodu `use(character)` kde argument je postava která tento předmět vlastní. Tato metoda se vždy zavolá jakmile se má předmět (obvykle z inventáře) použít, tedy při stisknutí tlačítka "USE ITEM", ale může to být i při jiné příležitosti například pokud si NPC vymění předmět který má aktuálně nasazený z jeho listu vlastněných předmětů. Funkcionalita této metody se samozřejmě odvíjí od daného předmětu - v případě vybavení jako je brnění nebo zbraně či itemy které mohou být drženy se nejdříve zjišťuje zda postava již nemá jiný předmět, který by bránil v užití nového, pokud ano je starý předmět přemístěn do inventáře a poté nasazen nový. V případě konzumních předmětů není potřeba zjišťovat příliš informací pouze navýšit potřebné vlastnosti postavy (ta je v argumentu metody) v závislosti na konzumujícím předmětu.

Rozlišují se dva typy zbraní a tím je střelná zbraň a zbraň krátkého dosahu tedy ruční. Ruční zbraně jsou v adresáři označené jako `melee_weapon` a poté děděny z tohoto objektu na specifičtější typy, avšak ty fungují v zásadě velmi podobně. Při zavolání metody `attack()` se aktivuje kolizní oblast která podle masky pozná objekt jež by měl být nějakým způsobem ovlivněn a zároveň spustí i jiné funkcionality jako je emitace signálů. Po nějakém čase je zavolána metoda `stop_attack()`, která je naopak použita k vypnutí útoku. Jakmile je zaznamenán objekt na kterém proběhla kolize a zároveň odpovídají patřičné masky je již na tomto kolidujícím předmětu přesně definováno co se s ním má nadále stát. To znamená že každý objekt u něhož je možno interagovat pomocí zbraně musí mít metodu `attakced_by_this_weapon(weapon)` a právě v této metodě se definuje co vše se má provést a zároveň pro zjištění případného rozsahu poškození a dalších informací je zbraň předána jako argument funkce. Definice této metody na kolidujícím objektu a né na samotné zbrani nám dává variabilitu co přesně se má s objektem stát. Střelné zbraně fungují na využívání obyčejných zbraní krátkého dosahu a to tím způsobem že střelivo je poděděno od `melee_weapon` takže detekuje objekty při letu. Střelné zbraně mají implementován seznam pro střelivo, v případě kuše je Node který charakterizuje šíp připojen na tuto zbraň takže je viditelný ve hře a při aktivaci útoku je odpojen od zbraně, připojen na uzel lokace a následně se již šíp pohybuje ve směru vystřelení pomocí metody `_process()` která je pravidelně volána při každém snímku hry. Směr vystřelení byl zadán již při aktivaci útoku podle natočení postavy.

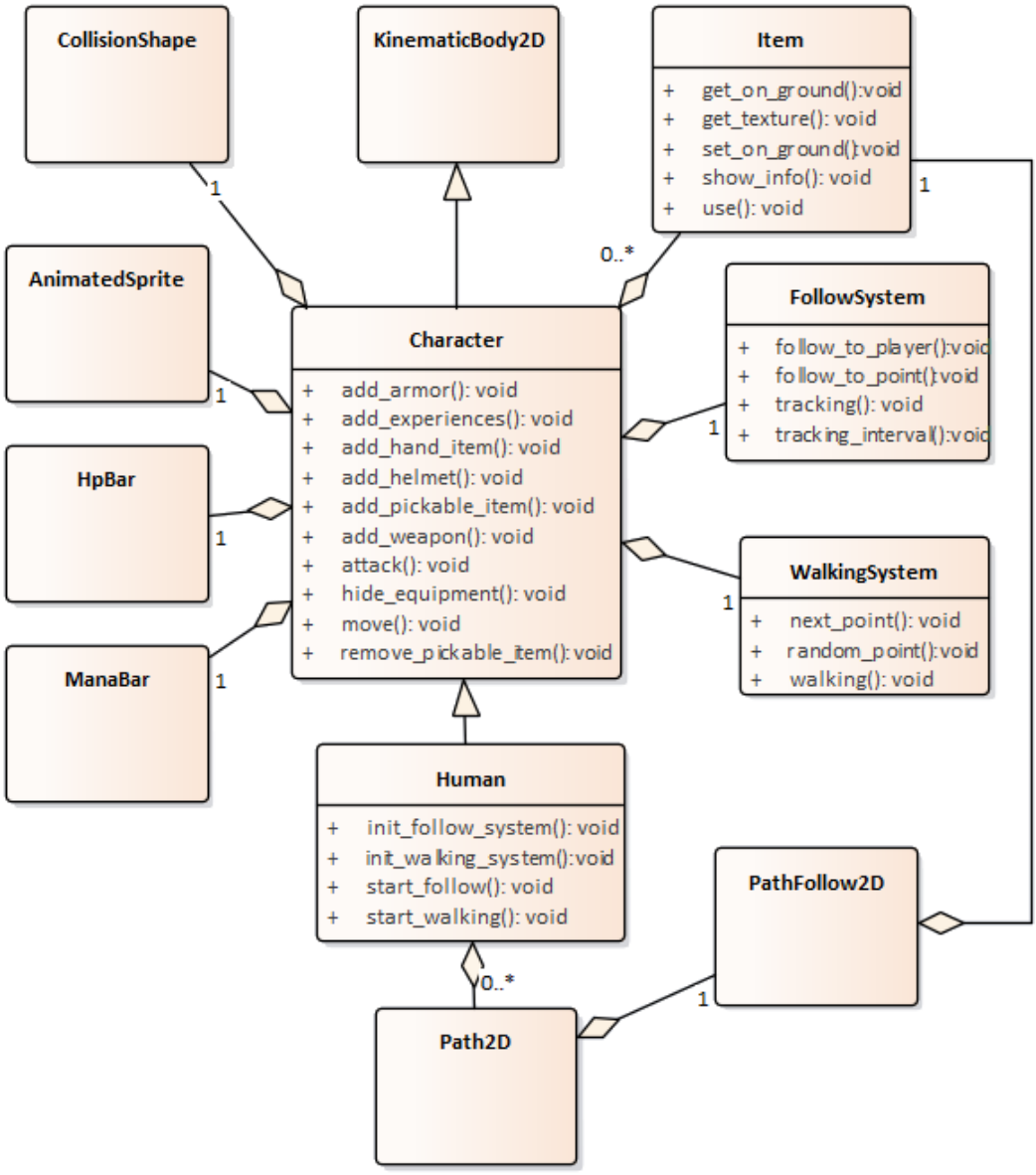


Obrázek 5.4. Diagram meče

Zvláštním řešením útoku je u charakterů kteří nemají možnost držet předměty v ruce, jelikož útok je implementován pouze předměty s detekcí kolize. Tento případ se objevuje například u vlků. Vlci mohou útočit pouze čelistmi tedy částí jejich těla, proto při vývoji byla hlava a tělo separováno do dvou textur kde hlava je poděděna od `melee_weapon` tudíž se chová jako zbraň a dá se nasadit na vlka. Zde je tedy stejný princip jako při nasazení kterékoliv jiné zbraně u člověka. Tělo vlka je zděděno od objektu `Character`. Předměty, které je možno držet v ruce a zároveň to nejsou zbraně se dědí od objektu `Hand_Item`, ten přepisuje metodu `use(Character)` a řídí podle ní případné kolize předmětů v ruce při nasazení, příklad itemu který mohu držet je pochodeň ta se využívá v lokaci podzemí kde z důvodu tmy není jiná možnost než chodit právě s pochodní která vyzařuje světlo. Konzumní předměty jsou tvořeny z objektu `Consumable` a fungují zase obdobně pomocí metody `use(Character)`. U questových itemů které jsou z většiny využívané při plnění úkolů jak název napovídá není nutno rozšiřovat další logikou při dědění od objektu `Item`, protože jsou tyto předměty obvykle nějakým způsobem pouze získány a přineseny zpět k zadavateli. Pokud má questový předmět nějakou další funkcionalitu je tato funkcionalita jednoduše hierarchicky přidána. To je viditelné u předmětu pitcher u kterého je potřeba interakce pomocí GUI prvků, tím pádem jsou tyto prvky vloženy v adresáři s předmětem.

5.6 Herní postavy

Postava by měla mít určité požadované schopnosti, které jsou na RPG hru běžné. To znamená pohyb v prostoru, možnost útoku, vlastnění předmětů, odhazování předmětů na zem, sbírání předmětů ze země, získávání zkušeností a zvyšování levelu, životy, mana a mnoho dalšího. Tyto schopnosti jsou běžné pro všechny typy charakterů. Z tohoto důvodu jsou všechny postavy zděděny od objektu `Character` který řeší tyto potřeby. `Character` je dále potomek Godot objektu `KinematicBody2D`. Tento Godot objekt je speciálně tvořen pro herní postavy zvláště kontrolované uživatelem, výhody jsou ale také v typu pohybu, který se využije v kolizním systému Godotu a v naší aplikaci zvláště pak pro systém trackování postav.



Obrázek 5.5. Class diagram postavy

5.6.1 Pohyb postav

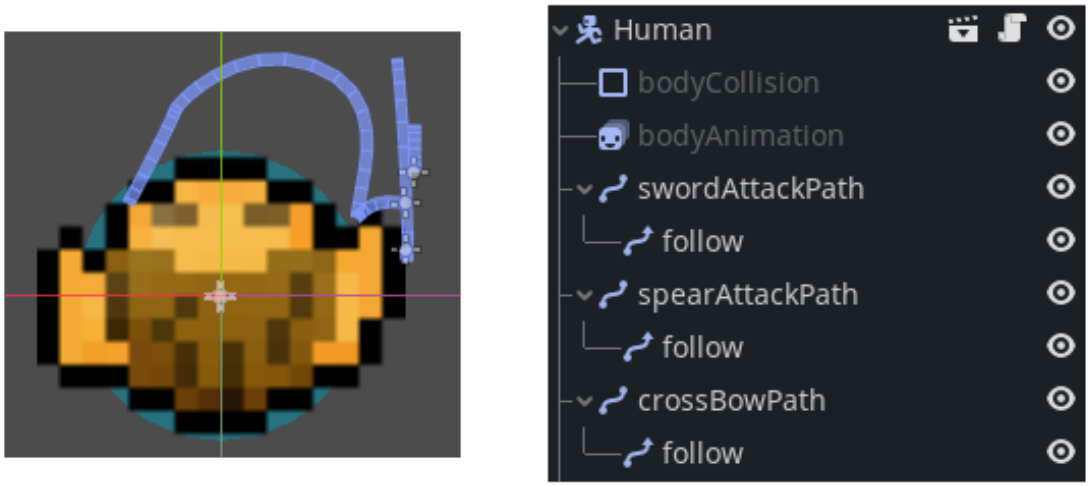
Postavy se pohybují po osách X a Y to znamená, že vektor pro pohyb je vždy rovnoběžný s hlavními osami, a zároveň je pronásoben s rychlostí. Pro pohyb postavy v

určitém směru je využita funkcionalita `move_and_slide`. Tato integrovaná funkce detekuje kolizi postavy s jiným neprůchodným objektem a pokud je to možné upřednostní sklouznutí po tomto objektu nad okamžitým zastavením. V praxi to vypadá tak že pokud postava koliduje pouze s rohem nějaké neprůchodného čtverce, tak se o tento roh nezastaví, ale dokáže jít dál podél rohu, naproti tomu pokud by se začalo kolidovat s celou stěnou postava by musela zastavit. Pokud by stěna byla zešikmená potom by charakter sklouznul podél této šikmé stěny což se využívá u side-scrolling her s gravitačními prvky.

5.6.2 Interakce s předměty

Vlastnění předmětů je u postav řešeno seznamem do kterého se mohou předměty vkládat, equipment - tedy předměty aktuálně nasazené na postavě jsou řešené předpřipraveným seznamem se 4 sloty. Každý slot je pro jiný typ vybavení, slot na indexu 0 je pro zbraň, na indexu 1 jsou předměty které je možnost držet v levé ruce, index 2 pro brnění a podobně. Je několik možností kdy postava může získat předmět, například obchodem, sebráním ze země nebo získáním z questu. K tomu aby postava mohla sebrat předmět, je potřeba zjistit zdá se vůbec u nějakého předmětu nachází. Tento problém řeší předmět samotný, protože každý item ve hře má kruhovou oblast do které pokud někdo vstoupí je automaticky identifikován jako postava která může předmět vzít. To znamená že po tom co postava přijde k předmětu se na kolizní oblasti itemu emituje signál `body_entered`, jako posluchač je předmět samotný takže se zavolá funkce na předmětu s argumentem respektive kolidující postavou ve hře. V takovém případě dokáže aplikace již informovat postavu o přítomnosti předmětu. Na postavě je tedy implementován další seznam, tentokrát pojmenovaný jako `pickable_list` do kterého se vkládají předměty jež je možné sebrat. Každý charakter může tedy sbírat jen ty předměty které má v tomto seznamu, přičemž je potřeba předmět který si postava vezme odpojit od rodičovského uzlu v tomto případě speciální uzel v lokaci na který se připojují itemy ležící na zemi. Po tom co postava odejde z kolidující oblasti se tento předmět odstraní z listu stejným způsobem jako se přidal.

Při nasazení předmětu do equipmentu musí být předmět v inventáři postavy. Zbraně se nasazují v několika krocích. Nejdříve je objekt přesunut z listu itemů do statického listu pro equipment, poté se nastavují kolizní masky v závislosti na tom zda je vlastnící postava hráč nebo NPC. Kolizní maska definuje komu item bude snižovat životy. Dalším krokem je připojení signálů ze zbraně do postavy, příkladem je signál `kill` který indikuje že hráč pomocí zbraně někoho zabil, tento signál se poté používá pro různé účely jako přidávání zkušeností postavě nebo kontrolování plnění questů. Následně se emituje signál `damage_changed` aby byly aktualizovány některé komponenty ve hře, které zobrazují damage. V případě hráče dostane tento signál HUD pro změnu textového labelu. V poslední řadě musí být zbraň připojena na postavu, avšak aby zbraň při útoky vykonávala určitý pohyb je vhodné ji připojit na speciální Godot objekt nazvaný `Path2D`. Pomocí tohoto objektu je možné se zbraní pohybovat v určité předem určené dráze. Postava těchto `Path2D` instancí může mít více, jelikož odlišné zbraně mají odlišnou dráhu, například kopí se pohybuje po přímce od hráče a zpět, naproti tomu meč má pohyb po jakémsi půlkruhu a podobně. Na levém obrázku jsou viditelné křivky znázorňující dráhy zbraní, vpravo je vidět stromová struktura, která ukazuje připojení těchto křivek na postavu.



Obrázek 5.6. Nalevo `Path2D` ve stromové struktuře, napravo graficky vyobrazeny křivky.

V případě nasazení předmětů, které reprezentují obranu jako je helma, brnění nebo štít je potřeba přidat tyto itemy do listu pro equipment a připojit je na uzel postavy. Následně emitovat signál `armor_changed` který má stejný význam jako signál `damage_changed` pro zbraň. Míra ochrany se poté z těchto předmětů počítá prostým sečtením všech armor bodů a využívá se při případném boji.

5.6.3 Level a zkušenosti

Zkušenosti jsou ve hře získávány pomocí questů nebo zabíjení nepřátel. Jakmile se dosáhne určitého množství zkušeností tyto body jsou transformovány na level. Čím vyšší level mám tím více zkušeností potřebuji pro dosažení další úrovně. Toto přidávání zkušeností a transformace na úrovně probíhá přes metodu na postavách a tím je `add_experiences(exp)` kde její argument je množství zkušeností. Tato metoda tedy přidá zkušenosti, poté zkontroluje podle vzorce, zda je dosaženo potřebné množství. Pokud je zkušeností dostatek zvýší se level, odečtou se potřebné zkušenosti a se zbytkem se metoda rekurzivně zavolá.

5.7 Questy

RPG hra má obvykle určitou dějovou linku která se zakládá na interakci s ostatními herními postavami, plnění úkolů a zabíjení nepřátel. Tento děj obsahuje určité dílčí části nazývající se questy. Ty je potřeba plnit postupně a často v určitém pořadí k dosažení nějakého cíle a úspěšného dohrání hry. Questy je tedy potřeba propojit do jakéhosi řetězce a zároveň umožnit určité rozvětvení tak aby příběh nebyl přímo lineární. Je potřeba se vypořádat s několika problémy, jako přiřazování textových dialogů k postavám, spouštění různých předpřipravených událostí, vytvořit variabilitu určitého rozhodování při akceptování úkolů a podobně.

Úkoly v naší aplikaci jsou řešeny spojením dvou typů souborů a tím je JSON formát obsahující data o questu a scriptovací soubor který má na starosti chod samotného úkolu a spouštění akcí v průběhu vykonávání.

Obrázek 5.7. Grafické okno questu.

5.7.1 Datový soubor JSON

Tento soubor obsahuje v rootu objekt skládající se z 6 hodnot: `title`, `characterName`, `text`, `reward`, `script` a `children`. Tyto hodnoty se poté využijí při zpracování úkolu. Při hraní jsou obvykle různé stavy postav se kterými se komunikuje v rámci vykonávání questů, možný příklad stavů postavy je chování před a po vykonání questu. Před vykonáním se získá informace od NPC o tom co by se mělo udělat, spustí se určité typy akcí pro to aby mohlo začít vykonání úkolu a případně další nezbytné úlohy, naproti tomu po vykonání akce od dané postavy dostanu určitou formu odměny a odlišný text s jinými informacemi o případném pokračování. Tento problém je řešen pomocí slovníku. Předpokládejme že náš hráč komunikuje s herní postavou která má potencionálně dva stavy `stav_0` a `stav_1`. Pokud budu v nějaké fázi questu kde postava by měla mít `stav_0` jednoduše z datového souboru vezmu všechny data kde klíč je `stav_0`, pokud se postava dostane do `stav_1` nastaví se dialogy a ostatní data do právě tohoto stavu. Tyto stavy jsou dále využívány ve scriptovacích souborech přiřazených k danému questu.

Dialogy postav, které se nacházejí ve slovníku `text` jsou rozděleny v seznamech, to má praktický význam při stránkování kde jedna položka v seznamu odpovídá jedné stránce textu. Hodnota `characterName` je využívána pro zobrazování jména s kterou postavou se aktuálně hovoří, konkrétně je zde slovník namapovaný na stavy. `Reward` určuje odměnu která se přiřadí hráči za úspěšné dokončení úkolu. Hodnota `script` je odkaz na Godot soubor který se stará o průběh úkolu a ostatní potřebné úlohy. Poslední hodnota `children` definuje které questy jsou navazující, vzhledem k tomu že je možné

rozvětvení, tak je potřeba aby tato hodnota obsahovala slovník s klíčema které se použijí podle rozhodnutí hráče ve hře, hodnoty v tomto slovníku jsou odkazy na JSON soubory obsahující další questy.

Na ilustrativním obrázku jsou viditelné tři typy stavů `gerald_0`, `gerald_1` a `skeleton_0`. To znamená, že postava Gerald má zde dva stavy v rámci questu a kostlivec pouze jeden. Kde při stavu `gerald_0` jsou k zobrazení tři stránky textu, jméno charakteru je v tomto stavu Gerald. Při dokončení úlohy ať už úspěšně nebo neúspěšně je možno dostat úkol buď z klíče `option_1` nebo `option_2`.

```
{
    "title": "Title",
    "characterName": {
        "default": "",
        "gerald_0": "Gerald",
        "gerald_1": "Gerald",
        "skeleton_0": "Skeleton"
    },
    "text": {
        "default": "",
        "gerald_0": [
            "Text in state gerald_0 (page 1)",
            "Text in state gerald_0 (page 2)",
            "Text in state gerald_0 (page 3)"
        ],
        "gerald_1": [
            "Text in state gerald_1 (page 1)",
            "Text in state gerald_1 (page 2)",
            "Text in state gerald_1 (page 3)",
        ],
        "skeleton_0": [
            "Text in state skeleton_0 (page 1)",
            "Text in state skeleton_0 (page 2)",
        ]
    },
    "reward": {
        "money": 150,
        "experiences": 10,
        "items": []
    },
    "script": "res://game_quests/scripts/main_story/assigned_script.gd",
    "children": {
        "option_1": [
            "main_story/location/continue_1.json"
        ],
        "option_2": [
            "main_story/location/continue_2.json"
        ]
    }
}
```

5.7.2 Scriptovací soubor

Scriptovací soubor je nahrán podle hodnoty script v JSON souboru ta obsahuje jedinečnou cestu v adresáři hry. Všechny tyto scripty se nacházejí v cestě `game_quests/scripts`. Každý soubor představuje obvykle funkcionalitu přesně pro jeden úkol ve hře a soubory jsou pojmenovány podle jména questu. Všechny scripty se dědí od objektu `Quest`. Ten dopředu deklaruje proměnné pro často používané prvky ve hře se kterými je potřeba interagovat, například uzel pro zobrazování textu nebo tlačítka. Objekt `Quest` ale také poskytuje často využívanou funkcionalitu jako je odpojení signálů na tlačítkách nebo jiných objektech či odstranění questových předmětů z inventáře při splnění úkolu. Jednotlivé specifické úkoly již obsahují určitý vzor jak tyto soubory psát. Při spuštění úkolu se zavolá jako první metoda `_ready()`. Ta prvotně nastaví texty a GUI prvky tak aby byla viditelná odměna za úkol, dále zjistí zda zainteresované NPC se kterými se má komunikovat mají připojený `QuestInteraction` což je v zásadě objekt, jež se připojuje na postavy a dá možnost s těmito postavami komunikovat při úkolech, to se ve hře projeví graficky rotujícím prstencem - ten dává hráči najevo, že je možnost k této postavě přijít a zmáčknout tlačítko Q pro otevření dialogu. Na objektech `QuestInteracion` se poté emitují signály `quest_open` a `quest_close` ty se připojí na metody jež se vykonají. Při signálu `quest_close` obykle stačí pouze zavřít dialogové okno s questem, při signálu `quest_open` se již vyhodnocuje u jakého charakteru byl tento signál spuštěn a zavolá se patřičná část kódu. Po vyhodnocení u koho nastala interakce se nejdřív odpojí signály u tlačítek aby se předešlo únikům kdy některé funkce (z předešlých interakcí s dialogem) budou na těchto prvcích stále viset. Poté se zjistí v jakém stavu je charakter a následně se nastavují a vykonávají úlohy spojené s tímto stavem u aktuálního charakteru - jako dobrý příklad může být přidáním prvku na mapě který musí být hráčem zničen. Při dokončení úkolu se vykoná proces který odpojí signály, odstraní interakce na NPC, vymaže předměty které byly použity v průběhu úkolu z inventáře hráče a další ukončovací procesy v závislosti na jednotlivých questech. Poté je zjištěno na jaký další úkol se pokračuje, ten je nahrán a jeho script zavolán.

5.8 GUI

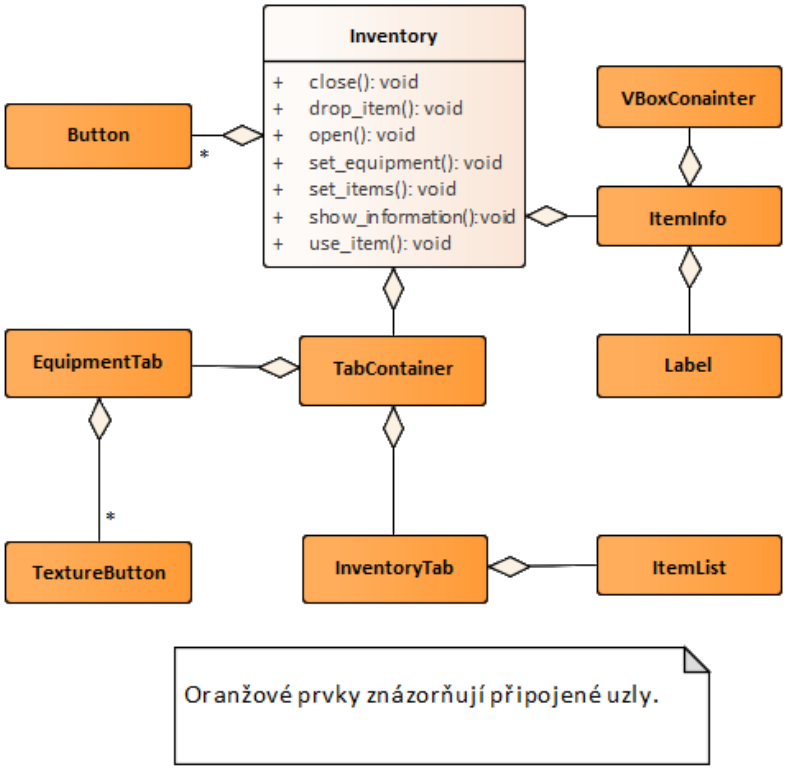
V rootu, což je v našem případě hlavní scéna, jsou připojeny dva uzly z nichž jeden je připojen staticky, tedy již při kompilaci a druhý v runtime - během chodu programu. Uzel který je připojen v runtime je lokace která není stálá a bude se měnit s jinými uzly definující odlišné lokace. Další uzel který je připojen staticky je pro GUI prvky a jeho typ musí být `CanvasLayer` z důvodu vykreslování po vrstvách. `CanvasLayer` nám tedy dovolí vytvořit další grafickou vrstvu ve hře. Tyto vrstvy jsou obdobné jako v populárním grafickém nástroji GIMP kde nejvyšší vrstva překrývá ty spodní. Důvody proč je potřeba pro GUI vytvářet další vrstvu jsou dva, jednak je potřeba aby GUI prvky byly vždy vidět a nepřekrýval je žádný objekt z lokace, ale také při chůzi kde je se všemi objekty posouváno což by při interaktivním prvku jako jsou tlačítka nebo texty nebylo žádoucí protože by při pohybu zmizely z viewportu.

Na uzel `CanvasLayer` je tedy přímo napojen uzel `HUD` a zde se připojují další grafické části které patří do interaktivní části tedy tlačítka, bary, dialogy, okno pro questy, inventář, okno pro obchod, informační tabule a další. Při inicializaci je potřeba nejdříve připojit jednotlivé části HUDu na herní prvky který tento HUD ovlivňují (posílají HUDu signály při nějaké změně) což je například inventář postavy hráče, jeho zdraví,

zkušenosti atd. V tomto případě je ve scriptu `HUD.gd` spuštěna funkcionalita pro prohledání všech připojených uzlů a to jak přímo tak nepřímo a na každém uzlu který má implementovanou metodu `init_connection()` je metoda zavolána, tím je zajištěno propojení mezi změnami ve hře a viditelným HUDem který dává hráči aktuální informace. Následně je možno s HUD prostředím interagovat jak myší tak klávesnicí. Některé primitivní prvky jako jsou ukazatele nebo texty zobrazující úroveň hráče jsou připojeny jako klasické uzly a je k nim přidělen pouze script který určuje kdy se má prvek aktualizovat. Ostatní složitější části jako dialogové okno, inventář, obchod nebo vyskakovací cedule jsou přímo scény, které se instancují a připojí na HUD.

5.8.1 Inventář

Okno inventáře je složeno ze tří částí, první část je uzel `TabContainer` sloužící na selekci předmětů jak v seznamu vlastněných předmětů tak v equipmentu (předměty které jsou nasazené na postavě), druhá část je pro zobrazení informací o aktuálně vybraném předmětu a následně třetí část jsou akční tlačítka, které mění svůj význam v závislosti na vybrané záložce v `TabContaineru`. Kontejner pro záložky obsahuje dva uzly (každý potomek tedy uzel představuje jednu záložku v kontejneru) první uzel je `ItemList` který umožňuje vkládat pod sebe ikonky předmětů a jejich název, druhý uzel je skladba čtyř texturovaných tlačítek znázorňující nasazené předměty. Při vybrání záložky "Inventory" je možnost předměty použít. Použití předmětu závisí na jeho typu. Také je možnost tlačítkem "DROP ITEM" předmět odhodit. U záložky "Equipment" je možnost nasazené předměty vracet zpět do inventáře tedy do seznamu vlastněných předmětů nebo předmět ihned odhodit na zem bez nutnosti ho přemisťovat nejdříve do inventáře. Část inventáře zobrazující data o vybraném předmětu má dva potomky, první je klasický text na zobrazení jména předmětu a druhý je `VBoxContainer` ten umožňuje při připojování dalších uzlů formátovat tyto potomky pod sebe. Důvod pro tento kontejner je že není předem zřejmé zda selektovaný předmět bude obsahovat pouze jednu nebo několik informací.



Obrázek 5.8. Schema uzlů k inventáři.

Při spuštění hry je inventář (stejně tak jako většina oken) již instancován a připojen na `HUD` uzel, avšak tyto scény jsou zneviditelněny. Jakmile aplikace zaznamená požadavek na otevření inventáře je hodnota visible nastavena na true a provedeny ostatní instrukce které vyplní inventář předměty - všechny akce se spustí při zavolání metody `open()`. Jako první se nastavuje informace o tom že vybraný předmět je na indexu 0 - tedy první předmět. Poté se vyčistí informační část o vybraném předmětu tím způsobem že jméno předmětu je nastaveno na defaultní hodnotu "NOTHING SELECTED"

a vertikálnímu kontejneru pro data jsou odstraněny všechny potomci. Další fáze je v naplnění `ItemListu` o vlastněné předměty. Každý charakter (tedy script `Character.gd`) v sobě obsahuje seznam předmětů - tento seznam se tedy vezme a pro každý předmět je zjištěno jméno v proměnné `NAME` a vrácena textura z metody `get_texture()`. Tyto data jsou následně použity pro vložení do `ItemListu` v záložce "Inventory" pomocí metody `set_item_text(index,text)` a `set_item_icon(index,icon)`. Indexy vkládaných předmětů jsou ve stejném pořadí jako v seznamu na objektu `Character` to nám dá možnost zjistit s jakým předmětem manipulujeme. Poslední fáze je v aktualizaci záložky "Equipment", jako první jsou všechny tlačítka a texty nastaveny na defaultní hodnoty a poté je zjišťováno zda v seznamu equipment na objektu `Character` jsou nějaké objekty mezi index 0 až 3. Pokud ano změní se textura daného tlačítka a nastaví se jméno předmětu. Interakce poté probíhá na způsobu poslechu a emitace signálů, kde například v `ItemListu` při vybrání předmětu se provolá metoda s argumentem nastaveným na daný index (tyto emitace a poslouchání signálů jsou velmi obdobné programovacímu patternu Listener/Observer), podle indexu je zjištěno s jakým předmětem se poté má pracovat, protože pořadí v `ItemListu` je stejné jako pořadí seznamu v `Charakteru`. Následně se čeká na akci, která si přečte v lokální proměnné poslední vybraný index. Texturovaná tlačítka, která jsou v equipmentu jsou přesně spojená s indexy tudíž při stisku tlačítka je automaticky jasné jaký předmět selektovat a zobrazit o něm data.

Obrázek 5.9. Grafická vizualizace inventáře.

5.8.2 Obchod

Obchod je ve hře implementován velmi podobně jako inventář. Jsou zde tedy tři části: `TabContainer`, okno pro detail selektovaného předmětu a poté akční tlačítko které mění chování v závislosti na vybrané záložce. Stromová struktura uzlů v této scéně je tedy v zásadě stejná, kromě odlišností jako přidání horizontálního kontejneru `HBoxContainer` pro obchodníka kde je zobrazeno jeho jmění. Logické fungování je tedy stejné s tím že u

obchodu je potřeba nastavit dvakrát `ItemList`: jeden pro hráče a druhý pro obchodníka. Důležitým rozdílem je přítomnost metody `trade()` která se zavolá při stisknutí tlačítka. Tato metoda má za cíl přemístit předmět z jednoho seznamu do druhého a zároveň s tím i aktualizovat jmění u obchodujících charakterů (plátcem je ten kdo předmět získá) a poté aktualizovat `ItemListy` tak aby odpovídali seznamům v `Charakterech`.

V definici metody `trade()` je nejdříve zjištěno, zda je vybrán list hráče nebo obchodníka, v závislosti na tom je zavolána metoda `sell_item(index, character)` která je definována ve scriptu `Character.gd`. Objekt který tuto metodu volá je prodávající a argument `character` je kupující. Vykonání metody `sell_item` je takové že se nejdříve zjistí zda kupující má dostatečné množství peněz. Pokud ne, předmět není přemístěn a metoda končí, pokud ano, předmět je odstraněn u prodávajícího a připojen ke kupujícímu s tím že je z předmětu zjištěna jeho cena a následně odečtena kupujícímu a přidána prodávajícímu. Poté jsou aktualizovány `ItemListy` a texty zobrazující jmění postav.

5.9 Dialog pro questy

Dialogové okno obsahuje tři části, oblast pro texty postav, oblast pro informaci o odměně a následně tlačítka, které se používají pro rozhodování a přijetí úkolu. Tato scéna je již instancována na začátku hry a poté pouze zneviditelněna.

Nejdůležitější funkcionalita v této scéně je nastavení textů a poté jejich stránkování. Pro toto slouží `set_dialogue(quest,key,page)` definována ve scriptu `Dialogue.gd`. Tato metoda očekává v argumentu `quest` slovník který odpovídá struktuře JSON souborů, ty jsou používány pro uložení dat o úkolu. Argument `key` je použit jako klíč pro texty z questu a `page` informuje na jaké aktuální stránce se hráč nachází. Jelikož jsou dialogy uloženy pomocí listu a tím pádem je možno relativně jednoduše stránkovat podle indexace. Ve slovníku jsou vyhledány data pomocí 3 klíčů : `text`, `characterName` a `title`. Poté se již využije `key` ke zjištění jaký konkrétní text z těchto dat se má přesně nastavit - což je vlastně zjištění stavu postavy se kterou se komunikuje. Pro zjištění stránky se vybere prvek v listu daného textu který byl vybrán podle klíče, tedy výsledný zobrazený text se získá tímto způsobem: `quest["text"][key][page]`. Pokud je hráč na první nebo poslední stránce, je také zneviditelněno tlačítko pro přechod na předchozí nebo následující stránku. Jakmile je hráč na poslední stránce má možnost začít interagovat s tlačítky v okně.

5.10 Interakce

Interakce jsou řešené připojováním individuálních uzlů nebo spíše scén na jednotlivé charaktery. Každá specifická interakce je děděna od objektu `Interaction.gd` se scénou `Interaction.tscn`. Ty jsou poměrně primitivní, protože jejich obsah je pouze předpřipravená kolizní oblast. Po zdědění tohoto objektu se připojí na oblast signál `body_entered` ten je emitován vstoupí-li někdo do této oblasti a je vykonávána určitá funkcionalita v závislosti na daném typu interakce, signál `body_exited` je naopak emitován po ukončení kolize a tím se volá proces, jež ukončí možnost interakce. Typů interakcí ve hře je několik: schopnost obchodu, interakce pro úkoly, interakce kde při vstupu je kolidovaný objekt sledován nebo agresivní interakce při které začne charakter vykonávat útok po dobu kolize.

5.11 Designer

Designér ve hře umožňuje uživatelům a programátorům vytvářet vlastní předměty do hry. Technicky je zamýšlen tak, aby dokázal přidávat zdrojové soubory a ty poté vkládal do správného umístění podle hierarchie projektu. Další vnitřní funkcionalitou je parsování scriptů kde se vyhledávají proměnné. Proměnné jsou dále nahrazovány požadovaným kusem kódu. Nové soubory se tvoří převážně přes předpřipravené šablony, jejichž obsah se nahradí právě z dat získaných od designéru.

Upravitelných vlastností v designéru je několik, ty jsou také závislé na konkrétním typu předmětu. Obecně po přidání vlastní textury lze upravovat předmět jako takový (root předmětu), poté oblast která indikuje kde mohu předmět zvednout ze země a následně texturu. V každém z těchto prvků lze provádět 3 operace a to je rotace, škálování a pozicování. U tvorby zbraně jsou další editační prvky a tím je kolizní oblast – zde je možnost zase všech operací a také výběr typu útočné dráhy, po které se zbraň posune při útoku postavy.

5.11.1 Proměnné

Pro to aby mohl designér vkládat nějaké informace do souborů je potřeba určit místo kam se má vložit daná informace. Toto je provedeno určitou sekvencí znaků kde součástí těchto sekvencí je název proměnné konkrétně `#$(NAZEV)`. Pokud je tedy vyhledávané místo s názvem `VARIABLE_1` poté se všechny místa kde se vyskytne `#$(VARIABLE_1)` nahradí stejnou informací. Vzhledem k tomu že se nahrazují jak scény tedy .tscn soubory tak Godot scripty .gd je potřeba začínat hashtagem, důvod tohoto znaku je aby se ve scriptovacím souboru tato "nežádoucí" proměnná pro Godot chovala jako komentář a aplikace tak mohla být v pořádku kompilovatelná a spustitelná. Je samozřejmostí že toto nahrazování může být využito i v jiných typech souborů.

5.11.2 Šablony

Šablony jsou soubory, které obsahují předpřipravený kus kódu - může to být Godot script nebo Godot scéna. Nejdříve se do šablony vloží informace z designéru a poté se již vyplněná šablona uloží do souboru nebo vloží do jiného existujícího scriptu který tento kód využije. Šablony svými proměnnými určují jaké informace zde budou uloženy - obvykle to je pozicování, škálování a rotace na upravitelné prvky ale také odkaz na texturu. Zde je příklad fragmentu šablony pro Godot scénu která je použita na tvorbu zbraně. Ukázka fragmentu pro šablonu:

```
[sub_resource type="RectangleShape2D" id=2]
extents = Vector2( #$(WIDTH_COLLISION), #$(HEIGHT_COLLISION) )

[sub_resource type="CircleShape2D" id=3]
radius = #$(RADIUS_PICKABLE)

[node name="#$(NAME)" instance=ExtResource( 1 )]
position = Vector2( #$(X_POS_ITEM), #$(Y_POS_ITEM) )
rotation = #$(ROTATION_ITEM)
scale = Vector2( #$(X_SCALE_ITEM) , #$(Y_SCALE_ITEM) )
script = ExtResource( 3 )

[node name="AnimatedSprite" parent="." index="0"]
frames = SubResource( 1 )
```

```
position = Vector2( #$(X_POS_SPRITE), #$(Y_POS_SPRITE) )
rotation = #$(ROTATION_SPRITE)
scale = Vector2( #$(X_SCALE_SPRITE), #$(Y_SCALE_SPRITE) )

[node name="DamageArea" parent="." index="1"]
collision_layer = 0
collision_mask = 0
```

5.11.3 Tvorba

Samotný postup při tvorbě je následovný: uživatel vybere jaký typ předmětu chce vytvářet. Poté vyplní jméno, cenu a další informace závislé na typu (pro brnění nebo helmu je to hodnota armor, pro zbraň naopak damage). Následně uživatel nahraje texturu předmětu a může začít s editací vlastností. Helma a zbraň jsou na charakteru ve hře viditelné tudíž je upravování těchto itemů v designéru taktéž na charakteru. V případě brnění není charakter potřeba, jelikož je viditelný pouze je-li odhozen na zem.

Po stisknutí tlačítka Create je zavolána metoda `_on_pressed_create_button()` ta má poté různorodé chování v závislosti na vytvářeném předmětu. V případě zbraně je jako první vytvořen adresář, následně je šablona jak pro Godot script tak pro Godot scénu parsována a jsou jim nahrazovány proměnné. Tuto práci obstarává metoda `rewrite_variable(variable,value,template_type)`, následně jsou vyplněné šablony vloženy do nového adresáře a také je zde připojena textura v podadresáři textures. V případě helmy nebo brnění je definice mírně pozměněná protože zde není potřeba vytvářet nový adresář. Tyto předměty mají jeden společný adresář s tím že jejich šablony jsou vkládány do již existujícího scriptu `Helmet.gd` nebo `BodyArmor.gd`. V takovém případě je poté ve hře instancován sice jeden a ten samý objekt, ale následně je ještě zpětně inicializován podle metody `item_init(type)` kde argument type určí jaká konkrétní helma má být použita a jsou nastaveny patřičné vlastnosti.

Uživatel který chce svoje předměty používat ve hře musí nově vytvořené zdrojové kódy sestavit do hry. Sestavení se provede otevřením složky se hrou v Godot interface a poté re-exportem patřičného binárního souboru. Pokud re-export není nutný, stačí pouze spustit v Godotu hru tlačítkem F5, tak je hra s novými předměty sestavena a ty jsou dostupné v lokaci tržiště u neznámého obchodníka.

Kapitola 6
Síťová komunikace ve hře

6.1 Myšlenka multiplayerové části hry

Multiplayerový prvek ve hře je implementován pro lokaci Arena která slouží pro spojení více hráčů v jedné lokální síti. Hráč může do této lokace vstoupit za účelem volného vydělávání peněz pro nákup předmětů. Za každou výhru ve skupinovém boji jsou tedy získávány peníze, výherce je právě ten hráč který zůstane na živu až do konce boje. Jakmile hráčův hrdina vstoupí do lokace Arena jsou mu dočasně odebrány všechny předměty které vlastní v single-playerové části hry, avšak jeho jmění mu zůstává. Za vydělané peníze si může koupit předměty se kterými poté bude bojovat proti ostatním hráčum na lokální síti. Jakmile se aréna opustí jsou multiplayerové předměty které byly nakoupeny pro boj v aréně odstraněny a zpětně se do inventáře nahrají původní předměty - částka hráče která byla utracena za předměty v aréně už je nevratná. Případná výhra za skupinový souboj se počítá podle koupených předmětů v aréně. Čím lepší předměty hráč použije tím větší je šance na výhru proti ostatním, to má za následek zmenšení odměny pro výherce.

Multiplayer je implementován v lokální síti na bázi client-server. Při přepnutí do multiplayerové části si hráč nejdříve zadá přezdívku která bude využita při odlišení hráčů v lobby a samotné hře. Poté je potřeba zvolit roli z výběru klienta nebo serveru. Pokud se hráč rozhodne být server, má možnost vybrat jméno serveru a počet hráčů ve skupině. Jakmile se připojí dostatečné množství hráčů, je zahájena hra ze které vyjde pouze jediný výherce. Pokud místo vytvoření serveru se hráč rozhodne připojit k již vytvořené čekající hře je v roli klienta a nerozhoduje o žádném nastavení serveru.

6.2 Multiplayerové možnosti v godotu

Godot nabízí několik možnosti v rámci síťování a to je RPC (remote procedure calls) - v tomto případě obě možnosti v transportní vrstvě tedy UDP i TCP, nízkoúrovňové posílání paketů přímo do sítě, WebSocket v případě her v prohlížeči a podobně. I přesto že Godot nabízí nízkoúrovňovou komunikaci, tak je často potřeba vynaložit velké množství úsilí pro synchronizaci hry manuálně, tato práce je nevyhnutelná například u vlastní implementace na backendové části serveru. V drtivé většině případů je lepší využít již připravené vysokoúrovňové API, které nám usnadní práci i přes mírnou ztrátu kontroly na nižší vrstvě. Godot využívá mid-level objekt `NetworkedMultiplayerPeer`. Není vytvořen pro přímé instancování, ale je tvořen pro potomky které již definují chování sítě. Je ještě dobré zmínit objekt `PacketPeer` který, je rodič třídy `NetworkedMultiplayerPeer` a poskytuje serializaci, posílání a získávání dat. Konečnými implementacemi pro správu sítě jsou již `NetworkedMultiplayerEnet`, `WebRTCMultiplayer` a `WebSocketMultiplayerPeer`. Ale pro naprostou většinu případů je přímé používání těchto objektů nedoporučené, jelikož engine nabízí vyšší možnosti jak s těmito objekty komunikovat.

6.2.1 Komunikace mezi hráči

Herní engine Godot má velmi příjemný typ komunikace na vyšší úrovni a tím je vzdálené volání metod. Tato možnost je v zásadě použitelná na kterémkoliv uzlu ve hře. Pro kterýkoliv prvek jsou tedy definovány následující metody

- `rpc(function_name, optional_args)`
- `rpc_id(peer_id, function_name, optional_args)`
- `rpc_unreliable(function_name, optional_args)`
- `rpc_unreliable_id(peer_id, function_name, optional_args)`

První metoda v seznamu zavolá tu funkci která se určí v argumentu a dále je možné do ní předat argumenty, druhá vykonná obdobnou akci, avšak je zde možnost nastavit pro kterého klienta se má volání vykonat (v Godotu má každý klient nebo server svůj identifikátor). Zbylé dvě metody jsou identické s předchozími s rozdílem že není zaručeno zavolání jelikož je použit protokol UDP. Taktéž je možné nastavovat přímo proměnné na daném ulzu s následujícími metodami

- `rset(variable, value)`
- `rset_id(peer_id, variable, value)`
- `rset_unreliable(variable, value)`
- `rset_unreliable_id(peer_id, variable, value)`

Informace na kterém uzlu se má vyžadovaná metoda vzdáleně zavolat je určena podle cesty uzlu. To znamená že pokud je uzel s názvem Player připojen na lokaci Arena tedy má například cestu `/Arena/Player` tak poté je přesně na této cestě zavolána remote metoda která se předá do rpc funkce. Na každý uzel se může nastavit master peer což značí že daný uzel klient vlastní - a to se může využít na typech vzdálených metod v Godotu. Vzdálené metody (remote methods) mají v Godotu čtyři typy.

- remote: Metoda bude spuštěna pouze vzdáleně.
- remotesync: Metoda bude spuštěna jak vzdáleně tak lokálně.
- master: Vzdálené vykonání metody je jen na klientovi který vlastní node.
- puppet: Vzdálené vykonání metody na klientech kteří node nevlastní.

6.3 Technická implementace

Ve hře je využíván jak transportní protokol UDP tak protokol TCP. Jakmile se hráč dostane do arény, spustí se nezbytné akce pro funkčnost multiplayeru. Jako první je vytvořen objekt `PacketPeerUDP` kterému je nastaven port. Tento objekt se stará o naslouchání na lokální síti, jakmile jsou k dispozici data tak se předpokládá že jsou to právě informace o již existujícím serveru. V první řadě se tyto data vezmou, konvertují z ASCII na string a následně podle separátorů porozdělí a vypíšou se informace o serveru do uživatelského rozhraní. S každým řádkem aktivního serveru je k dispozici tlačítko pro připojení. Pokud se tlačítko aktivuje je instancována scéna `Lobby.tscn` která v případě výběru serveru synchronizuje hráče mezi existujícím serverem. To znamená že připojení hráče na server po instancování Lobby proběhne následovně: nejdříve je vytvořen objekt pro správu sítě `NetworkedMultiplayerENet`. Díky tomuto objektu je možnost již přímo začít komunikovat se serverem a informovat ho o připojení. S tím tedy souvisí i definované tři metody: `_on_peer_connected(id)`, `_on_peer_disconnected(id)` a `_server_disconnected()`. Tyto metody jsou relativně samovystvětlující - metoda `_on_peer_connected(id)` se zavolá jak na serveru tak na každém klientu po tom co se

připojí nový hráč, obdobně metoda `_on_peer_disconnected(id)` je volána jakmile se někdo odpojí a v poslední řadě `_server_disconnected()` je využívání k informování klientů pokud se server zruší. Klienti ve fázy lobby tedy čekají než je server informuje o spuštění hry. V rámci serverové části je taktéž využíváno lobby, avšak s tím rozdílem že instancovaný objekt `NetworkMultiplayerENet` je aktivován jako server pomocí jeho implementované metody `create_server(server_port, max_players)`. Dále je vytvořen objekt pro vysílání UDP paketů do sítě. Tento objekt je využit k broadcastingu s informací že je k dispozici nový server. UDP pakety jsou vysílány po určitých intervalech. Data v těchto paketech mají formát jednoduchého řetězce, ten je rozdělován pomocí střednítek. Zahájení hry proběhne po aktivaci tlačítka start na serveru. Tato aktivace způsobí že je na každém klientovy zavolána metoda `instancing_player` pomocí RPC (remote procedure calls). Ta má za úkol vytvořit hrdiny hráčů na klientově straně a informovat zpět server o ukončení instancování. Na tuto zpětnou informaci se použije zase RPC s tím rozdílem že je nastaven argument id = 1 což značí že metoda je zavolána pouze na serveru. Vzhledem k tomu, že si server uchovává počet připojených hráčů, je schopen tedy zjišťovat zda jsou všichni hráči již připraveni. Pokud je zjištěno že ano, jsou zavolány na hráčovských instancích pomocí RPC volání metody `network_set_postion`, což je prvotní nastavení pozice na mapě a poté `network_set_items`, které slouží pro prvotní nastavení předmětů. Hráči v této fázi se již vidí v aréně a pomocí RPC metod definovaných na jednotlivých instancích jsou schopny vzájemné interakce.

Kapitola 7
Testování

Pro testování aplikace bylo vybráno pět účastníků s různými informačními vědomostmi a zkušenostmi s hraním her. Těmto učastníkům jakožto testerům byla poskytnuta sestavená aplikace a dotazník. Dotazník obsahoval tabulku s úkoly které má tester splnit a napsat zpětnou vazbu, poté je obsahem dotazníku větší úkol který vyzve pro průchod příběhem. V průběhu tohoto příběhu má tester zapisovat problémy a nejasnosti se kterými se potkal. Testerům byla předložena otázka jak často hráli nebo hrají hry a poté dotaz ohledně jejich informačních dovedností. Informační dovednosti jsou mapovány tabulkou.

Testování probíhalo pouze na platformě PC a to buď s operačním systémem Windows nebo Linux. Bylo potřeba tedy sestavit dvě spustitelné aplikace které jsou s těmito systémy kompatibilní. S jinými systémy aplikace nebyla vyzkoušena i přestože herní engine poskytuje sestavení aplikace například pro webový prohlížeč tak některé funkcionality by v tomto prostředí mohli přestat fungovat jako je režim multiplayeru protože hra využívá RPC volání metod místo WebRTC nebo WebSocket.

Znalosti	Popis
Základní znalosti	Převažuje pouze práce v prohlížeči.
Mírně pokročilé znalosti	Znalost klasických uživatelských programů.
Středně pokročilé znalosti	Dobrá znalost programů a uživatelského prostředí.
Pokročilé znalosti	Velmi dobrá znalost uživatelského rozhraní, uživatel zná informační pojmy jako HTML apodobné.
Programátorské dovednosti	Znalost různých informačních principů, schopnost vyvíjet software.

Tabulka 7.1. Popis informačních znalostí.

Účastník	Informační znalost	Zkušenosti s hrami
Tester 1	Programátorské dovednosti	PC hry nehraje.
Tester 2	Mírně pokročilé znalosti	PC Hry hrával příležitostně.
Tester 3	Mírně pokročilé znalosti	PC hry hraje příležitostně
Tester 4	Základní znalosti	PC Hry nikdy nehrál.
Tester 5	Pokročilé znalosti	PC hry hraje často.

Tabulka 7.2. Vybraní testeři.

7.1 Testování funkcionality

Pokud tester zaznamenal obtíže sepsal tyto problémy do tabulky, v případě bezproblémového splnění úkolu nechal zpětnou vazbu prázdnou. Jednotlivé vyplněné dokumenty se zpětnou vazbou testerů jsou přídány v příloze tohoto dokumentu. Z dotazníků vznikla

Úkol	Počet připomínek nebo prolémů
Přijmout první quest	1 / 5
Vyzkoušet pohyb postavy (WASD)	1 / 5
Sebrat předmět ze země	2 / 5
Otevřít inventář (tlačítko I) a nasadit předmět	2 / 5
Vyzkoušet útočit se zbraní	4 / 5
Zajít na tržiště	1 / 5
Otevřít obchod (tlačítko T) a koupit helmu	3 / 5
Nasadit helmu a poté ji odhodit na zem	2 / 5
Nechat se od někoho zranit, ale nezabít	2 / 5
Najít nebo koupit léčivý lektvar	1 / 5
Doplnit životy lektvarem	1 / 5
Najít nebo koupit pochodeň	0 / 5
Nasadit pochodeň	0 / 5
Vstoupit do lokace podzemí a vyzkoušet pochodeň	0 / 5
Nalezněte v pravé části vesnice dvě postavy s meči.	0 / 5
Pokuste se jednu z nich zabít nějakým předmětem.	1 / 5
Vstoupit do arény.	0 / 5
Vytvořit server a klienta.	2 / 5
Pokuste se zabít postavu v aréně.	0 / 5

Tabulka 7.3. Souhrn potíží při testování.

souhrná tabulka která zobrazuje nejčastější problémy u jednotlivých úkolů které měl tester splnit. Poté byly z následující tabulky identifikovány problémové části funkčnosti hry které byly po procesu testování analyzovány a opravovány.

7.2 Testování příběhu

Testeři museli procházet příběh hry od začátku do konce, individuální problémy v tomto průchodu jsou zaznamenány v přílohách. Na základě těchto problémů byla vypracována tabulka nejproblematičtějších částí hry. Tyto úkoly byly zpětně opraveny a upravovány tak aby nečinili příliš velké obtíže.

Úkol	Popis problému
Introduction	Close způsobuje nemožnost vrácení do příběhu.
Gerald Offer	Tlačítko Reject zbytečné.
	Po splnění se občas načte špatný text.
The Stolen Cape	Tlačítko Reject zbytečné.
	Goblini neútočili, měli by.
Crazy Skeleton	Tlačítko Reject zbytečné.
Pitcher	Džbán rozlit, příliš obtížné.
Skeleton Fiancee	Pokud ukol rychle dokončím, snoubnenka nepřijde.
Second Part Of Key	Text ukolu nenačten.
Meet With The Warrior	Text přetekl.
Dangerous Goblin	Tlačítko accept nereaguje.
	Goblin se nechová agresivně.
Rare Diamonds	Text úkolu přetekl.
Find princess	Text úkolu nenačten.
Burried Entrance	Vysoká pravděpodobnost uivíznutí hráče kvůli sledujícím postavám.

Tabulka 7.4. Potíže při testování příběhu.

7.3 Nalezené vážnější chyby a nedostatky

- Pronásledující postava mi zablokuje možnost změnu lokace pokud odejdu jinam.
- Dialogové okno multiplayeru se nezavírá po odchodu z Arény.
- Úkoly obsahují tlačítka která nefungují.
- Někdy nereaguje tlačítko Accept.
- Zabití postav spojených s příběhem způsobuje nemožnost dohrání.
- Texty u úkolů přetékají a nejdou přečíst.
- Na některých PC podivná synchronizace hráčů v multiplayeru.
- Reset hry není.
- Zaseknutí hráče kvůli pronásledujícím postavám.
- Nesmrtelnost po nasazení nevybalancovaného vybavení.

Kapitola 8
Závěr

Tato bakalářská práce odhalila vývoj jednoduchého RPG titulu v herním enginu Godot. Obsah tohoto dokumentu se tedy prvotně zabýval myšlenkou, historií tedy počátkem a vznikem titulu Role-play her a byly popsány různé typy - jak stolní hry, videohry tak kostýmové. Následně byl představen koncept, vlastnosti a příběh hry. Tyto témata byly zpracovány tak aby odpovídaly běžnému standardu RPG v herním průmyslu. To znamená že bylo do hry implementováno několik běžně známých standardních prvků, které se obvykle ve hrách tohoto typu vyskytují - příkladem může být trade-system, inventář, levlovací systém a podobně. Další důležitou informací bylo popsání přidané hodnoty pro samotnou aplikaci a tím je rozšíření které se do hry implementovalo pro rychlejší vývoj a možnost rozšířit hru bez znalosti programování. Následně byl vyobrazen příběh ve vývojových diagramech pro lepší přehlednost.

Nezbytnou součástí při vývoji je výběr správného herního enginu. Tomu bylo věnováno několik stránek tohoto dokumentu, kde se popsali a porovnali populární a často používané herní enginy. Z tohoto seznamu se vybral Godot Engine který se zdá být jako nejlepší kandidát pro 2D RPG hry. Také byly v krátkosti popsány některé kritické vývojové mechanismy jako je reprezentace uzlů nebo signály které měli uvést čtenáře aspoň velmi hrubě do problematiky Godotu.

Stěžejní částí této práce je téma “Vývoj a Herní mechanismy”. V tomto tématu byl do podrobnosti popsán celkově vývoj aplikace rozdělen do několika nezávislých částí. Jako první se popsala hierarchie souborů a adresářů v projektu hry, která měla ukázat myšlenku komponentního systému. Následně části lokace, předměty, herní postavy, questy, problematika GUI, interakce, designér a další nezbytné věci. Každá část je probrána z technické stránky tedy implementace. Vysvětlení myšlenky multiplayerového prvku a informace o její implementaci se zabývá kapitola 6.

V závěru bylo provedeno testování, které se snaží poukázat na slabé stránky aplikace avšak je nutno si uvědomit že herní vývoj je obvykle relativně komplexní a nelze pokrýt všechny problémy a chyby.

Literatura

[1] Role playing game [online]. Wikipedia. Dostupné z : https://en.wikipedia.org/wiki/Role-playing_game

[2] Andrew Rilstone. Role-Playing Games: An Overview [online]. Archived 2000-08-23 at the Wayback Machine. Dostupné z: https://web.archive.org/web/20000823050742/http://www.rpg.net/oracle/essays/rpgoverview.html.

[3] Laycock, Joseph. Chapter 1: The Birth of Fantasy Role-Playing Games. Dangerous Games: What the Moral Panic over Role-Playing Games Says about Play, Religion, and Imagined Worlds. University of California Press. pp. 31–50 ISBN 978-0520960565

[4] Martell, Carey. Interview with the creators of dnd (PLATO) [online]. RPGfanatic. Dostupné z: https://www.rpgfanatic.net/our-games-wiki-style-guide

[5] Trefonas, Peter. Dungeons and Dragons PDF [online]. CLOAD. Dostupné z: www.gametronik.com/site/rubriques/tandy/FAQs/CLOAD%20Magazine%201980-05%20(1980)(CLOAD%20Magazine%20Inc).pdf

[6] Barton, Matt. The History of Computer Role-Playing Games Part III: The Platinum and Modern Ages (1994–2004) [online]. Gamasutra. UBM Tech. Dostupné z: https://www.gamasutra.com/view/feature/1571/the_history_of_computer_.php

[7] Vestal, Andrew. The History of Console RPGs: Dragon Quest III [online]. GameSpot. Dostupné z: https://www.web.archive.org/web/20130203110651/http://uk.gamespot.com/features/vgs/universal/rpg_hs/nes7.html

[8] Kim, John. `Narrative` or `Tabletop` RPGs [online]. Archived from the original on 2008-08-29. Dostupné z: https://web.archive.org/web/20080829174633/http://www.darkshire.net/%7Ejhkim/rpg/whatis/tabletop.html

[9] Markus Montola, Jaakko Stenros. We Lost Our World and Made New Ones: Live Role-Playing in Modern Times. Playground Worlds. ISBN 978-952-92-3579-7 Ropecon. Také dostupné z : https://nordiclarp.org/w/images/c/c3/2008-Playground.Worlds.pdf

[10] Tychsen, Anders; Hitchens, Michael; Brolund, Thea; Kavakli, Manolya. The Game Master. The Second Australasian Conference on Interactive Entertainment. ISBN: 978-0-9751533-2-1 Creativity and Cognition Studios Press. Dostupné také z: https://dl.acm.org/doi/10.5555/1109180.1109214

[11] Introduction to SDL 2.0 [online]. wiki.libsdl.org. Dostupné z: https://wiki.libsdl.org/Introduction

[12] Posted by SoloGameStudios. Comparision of Game Engines 2020 [online]. IndieGameDev. Dostupné z:
https://indiegamedev.net/2020/02/11/comparison-of-game-engines-2020

[13] Murphy, Kevin. Unity Game Engine Review [online]. Game Sparks Technologies. Dostupné z:
https://www.gamesparks.com/blog/unity-game-engine-review

[14] How game engines create great 2D games [online]. Unity Technologies. Dostupné z:
https://unity.com/how-to/beginner/unity-good-2d-development

[15] Unreal Engine Review: Pros, Cons and Suitability [online]. newgenapps.com. Dostupné z:
https://www.newgenapps.com/blog/unreal-engine-review-pros-cons-suitability

[16] Drake, Jeff. 15 Great Games That Use The Unreal 4 Game Engine [online]. TheGamer. Dostupné z:
https://www.thegamer.com/great-games-use-unreal-4-game-engine

[17] Bryan, W. Unity vs Godot: Game Engine Show Down [online]. gamedesigning.org. Dostupné z:
https://www.gamedesigning.org/engines/unity-vs-godot/

[18] David Vinciguerra, Andrew Howell. What is GameMaker: Studio and Who Uses It? The GameMaker standard. ISBN 978-1-317-51469-5 CRC Press. Dostupné z:
https://books.google.cz/books?id=dM-9CgAAQBAJ&pg=PA3&redir_esc=y#v=onepage&q&f=false

[19] What are the best game engines ? [online]. slant.co. Dostupné z:
https://www.slant.co/topics/991/~best-game-engines#10

[20] Juan Linietsky, Ariel Manzur and the Godot community. Godot Docs – 3.2 branch [online]. Godot Engine. Dostupné z:
https://docs.godotengine.org/en/stable

Příloha A
Zkratky

RPG	Role-playing game - Hra na hrdiny.
LARP	Live action role-playing game - Typ RPG hry kde účastníci jsou přímo lidé.
MMORPG	Massively multiplayer online role-playing game - Počítačové RPG s možností multiplayeru.
NPC	Non-playable character - Postava která není ovládána hráčem.
SDL	Simple DirectMedia Layer - Multiplatformní multimediální knihovna poskytující nízkoúrovňový přístup k audiu, klávesnici, joysticku, 2D a 3D počítačové grafice.
SFML	Simple and Fast Multimedia Library - Knihovna poskytující rozhraní pro vývoj her a multimedionálních aplikací.
UE	Unreal Engine - Herní engine.
AAA	Triple-A game - Typ hry která je obvykle vyvýjena velkými společnostmi s velkým rozpočtem.
GUI	Graphical User Interface - Grafické rozhraní pro uživatele které je obvykle vždy viditelné.

Příloha B
Testovací dotazníky

B.1 Tester 1

Úkol	Připomínky a prolémy
Přijmout první quest	-
Vyzkoušet pohyb postavy (WASD)	-
Sebrat předmět ze země	-
Otevřít inventář (tlačítko I) a nasadit předmět	-
Vyzkoušet útočit se zbraní	Jak? Nasadit? Mezerník?
Zajít na tržiště	-
Otevřít obchod (tlačítko T) a koupit helmu	Musím se přiblížit k prodejci!
Nasadit helmu a poté ji odhodit na zem	U kterého prodejce?
Nechat se od někoho zranit, ale nezabít	Health a Mana špatně viditelné.
Najít nebo koupit léčivý lektvar	Problém s rozlišením many a lektvaru.
Doplnit životy lektvarem	-
Najít nebo koupit pochodeň	-
Nasadit pochodeň	-
Vstoupit do lokace podzemí a vyzkoušet pochodeň	-
Nalezněte v pravé části vesnice dvě postavy s meči.	-
Pokuste se jednu z nich zabít nějakým předmětem.	-
Vstoupit do arény.	-
Vytvořit server a klienta.	Co znamená nastartovat server? Tlačítko Join by měl být poblíž Nickname.
Pokuste se zabít postavu v aréně.	-

Tabulka B.1. Zpětná vazba na funkcionalitu, tester 1.

Napište co oceňujete na funkcionalitě.	Napište co se vám na funkcionalitě nelíbí.
Plynulá animace. Ovladatelnost. Úroveň grafických prvků.	Nepřátelé nejsou dostatečně agresivní. Komplikovanější práce s inventářem.

Tabulka B.2. Ocenění funkcionality, tester 1.

B.1.1 Zpětná vazba k průchodu příběhem

Quest s gobliny: Nejasné zadání od Derek: „Potion disappeared…“ Co dělat? Zabít každého goblina? Aha! Goblin není daleko a je zvýrazněn. Goblin zabit. Derek ale není spokojen. Zabil jsem všechny zelené gobliny, posbíral i koupil lektvary a Derek pořád

opakuje „Bring it…“ ale co ? AHA! Než jsem prvního goblina zabil, měl jsem s ním mluvit. Poznatek: Každá postava s rotující svatozáří se dá oslovit. Vysoká pravděpodobnost zaseknutí hráče pokud mě někdo pronásleduje. Podivná synchronizace v Multiplayeru způsobuje divnou chůzi postav.

B.2 Tester 2

Úkol	Připomínky a prolémy
Přijmout první quest	Close = nemožnost pokračovat v příběhu.
Vyzkoušet pohyb postavy (WASD)	-
Sebrat předmět ze země	-
Otevřít inventář (tlačítko I) a nasadit předmět	-
Vyzkoušet útočit se zbraní	-
Zajít na tržiště	-
Otevřít obchod (tlačítko T) a koupit helmu	Málo předmětů.
Nasadit helmu a poté ji odhodit na zem	Nevýrazná záložka pro vybavení.
Nechat se od někoho zranit, ale nezabít	Vybavení je nevybalancované.
Najít nebo koupit léčivý lektvar	-
Doplnit životy lektvarem	Mohu použít lektvar i při plném zdraví.
Najít nebo koupit pochodeň	-
Nasadit pochodeň	-
Vstoupit do lokace podzemí a vyzkoušet pochodeň	-
Nalezněte v pravé části vesnice dvě postavy s meči.	-
Pokuste se jednu z nich zabít nějakým předmětem.	-
Vstoupit do arény.	-
Vytvořit server a klienta.	GUI vypadá jinak než ostatní prvky.
Pokuste se zabít postavu v aréně.	-

Tabulka B.3. Zpětná vazba na funkcionalitu, tester 2.

Napište co oceňujete na funkcionalitě.	Napište co se vám na funkcionalitě nelíbí.
Standardní ovládání postav.	Jednoduché někoho zabít.
Funkcionalita hry.	Sbírám věci místo nákupu.
Několik lokací které se přepínají.	

Tabulka B.4. Ocenění funkcionality, tester 2.

B.2.1 Zpětná vazba k průchodu příběhem

Příběhem sem dokázal projít až do konce. Oceňuji na příběhu že na sebe dobře navazuje a skutečně funguje. Respektive zobrazují se předměty ve hře takové které s questem zrovna souvisí a hra si hlídá například pokud sem někoho skutečně zabil nebo ne či zda sem doopravdy donesl správný počet diamantů atd. Jako nevýhodu považuji že příběhem se dá projít jednoduše bez toho aniž bych nutně musel kupovat nějaký lepší předmět - respektive balanc hry je horší. Questy neměli někde dopsané texty. Close způsobuje na začátku nemožnost vrácení do příběhu. Tlačítka Reject jsou zbytečná pokud nefungují. Někde se občas načte špatný text úkolu. Některé postavy které by se měli bránit se nebrání nebo pomalu.

B.3 Tester 3

Úkol	Připomínky a prolémy
Přijmout první quest	-
Vyzkoušet pohyb postavy (WASD)	-
Sebrat předmět ze země	Jak? Nevýrazný informační panel.
Otevřít inventář (tlačítko I) a nasadit předmět	Složitější práce s inventářem.
Vyzkoušet útočit se zbraní	Jak se útočí ?
Zajít na tržiště	-
Otevřít obchod (tlačítko T) a koupit helmu	Tlačítko T nefungovalo.
Nasadit helmu a poté ji odhodit na zem	-
Nechat se od někoho zranit, ale nezabít	-
Najít nebo koupit léčivý lektvar	-
Doplnit životy lektvarem	Jak dolnit životy ?
Najít nebo koupit pochodeň	-
Nasadit pochodeň	-
Vstoupit do lokace podzemí a vyzkoušet pochodeň	-
Nalezněte v pravé části vesnice dvě postavy s meči.	-
Pokuste se jednu z nich zabít nějakým předmětem.	S helmou mi nikdo nic neubere.
Vstoupit do arény.	-
Vytvořit server a klienta.	GUI vypadá jinak než ostatní prvky.
Pokuste se zabít postavu v aréně.	-

Tabulka B.5. Zpětná vazba na funkcionalitu, tester 3.

Napište co oceňujete na funkcionalitě.	Napište co se vám na funkcionalitě nelíbí.
Vzhled, celkově grafika.	Boj postav je pomalý.
Intuitivní ovládání po vysvětlení.	Postavy nejsou inteligentní.
Funkcionalita hry.	Chaos.

Tabulka B.6. Ocenění funkcionality, tester 3.

B.3.1 Zpětná vazba k průchodu příběhem

Příběhem jsem dokázala projít.

Klady

- Příběh je dlouhý.
- Příběh je udělán napříč různými mapami.
- Rozmanitost questů (jsou zde i takové kde je potřeba například hledat kuličku v kelímcích).

Zápory

- Chyběli texty u ukolů.
- Rozvětvení úkolů není tak výrazné.
- Nevyužila jsem obchod se zbraněmi, zbraně na questy sem si mohla sebrat ze země.
- Pokud jsem nasadila helmu byla jsem neporazitelná.

B.4 Tester 4

Úkol	Připomínky a problémy.
Přijmout první quest	-
Vyzkoušet pohyb postavy (WASD)	-
Sebrat předmět ze země	-
Otevřít inventář (tlačítko I) a nasadit předmět	-
Vyzkoušet útočit se zbraní	Nevím jak útočit.
Zajít na tržiště	Nevím jak zajít na tržiště.
Otevřít obchod (tlačítko T) a koupit helmu	-
Nasadit helmu a poté ji odhodit na zem	Nevím jak vyhodit helmu.
Nechat se od někoho zranit, ale nezabít	-
Najít nebo koupit léčivý lektvar	-
Doplnit životy lektvarem	-
Najít nebo koupit pochodeň	-
Nasadit pochodeň	-
Vstoupit do lokace podzemí a vyzkoušet pochodeň	Nerozpoznaný vstup do podzemí.
Nalezněte v pravé části vesnice dvě postavy s meči.	-
Pokuste se jednu z nich zabít nějakým předmětem.	-
Vstoupit do arény.	-
Vytvořit server a klienta.	-
Pokuste se zabít postavu v aréně.	-

Tabulka B.7. Zpětná vazba na funkcionalitu, tester 4.

Napište co oceňujete na funkcionalitě.	Napište co se vám na funkcionalitě nelíbí.
Hra funguje pěkně.	Není zde manuál jak hru hrát.

Tabulka B.8. Ocenění funkcionality, tester 4.

B.4.1 Zpětná vazba k průchodu příběhem

Úkol pitcher byl příliš těžký, po sebrání se okamžitě rozlil džbán. Pokud sem byla nahatá tak sem byla velmi zranitelná. Chyby v textu, někde se text nenačetl. Někde text přetékal z okna úkolu. Pokud jsem přišla příliš rychle s princeznou až ke kostlivci tak zůstala v polovině mapy.

B.5 Tester 5

Úkol	Připomínky a problémy
Přijmout první quest	-
Vyzkoušet pohyb postavy (WASD)	Nemohu chodit diagonálně.
Sebrat předmět ze země	-
Otevřít inventář (tlačítko I) a nasadit předmět	Není drag and drop.
Vyzkoušet útočit se zbraní	Chybí vysvětlivky jak.
Zajít na tržiště	-
Otevřít obchod (tlačítko T) a koupit helmu	-
Nasadit helmu a poté ji odhodit na zem	-
Nechat se od někoho zranit, ale nezabít	-
Najít nebo koupit léčivý lektvar	-
Doplnit životy lektvarem	-
Najít nebo koupit pochodeň	-
Nasadit pochodeň	-
Vstoupit do lokace podzemí a vyzkoušet pochodeň	-
Nalezněte v pravé části vesnice dvě postavy s meči.	-
Pokuste se jednu z nich zabít nějakým předmětem.	-
Vstoupit do arény.	-
Vytvořit server a klienta.	-
Pokuste se zabít postavu v aréně.	-

Tabulka B.9. Zpětná vazba na funkcionalitu, tester 5.

Napište co oceňujete na funkcionalitě.	Napište co se vám na funkcionalitě nelíbí.
Žádné načítání ve hře.	Nemohu restartovat hru. Nemám pocit času ve hře. Rychlejší pohyb NPC.

Tabulka B.10. Ocenění funkcionality, tester 5.

B.5.1 Zpětná vazba k průchodu příběhem

Zabil jsem důležitou postavu spojenou s questem a potom sem nemohl dokončit příběh. Musel jsem restartovat několikrát hru - nebyl restart příběhu. Nemůžu si zobrazit informace k úkolu. Některá tlačítka fungují divně, reagují až po několikátém stisku nebo nefungují. Některé postavy se nebrání ač by měli.