FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Magitech - Modul interakce hráče s nehráčskými postavami
Student:	Štěpán Vejvoda
Vedoucí:	Ing. Jan Matoušek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Magitech je počítačová hra pro jednoho hráče žánru dungeon crawler, v níž se hráč zapojuje do střetu magie s technologií v procedurálně generovaných prostředích ve světě zabydleném nehráčskými postavami.

Cílem práce je vytvořit modul pro interakci hráče s nehráčskými postavami.

Pokyny pro vypracování:
1) Prozkoumejte podobné systémy v existujících počítačových hrách.
2) Analyzujte potřeby počítačové hry Magitech v souvislosti s interakcemi hráče s nehráčskými postavami.
3) Metodami softwarového inženýrství navrhněte modul pro interakci hráče s nehráčskými postavami.
4) Implementujte prototyp modulu.
5) Podrobte funkčnost modulu důkladnému testování.
6) Shrňte zjištěné výsledky a nastiňte další možnosti vývoje.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 3. února 2022 v Praze.

Bakalářská práce

MAGITECH - MODUL INTERAKCE HRÁČE S NEHRÁČSKÝMI POSTAVAMI

Štěpán Vejvoda

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Matoušek
9. května 2022

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat Ing. Janu Matouškovi za vedení této práce a za pravidelné konzultace, které byly velmi přínosné. Také bych rád poděkoval rodině a přátelům za podporu.

Prohlášení

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. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 9. května 2022 .....

Abstrakt

Tato práce se zabývá vývojem systému interakce mezi hráčem a NPC pro hru MagiTech.
Na základě analýzy her stejného žánru jsou navrženy jednotlivé moduly, které tvoří celý systém.
Práce také obsahuje komentář k implementaci a výsledky testování, kterého se účastnili vývojáři
a hráči.

Klíčová slova nehráčská postava, MagiTech, reputace, interakce, počítačová hra, Unity, izometrické RPG, společník, modul

Abstract

This bachelor's thesis deals with development of a player-npc interaction system. The individual modules are based on analysis of videogames from the same genre. The thesis also includes a commentary on implementation and developer/player testing.

Keywords non player character, MagiTech, reputation, interaction, video game, Unity, izometric RPG, companion, module

Seznam zkratek

NPC	Non Playable Character
ARPG	Action Role Playing Game
RPG	Role Playing Game
MMORPG	Massively Multiplayer Online Role Playing Game
DLC	DownLoadable Content
WoW	World of Warcraft
MaB	Mount and Blade

Kapitola 1
Úvod

Herní průmysl je jedním z nejrychleji rostoucích odvětví informačních technologií. Stejně tak proces softwarového inženýrství je s ním úzce spjat. Při vývoji počítačové hry, ať už malé nebo velké, většinou následují stejné kroky jako u jiných informatických projektů. Začíná se analýzou, ve které se zkoumají existující hry a proč jsou nebo nejsou úspěšné. Vypracovává se návrh, kde vývojáři dávají dohromady, jak bude celý systém propojený a co bude dělat každý modul a jak. Nakonec se provádí implementace, testování a údržba.

Hlavní motivace autora pro tuto práci je, že herní průmysl je mu velmi blízký. Hry byly hlavní důvod, proč autor s programováním začal, a do budoucna by se hernímu průmyslu chtěl věnovat. Ačkoliv byla týmová práce na MagiTechu v rámci předmětu BI-VHS někdy časově a jinak náročná, byla také velmi přínosná a podpořila autorův zájem pro vývoj počítačových her.

Tato práce se zabývá vývojem modulu pro hru MagiTech, která je do detailu popsána níže. MagiTech je hra žánru RPG a pro tyto hry je běžná existence takzvaných NPC. Tento modul se zaměřuje právě na tyto nehráčské postavy a jejich interakce s hráčem.

První kapitola se zabývá analýzou. Popisuje, co je nehráčská postava a jakým způsobem s hráčem může interagovat. Také jsou rozepsány faktory, které tuto interakci ovlivňují a řídí. Část analýzy je věnována zkoumání aktuálních her a jejich systémů interakce s NPC. Nakonec se provádí analýza funkčních a nefunkčních požadavků.

Návrh popisuje řešení problematiky a její následné dělení. Součástí jsou také návrhy obrazovek, které hráč během hry uvidí.

V kapitole o implementaci autor rozvádí, jakým způsobem byly problémy řešeny a jak vypadá výsledný produkt.

Závěr se věnuje míře splnění požadavků a podobu řešení z pohledu autora.

Kapitola 2
Cíl práce

Cílem této práce je vypracovat metodami softwarového inženýrství modul pro interakci hráče s nehráčskými postavami ve hře MagiTech. Analýza zahrnuje detailní rozbor problematiky interakce s NPC ve hrách, průzkum stávajících řešení této problematiky u známějších počítačových her a analýzu funkčních a nefunkčních požadavků pro hru MagiTech. Návrh vypracovává jednotlivé ucelené částí projektu a popisuje, jakým způsobem mají fungovat samy i mezi sebou navzájem. Důležitý je také návrh obrazovek, který hráč během hry uvidí, a s kterými bude interagovat. Cílem implementace je implementovat řešení na základě analyzovaných požadavků a návrhu. Nakonec se provede detailní otestování výsledné práce.

Kapitola 3
Analýza

Než je možné vytvořit kvalitní řešení daného problému, je potřeba v analýze popsat jednotlivé podproblémy. Dále je potřeba zvážit jaké řešení je pro dané podproblémy nejvhodnější a jak ho implementovat. Také je potřeba prozkoumat existující řešení a rozhodnout se, jak by se dala aplikovat pro hru MagiTech.

Analýza představí hru MagiTech a její mechaniky. Dále rozebere nehráčské postavy, způsoby, jakými interagují s hráčem a faktory, které tuto interakci ovlivňují. Poté popíše, jak je interakce s NPC řešena v některých známých hrách. Nakonec definuje jednotlivé funkční a nefunkční požadavky pro modul a pro hru MagiTech.

3.1 MagiTech

MagiTech je počítačová hra žánru ARPG, kde hráč bojuje proti nepřátelům v procedurálně generovaných scénách (dungeonech). Hra vznikla na Fakultě informačních technologií Českého vysokého učení technického v Praze v rámci předmětu BI-VHS, který vyučuje Ing. Radek Richtr, Ph.D.

O vytvoření hry se zasloužil sedmičlenný tým: Roman Španko, Daniel Breiner, Yuri Udavichenko, Jiří Macháček, Martin Slezák, Ivan Desiatov a Štěpán Vejvoda. Hra je stále ve fázi vývoje, a kromě této práce se jí zabývá bakalářská práce Jiřího Macháčka na téma procedurálního generování dungeonů.

Dungeon je herní prostor typický pro hry typu dungeon crawler a v MagiTechu je to jediné místo, kde hráč bojuje s nepřáteli. V MagiTechu jsou Dungeony procedurálně generované.

V mnoha ohledech se MagiTech podobá hrám typu roguelike a dungeon crawler [1], [2], [3].

3.1.1 Příběh hry

Hráč je vystudovaný učeň magie, který je povolán na pomoc vesnici, kde se začaly dít neobvyklé věci. Hráč zjistí, že na vesnici útočí takzvaní roboti a musí pomoci je porazit. Starosta a hospodský mají každý své vlastní zájmy a hráč se rozhoduje, komu je pomůže realizovat.

3.1.2 Herní cyklus

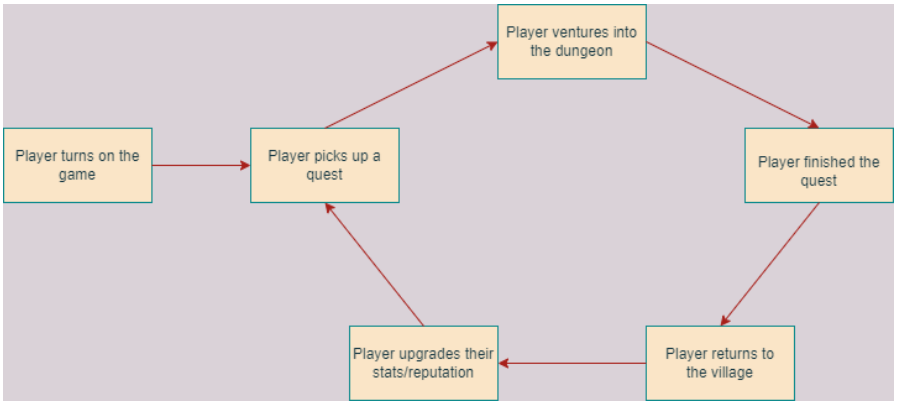
Hra spočívá ve dvou fázích, které se stále střídají. První fáze je ve vesnici, kde hráč potkává a konverzuje s NPC. Tyto postavy mu zadávají úkoly a vylepšují atributy (životy, mana, regenerace, udělené poškození, brnění, štěstí). Ve druhé fázi se hráč vydává do dungeonu (lesu, jeskyně

nebo města), který se generuje na základě aktivního úkolu a jeho parametrech. V dungeonu bojuje s nepřátelskými roboty a sbírá suroviny, které jsou po oblasti rozprostřené. Poté, co splní úkol nebo je poražen, se hráč vrací zpět do vesnice a cyklus se opakuje.

3.1.3 NPC v Magitechu

NPC ve vesnici se dělí na dvě skupiny. Interaktivní NPC (farmář, hospodský, starosta, čaroděj, kněz a kovář) jsou postavy, se kterými si hráč může popovídat a vylepšit si u nich atributy. Každé NPC vylepšuje jeden atribut. Hospodský a starosta zadávají hlavní úkoly. Pokud si hráč vezme třetí hlavní úkol od starosty nebo od hospodského, tak se mu tím zablokuje úkol pro toho druhého. Ostatní interaktivní NPC zadávají vedlejší úkoly. Neinteraktivní postavy s hráčem nijak neinteragují a pouze se pohybují po předdefinovaných trasách.

Obrázek 3.1 herní cyklus MagiTechu



3.2 Nehráčské postavy ve hrách

NPC jsou postavy, se kterými se hráč potkává ve světě, do kterého je hra zařazena. Mohou nějakým způsobem reagovat na hráče, nebo si ho nevšímat. Také mohou mít předem stanovené chování, kterého se drží a toto chování se může změnit, pokud do něj hráč zasáhne. NPC ve hrách tvoří nedílnou část světa, ve kterém se nachází, a hráč má díky tomu větší pocit realističnosti.

Přátelské postavy se často se shlukují ve vesnicích, městech nebo na jiných bezpečných místech a na nepřátelské hráč naráží při výpravách mimo tyto oblasti. V některých hrách je většina NPC nepohyblivá nebo se pohybují po definované trase, takže hráč vždy ví, kde na konkrétní NPC může narazit [4], [5]. V jiných hrách je pohyb postav naopak do jisté míry náhodný, protože se řídí tím, co dělá hráč a ostatní NPC [6].

Nehráčské postavy je možné v některých hrách vždy nalézt a žádným způsobem se nemohou ze hry odstranit. Stejně tak nemohou nové NPC vzniknout [4]. Jindy je životnost postav více dynamická a je tedy možné, aby NPC mizely kvůli akcím hráče nebo postupem herního času, a zase se vytvářely nové.

NPC jsou často hlavní prostředek pro vyprávění příběhu hry. Sice je běžné, aby hráč byl hlavní postava příběhu, ale NPC v něm hrají nedílnou roli. Některé postavy se aktivně snaží bránit hráči v dosažení jeho cíle a jiné se mu v tom snaží pomoci. Ostatní NPC mohou mít svůj cíl, nijak nesouvisející s hráčem, a je na hráči, jestli a jak s nimi naloží [7].

3.3 Interakce s hráčem

Interakce NPC s hráčem je hlavní účel jejich existence ve hrách. Díky interakci se posouvá děj hry. Do interakce patří například plnění úkolů zadanými postavami, obchodování, zvyšování atributů, ale i boj nebo okrádání. Některé hry mají atributy, které reflektují, jakým způsobem hráč interaguje s NPC. Například v Mount and Blade 2: Bannerlord dostává hráč takzvané traits (osobnostní rysy), které mohou být negativní, pokud byl hráč k NPC nemilosrdný, nebo pozitivní, pokud se choval velkoryse [6].

V mnohých hrách jsou faktory, které chování nehráčských postav vůči hráči ovlivňují. Tyto faktory mohou být například reputace (míra oblíbenosti hráče u konkrétního NPC nebo skupiny postav), rozhodnutí, která hráč učinil během hraní (volba zachránit jinou postavu) nebo fáze hry.

V některých hrách jsou NPC, která může hráč přímo ovládat. S těmito postavami se může hráč setkat po rozhodnutí ve hře, nebo tuto postavu ovládá od začátku. Mohou to být příběhové postavy, kterým se hráči rozhodli pomoci, nebo například ochočené zvíře. Ovladatelné postavy tradičně hráči pomáhají v boji, nebo poskytují služby (například oprava předmětů) i mimo místo, kde se tato služba běžně vykonává (kovárna) [4], [8].

3.4 Obvyklé faktory ovlivňující interakci s NPC

3.4.1 Reputace

Reputace měří, jak oblíbený je hráč u konkrétního NPC nebo celé skupiny. Často se měří číselně, tedy například 0–1000, a v některých případech jsou jednotlivé hodnoty reputace ohraničeny označeními, které slovně popisují momentální reputaci, jako například (v angličtině) hated, friendly, revered.

Podle stupně reputace mohou mít nehráčské postavy na hráče odlišné reakce a v některých příkladech se z normálně přátelských NPC stanou agresivní a na hráče začnou útočit. Stejně tak to může fungovat naopak a při získání dostatečně vysoké reputace se hráči odemknou nové možnosti interakce s jinak nepřátelskými NPC. Obecně může reputace ovlivňovat veškerou interakci mezi hráčem a nehráčskými postavami, tedy i dialogové možnosti, reakce na hráče nebo zamykání/odemykání možností vylepšení hráče.

Reputace se může měnit různými způsoby. Běžně se mění na základě rozhodnutí hráče. Konkrétně to může znamenat přijetí úkolu, splnění či nesplnění úkolu, konání přátelské či nepřátelské činnosti, nebo interakce s úplně jinými NPC než ovlivněné (například, pokud to jsou nepřátelé).

Reputace se tedy může změnit přímo, například věnováním dárku nebo splněním úkolu, ale může se změnit i nepřímo. Například postava A má špatný vztah s postavou B a pokud hráč zvýší svou reputaci u postavy A, tak se automaticky zhorší reputace s postavou B.

NPC mohou mít reputaci nejen s hráčem, ale i mezi sebou. Tato reputace se může sama měnit na základě toho, jak spolu interagují jednotlivé NPC nebo skupiny do kterých patří. [9].

3.4.2 Rozhodování hráče během hry

Steve Hoffman ve článku pro Game Developer rozděluje rozhodnutí hráče do devíti kategorií:

1. Hollow decision (prázdné rozhodnutí): žádný opravdový dopad
2. Obvious decision (předem jasné rozhodnutí): není doopravdy rozhodnutí
3. Uninformed decision (neinformované rozhodnutí): hráč nemá dostatek informací

4. Informed decision (informované rozhodnutí): hráč má dostatek informací
5. Dramatic decision (dramatické rozhodnutí): hraje na emoce hráče
6. Weighted decision (vyvážené rozhodnutí): každá možnost má nějaký dopad
7. Immediate decision (okamžité rozhodnutí): má dopad okamžitě
8. Long-term decision (dlouhodobé rozhodnutí): má dopad za nějakou dobu

Jako faktor ovlivňující hráčovu interakci s NPC a světem má pro účely analýzy smysl brát v potaz rozhodnutí, která mají okamžitý nebo dlouhodobý dopad.

Rozhodnutí, které hráč činí během hry, mohou mít pozitivní nebo negativní dopad v budoucnu. Už při vytváření postavy se hráč nepřímo rozhoduje, že některá NPC na něj nebudou reagovat pozitivně, nebo se s ním vůbec bavit nebudou a rovnou na něj zaútočí. Toto je běžné, pokud jsou ve hře jednotlivé frakce, které mezi sebou soupeří, a hráč si jednu z nich musí na začátku vybrat. V některých hrách naopak volby při vytváření postavy nemají žádný vliv na to, jak se ke hráči budou NPC chovat.

V průběhu hry hráč naráží na řadu rozhodnutí, která mohou mít efekt na to, jak je hráč ve hře vnímán ostatními NPC. Pokud například hráč zabije člena rodiny nějakého NPC, nemůže očekávat, že s ním toto NPC bude chtít obchodovat, nebo mu jinak pomáhat. Některá rozhodnutí mohou proti hráči poštvat celé skupiny NPC a může se i stát, že se v určitou chvíli stane pro hráče hra nehratelná. Naopak rozhodnutí ve prospěch postav mohou odemykat hráči nové úkoly, odměny a dialogové možnosti.

Konec hry může být také ovlivněn, nebo přímo určen, rozhodnutími hráče. V některých hrách se hráč na konci dozví, co se stalo se skupinami NPC, kterým pomohl, nebo naopak. [10].

3.4.3 Fáze hry

Fáze hry jsou vzájemně odlišné časové části hry. Jsou důležité pro zachování rozmanitosti v rámci hry. Hráči očekávají, že hra bude mít dostatečné množství fází, protože reálný život je do fází také rozdělen (dospívání, dospělost, stáří...). Každá hra sama určuje, jak jsou od sebe jednotlivé fáze rozlišitelné, a do jisté míry tkví rozlišování fází v perspektivě hráče.

V každé fázi se může chování NPC, a obecně světa, měnit. Pokud se jako fáze bere plynutí času ve hře, kde se postavy rodí a stárnou, tak každé NPC bude reagovat na hráče, kromě dalších faktorů, také na základě svého (a hráčova) věku. Ve hře, kde nová fáze začíná více očividně, například splněním důležitého úkolu, mohou NPC na hráče také reagovat různě, obzvlášť pokud na ně mělo splnění úkolu přímý dopad.

Některá NPC jsou ve hře pouze než nastane určitá fáze nebo se objeví až po jejím začátku. Ve hře se stárnutím je zřejmé, že některá NPC časem nebudou dostupná a jiná zase přibudou. Podobně mohou NPC zmizet kvůli přechodu do jiné fáze [11].

3.5 Systémy interakce s NPC ve hrách

3.5.1 World of Warcraft

World of Warcraft (WoW) je počítačová hra žánru MMORPG vytvořena společností Blizzard Entertainment. Byla vydána v roce 2004 a do dnešního dne se jedná o jednu z nejvíce populárních her svého žánru. Do hry se stále přidávají nové DLC.

3.5.1.1 Náplň hry

Hlavní náplní hry je plnění úkolů od NPC a získávání zkušeností, za které hráč dostává nové úrovně. Během toho hráči prozkoumávají svět, potkávají ostatní hráče a poráží nepřátele. Mimo to se nabízí velké množství dalších činností, ale pro účely této analýzy nejsou relevantní, a tedy tu nejsou uvedeny. Každý hráč si při tvorbě postavy vybírá kromě frakce také povolání (class), která určuje, jakým způsobem bude hrát a hlavně bojovat.

3.5.1.2 Rozdělení NPC

WoW má dvě hlavní frakce, které spolu soupeří, Hordu a Alianci. Mezi těmito frakcemi si hráč musí vybrat, když si vytváří postavu. Stejně jako hráči tak mnoho NPC patří do Hordy nebo Aliance, a tedy volba frakce automaticky vymezí, které postavy budou pro hráče přátelské a které na něj budou útočit. NPC bez frakce jsou takzvaně neutrální a ty útočí na všechny hráče bez rozdílu, pokud se ocitnou v jejich dosahu (tato NPC jsou označena červeně) nebo až tehdy, kdy na ně někdo zaútočí (označena žlutě). Všechna NPC přestanou na hráče útočit, pokud se dostane dostatečně daleko od místa, kde ho napadly [12], [13].

3.5.1.3 Městské NPC

Většina NPC ve městech a ostatních relativně bezpečných oblastech pouze stojí na místě. U těchto postav mohou hráči získávat úkoly (quest givers), nakupovat a opravovat výzbroj (traders/blacksmiths), učit se nové schopnosti (class trainers), učit se profesím (proffesion trainers) a pokud jsou ve městě, tak se ptát strážných na cestu. Na tyto NPC mohou útočit pouze hráči z nepřátelské frakce, ale nemají z toho žádnou reálnou odměnu (až na výjimky). Některé postavy se pohybují po předem dané trase, ale není jich oproti nehybným NPC mnoho.

3.5.1.4 Neutrální NPC

Mimo bezpečné oblasti se většinou vyskytují neutrální NPC. Nejčastěji se pohybují v malé omezené oblasti, do které patří. Některá NPC jsou nepohyblivá, nebo se pohybují po trase, podobně jako městské postavy. Z porážení těchto nepřátel mají tradičně hráči zkušenosti (experience) a předměty. Každé NPC má definovaný seznam předmětů, které z něj mohou hráči získat a šance na získání jednotlivých předmětů. Získané zkušenosti z jejich poražení se odvíjí od jejich síly a úrovně hráče, který je porazil. Pokud je hráč na moc vysoké úrovni, nedostane zkušenosti vůbec.

3.5.1.5 Reputace

Ve WoW je velké množství vedlejších frakcí, se kterými mohou hráči zvyšovat reputaci. Některé spadají pod dvě hlavní frakce a mohou s nimi zvyšovat reputaci pouze hráči z dané frakce. S ostatními může zvyšovat reputaci kdokoliv. Jeden způsob jak zvyšovat reputaci je plnění každodenních úkolů (daily quests). Tyto úkoly zvyšují reputaci po větším množství, ale hráč je může splnit pouze jednou za den. Další způsob je porážení nepřátel dané frakce. Tento způsob je možné využívat do nekonečna, ale každý poražený nepřítel běžně zvýší reputaci pouze o malé množství.

Reputace s každou frakcí je rozdělena do milníků. Po jejich dosažení se hráči odemkne odměna. Běžnou odměnou bývá možnost zakoupit specifický předmět u důstojníka (quartermaster) frakce [4], [14].

3.5.1.6 Pets

Některá povolání mají možnost vlastnit permanentního společníka (Pet). Tito společníci pomáhají hráči při boji a dají se ovládat pomocí ovládacího panelu, na kterém jsou příkazy na aktivaci

Obrázek 3.2 Nahoře: Hospodský ve WoW, vlevo dole: Žluté NPC, vpravo dole: Červené NPC

schopností a příkazy ovlivňující agresivitu společníka. Povolání lovec (Hunter) má jako společníky zvířata (beasts). Aby mohl dané zvíře mít jako společníka, musí ho nejdříve ochočit ve volném světě. Povolání černokněžník (Warlock) má jako společníky démony a může je používat, pokud se naučí konkrétní kouzlo na jejich vyvolání. Povolání rytíř smrti (Death knight) má jako společníka ghůla a aby ho mohl používat permanentně, tak musí mít bezbožnou specializaci (unholy).

Vzácnější, ale zato dostupní pro všechny povolání, jsou NPC poskytující služby jako oprava předmětů mimo město. Tyto postavy se běžně dají vyvolat specifickým předmětem, který hráč najde nebo vyrobí (profese inženýrství (engineering) těchto NPC odemyká celou řadu) [15].

Obrázek 3.3 Nahoře: Záložka reputace ve WoW, dole: Pet a jeho ovládací panel ve WoW

3.5.2 Mount and Blade 2: Bannerlord

Mount and Blade 2: Bannerlord je počítačová hra žánru ARPG od studia TaleWorlds. Byla vydána v roce 2020 a stále má titul *early access*, tedy není dokončena. Hra má kampaň pro jednoho hráče nebo online bitvy pro více hráčů. Pro účel této analýzy se zaměříme na kampaň, jelikož v módu pro více hráčů nejsou NPC.

3.5.2.1 Náplň hry

Hráč začíná kampaň jenom se základní výzbrojí a atributy, které si vybral při tvorbě postavy. Během hry má možnost stavět armádu, obchodovat, plnit úkoly pro NPC, bojovat s nepřáteli, spolupracovat s královstvími, nebo založit nové.

3.5.2.2 Lordi

Během kampaně se hráč potkává s množstvím různých NPC. V každém království jsou lordi, pro které může hráč dělat úkoly. Království má krále, který kromě úkolů umožňuje hráči přidat se k jeho království jako žoldák nebo vazal. Pokud se hráč přidá k nějakému království, tak si tím automaticky znepřátelí nepřátele toho království. Všichni lordi se dají napadnout a pokud nejsou nepřátelští je možné s nimi vyměňovat předměty. Pokud nepřátelští jsou, tak se může hráč pokusit před nimi utéct, nebo je podplatit, aby ho nechali jít.

3.5.2.3 Společníci

V městských hospodách může hráč potkat NPC, kteří se potulují po světě a nejsou vazalové žádného království. Tyto postavy je možné naverbovat do skupiny. Poté je může hráč posílat dělat úkoly, nechat je vytvořit vlastní armádu, poslat je s karavanou a pokud hráč vlastní hrad nebo město, tak je může jmenovat guvernérem daného hradu nebo města [16].

3.5.2.4 NPC ve městech (notables)

Ve městě potkává hráč další důležité osoby (notables), od kterých může nakupovat vojáky. Tyto NPC také někdy vlastní výrobnu (workshop) na specifický produkt. Tyto výrobny se dají kupovat a hráč z nich poté má stálý výdělek. Také může hráč vyslat z této výrobny karavanu, spolu s nějakým svým společníkem, a ta mu bude také vynášet peníze [17].

3.5.2.5 Banditi

Ve volném světě se pohybují skupiny banditů, kteří na hráče útočí, pokud vyhodnotí, že jsou schopni hráče porazit. Pokud má hráč dostatečně velkou armádu, tak mu bandité nabídnou, že se k němu přidají.

3.5.2.6 Reputace (relations)

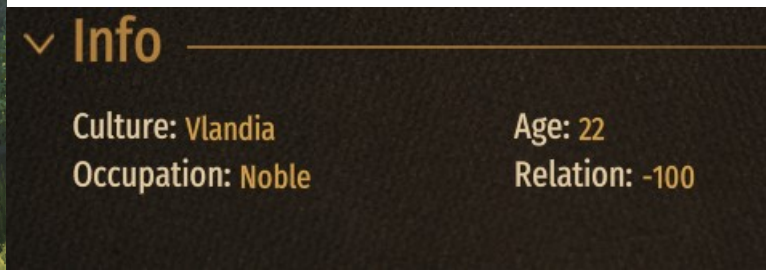
Každý lord a notable má s hráčem vedenou reputaci (ve hře se používá pojem relation – vztah, ale pro účely analýzy to bereme jako stejnou věc). U lordů úroveň reputace ovlivňuje způsob, jakým na hráče reagují. Pokud je reputace nízká, tak je z jejich reakcí snadno poznat, že hráče neradi vidí a pokud je hodně nízká, tak hráče vidí jako nepřítele a útočí na něj. Naopak pokud s nimi má hráč vysoký stupeň reputace, tak mu skládají komplimenty a je vyšší šance, že hráče podpoří.

U notables se kromě změněných dialogů reputace odráží v množství vojáků, které u nich může hráč nakoupit. Například při nízké reputaci si od konkrétního NPC nekoupí žádné vojáky,

nebo jenom ty nejhorší. Naopak při vysoké reputaci získá hráč možnost nakoupit větší množství kvalitnějších vojáků.

Přestože při tvorbě postavy si hráč vybírá kulturu své postavy, tak tato volba nemá efekt na reputaci s NPC, které budou s hráčem kulturu sdílet, nebo naopak.

Obrázek 3.4 Vlevo nahoře: Lord v Mount and Blade, Vpravo nahoře: Kupování vojáků u notables v Mount and Blade, Vlevo uprostřed: Bandita v Mount and Blade, Vpravo uprostřed: Ukázka vztahu v Mount and Blade, Vlevo dole: Traits v Mount and Blade



3.5.3 Stardew Valley

Stardew Valley je RPG z farmářského prostředí. Je vyvíjena Erikem Baronem alias "ConcernedApe"a vydaná studiem Chucklefish v roce 2016. Hra je hratelná v jednom nebo více hráčích, přičemž mezi oběma případy skoro není rozdíl ve fungování hry.

3.5.3.1 Náplň hry

Hráč začíná na zanedbané farmě a musí jí postupně uklidit a začít farmařit. Dále má možnost konverzovat s obyvateli vesnice, dávat jim dárky, rybařit a další činnosti. Mnoho aktivit se hráči odemkne až postupem času.

3.5.3.2 Vesničané

Ve vesnici se pohybují přátelská NPC, se kterými může hráč konverzovat a dávat jim dárky. Pohybují se po předem definovaných trasách a každý má specifické chování, které většinou není nijak ovlivněné hráčem. Některé vesničany má hráč možnost potkat až po nějakém příběhovém momentu. Stejně tak mohou jako důsledek hráčova rozhodnutí konkrétní NPC zmizet [18].

3.5.3.3 Dárky

Většina předmětů ve hře lze věnovat nějakému NPC jako dárek, které ovlivní jeho vztah s hráčem. Každý vesničan má definovanou tabulku, které dárky má a nemá rád. Hráč předem neví, který dárek se komu líbí, ale když ho zkusí darovat, tak se mu odemkne záznam, ze kterého se dá zjistit, jak na dárek NPC reagovalo. Každému vesničanovi lze darovat dárek pouze dvakrát během herního týdne a pokud se při předání strefí do narozenin NPC nebo Vánoc (Winter Star secret gifting event), tak se mu zlepší vztah o více než normálně.

Vztah s NPC je vyjádřen body přátelství (friendship points) a po dosažení určitého počtu se hráči s daným vesničanem/vesničankou odemkne část příběhu, během které je lépe pozná. Hráč má také možnost navázat s nezadaným NPC romantický vztah a později manželství [5], [19].

3.6 Analýza požadavků

Některé požadované funkcionality budou sloužit potřebám uživatele, tedy potenciálního vývojáře, který na Unity projektu bude pracovat a moduly používat při implementaci.

Ostatní požadavky jsou s ohledem na hráče, který bude s moduly interagovat v rámci hraní hry.

3.6.1 Funkční požadavky

3.6.1.1 FG - Modul cechů

3.6.1.1.1 FG1 - cechy Ve hře budou cechy, se kterými bude hráč mít určitou úroveň reputace. Cechy se skládají z vesničanů a s těmito vesničany bude hráč různými způsoby interagovat. Cechy budou mít spřátelené a znepřátelené cechy.

Vývojář bude mít možnost vytvářet nové cechy a jejich členy. U cechů bude určovat spřátelené a znepřátelené cechy.

3.6.1.1.2 FG2 - Dárky Dárky budou předměty, které bude hráč získávat během hraní hry a pokud se je rozhodne věnovat určitému NPC, tak se mu s jeho cechem změní reputace. Hráč bude mít přístup k deníku, ve kterém
je uloženo, který dárek se líbil, kterému NPC. Ze začátku je deník nevyplněný. Pokud hráč změní reputaci s cechem, změní se jeho reputace i s jejími přáteli a nepřáteli.

Modul bude vývojáři umožňovat přidávat nové dárky a určovat, kterému NPC se líbí nebo nelíbí.

3.6.1.1.3 FG3 - Vylepšování Ve hře bude možnost získat suroviny, za které si hráč bude vylepšovat atributy u konkrétních NPC. Cena vylepšení se bude s každou výměnou zvyšovat, aby nebylo pro hráče snadné daný atribut vylepšovat příliš často. Cena bude také ovlivněna reputací hráče s cechem NPC, tedy při nízké reputaci bude hráč vylepšovat za více surovin a naopak.

Vývojář bude mít možnost určovat, které NPC bude poskytovat které vylepšení. Bude měnit kterými surovinami bude hráč platit, kolik bude vylepšení stát a jak bude velké.

3.6.1.1.4 FG4 - Milníky reputace Milníky reputace budou reprezentovány hodnotou reputace s cechem, kterých když hráč dosáhne, tak bude pozitivně nebo negativně odměněn v závislosti na tom, jestli je jeho reputace s cechem vysoká nebo nízká.

Vývojář bude mít možnost přidávat nové milníky a definovat odměnu za jejich dosažení.

3.6.1.1.5 FG5 - Kill squad Každý cech bude mít definovanou takzvanou kill squad, což je skupina nepřátel, kteří se začnou objevovat v dungeonech, pokud má hráč s konkrétním cechem špatnou reputaci. Squady budou tím silnější, čím nižší bude reputace s jejich cechem.

Vývojář bude moci přidávat a upravovat kill squady pro cechy.

3.6.1.1.6 FG6 - Příběhové momenty závislé na reputaci Hráč získá přístup ke specifickým příběhovým momentům, pokud jeho reputace s cechem přesáhne určitý milník.

Vývojář bude moci vytvářet příběhové momenty závislé na reputaci s cechy.

3.6.1.1.7 FG7 - Proměnlivé dialogy závislé na reputaci NPC budou s hráčem komunikovat s ohledem na jakém úrovni reputace se s nimi nachází.

Vývojář bude mít možnost vytvářet dialogy v závislosti na reputaci konkrétních NPC.

3.6.1.1.8 FG8 - User Interface cechů Vztahy mezi cechy a hráčem budou přehledně viditelné pro hráče pomocí nové záložky do existujícího UI.

3.6.1.1.9 F1.9 - User Interface dárků Deník dárků bude hráči dostupný ve formě záložky do, existujícího UI.

3.6.1.2 FS - Modul společníků

Společník je NPC, které následuje hráče a pomáhá mu. Stejně jako hráč má atributy a může bojovat. Každý společník má definovaný typ, který shrnuje jeho možnosti (Tank, damage, healer).

3.6.1.2.1 FS1 - Společníci Hráč bude mít přístup ke společníkům. Společníka bude umět ovládat, aby ho následoval nebo zastavil na místě. Také u něj bude umět nastavovat stupně agresivity. Hráč si bude moci vybrat, které společníky použije do dungeonu ze seznamu dostupných společníků.

Vývojář bude umět vytvářet nové společníky a definovat, jakým způsobem je bude hráč odemykat.

3.6.1.2.2 FS2 - User Interface společníků Seznam dostupných společníků bude viditelný pomocí záložky do existujícího UI. V seznamu budou informace o atributech a typu společníků a bude obsahovat tlačítka na jejich aktivování.

Hráč bude pro každého aktivovaného společníka vidět panel příkazů, kterým bude společníka ovládat.

3.6.1.3 FA - Aplikace pro Magitech

Každý existující cech bude mít definované pozitivní a negativní dopady na hráče v závislosti na dosažených milnících reputace. Milníky vzestupně: Nemesis, Enemy, Unfriendly, Neutral, Friendly, Ally, Beloved.

3.6.1.3.1 FA1 - Cech kovářů Pro všechny negativní milníky platí, že pokud jich hráč dosáhne, tak budou roboti v dungeonu silnější (větší damage a životy).

Friendly: Šance na zrušení damage, které hráč obdrží.
Ally: Předchozí bonus a k němu navíc ohnivá aura, která poškozuje nepřátele kolem sebe.
Beloved: Předchozí bonusy plus jednorázová záchrana života, pokud životy hráče klesnou na nulu.

3.6.1.3.2 FA2 - Cech farmářů Všechny negativní milníky způsobují snížení atributů hráče.
Všechny pozitivní milníky způsobují zvýšení atributů hráče.

3.6.1.3.3 FA3 - Cech čarodějů Unfriendly: Šance na odebrání životů a many při kouzlení.
Enemy: Předchozí mínus plus šance na to, že se kouzlo nepoužije.
Nemesis: Předchozí mínusy, ale s vyššími šancemi.

Pozitivní účinky Nabízejí hráči možnost stavět si vlastní unikátní kouzlo. Každý milník nabízí možnosti modifikace, ze kterých si hráč vybírá.

Friendly:
1) Kouzlo zůstane jak efekt na zemi a když na něj stoupne nepřítel, tak dostane damage.
2) Kouzlo udělí damage nepřátelům v daném rozsahu.
3) Kouzlo udělí damage nepřátelům kolem hráče.

3.6.1.3.4 FA4 - Guilda kněží Když bude hráč v dungeonu, tak bude aktivní "Kolo Bohů". Kolo Bohů bude mít šest efektů, které se budou střídat a každý pozitivně nebo negativně ovlivní hráče.
Negativní reputace s cechem kněží bude způsobovat, že častěji budou aktivní negativní efekty.
Pozitivní reputace naopak způsobí, že častěji bude aktivní pozitivní efekt.

3.6.1.3.5 FA5 - Cech starostů Negativní reputace s cechem starostů způsobí, že když hráč sebere předmět v dungeonu, tak je šance, že se místo předmětu objeví nepřítel. Horší reputace znamená větší šance na objevení silnějších robotů.
Pozitivní reputace s cechem starostů dá hráči možnost získat nového robota společníka. Při každém dosaženém milníku dostane hráč na výběr z nových robotů. Roboti se budou lišit v atributech a typech.

3.6.1.3.6 FA6 - Cech hospodských Negativní reputace s hospodskými způsobí, že při vypití lektvaru má hráč šanci na získání efektu "opití", který ovlivňuje hráčův pohyb a některá kouzla.
Pozitivní reputace umožní hráči postavit si vlastní lektvar. Lektvar bude mít po vypití na hráče pozitivní efekt. Efekt si bude hráč moci vybrat.

3.6.2 Nefunkční požadavky

3.6.2.1 N1 - Použití Unity

Moduly budou použité v systému Unity pro 3D hru jelikož MagiTech je vytvořen v Unity.

3.6.2.2 N2 - Snadná ovladatelnost

S moduly se bude uživateli dobře pracovat a půjde bez obtíží přidávat nové objekty. Pro hráče bude ovládání intuitivní a snadno pochopitelné.

3.6.2.3 N3 - Rozšiřitelnost

V rámci jednotlivých modulů nebude složité přidávat nové implementace využívající stávající rozhraní.

3.6.2.4 N4 - Optimalizace

Hra pro hráče bude běžet plynule.

3.6.2.5 N5 - Znovupoužitelnost

Moduly bude možné znovu použít v jiných Unity projektech.

3.6.2.6 N6 - Uživatelský manuál

Pro vývojáře bude v příloze manuál na práci s moduly.

Kapitola 4
Návrh

Předtím než se začne s implementací, musí se vypracovat návrh. V něm musí být diagramy tříd, doménové modely a modely případů užití. Pro přehlednost se jednotlivé diagramy rozdělily do samostatných modulů. Pokud by byl celý systém popsaný v jednom modelu, byl by příliš nepřehledný.

Hlavní moduly řešící daný problém jsou tři: Vylepšování hráče, Reputace s cechy a Interakce se společníky. Ke každému modulu je přiřazený diagram tříd, doménový model a diagram případů užití. Jednotlivé elementy uživatelského rozhraní, které se budou v práci vytvářet, jsou také navrženy. Konkrétně se jedná o ovládací panely společníků, panel reputace s cechy, panel deníku dárků a panel aktivování společníků.

Diagramy domény, tříd a případů užití pokrývají požadavky týkající se vývojáře i hráče. Na konci je přidaný model případů užití, který se týká nastavování scény v Unity a je pouze pro vývojáře.

Při vytváření návrhu počítáme s tím, že část hry je již dokončena. Tato práce se tedy v některých případech bude spoléhat na hotové části a bude do nich, pokud to bude potřeba, zasahovat se svolením původního autora.

4.1 Vylepšování hráče

Modul vylepšování hráče zahrnuje proces, při kterém hráč sbírá v dungeonech předměty (konkrétně suroviny), které se mu budou ukládat do inventáře a po návratu do vesnice je bude vyměňovat za vylepšení atributů.

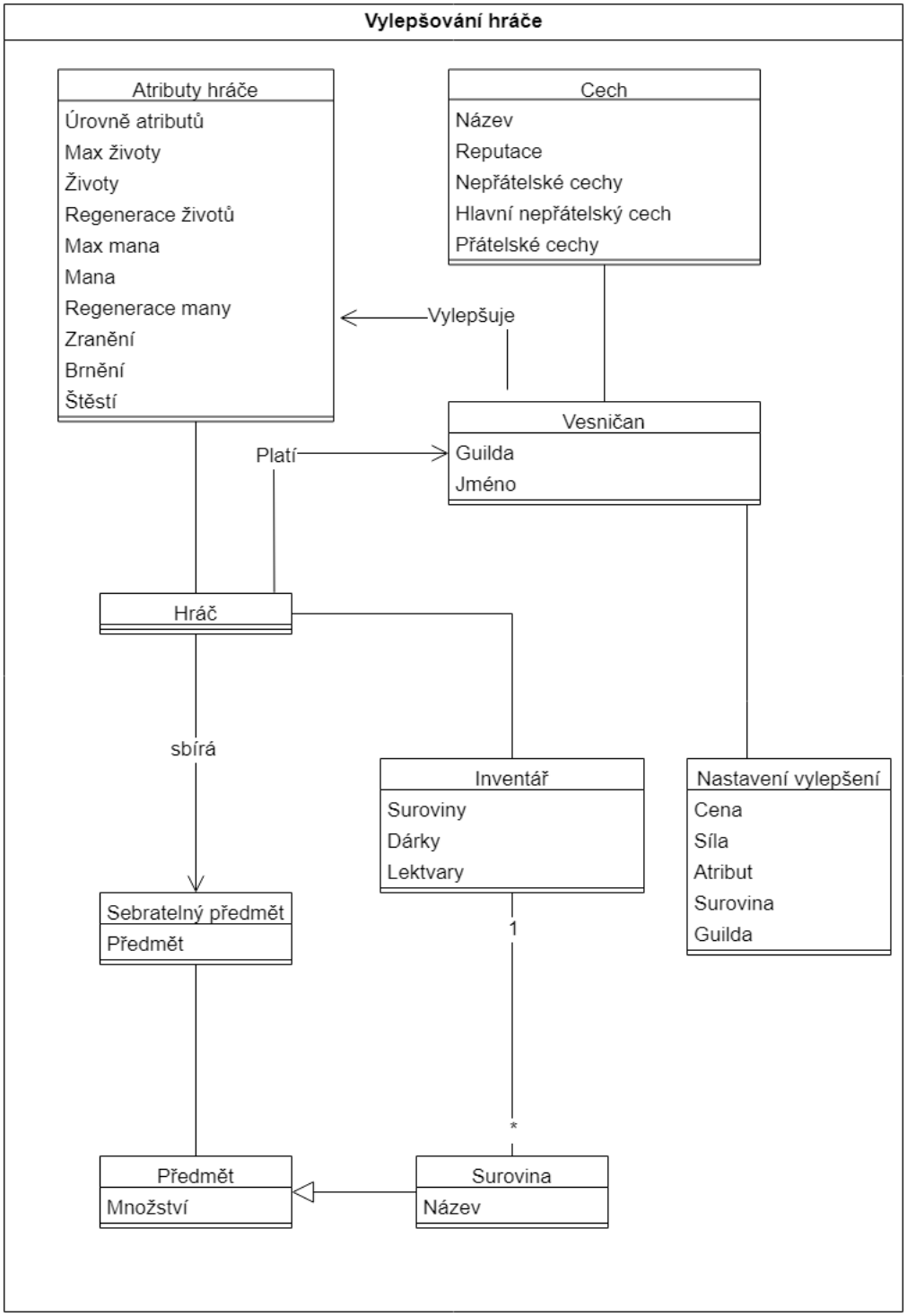
Postava hráče bude mít atributy. Kromě životů, many a podobně se bude ukládat informace o úrovni atributů. Díky tomu se bude moci ovlivňovat cena každého dalšího vylepšení, tedy za vysoké úrovně atributů se bude platit více surovin než za nižší.

Atributy se budou vylepšovat u vesničana, který bude mít informaci o svém cechu. To je potřeba jelikož cena vylepšení se bude také měnit podle hodnoty reputace s cechem.

Každý vesničan bude mít uložené jakým způsobem se u něj bude vylepšovat atribut. Tedy kolik bude stát jaké suroviny, o jaký atribut se jedná a síla vylepšení o kterou se daný atribut zvýší.

Hráč bude suroviny nacházet jako předměty, které může posbírat v dungeonech a po sebrání se mu uloží do inventáře.

Obrázek 4.1 Doménový diagram vylepšování hráče

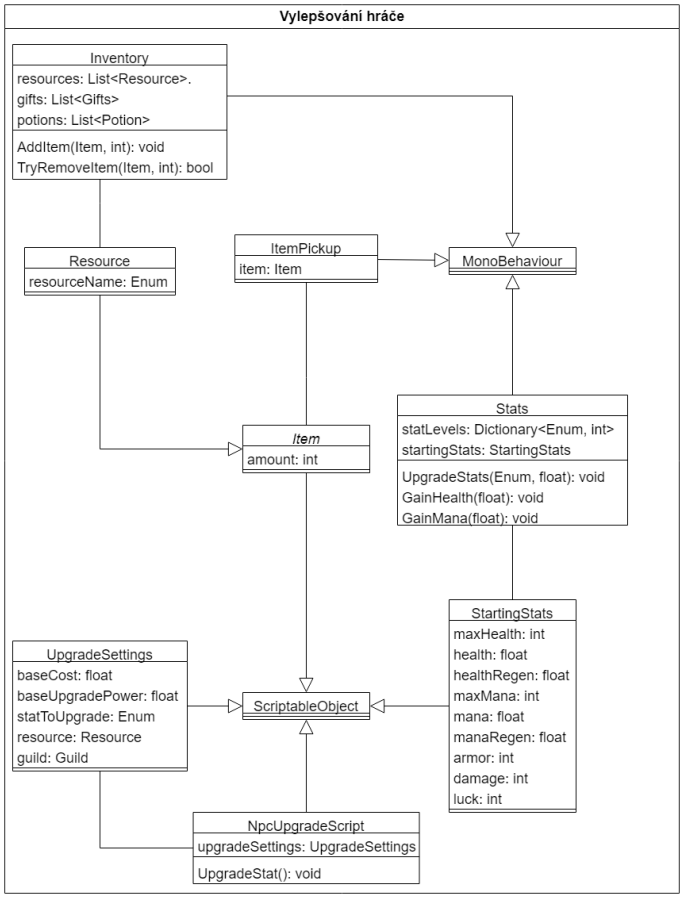


Hráč bude mít v Unity scéně jako komponenty připojené skripty, které se budou starat o různé aspekty jeho fungování. Tyto skripty budou zpravidla dědit ze třídy MonoBehaviour, kterou poskytuje Unity. Ta poskytuje metody, které se používají pro průběh hry.[20]

Některé skripty budou dědit ze třídy ScriptableObject. Tato třída se často používá na ukládání dat, které se nebudou často měnit.[21]

1. Stats je skript, který se bude starat o atributy hráče a práci s nimi. Dědí ze třídy MonoBehaviour.
2. Inventory v sobě bude mít uloženy všechny suroviny, dárky a lektvary, které hráč nasbíral a bude je umět přidávat a odebírat. Dědí ze třídy MonoBehaviour.
3. ItemPickup bude skript, který bude mít jako komponentu objekt reprezentující předmět, který se může sebrat ve scéně. Starat se bude o to, aby se po interakci s hráčem předmět odebral ze scény a přidal do hráčova inventáře. Dědí ze třídy MonoBehaviour.
4. Item je abstraktní třída, ze které dědí Resource, Potion a Gift. Dědí ze třídy ScriptableObject.
5. StartingStats je třída, ve které jsou uloženy startovní atributy hráče nebo nepřátel. Dědí ze třídy ScriptableObject
6. NpcUpgradeScript umí vylepšit atribut hráče a má uložené nastavení vylepšování. Dědí ze třídy ScriptableObject
7. UpgradeSettings je třída, která ukládá informace o vylepšování atributu pro konkrétní cech. Dědí ze třídy ScriptableObject

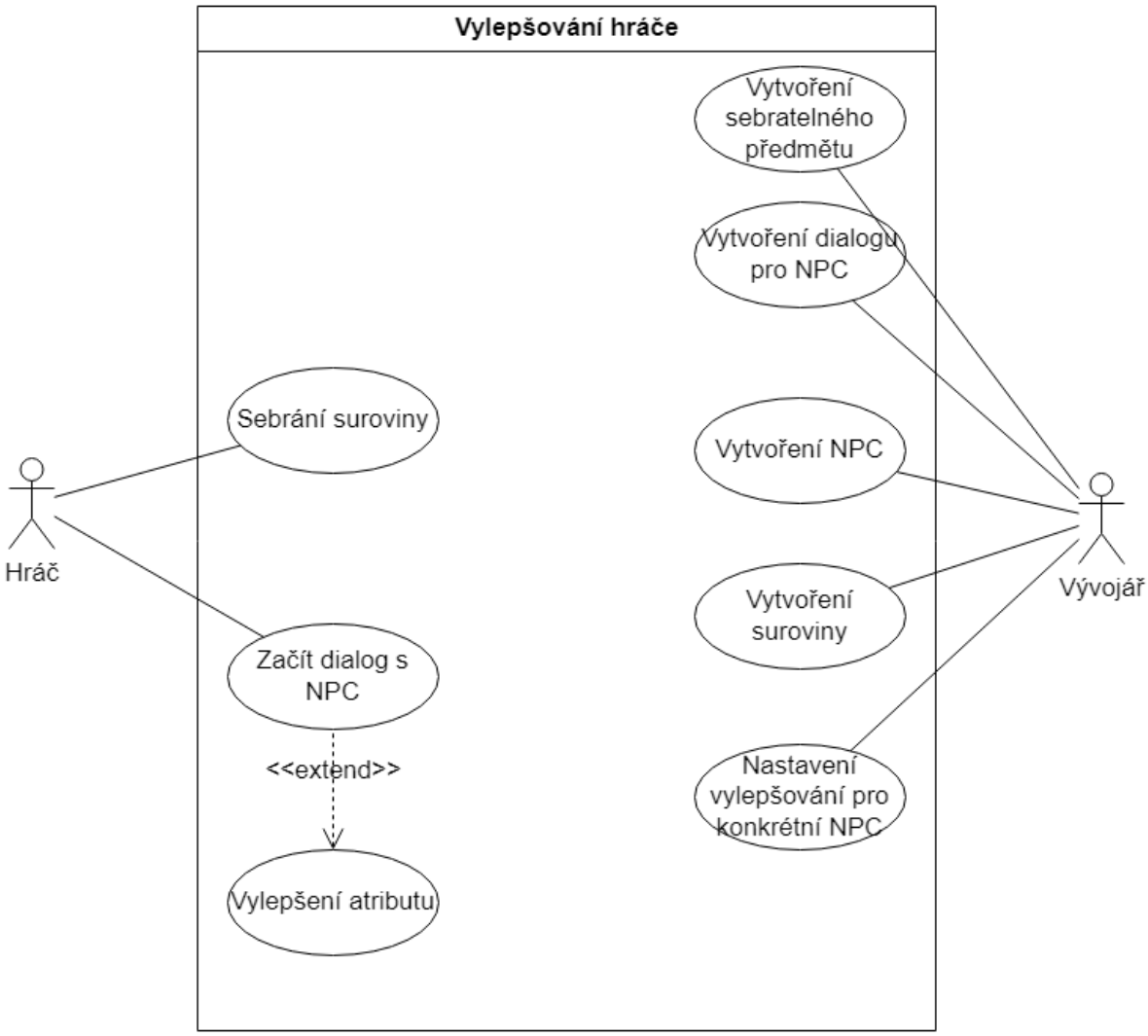
Obrázek 4.2 Class diagram vylepšování hráče



4.1.1 Případy užití

1. Hráč sbírá suroviny a ukládá je do inventáře.
2. Hráč začíná dialog s NPC a vylepšuje u něj atributy výměnou za suroviny. (Samotný start dialogu je již hotový v aktuální verzi MagiTechu. Na dialogovém systému dělal převážně Yuri Udavichenko).
3. Vývojář vytváří NPC, se kterým bude hráč interagovat.
4. Vývojář vytváří předměty, které se dají sebrat.
5. Vývojář vytváří dialogy pro NPC zahrnující vylepšení atributů.
6. Vývojář vytváří nové suroviny.
7. Vývojář nastavuje, jakým způsobem se bude vylepšovat atribut u konkrétního NPC.

Obrázek 4.3 Případy užití vylepšování hráče



4.2 Reputace s cechy

Modul reputace s cechy zahrnuje sbírání dárků v dungeonech, které hráč bude dávat NPC a tím se mu s jejich cechy bude měnit reputace. Dále zahrnuje deník, do kterého se budou hráči zapisovat transakce po předání dárku. Také do tohoto modulu spadají reputační milníky, které provedou definovanou akci poté co hráč dosáhne určité reputace s cechem.

4.2.1 Doménový diagram

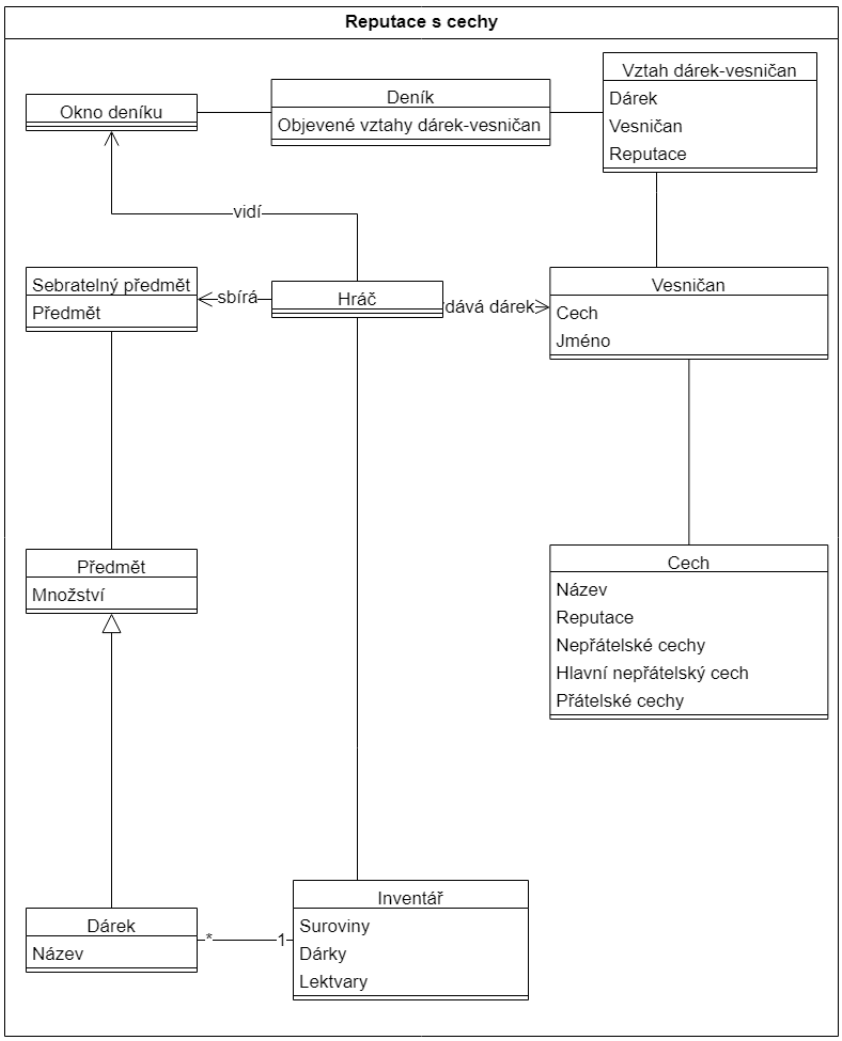
Každý vesničan bude mít přiřazený svůj cech. S těmito cechy bude hráč udržovat hodnotu reputace. Každý cech bude mít nastavené také nepřátelské a přátelské cechy.

Reputace se bude měnit tím, že bude hráč dávat vesničanům dárky. Podle toho, jak má vesničan dárek rád se reputace zvýší nebo sníží. Tato skutečnost je reprezentována vztahem dárek-vesničan, který má uložený dárek, vesničana a hodnotu o kolik se má reputace posunout.

Tyto vztahy bude hráč pozorovat v deníku, který bude mít přiřazené UI okno.

Před tím, než se dárek věnuje vesničanovi se nejdřív musí sebrat ze země a uložit do inventáře.

Obrázek 4.4 Doménový diagram reputace s cechy



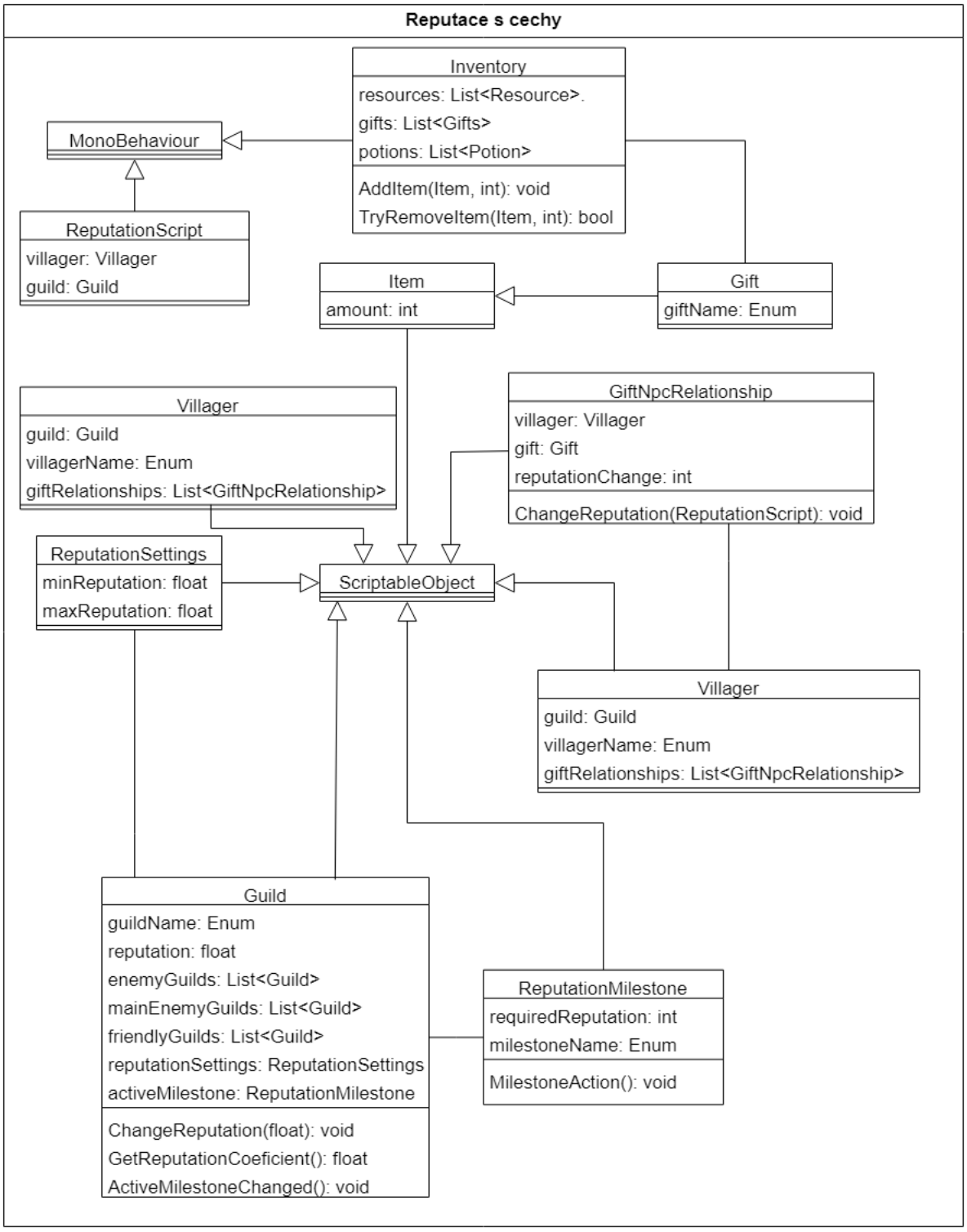
4.2.2 Diagram tříd

1. ReputationScript je skript, který bude mít jako komponentu připojené NPC a bude se starat o interakci s hráčem. Dědí ze třídy MonoBehaviour.
2. Inventory v sobě bude mít uloženy všechny suroviny, dárky a lektvary, které hráč nasbíral a bude je umět přidávat a odebírat. Dědí ze třídy MonoBehaviour.
3. Item je abstraktní třída, ze které dědí Resource, Potion a Gift. Dědí ze třídy ScriptableObject.
4. Villager bude mít nastavené jméno, cech a své vztahy k jednotlivým dárkům. Dědí ze třídy ScriptableObject.
5. GiftNpcRelationship je skript, který drží informaci o změně reputace vzhledem k Villager a Gift. Také umí zavolat na cechu informaci o změně reputace. Dědí ze třídy ScriptableObject.
6. ReputationSettings je třída, která omezuje dosažitelnou reputaci na definované minimum a maximum. Je potřeba například při zobrazování reputace na obrazovce. Dědí ze třídy ScriptableObject.
7. Guild třída reprezentuje cech. Obsahuje reputaci, spřátelené a znepřátelené cechy a aktuální milník, kterého hráč dosáhl. Také dokáže na změnu reputace reagovat změnou milníku a aktivováním akce s ním spojenou. Dědí ze třídy ScriptableObject.

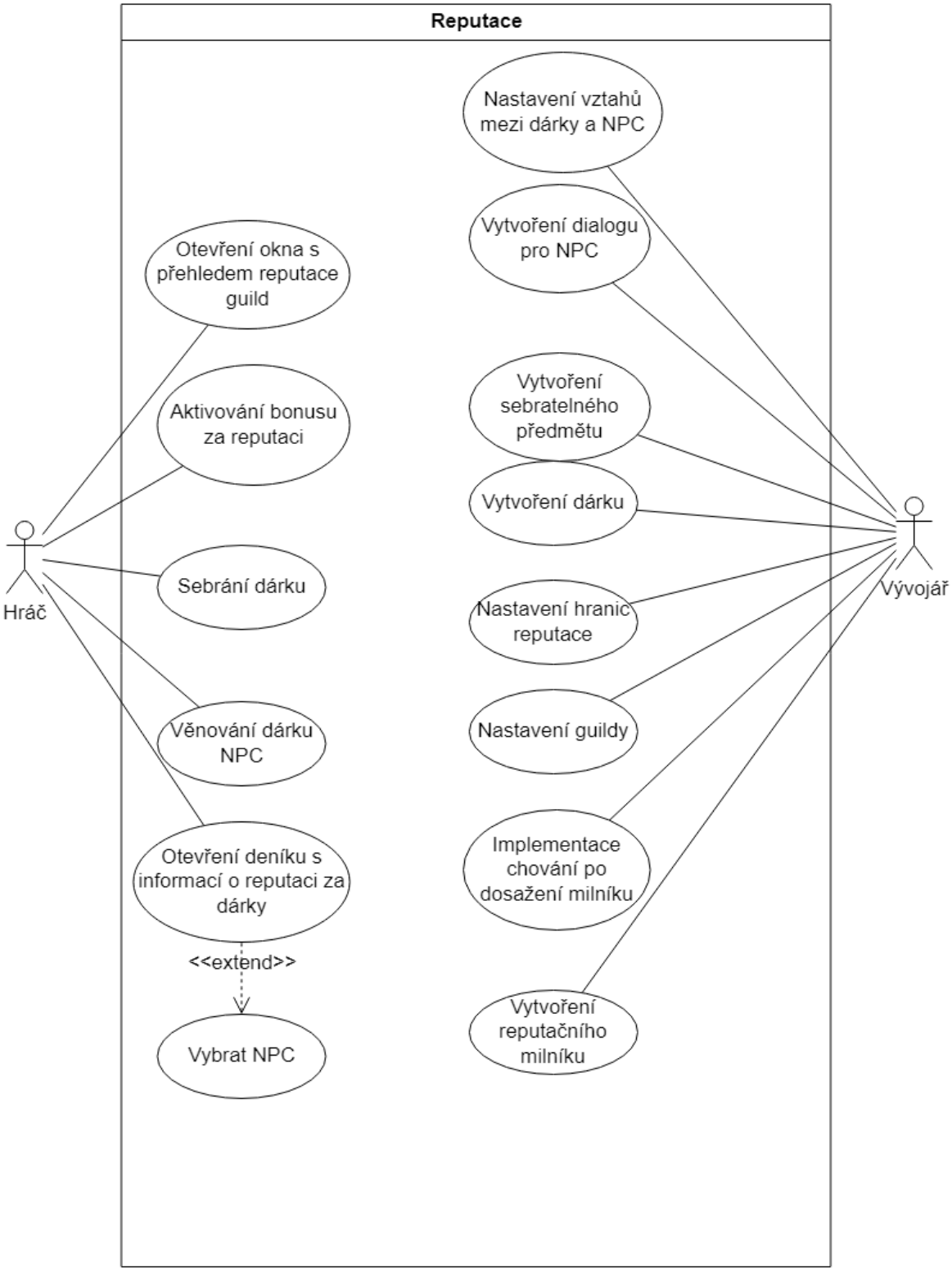
4.2.3 Diagram případů užití

1. Hráč sbírá dárky a ukládá je do inventáře.
2. Hráč věnuje v rámci dialogu dárky NPC.
3. Hráč aktivuje u vesničana bonus za dosažení specifického milníku.
4. Hráč pozoruje okno s přehledem reputace s cechy.
5. Hráč pozoruje okno deníku a vybírá v něm informace o konkrétním NPC.
6. Vývojář vytváří dárky a předměty, které se dají sebrat, s ním spojené.
7. Vývojář vytváří dialogy pro NPC zahrnující darování dárků a získání bonusu za reputaci.
8. Vývojář nastavuje vztahy mezi dárky a NPC.
9. Vývojář nastavuje hranice reputace a cechy.
10. Vývojář vytváří reputační milníky a nastavuje u nich chování.

Obrázek 4.5 Class diagram reputace s cechy



Obrázek 4.6 Případy užití reputace s cechy



4.3 Interakce se společníky

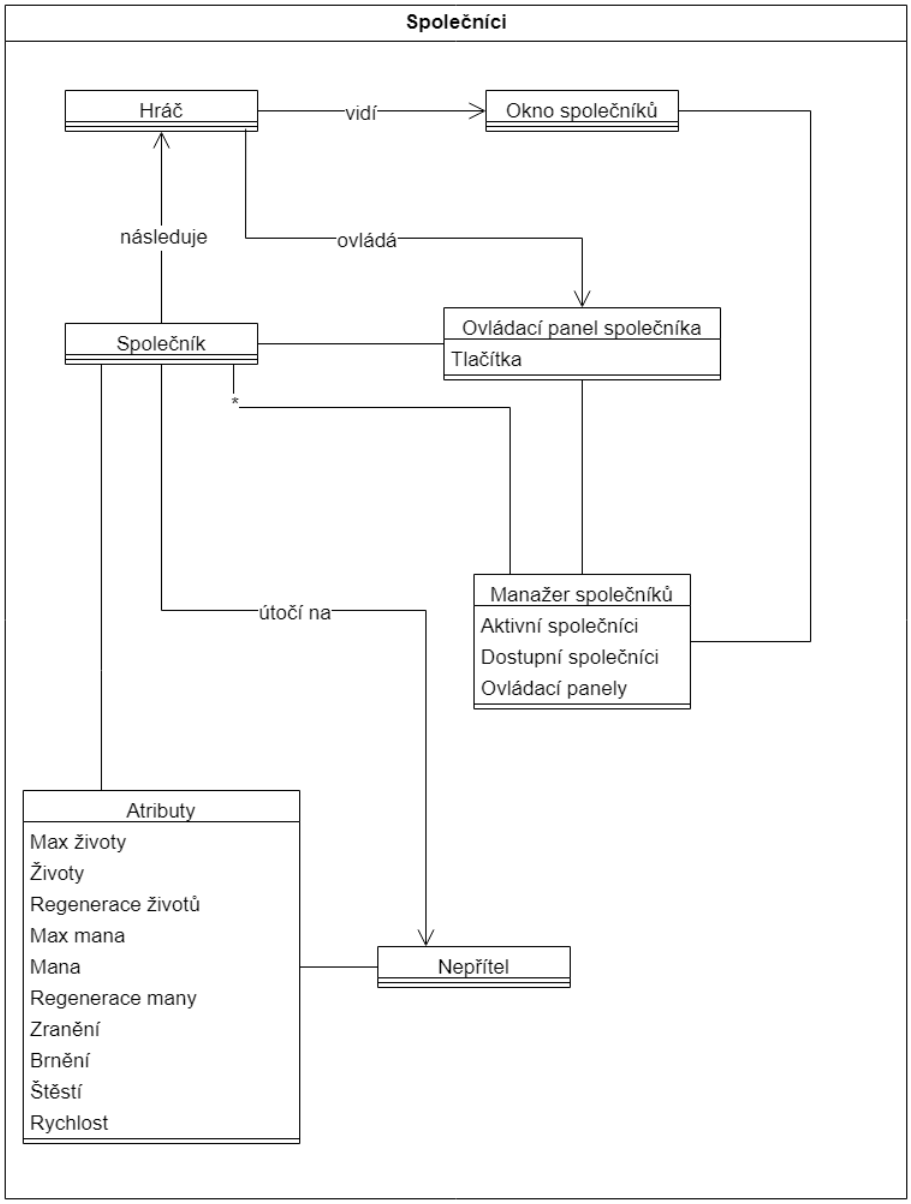
Návrh interakce se společníky zahrnuje získávání, aktivování a ovládání nových společníků hráčem. Také zahrnuje chování společníků popsané stavovým automatem.

4.3.1 Doménový diagram

Společníci stejně jako nepřátelé, kteří již ve hře jsou implementováni Danielem Breinerem, mají atributy, které je charakterizují. Společníci budou útočit na nepřátele a naopak.

Hráč bude aktivovat společníky pomocí okna společníků a ovládat společníky pomocí ovládacích panelů. Manažer společníků bude udržovat společníky a ovládací panely ve scéně.

Obrázek 4.7 Doménový diagram interakce se společníky



4.3.2 Diagram tříd

1. AICombatEntity je abstraktní třída, ze které budou dědit třídy Companion a Enemy. Dědí ze třídy MonoBehaviour.
2. Companion dědí z AICombatEntity a je připojená jako komponenta na objektu společníka. Stará se o chování společníka a uchovává jeho atributy. Dědí ze třídy MonoBehaviour.
3. Stats je skript, který se bude starat o atributy společníků a práci s nimi. Dědí ze třídy MonoBehaviour.
4. UICompanionController je třída, která se bude umožňovat ovládání společníka ve scéně. Dědí ze třídy MonoBehaviour.
5. CompanionManager je třída, kde se uchovávají dostupní a aktivní společníci a jejich ovládací panely. Také umí přidávat nové dostupné společníky a starat se o jejich aktivaci a deaktivaci. Dědí ze třídy MonoBehaviour.
6. StartingStats je třída, ve které jsou uloženy startovní atributy společníka nebo nepřátel. Dědí ze třídy ScriptableObject.

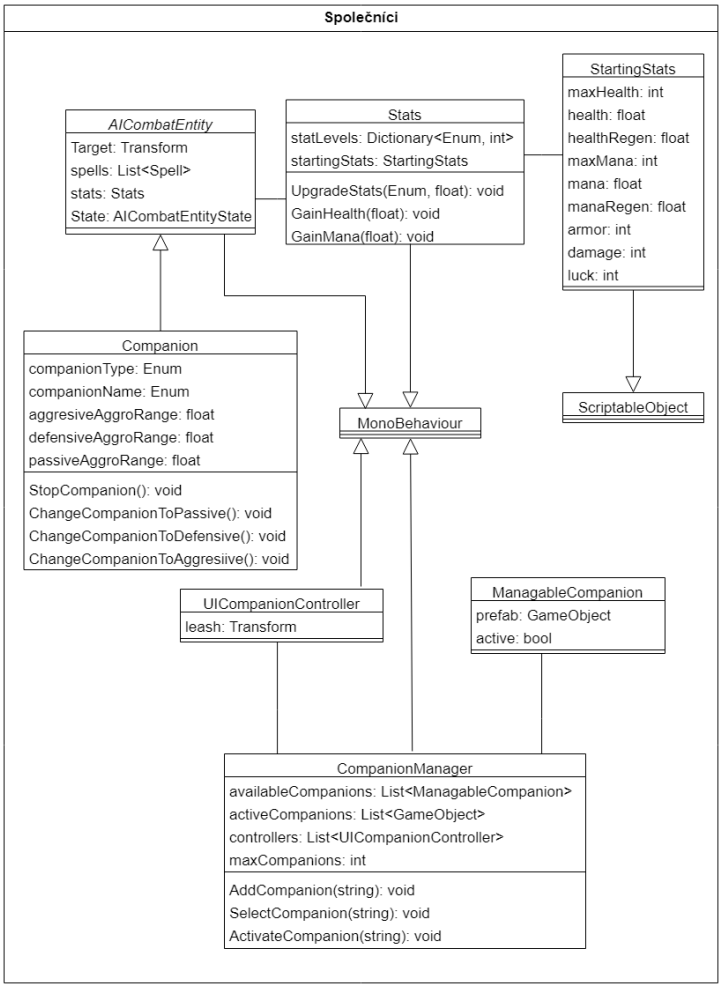
4.3.3 Diagram tříd stavů společníka

1. AICombatEntityState je abstraktní třída, ze které budou dědit všechny stavy společníka (i nepřítele).
2. StayCompanionState je stav, při kterém společník stojí na místě.
3. ChaseCompanionState je stav, při kterém společník pronásleduje nepřítele.
4. FollowCompanionState je stav, kdy společník následuje hráče.
5. KeepDistanceCompanionState je stav, během kterého se společník vzdaluje od nepřítele.
6. RangedAttackCompanionState je stav, při kterém společník útočí na nepřítele.

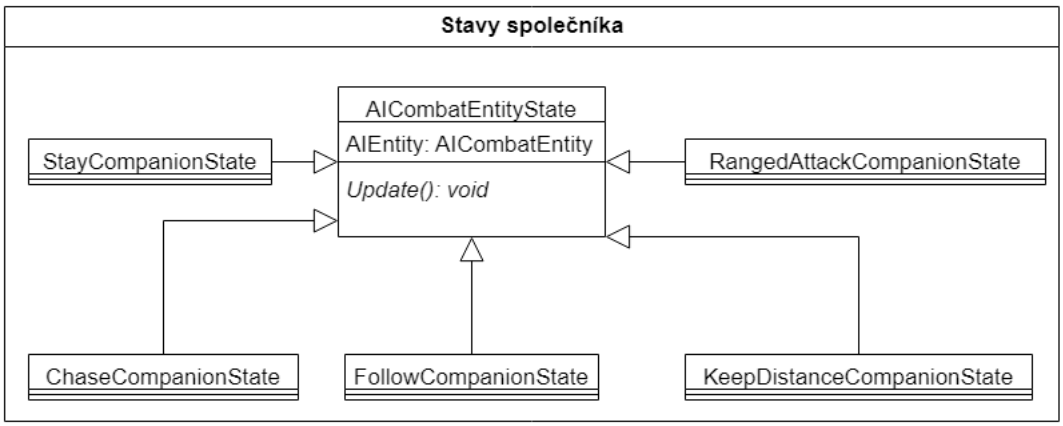
4.3.4 Diagram případů užití

1. Hráč otevírá okno s odemknutými společníky.
2. Hráč vybírá konkrétního společníka a toho aktivovat nebo deaktivovat.
3. Hráč zastavuje nebo volá aktivovaného společníka.
4. Hráč mění agresivitu společníka.
5. Vývojář vytváří nového společníka.
6. Vývojář nastavuje manažera společníků.
7. Vývojář přidává manažera společníků do scény.
8. Vývojář vytváří atributy pro daného společníka.

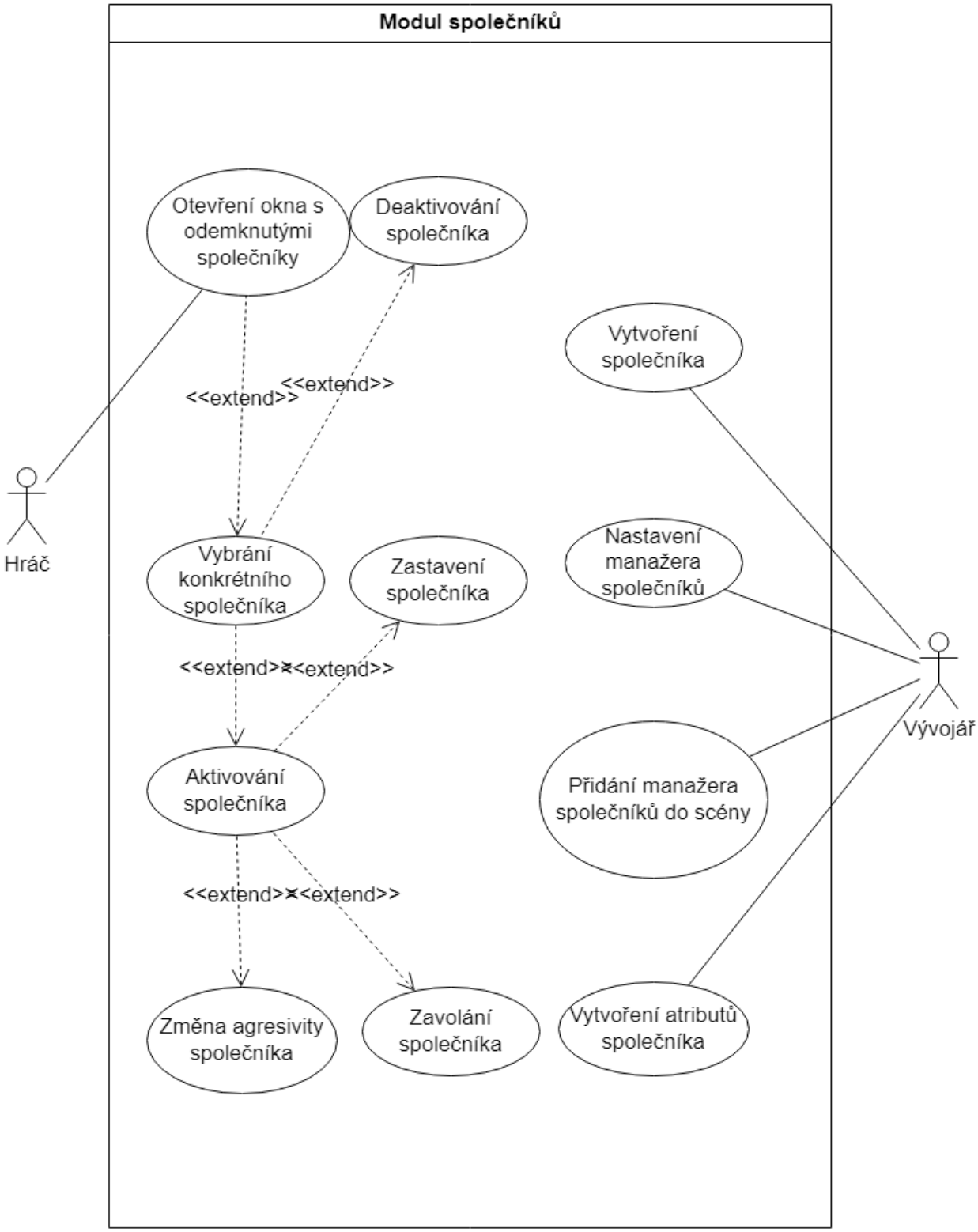
Obrázek 4.8 Class diagram interakce se společníky



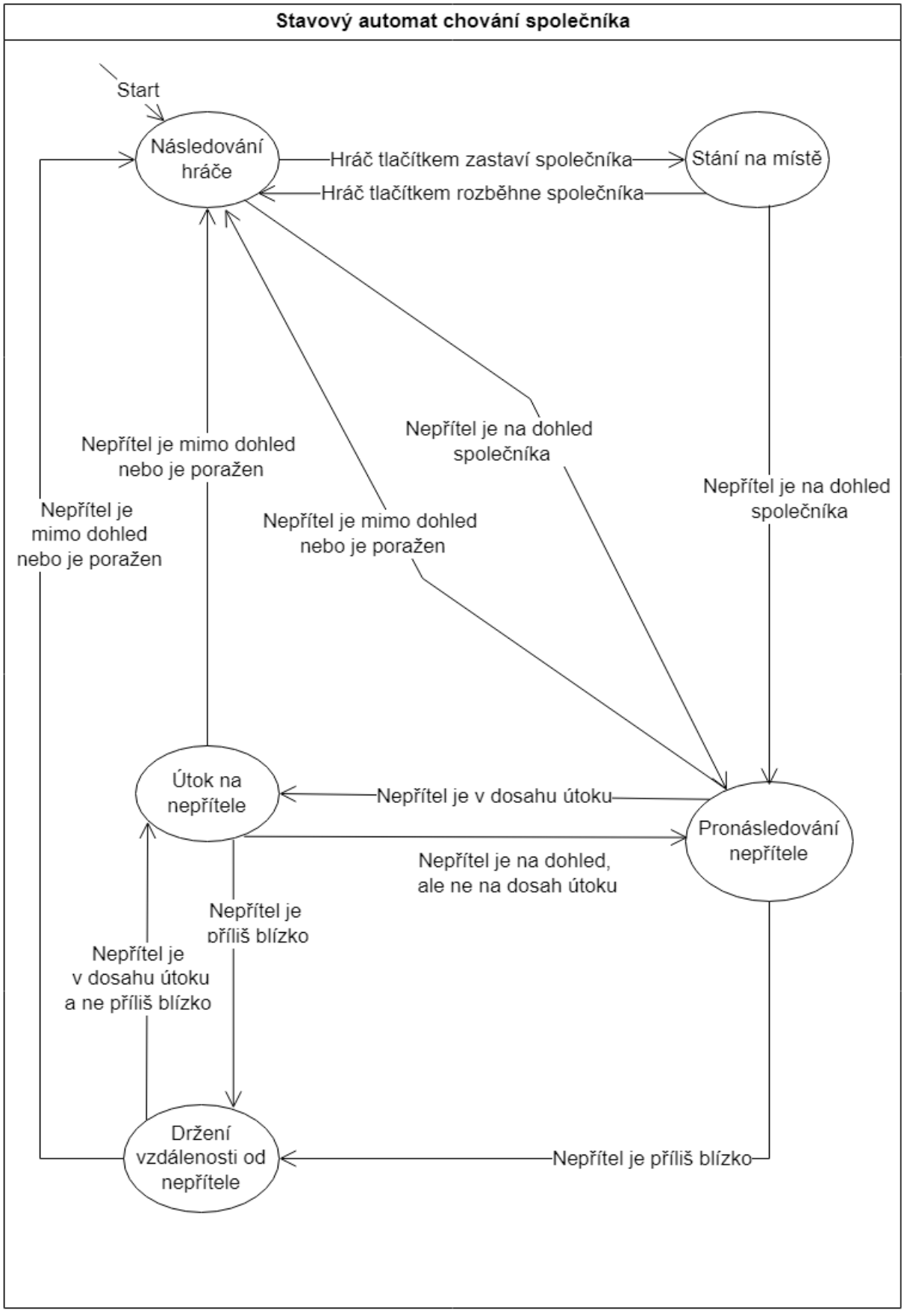
Obrázek 4.9 Class diagram stavů společníků



Obrázek 4.10 Případy užití interakce se společníky



Obrázek 4.11 Stavový automat chování společníků



Obrázek 4.12 Původní user interface MagiTechu

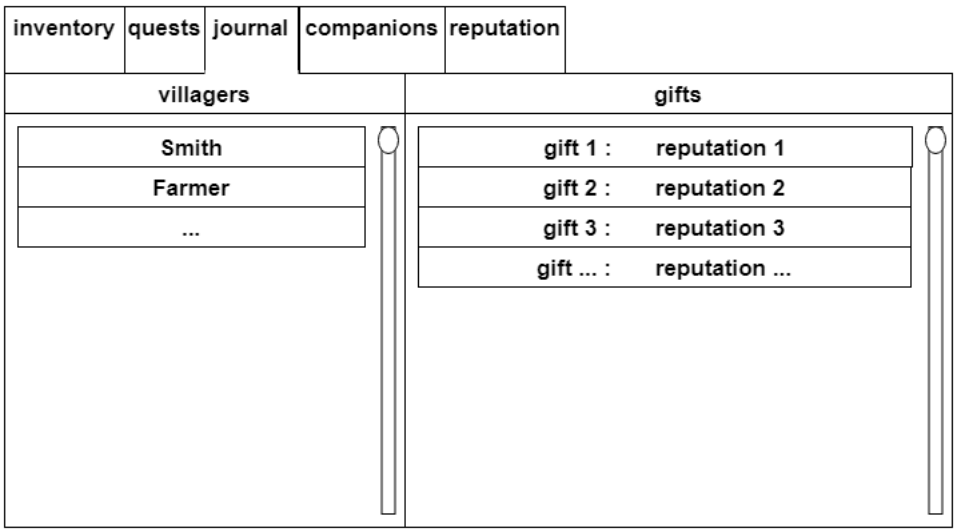
4.4 Návrh obrazovek pro UI

Důležitou částí návrhu je uživatelské rozhraní a návrh jednotlivých obrazovek, které budou spolu s herními systémy vznikat. Při návrhu se vychází z původního rozhraní v MagiTechu.

4.4.1 Obrazovka deníku dárků

Obrazovka, kde se zobrazují vesničané a po kliknutí na tlačítko vesničana se zobrazí dárky a reputace, kterou dávají.

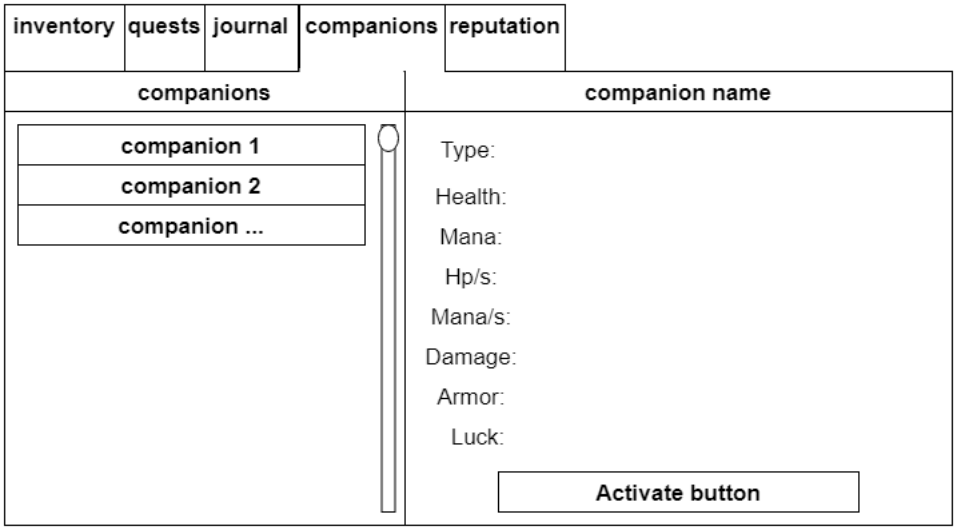
Obrázek 4.13 Návrh obrazovky deníku dárků



4.4.2 Obrazovka společníků

Obrazovka kde se zobrazují dostupní společníci. Po kliknutí na společníka se otevře napravo jeho detail a hráč ho může aktivovat a deaktivovat tlačítkem.

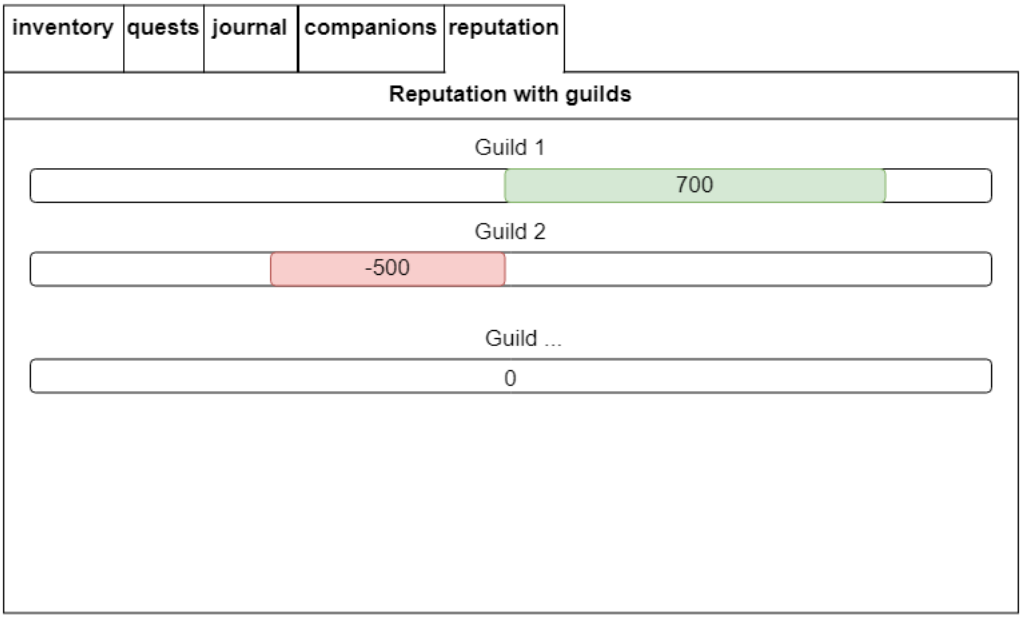
Obrázek 4.14 Návrh obrazovky přehledu společníků



4.4.3 Obrazovka reputací s cechy

Obrazovka kde se zobrazují aktuální reputace s každým cechem.

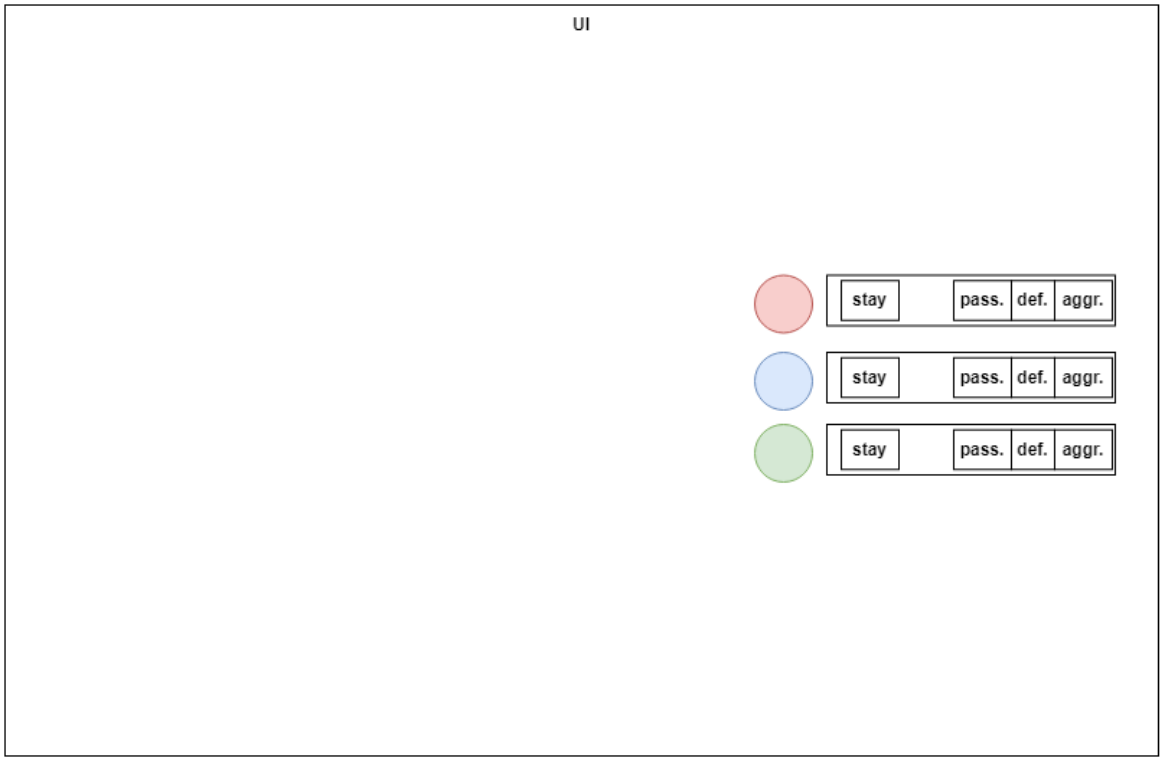
Obrázek 4.15 Návrh obrazovky reputací



4.4.4 Obrazovka ovládacích panelů společníků

Ukázka ovládacích panelů společníka. Jejich pozice je na pravé straně obrazovky z toho důvodu, aby nepřekážely ostatním elementům uživatelského rozhraní.

Obrázek 4.16 Návrh ovládacích panelů společníků



Kapitola 5
Implementace

V této kapitole se rozebírá implementace modulu, jaké problémy se vyskytly a jaké se našlo řešení.

Implementace modulu začala už předtím, než se začal psát tento text. Je to proto, že modul byl potřeba pro účely předmětu BI-VHS, Virtuální herní světy, kde se první verze hry MagiTech vytvářela. V první verzi byla implementace modulu velmi omezena z důvodu nedostatku času a obsahovala hlavně část vylepšování hráče a částečně reputaci s NPC (tehdy se uvažovalo o reputaci s vesničany, a ne s cechy). Později se při vytváření modulu některé dříve implementované části zachovaly a některé se úplně předělaly.

5.1 Kontrola vstupu vývojáře

Na některých místech bylo potřeba ošetřit vstupy, aby vývojář nezadal například záporné číslo, kde by nedávalo smysl a nechtěně tak nezpůsobil chybu.

V jednodušších případech se toho docílilo použitím anotace Range(), která bere jako argumenty minimální a maximální hodnotu a poté se do proměnné s touto anotací nedá vložit hodnota mimo tyto hranice (viz. Třída nastavení vylepšování). Pro složitější případy se používá metoda OnValidate(), která se volá vždy, kdy je provedena změna v Unity inspectoru, a dá se jí definovat chování, které omezuje vstup. V následující ukázce se v metodě OnValidate() nastavuje vzdálenost agrese společníka tak, aby defensivní vzdálenost nebyla nižší než pasivní a agresivní vzdálenost aby nebyla nižší než defensivní a pasivní.

Výpis kódu 5.1 Metoda OnValidate() v CompanionObject

```
//defensive and aggresive must be larger range than passive
    private void OnValidate()
    {
        defensiveAggroRange = Mathf.Clamp(defensiveAggroRange,
                                          passiveAggroRange, float.MaxValue);
        aggresiveAggroRange = Mathf.Clamp(aggresiveAggroRange,
                                          defensiveAggroRange, float.MaxValue);
    }
```

5.2 Inventář hráče

Jedna z prvních věcí, které bylo potřeba implementovat, byl inventář hráče. V inventáři si hráč ukládá suroviny, dárky a lektvary, které jsou reprezentovány jako List of Resource, List of Gift

a List of Potion. Informace o obsahu inventáře se zobrazují na obrazovce inventáře. To se dělá jednoduše tak, že se v cyklu prochází listy předmětů v Inventory skriptu a vypisuje se, kolik jich je. Aby tohle bylo možné, tak je potřeba v metodě Awake() inicializovat listy. Na to se používá metoda FindObjectsOfTypeAll(), která najde všechny scriptableObjects, které odpovídají typu. Díky tomu se každý předmět, který vývojář vytvoří automaticky načte při spuštění hry do inventáře s hodnotou 0. Jednotlivé předměty se z inventáře odebírají metodou TryRemoveItem() a přidávají AddItem().

Výpis kódu 5.2 Metoda Awake() v Inventory skriptu

```
private void Awake()
    {
        playerStats = gameObject.GetComponent<Stats>();

        if (resources.Count == 0)
            foreach (Resource resource in
            UnityEngine.Resources.FindObjectsOfTypeAll(typeof(Resource))
                                                    as Resource[])
            {
                resources.Add(Instantiate(resource));
            }
        if (gifts.Count == 0)
            foreach (Gift gift in
            UnityEngine.Resources.FindObjectsOfTypeAll(typeof(Gift))
                                                    as Gift[])
            {
                gifts.Add(Instantiate(gift));
            }
        if(potions.Count == 0)
            foreach (Potion potion in
            UnityEngine.Resources.FindObjectsOfTypeAll(typeof(Potion))
                                                        as Potion[])
            {
                potions.Add(Instantiate(potion));
            }
    }
```

5.3 Zásahy do dialogového systému

Vzhledem k tomu, že téměř veškerá interakce s NPC pro hráče je vedena pomocí dialogu, tak se muselo implementovat několik nových funkcionalit do dialogového systému (dialogový systém ale nespadá do rámce této práce a byl pro MagiTech vytvořen Yuri Udavichenkem).

Hlavní nová funkcionalita byla SkipDialogueNode. Tato třída stejně jako BasicDialogueNode a ChoiceDialogueNode dědí ze třídy DialogueNode a je vytvořena tak, aby se po jejím navštívení v rámci dialogu, provedl přiřazený UnityEvent a skočilo se na další DialogueNode. Docílit se toho dalo pomocí přidání nového UnityEventu, který zavolá outerCallback (akci, která byla přiřazena této DialogueNode na zavolání), zavolá callback další DialogueNode a na tu přeskočí. Díky tomu hráč, například při úspěšném vylepšení atributu, rovnou přeskočí na hlavní ChoiceDialogueNode a může pokračovat v dialogu.

Výpis kódu 5.3 Metoda CallbackAndSkip() u SkipDialogueNode

```
public void CallbackAndSkip()
    {
        outerCallback.Invoke();
        nextNode.callback.Invoke();
        dialogueChannel.RaiseRequestDialogueNode(nextNode);
    }
```

5.4 Implementace vylepšování

Hlavní dva skripty, na kterých je postaveno vylepšování hráče jsou NpcUpgradeScript a UpgradeSettings, které oba dědí ze scriptableObject, takže se pro každé NPC vytváří jako asset (viz. příručka pro vývojáře). V první verzi byl UpgradeScript připojený jako komponenta na každém NPC, takže díky použití scriptableObject se tomu dalo vyhnout a omezit tak počet skriptů, které každé NPC ve scéně mělo připojených.

NpcUpgradeScript má metody CheckUpgrade(), která kontroluje, jestli má hráč dostatek surovin na vylepšení a UpgradeStat(), která vylepší atribut hráče. Obě metody se volají jako UnityEvent při dialogu se specifickým NPC.

UpgradeSettings si ukládá informace potřebné k provedení vylepšení.

Výpis kódu 5.4 Třída nastavení vylepšování

```
public class UpgradeSettings : ScriptableObject
{
    [Range(0, float.MaxValue)]
    public float baseCost;
    [Range(0, float.MaxValue)]
    public float baseUpgradePower;
    public Stat statToUpgrade;
    public Resource resource;
    public Guild guild;
}
```

Výpis kódu 5.5 Výpočet výsledné ceny vylepšení

```
var finalCost = (int) (level * upgradeSettings.baseCost * repCoeficient);
```

5.4.1 Cena vylepšování

Při implementaci vylepšování pro MagiTech bylo potřeba myslet na to, aby bylo co nejjednodušší, ale zároveň dobře zapadalo do hry. Jedna možnost, jak se dívat na vylepšování bylo jako na rovnoměrnou výměnu surovin za atributy. To ale přineslo problém s tím, že by se hráč mohl stát celkem jednoduše velmi silný tak, že nasbírá mnoho surovin a vylepší atribut do nekonečna.

Tento problém se vyřešil zavedením úrovně atributů, které reprezentují, kolikrát hráč atribut vylepšil. Cena vylepšení atributu se potom odvíjí nejenom od základní ceny a reputace, ale i od úrovně atributu. Úrovně jsou vedené jako Dictionary¡Stat, int¿ kde Stat je enum označující daný atribut a hodnota int značí úroveň atributu.

Výpis kódu 5.6 Atributy třídy Stats

```
public class Stats : MonoBehaviour
{
    public Dictionary<Stat, int> statLevels = new Dictionary<Stat, int>();

    [SerializeField] StartingStats startingStats;

    private float nextActionTime = 0.0f;
    [SerializeField] float period = 1f;

    [SerializeField] Slider healthBar;
    [SerializeField] Slider manaBar;

    [SerializeField] StatusEffects statusEffects;

    public delegate void DeathEventHandler();
    public event DeathEventHandler DeathEvent;
}
```

Další problém byl, jakým způsobem by se měla cena vylepšení odvíjet od reputace s NPC, u kterého si hráč vylepšení pořizuje. Vylepšení nesmělo být úplně zadarmo, ani příliš drahé, takže se přišlo na vzorec, který způsobí, že při nejnižší reputaci bude hráč platit dvojnásobek ceny, při nejvyšší polovinu a když je reputace neutrální tak se cena nezmění. Reputation ve vzorci značí aktuální reputaci s guildou, MaxReputation značí maximální dosažitelnou reputaci.

Výpis kódu 5.7 Metoda na vrácení koeficientu ceny reputace (ve skriptu Guild)

```
public float GetReputationCoeficient()
    {
        // formula to calculate coeficient to alter upgrade costs
        // -100% rep -> times 2 | 100% rep -> times 0.5
        return reputation < 0 ?
        ((float) Mathf.Abs(reputation) + reputationSettings.MaxReputation)
        / reputationSettings.MaxReputation :
        1 / ((reputation + reputationSettings.MaxReputation)
        / reputationSettings.MaxReputation);
    }
```

5.5 Implementace cechů a reputace

V rámci celé práce byla snaha používat co nejvíce scriptableObject, aby objekty nebyly omezeny na konkrétní scénu a vývojář mohl snadno vytvářet další instance.

Guild je hlavní třída systému cechů. Má jako argumenty reputaci s hráčem a seznamy znepřátelených a spřátelených cechů. V rámci systému reputace byl požadavek, aby se ukázala rivalita mezi cechy, což je součástí příběhu hry. To je řešeno tak, že když hráč vylepší reputaci s jedním cechem, tak se mu automaticky zhorší reputace se znepřátelenými cechy a naopak. Díky tomu se bude muset hráč rozmýšlet, se kterými cechy reputaci zvyšovat, protože každý cech mu dává za nízkou nebo vysokou reputaci mínusy nebo bonusy. Tomu ještě přispívá implementace hlavního nepřátelského cechu, se kterým se reputace mění o 100 procent změněné reputace, zatímco se spřátelenými a ostatními znepřátelenými jenom o 50 procent.

Výpis kódu 5.8 Metoda na změnu reputace s cechem

```
public void ChangeReputation(float reputation)
{
    var guildManager = DataManager.Instance.GuildManager
                       .GetComponent<GuildManager>();
    guildManager.ChangeReputation(this, reputation);
    foreach(Guild guild in enemyGuilds)
    {
    if (mainEnemyGuilds.Contains(guild))
    guildManager.ChangeReputation(guild, -reputation);
    else
    guildManager.ChangeReputation(guild, -(reputation / reputationChangeModifier));
    }

    foreach (Guild guild in friendlyGuilds)
    guildManager.ChangeReputation(guild, reputation / reputationChangeModifier);
}
```

5.5.1 Proces změny reputace

Vzhledem k tomu, že Guild se vytváří jako asset, nemůže se s tímto objektem pracovat přímo, protože by se zachovaly změny při každém vypnutí a spuštění hry. Proto bylo potřeba zavést GuildManager, což je skript na prázdném objektu ve scéně a stará se o instance cechů během hry. Při startu hry si GuildManger najde všechny cechy, které předtím vývojář vytvořil jako scriptableObject, a pracuje s jejich instancemi. Tyto instance se při vypnutí hry zničí, takže pokaždé hra začíná se stejně nastavenými cechy. GuildManager se také stará o milníky reputace a kontroluje, jestli se při změně reputace nějaký neaktivoval.

Výpis kódu 5.9 Metoda Awake() u GuildManager

```
private void Awake()
{
    foreach (Guild guild in Resources.FindObjectsOfTypeAll(typeof(Guild)) as Guild[])
        guilds.Add(Instantiate(guild));

    milestones.Sort(MilestoneComparer);
}
```

Pro změnu reputace je důležitý ReputationScript, který je jako komponenta připojený na každém NPC, který patří k nějakému cechu. Tento skript obsahuje metodu InitRepuation(), která se zavolá, když hráč vyvolá dialog s vesničanem a dělá to, že nastavuje dialogové možnosti pro darování dárků. Za každý dárek, který hráč vlastní, se vytvoří nová dialogová možnost s vesničanem spolu s možností nedat žádný dárek. ReputationScript se také stará o nastavování sliderů reputace ve scéně, a proto není implementován jako scriptableObject (který se přímo ve scéně nenachází).

Výpis kódu 5.10 Metoda InitReputation() u ReputationScriptu

```
/// <summary>
/// This is called when listing main dialogue choices with npc
/// It creates new dialogue choice for each gift that
/// can be gifted to the npc + default no gift choice
/// </summary>
public void InitReputation()
{
SetRepSlider(guild);

var playerInventory = GameObject.FindGameObjectWithTag("Player")
                                    .GetComponent<Inventory>();
giftChoices.Choices.Clear();
returnNode.callback.RemoveAllListeners();

DialogueChoice dialogueChoice;
for (int i = 0; i < playerInventory.Gifts.Count; i++)
{
var gift = playerInventory.Gifts[i];
if (gift.Amount != 0)
{
dialogueChoice = new DialogueChoice();
var skipNode = ScriptableObject.CreateInstance<SkipDialogueNode>();
var relationship = villager.GiftRelationships
.Find(r => r.Gift.GiftName == gift.GiftName && r.Villager
.VillagerName == villager.VillagerName);
if(relationship != null)
{
skipNode.outerCallback = new UnityEvent();
skipNode.callback = new UnityEvent();
skipNode.DialogueChannel = dialogueChannel;
skipNode.NextNode = returnNode;
skipNode.outerCallback.AddListener(
    delegate { relationship.ChangeReputation(this); }
);
skipNode.callback.AddListener(skipNode.CallbackAndSkip);

if (JournalManager.IsDiscovered(relationship))
    dialogueChoice.ChoicePreview = "Give " +
    gift.GiftName.ToString() + " as gift (" +
    relationship.ReputationChange + " reputation)";
else
    dialogueChoice.ChoicePreview = "Give " +
    gift.GiftName.ToString() +
    " as gift (" + "+??? reputation)";
}

dialogueChoice.ChoiceNode = skipNode;
giftChoices.Choices.Add(dialogueChoice);
}
}
dialogueChoice = new DialogueChoice();
dialogueChoice.ChoicePreview = "No gift for you";
returnNode.callback.AddListener(InitReputation);
dialogueChoice.ChoiceNode = returnNode;
giftChoices.Choices.Add(dialogueChoice);
}
```

Změna reputace začíná u třídy GiftNpcRelationship. Tato třída je také scriptableObject a uchovává informace o vesničanovi, dárku a o tom jak se reputace s hráčem změní po darování dárku. Také obsahuje metodu ChangeReputation(), která se volá vybráním odpovídající dialogové možnosti, sebere hráči správný počet dárků a zavolá metodu ChangeReputation() na skriptu Guild podle toho, komu hráč dárek daroval.

Výpis kódu 5.11 Metoda ChangeReputation() u GiftNpcRelationship

```
public void ChangeReputation(ReputationScript reputationScript)
{
    var playerInventory = GameObject.FindGameObjectWithTag("Player")
                                        .GetComponent<Inventory>();

    if (playerInventory.TryRemoveItem(gift, 1))
    {
        villager.Guild.ChangeReputation(reputationChange);
        reputationScript.SetRepSlider(villager.Guild);
        JournalManager.Discover(this);
    }
    else
        Debug.Log("Not enough: " + gift.GiftName.ToString());
}
```

Na skriptu Guild poté metoda Guild.ChangeReputation() zavolá GuildManager.ChangeReputation() na sobě a na znepřátelených ceších s odpovídajícími hodnotami reputace (negativní pro znepřátelené apod.) (viz. Metoda na změnu reputace s cechem).

GuildManager poté v mětodě GuildManager.ChangeReputation() změní reputaci (pokud je větší než maximum nebo menší ne minimum tak ji nastaví na maximum nebo minimum).

5.5.2 Milníky reputace

Milníky reputace jsou část modulu, která není z časových důvodů plně dokončena. Nicméně je hotový systém reputačních milníků, na kterém se dá dál stavět. Také je plně dokončena implementace pro cech kovářů. Implementace konkrétních mínusů a bonusů pro ostatní cechy prozatím chybí.

Základem systému milníků je třída ReputationMilestone. Tato třída reprezentuje pro hráče milník, kterého když dosáhne, tak se aktivuje negativní nebo pozitivní efekt. Dědí ze scriptableObject a obsahuje číslo hranice reputace pro dosažená a název reprezentovaný jako enum. Název je zatím pouze pro přehlednost.

Logika aktivování milníků je ve třídě GuildManager. Pokaždé, když se změní reputace, tak GuildManager zkontroluje, jestli hráč nedosáhl nějakého milníku, a pokud ano, tak aktivuje příslušný UnityEvent. Tyto volání UnityEvent musí vývojář implementovat sám, a to pomocí rozhraní IMilestoneActions. Vývojář tedy musí vytvořit třídu, která dědí ze scriptableObject a implementuje toto rozhraní. Metody tohoto rozhraní se poté volají.

Výpis kódu 5.12 Rozhraní IMilestoneActions

```
    public interface IMilestoneActions
{
    public abstract void ResetAllActions();
    public abstract void NemesisAction();
    public abstract void EnemyAction();
    public abstract void UnfriendlyAction();
    public abstract void NeutralAction();
    public abstract void FriendlyAction();
    public abstract void AllyAction();
    public abstract void BelovedAction();
}
```

Třída, která slouží jako prostředník mezi ReputationMilestone a implementovanými metodami IMilestoneActions, je MilestoneActionObject. Tato třída obsahuje ReputationMilestone a UnityEvent, který se má pro daný milestone vyvolat. Instance této třídy jsou specifické pro každé NPC a jejich seznam má příslušný Guild skript.

5.6 Implementace společníků

Systém společníků je silně inspirovaný systémem společníků ze hry World of Warcraft. Hlavní rozdíl je,
že ve hře MagiTech může hráč mít najednou aktivní až tři společníky. Toto rozhodnutí je z toho důvodu, aby na obrazovce hráče nebylo příliš mnoho entit vzhledem k tomu, že v dungeonech může být velké množství nepřátel.

Také z důvodu nedostatku modelů vypadají momentálně společníci stejně jako nepřátelé, takže jsou označeni zelenou, modrou a červenou barvou. Toto označení navíc poskytuje vizuální spojení aktivního společníka s jeho ovládacím panelem, který má stejnou barvu.

5.6.1 Companion skript a zásahy do Enemy skriptu

Aby prefab mohl fungovat jako společník, musí mít jako komponentu připojený skript Companion. Tento skript je silně inspirovaný skriptem Enemy, který pro předmět BI-VHS programoval Daniel Breiner. Také implementace stavového automatu chování společníků se inspirovala stavovým automatem chování nepřátel.

Když se přidával skript Companion, bylo zřejmé, že společníci budou mít hodně společného s nepřáteli. Proto byla z původního Enemy skriptu vytvořena třída AICombatEntity, ze které změněná Enemy třída a Companion třída dědí. Společné mají například atributy, kouzla, která mohou použít, animaci zničení a podobně.

V systému nepřátel bylo také potřeba provést několik změn. Atributy související s navigací ve scéně se z třídy připojené na modelu přesunuly do samostatného scriptableObject AICombatObject. To bylo potřeba, protože i po této změně má AICombatEntity mnoho pomocných atributů a nebyla by přehledná.

Výpis kódu 5.13 Třída AICombatObject

```
public class AICombatObject : ScriptableObject
{
    public float AwarenessRadius;
    public float MovementSpeed;
    public float MovementAcceleration;
    public float RotationSpeed;
    public float AttackRange;
    public float AttackCooldown;
    public float AimDuration;
    public float RetreatRadius;
    public bool isEnemy;
}
```

Útok na hráče v původním Enemy skriptu fungoval tak, že vývojář přidal v inspectoru Transform hráče na nepřátele a ti se podle něj mohli orientovat a útočit na něj. S přidáním společníků toto nebylo možné, protože bylo žádoucí, aby nepřátelé útočili na hráče i jeho společníky. Proto se do AICombatEntity přidal atribut potentialTargets jako List¡Transform¿. Tento atribut se v metodě FindPotentialTargets() naplní potenciálními cíli ve scéně a pokud se nějaký z cílů objeví v dosahu, tak se na něj zaměří. FindPotentialTargets() je abstraktní metoda v AICombatObject, takže Enemy i Companion třídy ji musí implementovat. Enemy jako cíle mají společníky a hráče, Companion má jako cíle nepřátele.

Výpis kódu 5.14 FindPotentialTargets() pro Companion skript

```
public override void FindPotentialTargets()
{
    var tmp = GameObject.FindGameObjectsWithTag("Enemy");
    for (int i = 0; i < tmp.Length; i++)
    {
        potentialTargets.Add(tmp[i].transform);
        tmp[i].GetComponent<Enemy>().potentialTargets.Add(transform);
    }
}
```

Výpis kódu 5.15 FindPotentialTargets() pro Enemy skript

```
    public override void FindPotentialTargets()
{
    var players = GameObject.FindGameObjectsWithTag("Player");
    var helpers = GameObject.FindGameObjectsWithTag("Companion");
    var targets = new GameObject[players.Length + helpers.Length];
    players.CopyTo(targets, 0);
    helpers.CopyTo(targets, players.Length);

    potentialTargets.Clear();
    for (int i = 0; i < targets.Length; i++)
    {
        potentialTargets.Add(targets[i].transform);
    }
}
```

Společníci se ve scéně ovládají pomocí třídy CompanionManager, která je ve scéně připojena na prázdném objektu a obsahuje informace o dostupných a aktivovaných společnících. Ke kontrolování, který společník je aktivní pomáhá pomocná třída ManagableCompanion, která má jako atributy GameObject společníka a true/false informaci o tom, jestli je aktivní.

CompanionManager se také stará o aktivaci ovládacích panelů společníků, takže pokaždé, kdy se aktivuje nebo deaktivuje společník, tak se aktivuje nebo deaktivuje jeho ovládací panel. Tlačítka na ovládacích panelech aktivují metody ve třídě Companion na změnu agresivity a zastavení společníka. Tlačítka se nastavují ve skriptu CompanionPanelController, který je připojený na panelu ve scéně.

Výpis kódu 5.16 Třída CompanionPanelController

```
public class CompanionPanelController : MonoBehaviour
{
    [SerializeField] Button stopButton;
    [SerializeField] Button passiveButton;
    [SerializeField] Button defensiveButton;
    [SerializeField] Button aggressiveButton;
    [SerializeField] GameObject colorCircle;

    public void Init(Companion companion, Material material)
    {
        colorCircle.GetComponent<Image>().material = material;
        stopButton.onClick.AddListener(delegate { companion.StopCompanion(); } );
        passiveButton.onClick.AddListener(delegate {
            companion.ChangeCompanionToPassive();
        });
        defensiveButton.onClick.AddListener(delegate {
            companion.ChangeCompanionToDefensive();
        });
        aggressiveButton.onClick.AddListener(delegate {
            companion.ChangeCompanionToAggresive();
        });
    }
}
```

Změna agresivity je implementována jako změna atributu awarenessRadius, který určuje v jakém rozsahu nachází společník nepřátele.

Obrázek 5.1 Aktivovaní společníci a jejich ovládací panely

5.7 Implementace uživatelského rozhraní

Implementace uživatelského rozhraní pro modul byla silně inspirována uživatelským rozhraním, které pro MagiTech vytvořil Roman Španko. Díky tomu se zachovala konzistence a všechny implementované obrazovky vypadají podobně.

Obrazovka deníku je ovládána skriptem JournalManager, který si při startu hry najde všechny vytvořené vesničany a jejich vztahy s dárky.

Výpis kódu 5.17 Awake() třídy JournalManger

```
private void Awake()
{
foreach (Villager villager in Resources.FindObjectsOfTypeAll(typeof(Villager))
                                                              as Villager[])
{
    VillagerButtonController newButton = Instantiate(villagerButtonPrefab);
    newButton.SetButton(villager);
    newButton.SetManager(this);
    newButton.transform.parent = villagerList;
}

foreach (GiftNpcRelationship relationship in Resources.FindObjectsOfTypeAll(
                        typeof(GiftNpcRelationship)) as GiftNpcRelationship[])
{
    if(!discoveredGifts.ContainsKey((relationship.Villager, relationship.Gift)))
    discoveredGifts.Add((relationship.Villager, relationship.Gift), false);
}
}
```

Díky tomu je může vypsat na obrazovku a zároveň pro každého vesničana vytvořit tlačítko, které ovládá, jaké vztahy se mají momentálně zobrazit.

Obrázek 5.2 Obrazovka deníku

Obrazovka společníků obsahuje Dostupné společníky a informace o nich. Ovládá ji CompanionManager, o kterém se mluví výše.

Obrázek 5.3 Obrazovka společníků

Obrazovka reputací ukazuje reputace guild, které jsou uložene ve třídě GuildManager.

Obrázek 5.4 Obrazovka reputací

5.8 Implementace kill squadů

Kill squady jsou řešeny třídou KillSquadManager, která obsahuje seznam KillSquadMembers a veřejné metody, které do tohoto seznamu přidávají a odebírají. Se třídou KillSquadManager pracuje systém generování dungeonů, na který dělá práci Jiří Macháček a ten ze seznamu vezme jednotlivé KillSquadMembers a vygeneruje je v dungeonu.

KillSquadMembers je pomocná třída, která obsahuje GameObject a počet kolik se ho má vygenerovat.

Výpis kódu 5.18 Třída KillSquadManager

```
public class KillSquadManager : MonoBehaviour
{
    [SerializeField] List<KillSquadMembers> killSquad = new List<KillSquadMembers>();

    public void AddKillSquad(GameObject prefab, uint count)
    {
        var squadMembers = killSquad.Find(member => member.squadMemberPrefab == prefab);
        if (squadMembers == null)
            killSquad.Add(new KillSquadMembers(prefab, count));
        else
            squadMembers.squadMemberCount += count;
    }

    public void RemoveKillSquad(GameObject prefab, uint count)
    {
        var squadMembers = killSquad.Find(member => member.squadMemberPrefab == prefab);
        if (squadMembers == null)
            killSquad.Remove(squadMembers);
        else
            squadMembers.squadMemberCount -= count;
    }

    public void ClearKillSquads()
    {
        killSquad.Clear();
    }
}
```

Kapitola 6
Testování

Poslední částí této práce je testování. Během testování se musí prověřit, jestli systém funguje, jak má, a jestli splňuje vytyčené požadavky. Také se při něm musí nalézt a zdokumentovat případné chyby. Uživatelské testování je rozděleno na dvě části: hráčské testování a vývojářské testování. V rámci obou testování hráči a vývojáři budou testovat implementovaný modul na základě testovacích scénářů. Poté budou odpovídat na předem připravené otázky, které se týkají průběhu testování.

6.1 Vývojářské testování

Vývojářské testování bude probíhat na základě testovacího scénáře pro vývojáře a jako pomůcka bude sloužit příručka pro vývojáře, která je v příloze této práce. V otázkách na konci testování se bude rozebírat, jak se vývojáři se systémem pracovalo, jestli narazil na chyby a případně na které.

6.1.1 Testovací scénář

Testovací scénář je rozepsán do jednotlivých kroků, během kterých bude vývojář postupně vytvářet nový cech a nastavovat nové vylepšování pro NPC, které již ve hlavní scéně existuje. Také si vyzkouší vytváření předmětů a vztahů mezi dárky a novým NPC.

V průběhu celého testování má vývojář k dispozici příručku pro vývojáře.

6.1.1.1 Nastavování dialogu pro vesničana

1. Vyberte si ve scéně jednoho z neinteraktivních NPC. Pro něj budete vytvářet dialog a nastavovat ho.
2. Přejděte do složky Assets/ScriptableObjects/Narration/Dialogues a vytvořte složku podle názvu vašeho budoucího vesničana.
3. Do složky pro zjednodušení zkopírujte obsah jiné složky například Assets/ScriptableObjects/Narration/Dialogues/Smith (Dialogový systém se netestuje).
4. Ve složce přejmenujte NarrationCharacter původního NPC na jméno vašeho vesničana.
5. Do Dialogue_start nastavte Intro_choices ze složky Nodes.
6. Přejděte do složky Lines.

7. Všude kromě SideQuestLine změňte Speaker na NarrationCharacter z předchozí složky. Texty můžete libovolně měnit. (Původní EmptyLine a GiftChoicesLine přejmenujte, aby název souhlasil s vaším NPC).
8. Přejděte do složky Assets/ScriptableObjects/Narration/Dialogues/*váš vesničan*/Nodes.
9. V SideQuestNode změňte Dialogue Line na SideQuestLine, které jste upravovali dříve a Next Node na Repeatable_choice z této složky.
10. Přejděte do složky Reputation a v GiftChoiceNode změňte DialogueLine na GiftChoicesLine, kterou jste upravovali dříve.
11. NotEnoughTools změňte na NotEnough*název vytvořené suroviny*, do Dialogue Line přidejte Empty Line, kteý jste předtím měnili a do Next Node Repeatable_choice z předchozí složky.
12. V ConfirmUpgrade změňte Dialogue Line stejně jako v předchozím bodě. Poté do Choices Yes dejte SkipSuccessUpgrade z této složky a do No Repeatable_choice z předchozí složky.
13. Vraťte se do složky Assets/ScriptableObjects/Narration/Dialogues/*váš vesničan*/.
14. V Intro_choices dejte do Dialogue Line vámi upravenou Entry_line, v Choices změňte Upgrade Armor na Upgrade *atribut který váš vesničan vylepšuje* a CheckStat nastavte na CheckStat ze složky Upgrade.
15. V Give Gift dejte GiftChoiceNode ze složky Reputation a do SideQuestNode dejte SideQuestNode z této složky.
16. To samé udělejte pro Repeatable_choice.

6.1.1.2 Vytváření vesničana

1. NPC, které jste si vybrali dříve přidejte skripty Interactable a ReputationScript.
2. U ReputationScriptu nastavte DialogueChannel, který již existuje a ReputationSlider podle cesty ve scéně UI/Dialogues/NameAndReputation/ReputationSlider.
3. Do Interactable do OnInteraction() dejte DialogueChannel.RaiseRequestDialogue a jako atribut Dialogue_start z nově vytvořeného dialogu.
4. V záložce Project přejděte do složky Assets/ScriptableObjects/Reputation/Guilds.
5. Vytvořte cech a nastavte mu atributy. Start Node a Repeatable Node budou z předtím vytvořeného dialogu (Intro_choices a Repeatable_choice). Milestone Objects se nebude nastavovat v rámci tohoto testování.
6. Přejděte do složky Assets/ScriptableObjects/Reputation/Villagers a vytvořte vesničana.
7. Nastavte mu cech, který jste vytvořili a jméno (ke vztahům s dárky se vrátíte později).

6.1.1.3 Vytváření předmětů a vztahů mezi dárky a vesničanem

1. Přejděte do složky Assets/ScriptableObjects/Items/Gifts, vytvořte dárek a dejte mu jméno.
2. Přejděte do složky Assets/ScriptableObjects/Items/Resources, vytvořte surovinu a dejte jí jméno.
3. Ve scéně vytvořte dva 3D objekty (například 3D cube) a přidejte jim skript ItemPickup (dejte je do scény tak aby je hráč mohl sebrat).

4. Do jednoho ItemPickup nastavte nově vytvořenou surovinu a počet. Do druhého dárek a počet.

5. Přejděte do složky Assets/ScriptableObjects/Reputation/GiftNpcRelationships a vytvořte složku s názvem vesničana, kterého jste vytvořili dříve.

6. V této složce vytvořte GiftNpcRelationship za každý dárek ve hře (momentálně 5 + vámi vytvořený) a nastavte mu dárek, změnu reputace a vámi vytvořeného vesničana.

7. Tyto vztahy přidejte do Gift Relationships vytvořeného vesničana.

6.1.1.4 Nastavení vylepšování

1. Přejděte do složky Assets/ScriptableObjects/Upgrades a vytvořte složku se jménem vašeho vesničana.

2. V této složce vytvořte UpgradeSettings a NpcUpgradeScript.

3. V UpgradeSettings nastavte atributy (cech a surovina byly vámi vytvořeny).

4. V NpcUpgradeScript nastavte atributy. UpgradeSettings budou vámi vytvořené.

5. Přesuňte se do složky Assets/ScriptableObjects/Narration/Dialogues/*váš vesničan*/Nodes/Upgrade.

6. V CheckStat změňte Callback na NpcUpgradeScript.CheckStat() z NpcUpgradeScript, který jste vytvořili dříve.

7. Ve SkipSuccessUpgrade změňte Next Node na Repeatable_choice z předchozí složky a Outer Callback na NpcUpgradeScript.UpgradeStat() z NpcUpgradeScript, který jste vytvořili dříve.

8. Nakonec spusťe hru, vyzkoušejte si sebrání předmětů, prohlédněte si, jak se změnili obrazovky reputace, dárků s vesničany a inventář. Ověřte, že se vaše změny na těchto obrazovkách projevili.

9. Najděte ve scéně nově nastaveného vesničana, zkuste mu dát dárek a vylepšit si u něj atribut.

6.1.2 Dotazník

1. Byla vám při testování užitečná příručka pro vývojáře?

2. Byla přehledná?

3. Dalo se v ní snadno najít co jste potřebovali?

4. Dalo se z ní pochopit, jak se se systémem pracuje?

5. Jaký byl váš celkový dojem z práce se systémem?

6. Našli jste při testování nějaké chyby a jaké?

6.1.3 Výsledky testování

Vzhledem k tomu, že jako vývojáři testovali pouze Roman Španko a Jiří Macháček, bylo testování pojato více osobní formou než testování hráčské. Testování probíhalo osobně a otázky z dotazníku byly zodpovězeny verbálně.

Hlavní problémy, které se našly během vývojářského testování, byly s testovacím scénářem. První problém byl s pochopením, která scéna je hlavní. Autor scénáře měl na mysli MainHubScene v cestě Assets/Scenes/CURRENT.

Další problém, na který poukazoval hlavně Roman Španko, byl s prací s dialogovým systémem, které bylo relativně zdlouhavé. Tento problém není až tak závažný, protože dialogový systém nepatří do rozsahu této práce (ačkoliv je pro fungování systému potřebný).

Jedna z připomínek při testování byla, že testovací scénář není popsaný dostatečně podrobně. Například, že v jedné instrukci se něco mění v jedné složce, ve druhé instrukci se něco mění ve druhé složce, ale mezi nimi není popsán přesun mezi těmito složkami.

Ve kroku 16 u Nastavování dialogu pro vesničana nebylo jasné, co se myslí instrukcí "To samé udělejte pro Repeatable choice".

Během nastavování dialogu se mluví o názvu vytvořené suroviny, která se ale předtím nezmiňuje a tester si musí domyslet co tím autor zamýšlel.

Některé instrukce by mohli být popsány více podrobně. Například, že skripty na NPC musí být nastaveny jako enabled a po změně ve scéně se musí pomocí kláves ctrl+s scéna uložit.

Při nastavování částí dialogu by bylo přehlednější, kdyby byl název vesničana v názvu jednotlivých částí (například SmithRepeatableChoice) pro lepší vyhledávání.

Uživatelská příručka byla pro oba testery užitečná až, kvůli povaze instrukcí, potřebná. Pro jednoho testera byla přehledná, pro druhého méně. Při testování se našli nějaké chyby, ale po přezkoumání nešlo o chyby v systému, ale přeskočení nebo špatné pochopení instrukcí. Jednou přestalo fungovat Unity, ale pravděpodobně byl tento problém věcí Unity, a ne testovaného systému.

Práce se systémem byla pro oba testery, až na malé výhrady ohledně práce s dialogovým systémem, jednoduchá. Nicméně se vyskytl návrh, že by bylo příjemnější, aby byla práce se systémem zaměřena na méně Inspector oken, ve kterých by se nastavovalo více věcí.

6.2 Hráčské testování

Hráčské testování bude probíhat na základě testovacího scénáře pro hráče. K dispozici budou hráči mít nejnovější verzi MagiTechu. Do dotazníku po splnění testovacího scénáře vyplní, jestli bylo vše ve hře jasné a jak na ně hra působila.

6.2.1 Testovací scénář

1. Zapněte hru. V dialogu pokračujete stiskem mezerníku a klávesou escape dostanete přístup k popisu ovládání a vysvětlivkám ke hře.
2. Prohlédněte si dostupné obrazovky pomocí klikání a klávesových zkratek.
3. Najděte NPC, se kterými si můžete povídat. Je jich ve vesnici 6 a mají nad hlavou šipku, když se přiblížíte.
4. Promluvte si s NPC (klávesa F). Zkuste si vylepšit atribut a dát mu dárek (i když zatím nemáte žádné předměty).
5. Seberte vedlejší quest abyste mohli do dungeonu. Alternativně najděte hospodského nebo starostu a vezměte si od nich hlavní quest.

6. Pomocí obrazovky společníků zkuste vybrat a aktivovat jednoho nebo více společníků.
7. Podle mapy (klávesa M) najděte cestu do dungeonu a vejděte do něj.
8. Projděte si dungeon a po cestě sbírejte předměty, které uvidíte.
9. Vraťte se zpět do vesnice (stejnou cestou, nebo splňte quest).
10. Znovu si promluvte s vesničany, zkuste si vylepšit za nasbírané předměty atributy. Zkuste jim dát dárky co jste sebrali.
11. Ověřte na obrazovkách atributů, že se změny po vylepšení projevily.
12. Po věnování dárku ověřte, že se zapsala transakce do deníku dárků a že se reputace změnila.
13. Po skončení testování prosím vyplňte dotazník. Děkuji za testování!

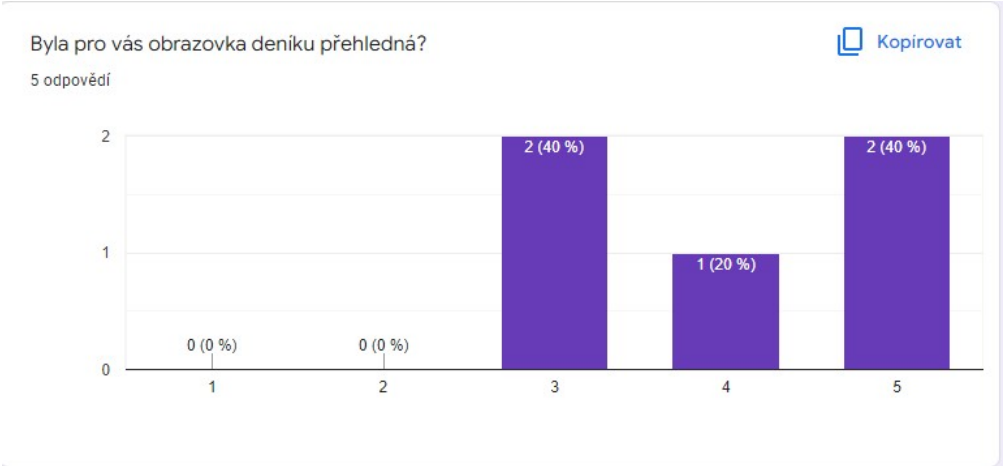
6.2.2 Dotazník

1. Byla pro vás obrazovka deníku přehledná (1-ne, 5-ano)?
2. Máte nějaké připomínky k obrazovce deníku? Našli jste v ní nějaké chyby?
3. Byla pro vás obrazovka společníků přehledná (1-ne, 5-ano)?
4. Máte nějaké připomínky k obrazovce společníků? Našli jste v ní nějaké chyby?
5. Byla pro vás obrazovka reputací přehledná (1-ne, 5-ano)?
6. Máte nějaké připomínky k obrazovce reputací? Našli jste v ní nějaké chyby?
7. Bylo pro vás jednoduché vylepšit svou postavu pomocí dialogu s NPC (1-ne, 5-ano)? (včetně sebrání suroviny v dungeonu)
8. Máte nějaké připomínky k vylepšování hráče? Našli jste nějaké chyby?
9. Bylo pro vás jednoduché darovat dárek pomocí dialogu s NPC (1-ne, 5-ano)? (včetně sebrání dárku v dungeonu)
10. Máte nějaké připomínky k dávání dárků? Našli jste nějaké chyby?
11. Bylo pro vás jednoduché aktivovat a ovládat společníky (1-ne, 5-ano)?
12. Máte nějaké připomínky k ovládání společníků? Našli jste nějaké chyby?
13. Jaký byl váš celkový dojem z hraní MagiTechu (1-hra mě nebavila, 10-hra mě bavila)? (zaměřte se na interakci s NPC)
14. Máte nějaké obecné připomínky? Návrhy na vylepšení? Našli jste nějaké další chyby?

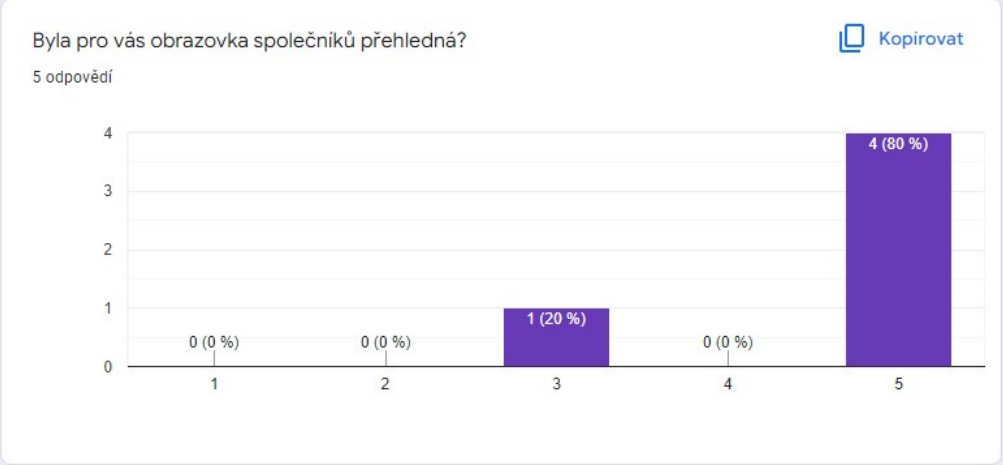
6.2.3 Výsledky dotazníku

Hráčský dotazník vyplnilo 5 testerů. Odpovědi na otevřené otázky i na otázky s lineární stupnicí byly vesměs různé. V otevřených otázkách se testeři často vyjadřovali i k problémům nebo chybám, které ve hře jsou, ale nejsou v rozsahu této práce. Lineární stupnice měla hodnoty 1 až 5 s tím, že 1 je nejvíce negativní a 5 je nejvíce pozitivní odpověď. Stejně to je i u poslední otázky na celkový dojem z MagiTechu, ale tam je stupnice od 1 do 10 (10 je nejvíce pozitivní).

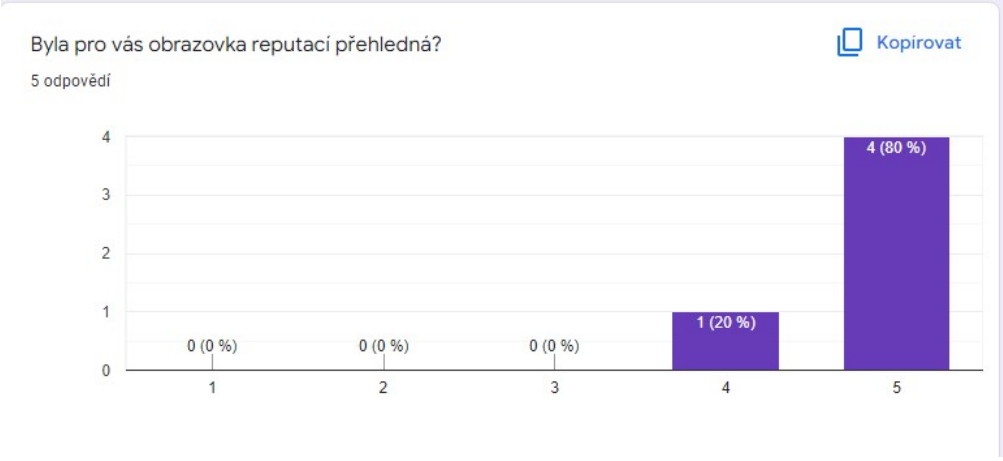
Obrázek 6.1 Otázka na přehlednost deníku



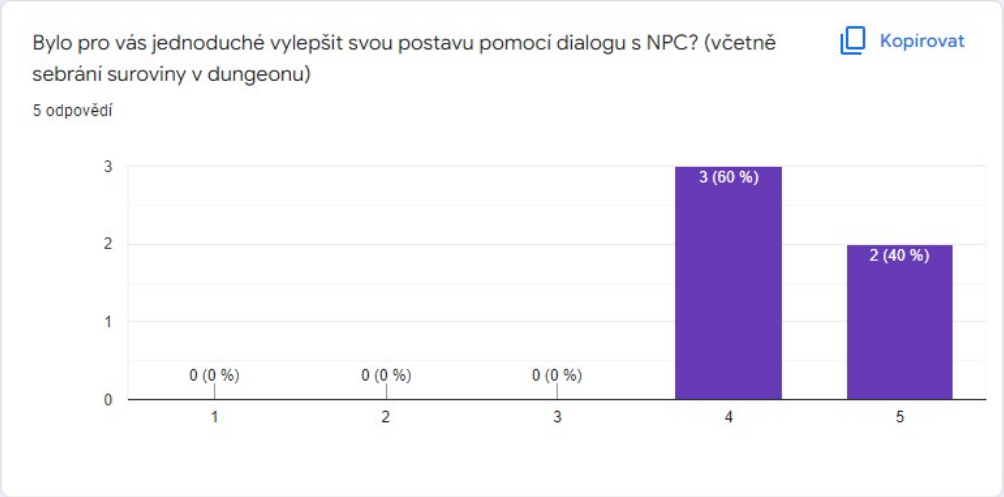
Obrázek 6.2 Otázka na přehlednost obrazovky společníků



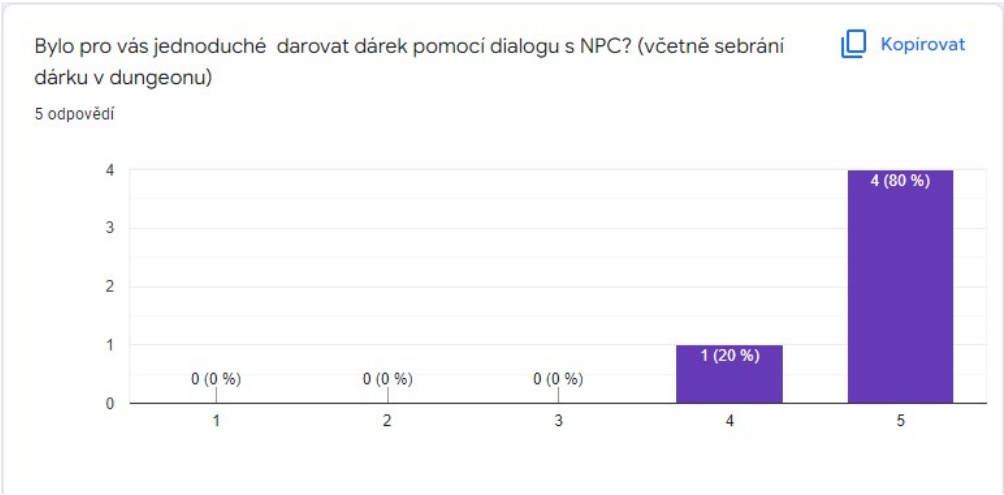
Obrázek 6.3 Otázka na obrazovku reputací



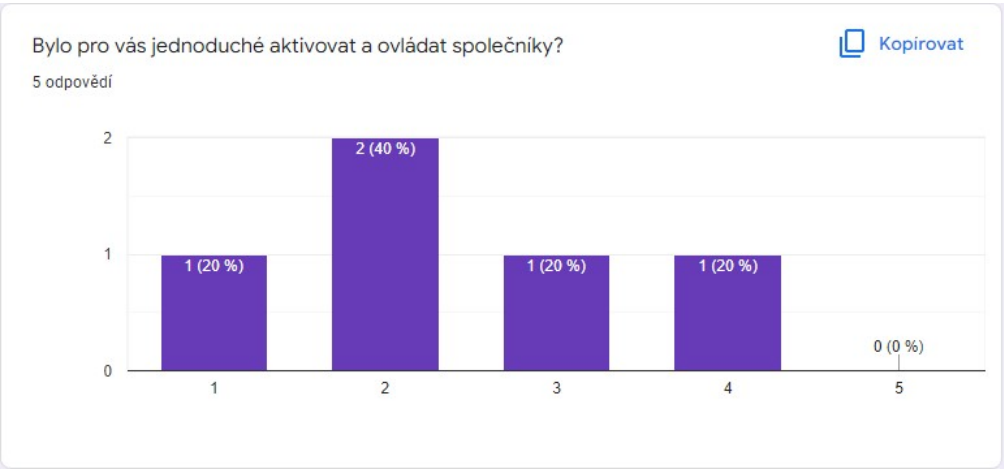
Obrázek 6.4 Otázka na vylepšování hráče



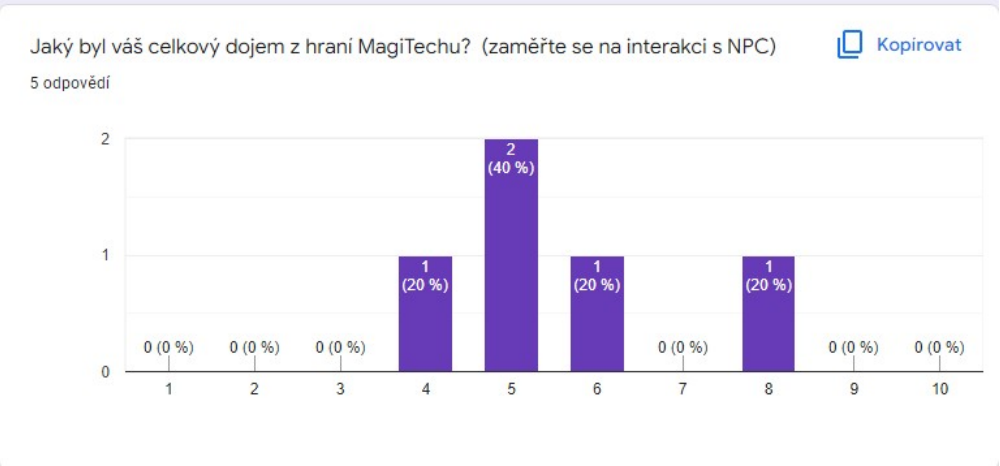
Obrázek 6.5 Otázka na věnování dárků



Obrázek 6.6 Otázka na ovladatelnost společníků



Obrázek 6.7 Otázka na celkový dojem z MagiTechu



6.2.3.1 Odpovědi na otevřené otázky

Některé odpovědi se netýkaly tolik fungování systému implementovaného v rámci této práce, ale fungování zbytku MagiTechu.

První chyba, ke které se vyjádřili dva testeři je, že během klikání na obrazovky v menu postava hráče útočí. O této chybě se ví, ale nepřikládá se jí velká hodnota, protože hru nijak nerozbíjí a jenom mate hráče.

V jedné otevřené otázce se vyskytla připomínka, že nebylo snadné ve vesnici nalézt vesničany, se kterými by se dalo interagovat. Tento problém by se dal vyřešit větším množstvím NPC ve vesnici, ale pokud by hráč hrál delší dobu, tak by neměl mít problém si pozice vesničanů zapamatovat.

Jeden nápad k uživatelskému rozhraní byl přidat tlačítko pro zavření obrazovek. Momentálně se obrazovky dají zavřít tak, že se klikne na tlačítko momentálně aktivní obrazovky.

Další problém s obrazovkou inventáře byl, že scrollování v seznamu předmětů není dostatečně rychlé.

Několik testerů hlásilo problémy s kamerou. Při přecházení ze scén se kamera natáčí směrem, aby byla směrem od místa kam hráč přešel. To je udělané záměrně, ale stálo by za to udělat změnu, aby hráč neměl problémy s orientací ve scéně. V jeskyním dungeonu překážel ve výhledu strop a stěny. Momentálně jsou průhledné všechny stěny, které překážejí v pohledu na hráče, ale ne stěny, které jsou například těsně vedle hráče.

V dungeonu vygenerovaným hlavním úkolem pro hospodského nejsou předměty. Tento problém byl již opraven, ale v pozdější verzi, než která se používala na testování.

Testerům vadilo, že se nechtěně přemístili z dungeonu, pokud se přiblížili k místu, kterým do dungeonu přišli. O tomto problému se ví a do budoucna by se řešil vytvořením dialogového okna pro vstup a výstup z dungeonu.

Vyvažování není dostatečně promyšlené (po několika vylepšeních je hráč příliš silný). Vyvažování síly hráče, nepřátel a společníků nemá tak vysokou prioritu, vzhledem k tomu, že MagiTech hrálo velmi omezené množství lidí.

Některá kouzla mají cooldown, ale ten není hráči nijak komunikován. Práci na některých elementech uživatelského rozhraní bylo věnováno méně času než práci na logice a fungování hry. To se týká i tlačítek na ovládání společníků, které nedávají hráči vizuální nebo zvukový feedback, jestli byly stisknuty.

V obrazovce se společníky by bylo přehlednější, kdyby se tlačítko ”activate companion”měnilo na ”deactivate companion”, když se má společník deaktivovat.

Předměty, které lze sebrat, by měly být pro přehlednost rozeznatelné od textur v dungeonech.

Ovládání společníků by mělo být lépe vysvětlené.

Ve hře chybí save/load a nastavování hlasitosti. Tyto features v MagiTechu zatím nejsou implementovány.

Obecně se během testování našlo hodně chyb, které nesouvisí se systémem implementovaným v rámci této práce, ale i tak byly pro potenciální budoucí vývoj MagiTechu velmi přínosné. Hlavní problémy, které hráči měly s tímto systémem, se týkaly uživatelského rozhraní a ovládání společníků, které by mělo být hráči lépe vysvětleno.

Kapitola 7
Závěr

Cílem práce bylo analyzovat, navrhnout, implementovat a testovat modul pro interakci hráče s nehráčskými postavami pro hru MagiTech.

V rámci analýzy se rozebraly existující řešení her World of Warcraft, Stardew Valley a Mount and Blade. Dále se rozebraly jednotlivé aspekty interakce hráče s NPC a faktory, které ji ovlivňují.

V návrhu se vypracovaly doménové a třídní diagramy a diagramy případů užití pro jednotlivé části modulu. Také byly vytvořeny návrhy obrazovek, se kterými hráč ve hře interaguje.

V implementaci se vypracovalo řešení pokrývající většinu požadavků, ale požadavky FA2 – FA6 nebyly vypracovány kvůli nedostatku času, a protože měli nižší prioritu. Požadavek FA1 byl pokryt z důvodu otestování systému milníků reputace. Také se nevyhotovily požadavky FG6 a FG7 ze stejných důvodů jako požadavky FA2 – FA6.

Do budoucna se kromě implementace nesplněných požadavků může pokračovat komplexnějším systémem interakce, nebo hledáním způsobu, jak systém zobecnit pro širší spektrum her.

Literatura

[1] *MagiTech repozitář* [online]. [cit. 2022-04-04]. Dostupné z: https://gitlab.fit.cvut.cz/BI-VHS/b221projects/ardastea/tree/master/doc/

[2] *Roguelike* [online]. Gamer Network Limited, a ReedPop company., c2022 [cit. 2022-04-03]. Dostupné z: https://www.usgamer.net/articles/the-gateway-guide-to-roguelikes.

[3] *Dungeon crawler* [online]. IEEE, c2021 [cit. 2022-04-03]. Dostupné z: https://insight.ieeeusa.org/articles/going-rogue-a-brief-history-of-the-computerized-dungeon-crawl/.

[4] *World of Warcraft* [online]. BLIZZARD ENTERTAINMENT, c2022 [cit. 2022-04-04]. Dostupné z: https://worldofwarcraft.com/en-us/.

[5] *Stardew Valley* [online]. ConcernedApe, c2016-2022 [cit. 2022-04-04]. Dostupné z: https://www.stardewvalley.net/

[6] *Mount and Blade 2: Bannerlord* [online]. TaleWorlds Entertainment, c2005-2022 [cit. 2022-04-04]. Dostupné z: https://www.taleworlds.com/en/Games/Bannerlord

[7] *What is NPC* [online]. c2022 [cit. 2022-04-03]. Dostupné z: https://www.businessinsider.com/npc-meaning.

[8] *NPC Interaction* [online]. 2021 [cit. 2022-05-01]. Dostupné z: https://www.roleplayingtips.com/npcs-roleplaying/rpt265-crafting-npc-interaction-plans-designing-encounters-players-mind/

[9] *Reputation* [online]. GIANT BOMB, c2022 [cit. 2022-05-01]. Dostupné z: https://www.giantbomb.com/reputation/3015-2741/

[10] *Hráčské rozhodnutí* [online]. London: Informa PLC Informa UK Limited, c2022 [cit. 2022-04-04]. Dostupné z: https://www.gamedeveloper.com/design/improving-player-choices

[11] *Fáze hry* [online]. London: Informa PLC Informa UK Limited, c2022 [cit. 2022-04-04]. Dostupné z: https://www.gamedeveloper.com/design/phases-in-games

[12] *NPC ve World of Warcraft* [online]. [cit. 2022-05-01]. Dostupné z: https://wowpedia.fandom.com/wiki/NPC

[13] *F.A.Q k NPC ve World of Warcraft* [online]. [cit. 2022-05-01]. Dostupné z: http://web.archive.org/web/20070602140349/http://www.worldofwarcraft.com/info/faq/npcs.html

[14] *Reputace ve World of Warcraft* [online]. [cit. 2022-05-01]. Dostupné z: https://wowpedia.fandom.com/wiki/Reputation

[15] *Pets ve World of Warcraft* [online]. [cit. 2022-05-01]. Dostupné z: https://wowpedia.fandom.com/wiki/Pet?so=search

[16] *Companions v Mount and Blade 2: Bannerlord* [online]. [cit. 2022-05-01]. Dostupné z: https://mountandblade.fandom.com/wiki/Companions_(Bannerlord)

[17] *Notables v Mount and Blade 2: Bannerlord* [online]. [cit. 2022-05-01]. Dostupné z: https://mountandblade.fandom.com/wiki/Notables

[18] *Vesničané ve Stardew Valley* [online]. [cit. 2022-05-01]. Dostupné z: https://stardewvalleywiki.com/Villagers

[19] *Dárky ve Stardew Valley* [online]. [cit. 2022-05-01]. Dostupné z: https://stardewvalleywiki.com/Friendship

[20] *Unity MonoBehaviour* [online]. Unity Technologies, c2022 [cit. 2022-04-23]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[21] *Unity ScriptableObject* [online]. Unity Technologies, c2022 [cit. 2022-04-23]. Dostupné z: https://docs.unity3d.com/Manual/class-ScriptableObject.html

Příloha A

Příručka pro vývojáře

Tato příručka popisuje, jakým způsobem pracovat s modulem pro interakci s NPC pro hru MagiTech. Obsahuje informace o tom, co musí vývojář vytvořit a nastavit, aby dosáhl výsledků. Je tedy určena pro vývojáře hry MagiTech, ale potenciálně pokud se modul rozšíří, tak i pro vývojáře jiných her.

Příručka je rozdělena na tři hlavní části: Vylepšování hráče, Reputace s cechy a společníci. Modul není, až na obrazovky, zaměřen na grafiku, takže se předpokládá, že ve hře již jsou objekty (například modely NPC), na které se mohou případné skripty připojit.

Také se v příručce spoléhá na to, že existují a jsou správně nastaveny systémy, které byly do MagiTechu přidány dříve, například dialogový systém.

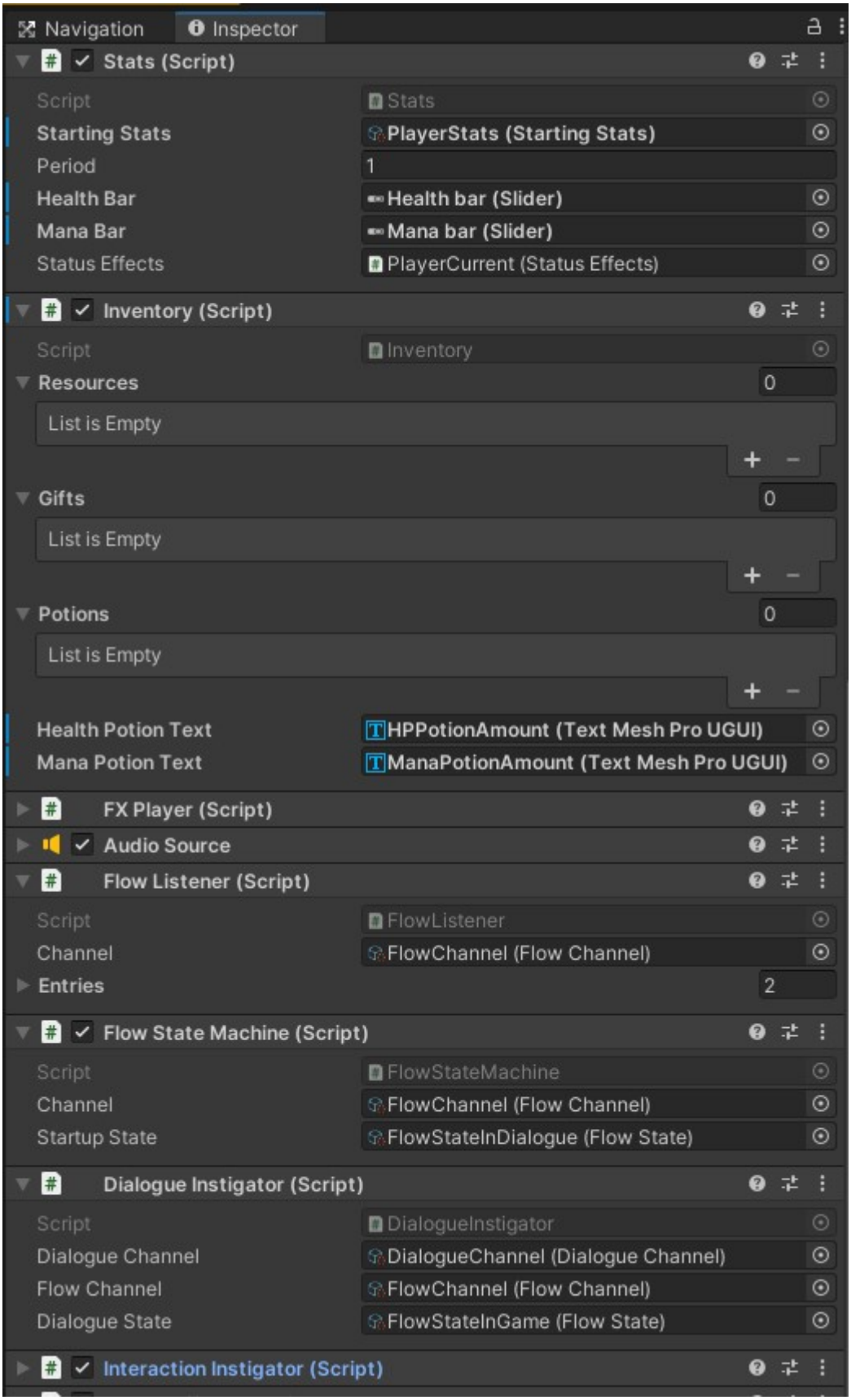
A.1 Vylepšování hráče

V této části se vysvětluje, jak dosáhnout vylepšování hráče u NPC. Počítá se předem s tím, že existuje postava hráče, která má jako komponentu připojené a nastavené skripty:

1. Stats, aby hráč měl atributy, které by se daly vylepšovat
2. Inventory, aby se hráči ukládaly suroviny a mohl je dávat NPC
3. FlowListener, FlowStateMachine, DialogueInstigator, InteractionInstigator. Tyto skripty jsou potřeba pro začínání a průběh dialogu s NPC

Dále se počítá s tím, že NPC, které se nastavuje má vytvořený model a na něm jako komponentu připojený skript Interactable, který má nastavený začátek dialogu.

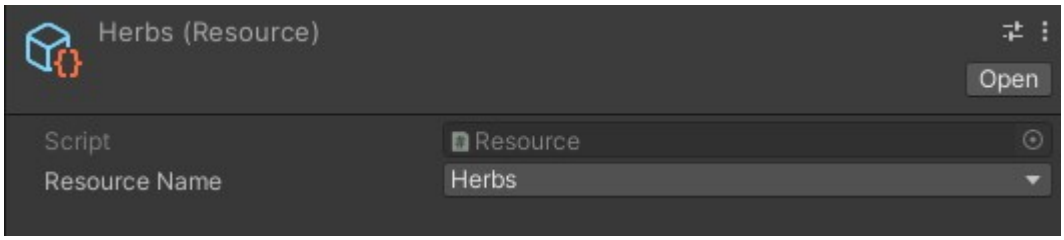
Obrázek A.1 Unity inspektor hráče



A.1.1 Vytváření surovin

Suroviny jsou v MagiTechu implementované jako scriptableObject, tedy se vytváří jako nový asset. Nová surovina se vytvoří pomocí cesty Create/Scriptable Objects/Item/Resource. Surovina má název, který je implementovaný jako Enum, takže do skriptu Resource je potřeba do enumu ResourceName přidat název nové suroviny. Tento název poté vybrat v nastavení suroviny.

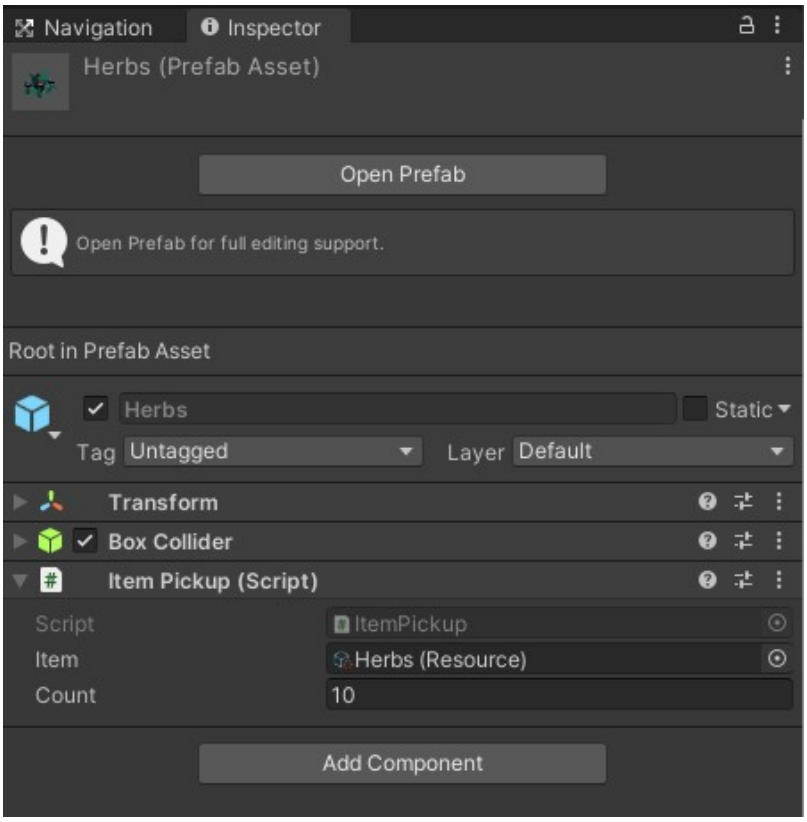
Obrázek A.2 Unity inspektor suroviny



Další krok je nastavit předmět, který bude hráč sbírat v dungeonu. Je předpoklad, že model předmětu je již vytvořen a má nastavený boxCollider. Na tento předmět je potřeba jako komponentu připojit skript ItemPickup. Tento skript má atributy:

1. Item – tam vyplnit surovinu, která se má sebrat
2. Count – vyplnit kolik z dané suroviny se má sebrat

Obrázek A.3 Unity inspektor sebratelné suroviny



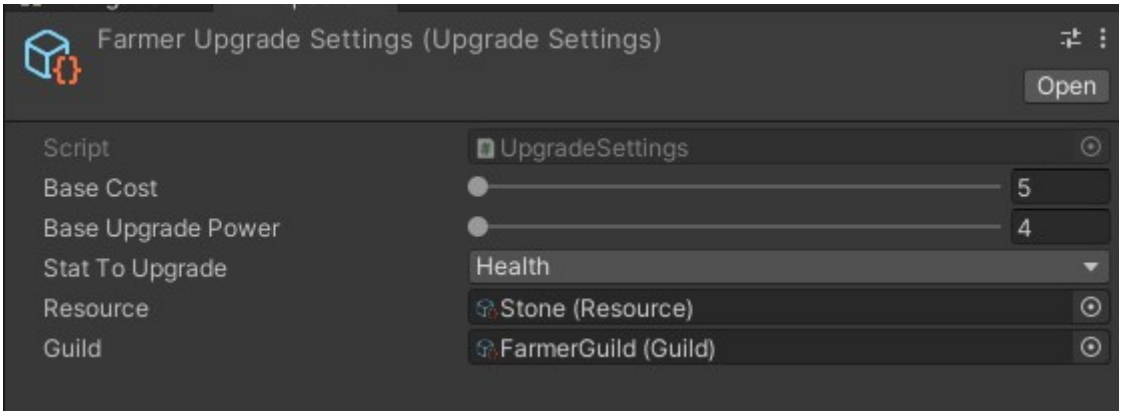
A.1.2 Vytváření a nastavování objektů pro vylepšování

Pro vylepšování u konkrétního NPC je potřeba vytvořit jako scriptableObject NpcUpgradeScript a UpgradeSettings.

UpgradeSettings se vytvoří pomocí cesty Create/Scriptable Object/Upgrades/Upgrade Settings. Je potřeba vyplnit následující atributy:

1. Base Cost – základní cena za kolik surovin se vylepšení pořídí
2. Base Upgrade Power – o kolik se zvýší atribut
3. Stat To Upgrade – který atribut se má zvýšit (reprezentováno jako Enum)
4. Resource – surovina za kterou se vylepšuje
5. Guild – cech, do kterého NPC patří (viz. vytváření a nastavování cechů)

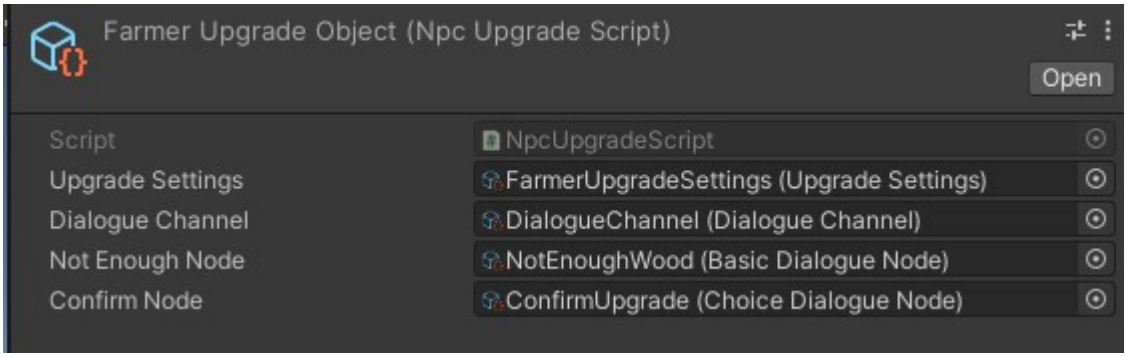
Obrázek A.4 Unity inspektor nastavení vylepšování



NpcUpgradeScript se vytvoří pomocí cesty Create/Scriptable Object/Upgrades/NPC Upgrade. Je potřeba vyplnit následující atributy:

1. Upgrade Settings – UpgradeSettings odpovídající danému NPC
2. Dialogue Channel – potřeba pro dialogový systém, počítá se s existencí
3. Not Enough Node – potřeba pro dialogový systém, jedná se o BasicDialogueNode, která reprezentuje část dialogu, kdy hráč nemá dostatek surovin
4. ConfirmUpgrade – potřeba pro dialogový systém, jedná se o ChoiceDialogueNode, která reprezentuje část dialogu, kdy se hráč rozhoduje jestli opravdu koupí vylepšení

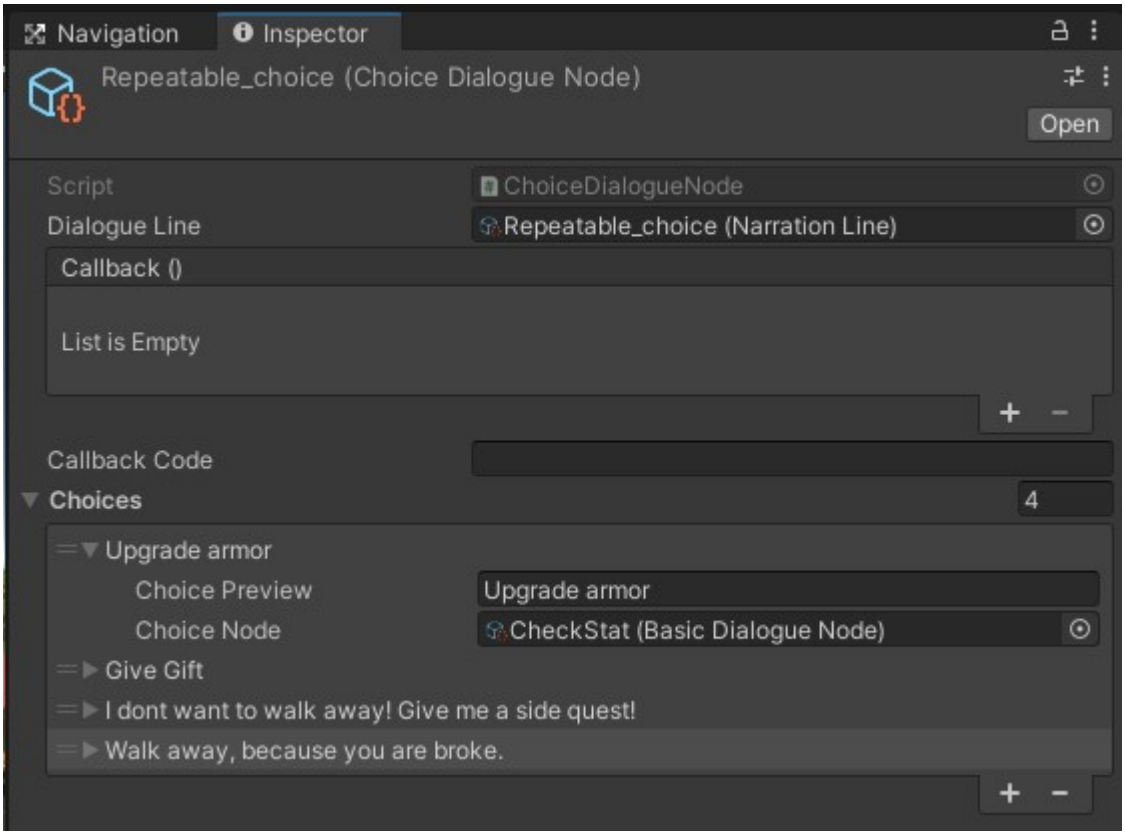
Obrázek A.5 Unity inspektor skriptu vylepšování



A.1.3 Vytváření a nastavování dialogů

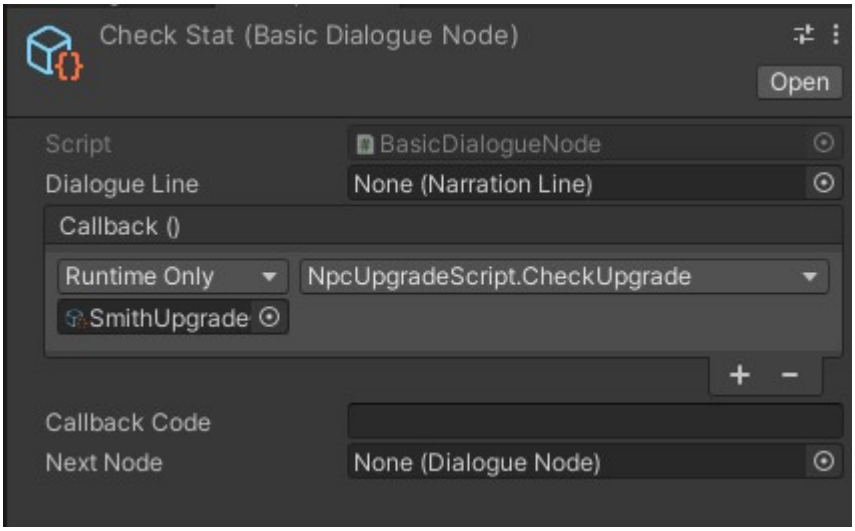
V této části se počítá s tím, že NPC má fungující dialog a je potřeba přidat dialogovou možnost k vylepšení hráče. Když se dále zmiňuje ”hlavní ChoiceDialogueNode”myslí se tím node, ke které se hráč v rámci dialogu vrací a kterou vidí při startu dialogu.

Obrázek A.6 Unity inspektor hlavního dialogu NPC



Nejprve je potřeba vytvořit BasicDialogueNode, které se jako Callback() nastaví metoda NpcUpgradeScript.CheckUpgrade(). NpcUpgradeScript musí odpovídat NPC, kterému patří vytvářený dialog. Tato BasicDialogueNode se poté přidá jako nová choice do hlavní ChoiceDialogueNode.

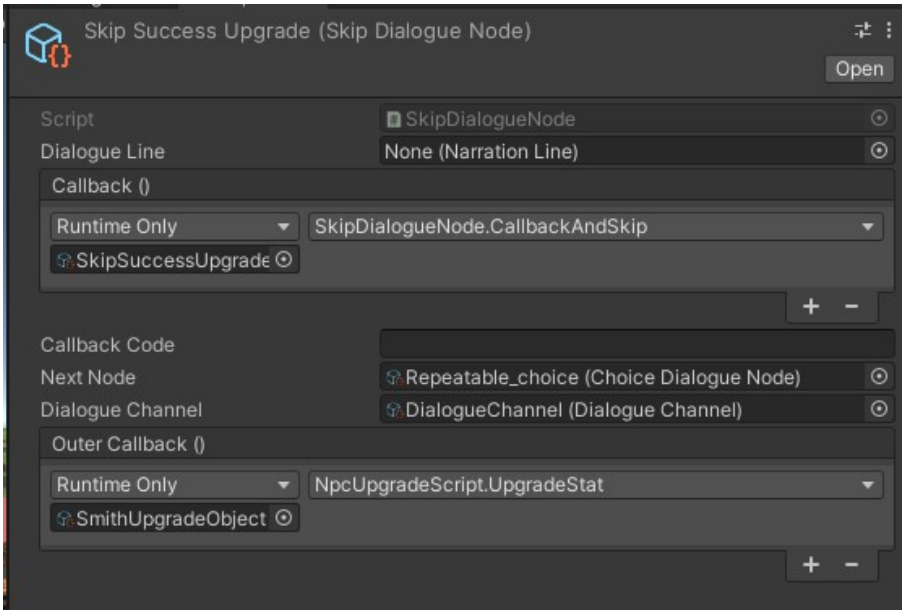
Obrázek A.7 Unity inspektor vylepšení v dialogu



Poté se musí vytvořit SkipDialogueNode která bude mít následující nastavení:

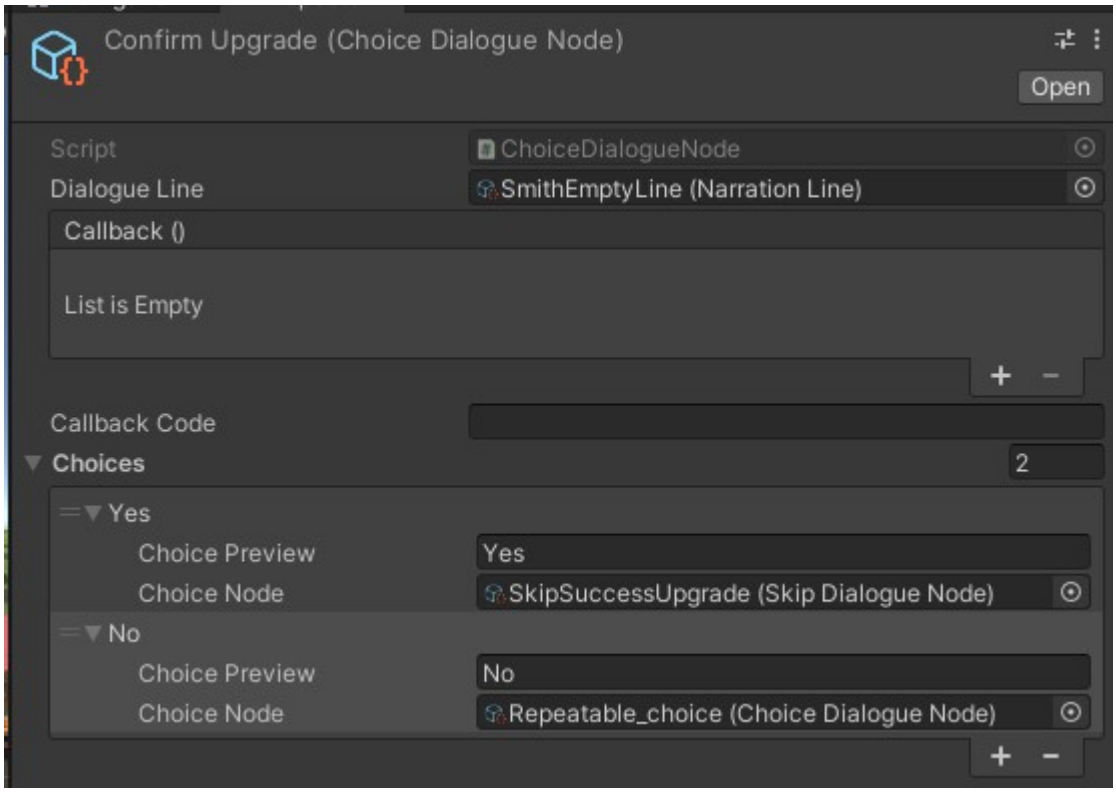
1. Callback() bude nastaven na SkipDialogueNode.CallbackAndSkip s tím, že tato metoda musí být volána z té samé SkipDialogueNode
2. NextNode – hlavní ChoiceDialogeChoice, do které se má dialog vrátit
3. DialogueChannel – potřeba pro dialogový systém
4. OuterCallback() bude nastaven na NpcUpgradeScript.UpgradeStat. NpcUpgradeScript musí odpovídat NPC, kterému patří vytvářený dialog.

Obrázek A.8 Unity inspektor dialogu pro aktivaci vylepšení



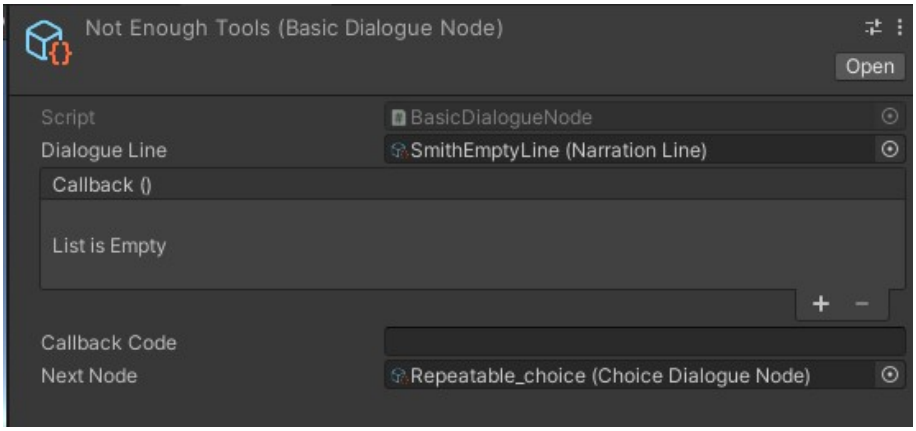
Dále je potřeba vytvořit ChoiceDialogueNode, kde si hráč může vybrat, jestli chce doopravdy koupit surovinu. Té je potřeba přidat prázdnou NarrationLine (text se nastaví sám ve skriptu) a do Choices možnost pro zamítnutí a potvrzení. Pro potvrzení to bude SkipDialogueNode vytvořena výše, která se postará o vylepšení atributu a pro zamítnutí to bude hlavní ChoiceDialogueNode.

Obrázek A.9 Unity inspektor dialogu pro potvrzení vylepšení



Nakonec je potřeba vytvořit BasicDialogueNode, která reprezentuje, že hráč nemá dost surovin. Bude mít nastavenou prázdnou NarrationLine (text se nastaví sám ve skriptu) a Next Node musí být hlavní ChoiceDialogueNode.

Obrázek A.10 Unity inspektor dialogu pro nedostatek suroviny



A.2 Reputace s cechy

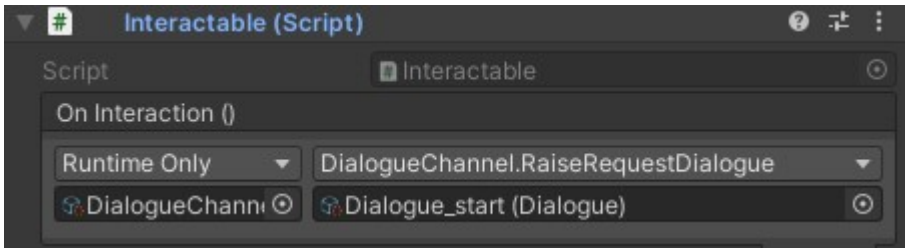
V této části se popíše, jak se vytváří cechy a ostatní části modulu, které s nimi souvisí.

Počítá se předem s tím, že existuje postava hráče, která má jako komponentu připojené a nastavené skripty:

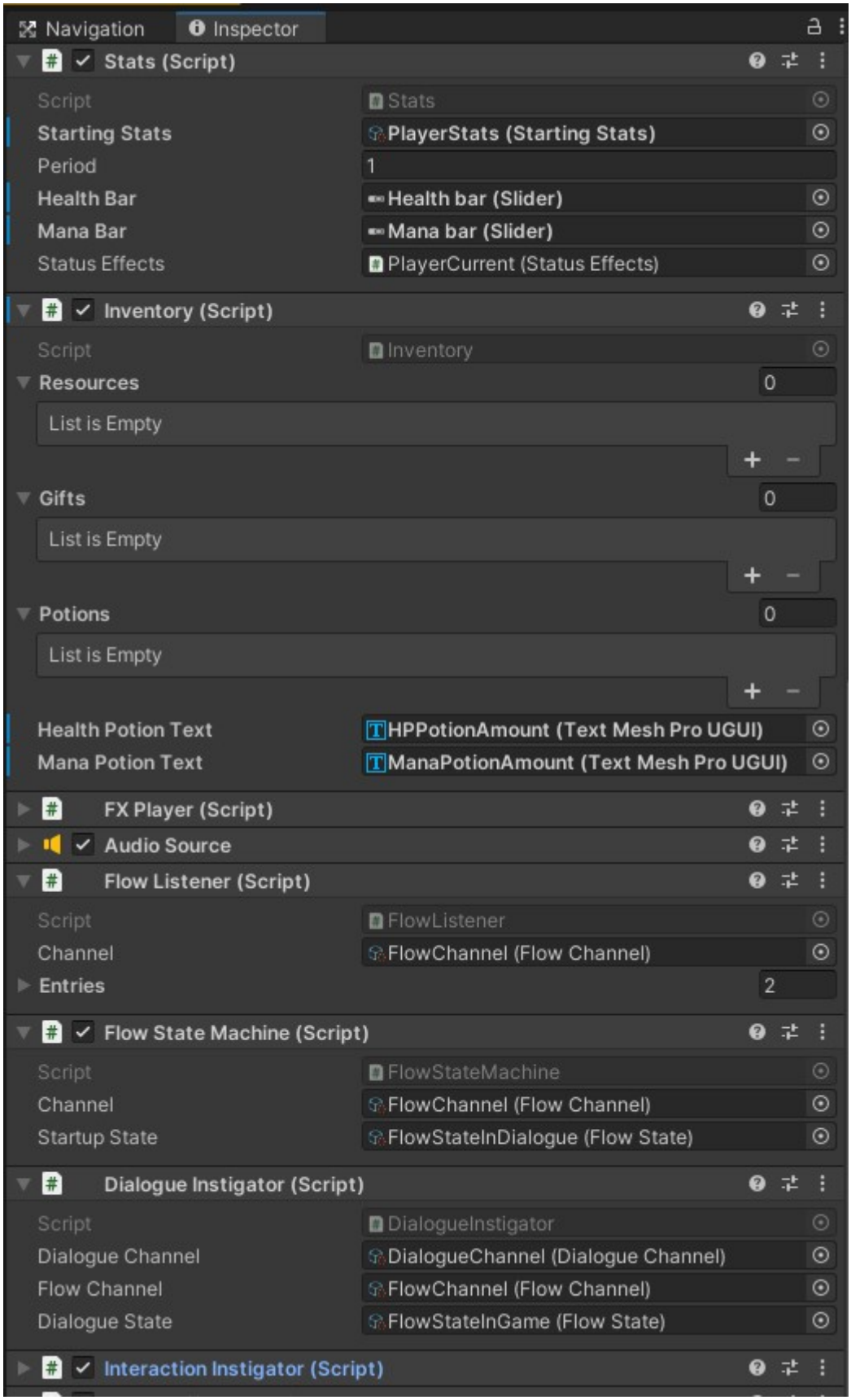
1. Inventory, aby se hráči ukládaly dárky a mohl je dávat NPC
2. FlowListener, FlowStateMachine, DialogueInstigator, InteractionInstigator. Tyto skripty jsou potřeba pro začínání a průběh dialogu s NPC

Dále se počítá s tím, že NPC, které se nastavuje má vytvořený model a na něm jako komponentu připojený skript Interactable, který má nastavený začátek dialogu.

Obrázek A.11 Unity inspektor Interactable u NPC



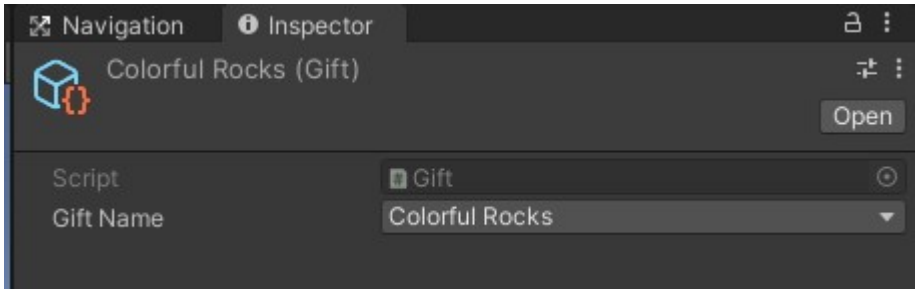
Obrázek A.12 Unity inspektor hráče



A.2.1 Vytváření dárků

Nový dárek se vytvoří pomocí cesty Create/Scriptable Objects/Item/Gift. Dárek má název, který je implementovaný jako Enum, takže do skriptu Gift je potřeba do enumu GiftName přidat název nového dárku. Tento název poté vybrat v nastavení dárku.

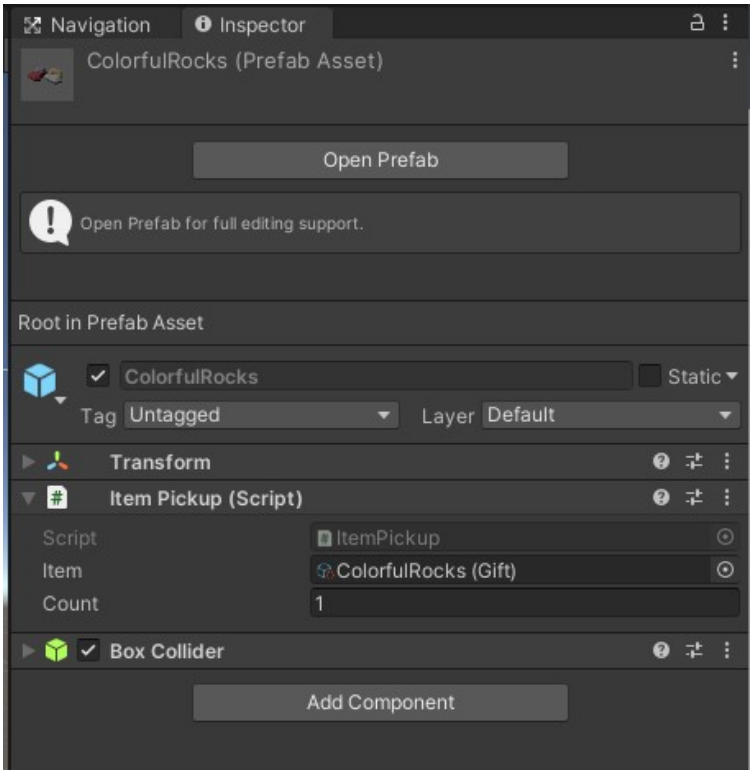
Obrázek A.13 Unity inspektor dárku



Další krok je nastavit předmět, který bude hráč sbírat v dungeonu. Je předpoklad, že model předmětu je již vytvořen a má nastavený boxCollider. Na tento předmět je potřeba jako komponentu připojit skript ItemPickup. Tento skript má atributy:

1. Item – vyplnit dárek, který se má sebrat
2. Count – vyplnit kolik z daného dárku se má sebrat

Obrázek A.14 Unity inspektor sebratelného dárku

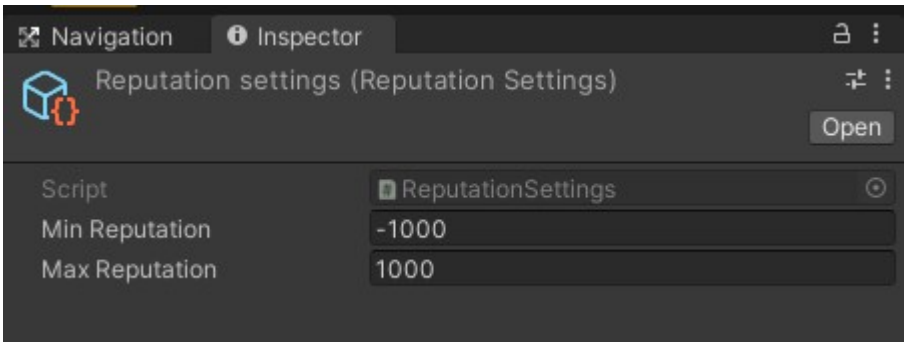


A.2.2 Nastavení hranic reputace

Aby fungovaly ukazatele reputace ve hře, je potřeba vytvořit ReputationSettings pomocí cesty Create/Scriptable Objects/Reputation/Settings.

ReputationSettings má MinReputation a MaxReputation, které je potřeba nastavit jako dosažitelné minimum a maximum.

Obrázek A.15 Unity inspektor nastavení reputace

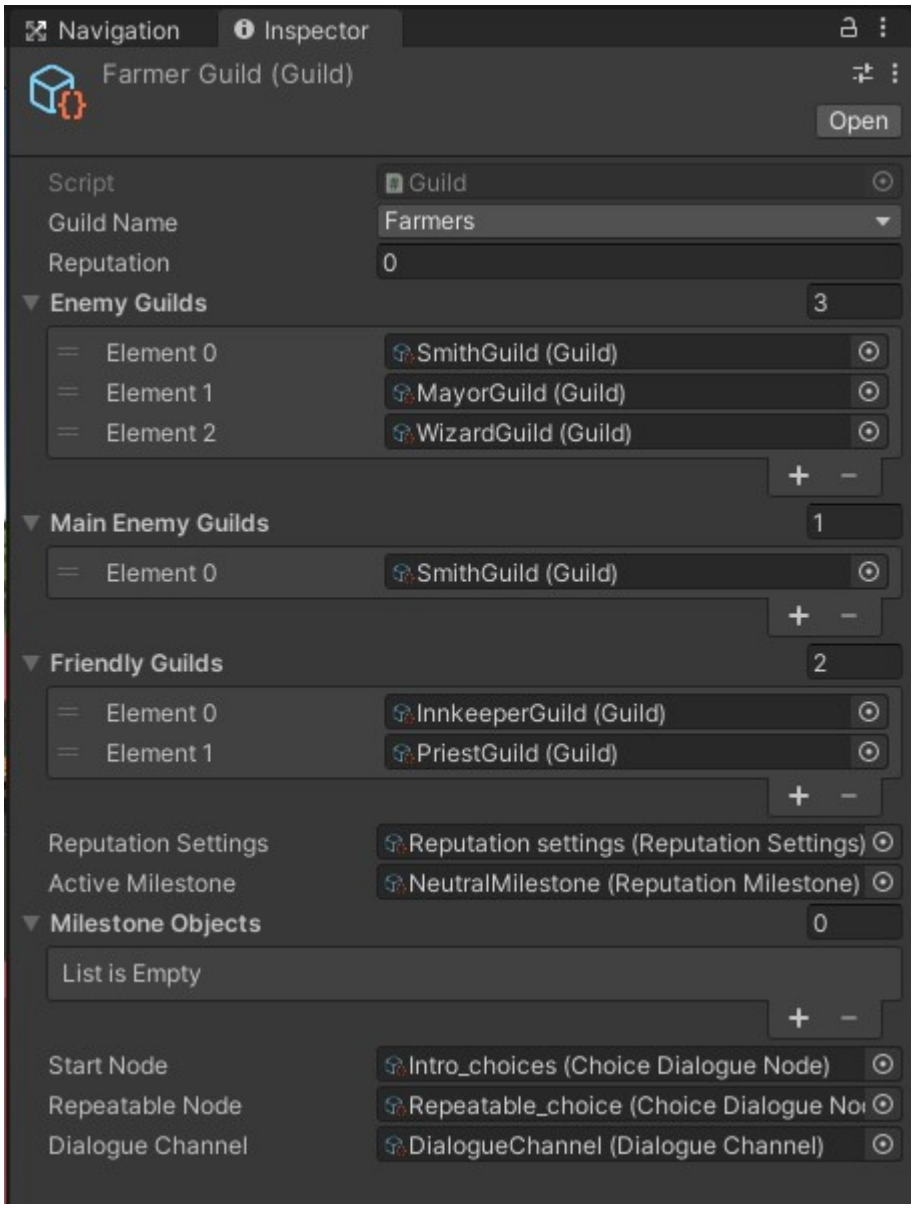


A.2.3 Vytváření a nastavování cechů

Nový cech se vytvoří pomocí cesty Create/Scriptable Objects/Guild a má následující atributy:

1. Guild Name – jméno cechu reprezentované jako Enum. Nový název je třeba přidat do enumu GuildName v Guild skriptu
2. Reputation – vyplnit startovní reputaci s cechem
3. EnemyGuilds – vyplnit znepřátelené cechy
4. MainEnemyGuilds – vyplnit hlavní znepřátelené cechy
5. FriendlyGuild – vyplnit spřátelené cechy
6. ReputationSettings – vyplnit výše vytvořené ReputationSettings
7. ActiveMilestone – vyplnit aktivní milník reputace (viz. Vytváření a nastavování milníků reputace)
8. MilestoneObjects – vyplnit objekty milníků (viz. Vytváření a nastavování milníků reputace)
9. StartNode – první ChoiceDialogueNode, která se aktivuje při dialogu s NPC
10. RepeatableNode – hlavní opakující se ChoiceDialogueNode, která se aktivuje při dialogu s NPC
11. DialogueChannel – potřeba pro dialogový systém

Obrázek A.16 Unity inspektor cechu

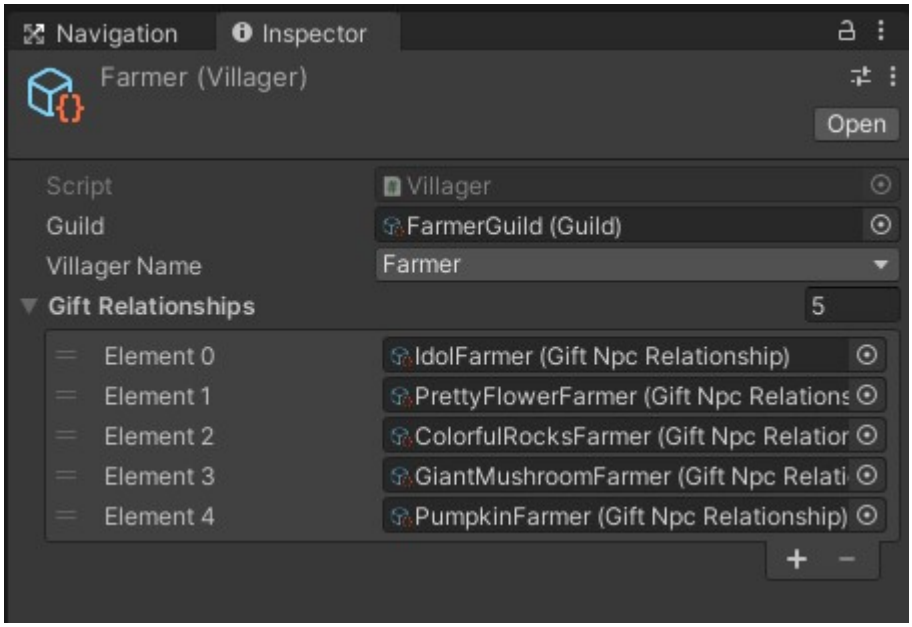


A.2.4 Vytváření a nastavování vesničanů

Nový vesničan se vytvoří pomocí cesty Create/Scriptable Objects/Villager a má následující atributy:

1. Guild – nastavit cech do kterého patří
2. Villager Name – jméno vesničana reprezentované jako Enum. Nový název je třeba přidat do enumu VillagerName ve Villager skriptu
3. GiftRelationships – nastavit vztahy mezi dárky a vesničanem (viz. Vytváření a nastavování vztahů mezi dárky a vesničany)

Obrázek A.17 Unity inspektor vesničana



A.2.5 Vytváření a nastavování dialogů

V této části se počítá s tím, že NPC má fungující dialog a je potřeba přidat dialogovou možnost k darování dárků. Když se dále zmiňuje ”hlavní ChoiceDialogueNode”myslí se tím node, ke které se hráč v rámci dialogu vrací a kterou vidí při startu dialogu.

Nejprve je potřeba vytvořit ChoiceDialogueNode, která bude mít pouze nastavenou NarrationLine s textem souvisejícím s přijímáním dárků (zbytek atributů se nastavuje ve skriptu).

Tato ChoiceDialogueNode se poté přidá jako choice do hlavní ChoiceDialogueNode.

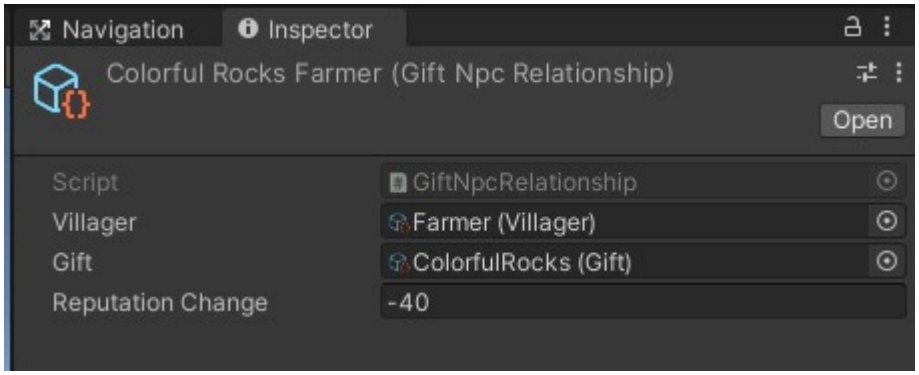
A.2.6 Vytváření a nastavování vztahů mezi dárky a vesničany

Nový vztah mezi vesničanem a dárkem se vytváří pomocí cesty Create/Scriptable Objects/Reputation/GiftNpcRelationship a má atributy:

1. Villager – nastavit vesničana
2. Gift – nastavit dárek

3. Reputation Change – nastavit o kolik se má změnit reputace při darování

Obrázek A.18 Unity inspektor vztahu vesničan-dárek

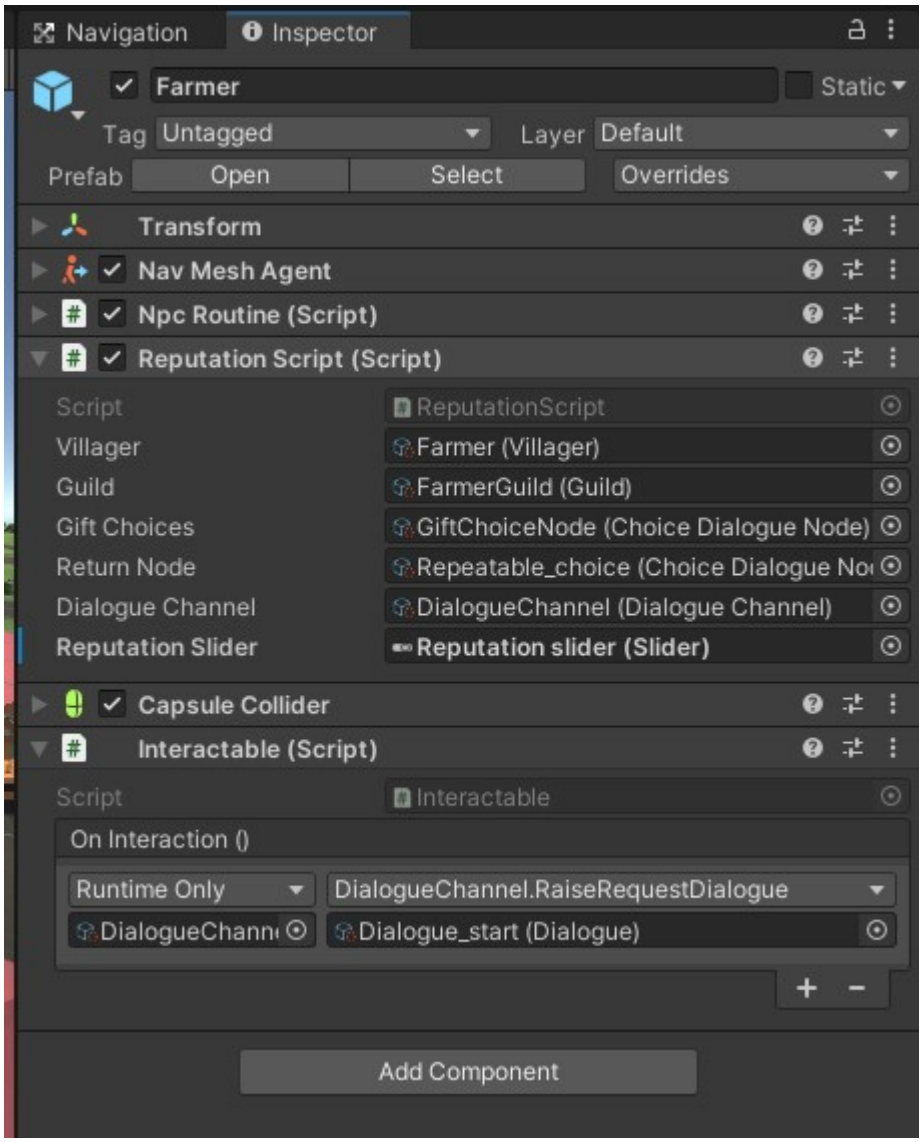


A.2.7 Nastavování reputačního skriptu na NPC

Vesničan, se kterým má hráč interagovat musí mít nastavený na svém modelu jako komponentu ReputationScript, který má následující atributy:

1. Villager – odpovídající Villager scriptableObject
2. Guild – odpovídající cech
3. Gift Choices – ChoiceDialogueNode vytvořena v rámci dialogu (viz. Vytváření a nastavování dialogů)
4. Return Node – hlavní opakující se ChoiceDialogueNode, která se aktivuje při dialogu s NPC
5. Dialogue Channel – potřeba pro dialogový systém
6. Reputation Slider – ukazatel reputace při dialogu v rámci uživatelského rozhraní

Obrázek A.19 Unity inspektor NPC

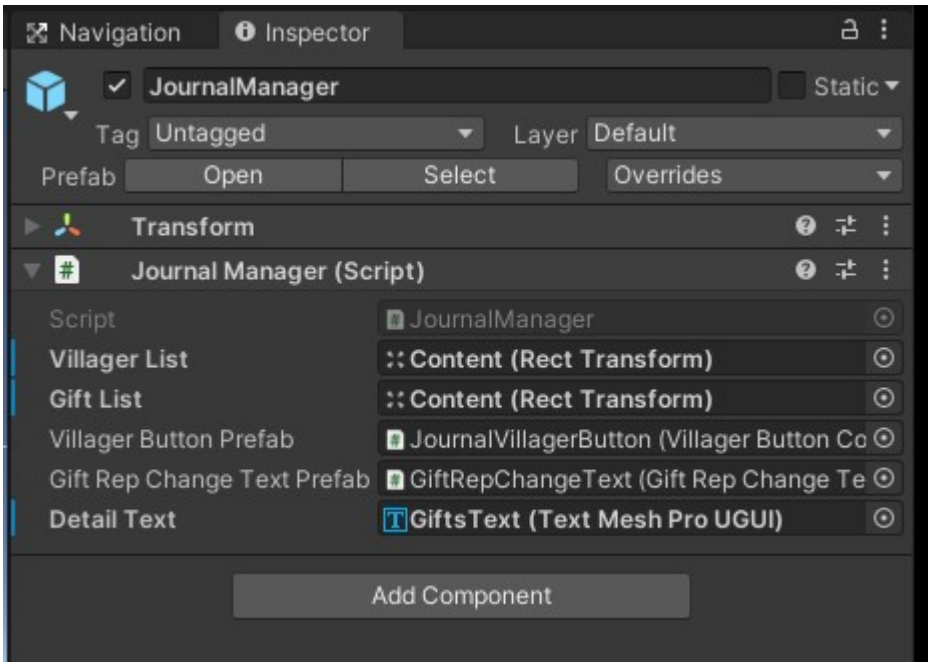


A.2.8 Vytváření a nastavování manažera deníku dárků ve scéně

Aby fungoval deník vztahů mezi dárky a vesničany, tak musí být ve scéně prázdný objekt se skriptem JournalManager, který má následující atributy:

1. Villager List – Rect Transform, který reprezentuje místo na UI kde se zobrazují vesničané
2. Gift List – Rect Transform, který reprezentuje místo na UI kde se zobrazují dárky
3. Villager Button Prefab – skript napojený na předpřipraveném tlačítku vesničana
4. Gift Rep Change Text Prefab – skript napojený na předpřipraveném textu, který ukazuje změnu reputace
5. Detail Text – text, který ukazuje jméno vybraného vesničana

Obrázek A.20 Unity inspektor manažeru deníku

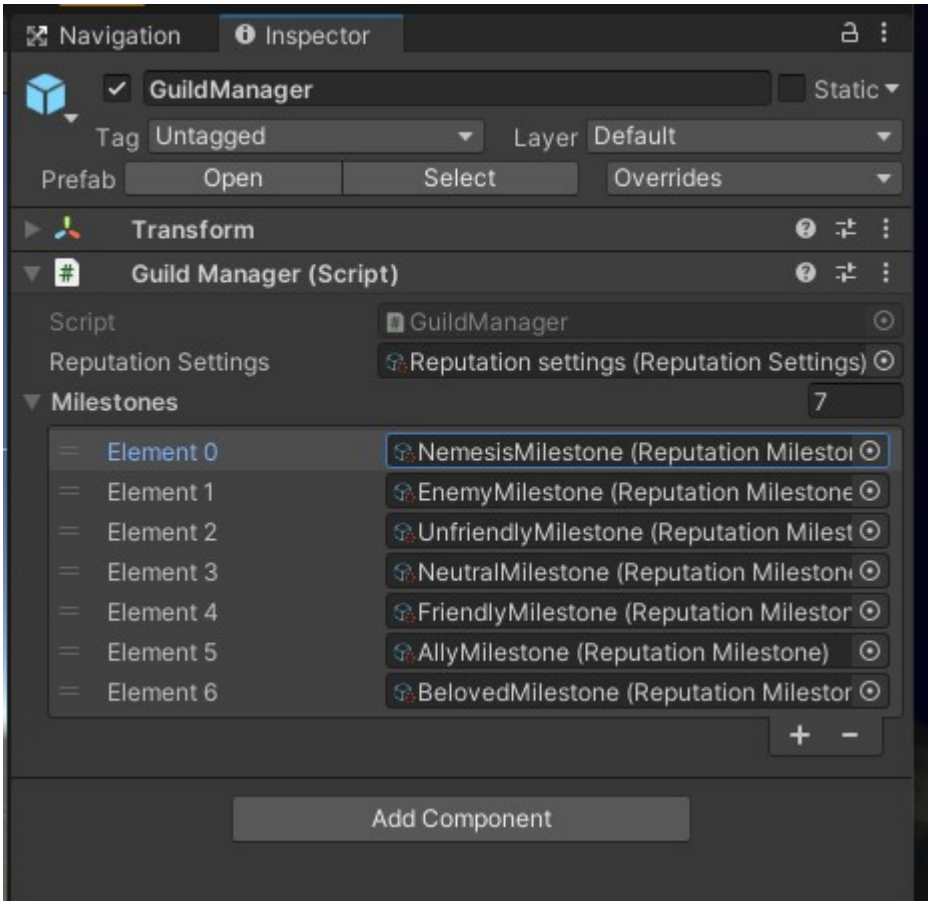


A.2.9 Vytváření a nastavování manažera cechů ve scéně

Aby fungovaly cechy je potřeba aby byl ve scéně prázdný objekt se skriptem GuildManager, který má následující atributy.

1. Reputation Settings – nastavení hranic reputace
2. Milestones – seznam ReputationMilestone, kterým lze dosáhnout ve hře

Obrázek A.21 Unity inspektor manažeru cechů



A.3 Společníci

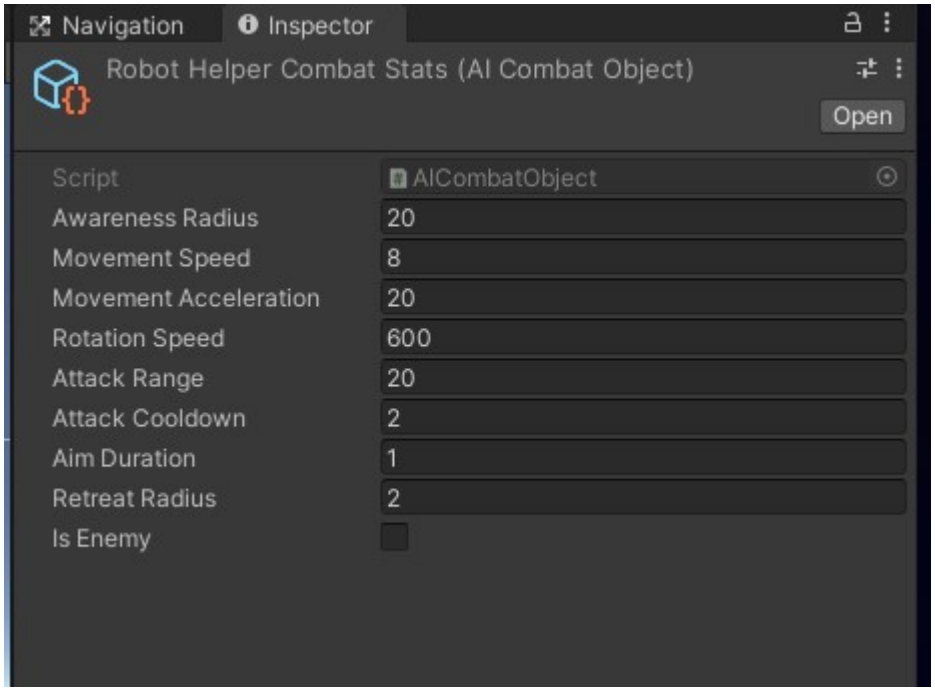
V této části se ukazuje, jak se vytváří a nastavují noví společníci. Předpokládá se, že je vytvořen model společníka, a že hráč má na svém modelu vytvořené prázdné objekty reprezentující místa, která mají společníci následovat (vodítka).

A.3.1 Vytváření a nastavování společníka

Nejprve je potřeba vytvořit scriptableObject AICombatObject, který definuje údaje o pohybu a útoku společníka. Vytvoří se pomocí cesty Create/Scriptable Objects/AI Combat Object a má následující atributy:

1. Awareness Radius – rozsah, ve kterém vidí nepřátele
2. Movement Speed – rychlost pohybu
3. Movement Acceleration – hodnota zrychlení
4. Rotation Speed – rychlost rotace
5. Attack Range – dosah útoku
6. Attack Cooldown – čas mezi útoky
7. Aim Duration – Doba míření
8. RetreatRadius – Vzdálenost, kterou se bude snažit držet od nepřítele
9. Is Enemy – true/false hodnota, která říká jestli je nepřítel

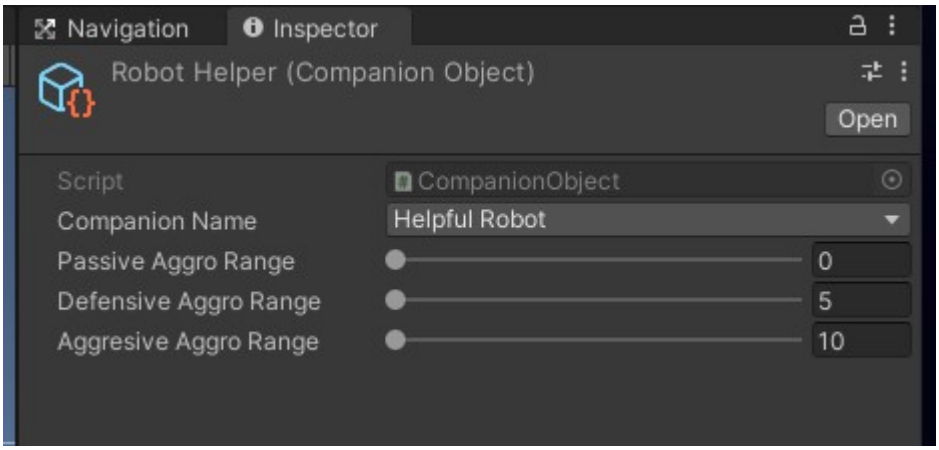
Obrázek A.22 Unity inspektor bojového objektu společníka



Poté je potřeba vytvořit scriptableObject CompanionObject pomocí cesty Create/Scriptable Objects/Companion a má následující atributy:

1. Companion Name – jméno reprezentováno jako Enum. Nový název je třeba přidat do enumu CompanionName v CompanionObject skriptu
2. Passive Aggro Range – vzdálenost na kterou bude na nepřátele útočit, když je společník pasivní
3. Defensive Aggro Range – vzdálenost na kterou bude na nepřátele útočit, když je společník defensivní
4. Aggresive Aggro Range – vzdálenost na kterou bude na nepřátele útočit, když je společník agresivní

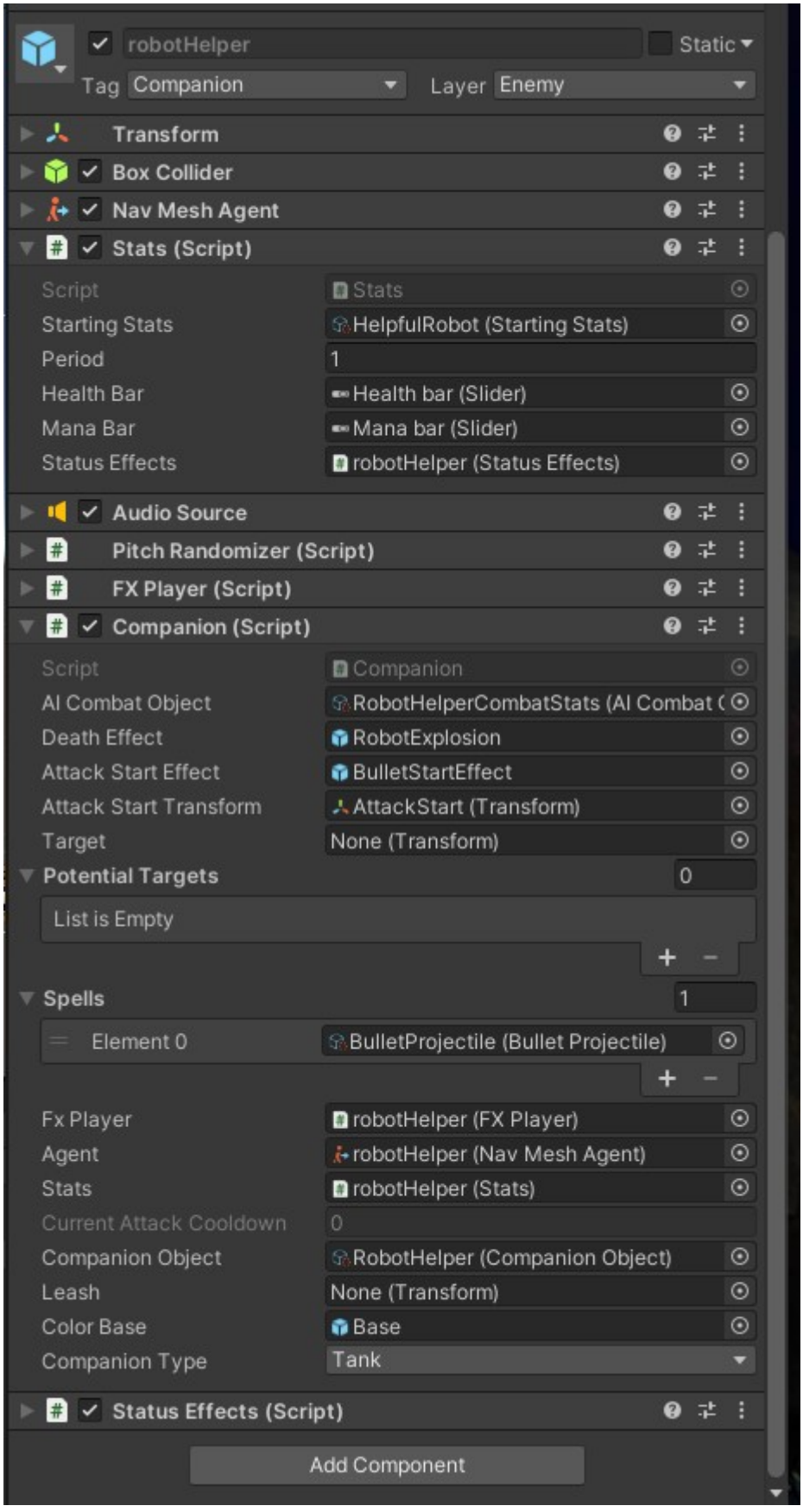
Obrázek A.23 Unity inspektor objektu společníka



Nakonec musí mít model společníka na sobě jako komponentu napojený skript Companion, který má následující atributy:

1. AI Combat Object – AICombatObject vytvořený výše
2. Death Effect – model efektu, který se aktivuje při zničení společníka
3. Attack Start Effect – model efektu, který se aktivuje při výstřelu společníka
4. Attack Start Transform – Transform místa, ze kterého má společník vystřelit
5. Target – aktuální cíl společníka (nastavuje se ve skriptu)
6. Potential Targets – potenciální cíle společníka (nastavují se ve skriptu)
7. Spells – kouzla, která společník používá
8. Fx Player – potřeba pro audio systém
9. Agent – Nav Mesh Agent společníka
10. Stats – skript Stats napojený na společníka
11. Companion Object – CompanionObject vytvořený výše
12. Leash – místo za kterým se companion pohybuje při následování hráče (nastavuje se ve skriptu)
13. Color Base – model barevného identifikátoru, který je na modelu společníka napojený
14. Companion Type – typ společníka reprezentovaný jako Enum

Obrázek A.24 Unity inspektor modelu společníka

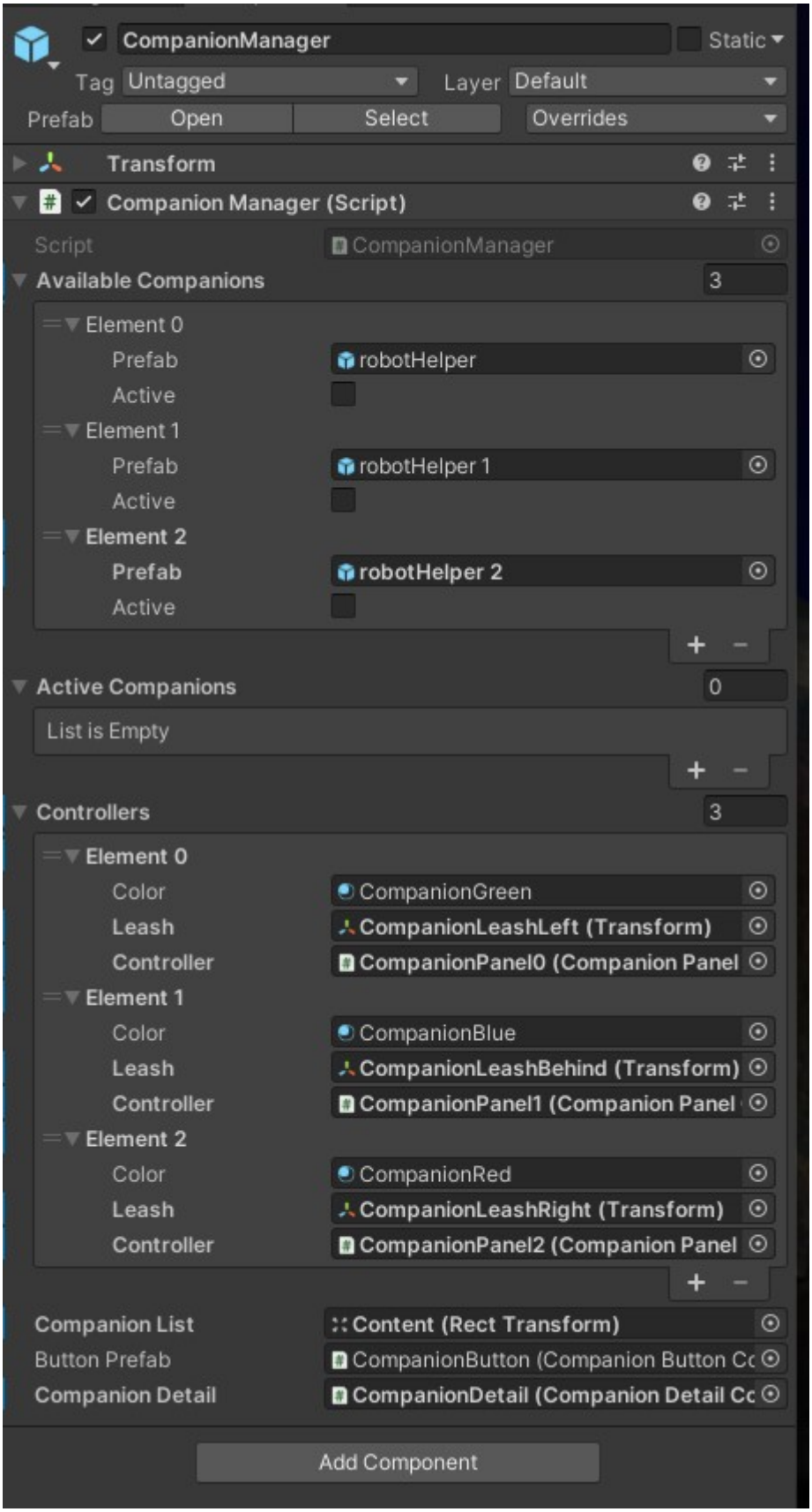


A.3.2 Vytváření a nastavování manažeru společníků

Aby hráč mohl získávat, aktivovat a ovládat společníky tak je potřeba aby ve scéně byl na prázdném objektu napojen jako skript CompanionManager, který má následující atributy:

1. Available Companions – seznam ManagableCompanion (jednoduchá třída obsahující model společníka a informaci, jestli je aktivní), tedy dostupných společníků
2. Active Companions – seznam aktivních společníků (nastavuje se ve skriptu)
3. Controllers – seznam UICompanionController což třída obsahující barvu identifikátoru společníka, pozici, kterou společník má následovat a skript ovládající panel společníka
4. Companion List – Rect Transform, který reprezentuje seznam společníků v rámci uživatelského rozhraní
5. Button Prefab – prefab tlačítka, kterým se vybírá konkrétní společník ze seznamu
6. Companion Detail – prefab detailu společníka, který obsahuje informace o jeho atributech a tlačítko aktivace

Obrázek A.25 Unity inspektor manažeru společníků



A.4 Vytváření a nastavování milníků reputace

Aby ve hře fungovaly milníky reputace, musí se udělat vytvořit reputační milníky, kterých může hráč dosáhnout.Poté objekty milníků pro konkrétní NPC pomocí kterých se budou volat metody, které budou pomáhat nebo škodit hráči. Nakonec se musí vytvořit pro konkrétní NPC objekt, který bude tyto metody implementovat.

Reputační milníky nejsou pro konkrétní NPC, ale unikátní pro celou hru, vytváří se pomocí cesty Create/Scriptable Objects/Reputation/Milestone a má následující atributy:

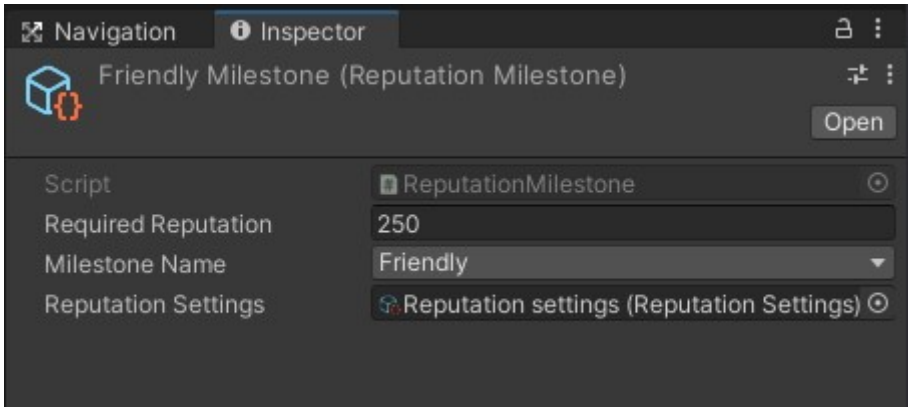
1. Required Reputation – nastavit hodnotu reputace při které se má milník aktivovat
2. Milestone Name – jméno milníku reprezentováno jako Enum. Nový název je třeba přidat do enumu MilestoneName v ReputationMilestone skriptu
3. Reputation Settings – vytvořené výše, potřeba pro kontrolu reputace

Poté se vytváří scriptableObject, který musí implementovat rozhraní IMilestoneActions. Implementace metod tohoto skriptu je plně v režii vývojáře.

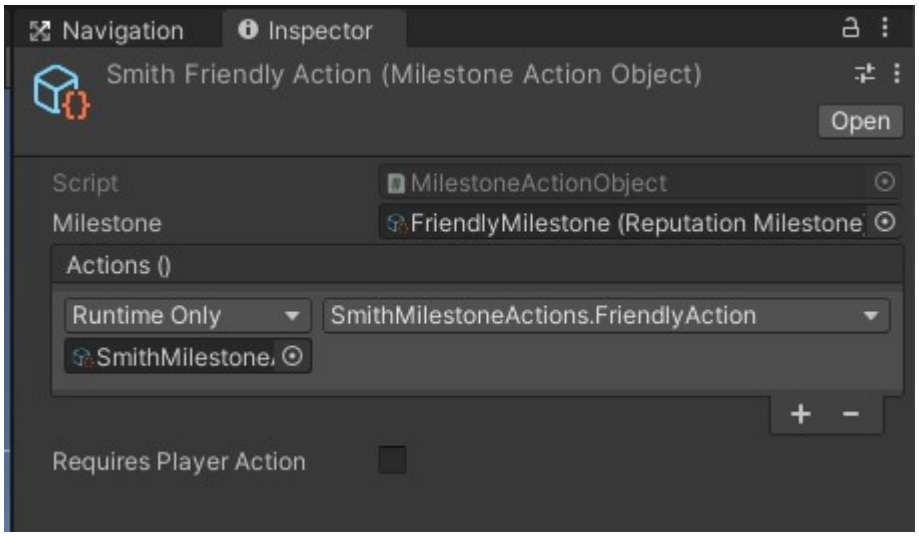
Nakonec se vytváří milestone objekt pro konkrétní NPC pomocí cesty Create/Scriptable Objects/Reputation/Milestone Action Object a má následující atributy:

1. Milestone – ReputationMilestone vytvořený výše
2. Actions() - callback na metodu ze skriptu, který implementuje IMilestoneActions, která se má zavolat při dosažení milníku
3. Requires Player Action – true/false hodnota. Pokud true tak je potřeba hráčova reakce v dialogu s NPC, jinak se aktivuje automaticky

Obrázek A.26 Unity inspektor reputačního milníku



Obrázek A.27 Unity inspektor objektu milníku pro NPC



Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `MagiTech_build` ..... adresář se spustitelnou formou implementace
- `project-magitech` ..... adresář s unity projektem a zdrojovými kódy
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX