Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Využití evolučních schémat a procedurálně generovaných modelů ve vývoji her

Monika Majorová

Diplomová práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Monika Majorová**
Osobní číslo:	**I15218**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Využití evolučních schémat a procedurálně generovaných modelů ve vývoji her**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V teoretické části se práce zaměří na techniky vývoje software s důrazem na vývoj počítačových her, které se využitými technikami odlišují. Kromě obecných technik a koncepcí vývoje bude kladen důraz na vysvětlení dvou konkrétních softwarových technik, kterými jsou: Tvorba adaptabilních evolučních schémat a využití procedurálního modelování.
Práce samotná bude mít tři nosná témata, která budou realizována prostřednictvím případové studie. Prvním okruhem bude vlastní implementace software podle design dokumentu. Druhým nosným tématem je využití evolučních schémat, které přispěje komplexnosti vlastního řešení. Důraz bude kladen na téma na aplikaci procedurálně generovaných modelů (a schémat). Právě procedurální modelování patří k významným doménám poslední doby, je nutné tedy na ně klást patřičný důraz a představit alespoň základní třídy algoritmů, které se v procedurálním modelování využívají.

Vlastní praktickým přínosem autora bude proof-of-concept počítačové hry typu multi RPG survival simulace. Očekává se hratelnost po dobu alespoň 30 minut a otevřená rozhraní pro další rozvoj. Zároveň v produktu musí být použity výše zmíněné techniky a technologie.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

***SCHELL, Jesse. The art of game design: a book of lenses. Boca Raton: CRC, c2008. ISBN 978-0-12-369496-6.**

Rapid development: taming wild software schedules. Redmond: Microsoft Press, c1996. ISBN 1-55615-900-5.

***GIBSON, Jeremy. Introduction to game design, prototyping, and development: from concept to playable game-with Unity? and C#. Upper Saddle River, NJ: Addison-Wesley, 2015. ISBN 0321933168.**

Vedoucí diplomové práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2016**

Termín odevzdání diplomové práce: **17. května 2017**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2016

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 29. 8. 2018

Monika Majorová

PODĚKOVÁNÍ

Ráda bych poděkovala svému vedoucímu práce Ing. Josefu Brožkovi za vedení mé práce, pomoc a věnovaný čas. Dále bych chtěla poděkovat svým rodičům, prarodičům a mému partnerovi za podporu nejen při psaní diplomové práce, ale i během celého studia. Nakonec chci poděkovat i všem vyučujícím za trpělivé předávání znalostí a vědomostí.

ANOTACE

Tato diplomová práce se zabývá problematikou vývoje software s důrazem na vývoj počítačových her. Zkoumá především techniky tvorby evolučních schémat a využití procedurálního modelování. Cílem diplomové práce je též implementace software (počítačové hry) podle design dokumentu s důrazem na využití evolučních schémat a aplikaci procedurálně generovaných modelů.

KLÍČOVÁ SLOVA

software, hry, evoluční schémata, procedurální modelování, procedurální generování, design dokument

TITLE

Usage of evolution schemes and procedurally generated models in game development

ANNOTATION

This thesis deals with problems of software development with emphasis on computer game development. It probes especially into techniques of creating evolution schemes and usage of procedural modelling. The goal of the diploma thesis is also implementation of software (computer game) according to a design document with emphasis on usage of evolution schemes and application of procedurally generated models.

KEYWORDS

software, games, evolution schemes, procedural modeling, procedural generation, design document

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

2D	2-Dimensional (dvojrozměrný)
3D	3-Dimensional (trojrozměrný)
AI	Artificial Intelligence (umělá inteligence)
CD	Compact Disc (kompaktní disk)
DVD	Digital Versatile Disc (všestranný digitální disk)
GUI	Graphical User Interface (grafické uživatelské rozhraní)
ID	Identifikátor
IDE	Integrated Development Environment (integrované vývojové prostředí)
IT	Informační technologie
LOD	Level of Detail (úroveň detailu)
MSF	Microsoft Solution Framework
NPC	Non-Player Character (ne-hráčská postava)
RPG	Role Playing Game (hra na hraní rolí)
RUP	Rational Unified Process
VR	Virtuální realita

ÚVOD

Počítačové hry, jejich vývoj a celkově herní průmysl je i v dnešní době některými lidmi jak z laické, tak i z odborné veřejnosti stále podceňovanou oblastí. Přesto se jedná o velmi rychle rostoucí průmyslové odvětví. Lidé se potřebují bavit a počítačové hry tuto potřebu dokáží naplnit. A nemusí jít jen o zábavu, jelikož počítačové hry, podobně jako např. filmy či knihy, dokáží poskytnout různé zážitky a pocity, které v běžném životě člověk tak snadno nezažije.

Krom zábavy a zážitků mohou hry též učit různým znalostem nebo zlepšovat dovednosti jako např. postřeh, kreativitu nebo logické myšlení. Není tak divu, že stále přibývá jak hráčů, tak počítačových her dostupných na trhu a tím tento trh roste.

Cílem této diplomové práce je popsat techniky vývoje počítačových her jakožto software, především takové techniky, které nejsou ve vývoji obecného software běžné a jsou specifické právě pro vývoj počítačových her. Větší důraz bude kladen na vysvětlení technik procedurálního modelování a evolučních schémat.

Práce v první řadě poskytne čtenáři úvod do problematiky vývoje počítačových her včetně vysvětlení pojmů používaných v této oblasti. Dále popíše metodiky vývoje software s důrazem na výše zmíněné techniky specifické pro vývoj počítačových her. Krátce bude též popsána problematika vydavatelství a distribuce v rámci herního trhu.

Druhou, praktickou částí této diplomové práce je implementace software na základě design dokumentu, přesněji proof-of-concept počítačové hry typu multi RPG survival simulace s očekávanou hratelností po dobu alespoň 30 minut. Při implementaci bude využito popsaných technik procedurálního modelování a evolučních schémat. Implementace bude provedena pomocí herního engine Unity, přičemž skripty budou psány v jazyce C#.

Téma diplomové práce jsem si zvolila, jelikož mi vývoj počítačových her připadá jako zajímavá činnost s velkým potenciálem, ve které mohu využít značnou část znalostí získaných při studiu na vysoké škole od matematiky, statistiky či algoritmizace, přes programování, datové struktury a počítačovou grafiku až po simulace, 3D grafiku a umělou inteligenci. Pro vývoj online webových her též využiji znalosti počítačových sítí, databázových systémů či webových aplikací. Zároveň se při psaní této práce dozvím značné množství nových informací, např. o procedurálním modelování nebo evolučních schématech a celkově získám přehled o vývoji počítačové hry, což mohu v budoucnu využít pro tvorbu dalších počítačových her.

1 VÝVOJ POČÍTAČOVÝCH HER

Vývoj počítačových her jakožto obor tvůrčí lidské činnosti se za svou přibližně sedmdesátiletou existenci proměnil v jeden z nejrychleji rostoucích průmyslů: videoherní průmysl. Tato práce se zabývá samotným vývojem počítačových her, které se vyznačují tím, že jsou tvořeny pro desktopové počítače, zatímco pojem videohry zahrnuje též například hry pro konzole či mobilní zařízení.

1.1 Základní pojmy

V oblasti vývoje počítačových her existují termíny, které jsou pro tuto oblast typické, případně termíny, které mají v této oblasti speciální význam. Některé z těchto termínů jsou popsané níže. Většina termínů v tomto oboru pochází z angličtiny a v tomto tvaru se běžně používají. Pro lepší srozumitelnost je u některých termínů v závorce uveden překlad do češtiny. V této diplomové práci jsou ve většině případů použity právě tyto běžně používané anglické termíny nebo případně jejich počeštěné varianty.

Game design (návrh hry) je činnost, při které se vymýšlí budoucí hra, včetně jejích funkcí, chování, grafické podoby, zvuků atd. Výsledkem této činnosti bývá Game design dokument, podle kterého se dále hra tvoří.

Game design dokument (dokument návrhu hry) je dokument, který je výsledkem Game designu. Na základě tohoto dokumentu pak probíhá samotná tvorba hry (programování, tvorba grafiky, zvuků atd.), přičemž je možná zpětná úprava Game design dokumentu v případě výskytu komplikací (přílišná obtížnost, nedostatek času, nedostatek financí, nové nápady apod.) při tvorbě hry.

Herní engine (jádro hry) je označení pro software, na kterém je hra založena. Velmi často jde o editor či IDE, ve kterém autor hru vytvoří a zkompiluje. Může se ale též jednat o softwarový základ vytvořený přímo autorem hry (či společností vyvíjející hru), na kterém může stavět více svých her. V obou případech použití enginu šetří čas, jelikož obsahuje značné množství funkcí a mechanik ve hrách používaných. V případě vývoje her je použití enginu značná úspora, jelikož kompletní tvorba hry je velmi náročná a nákladná.

Herní svět (mapa) označuje celé prostředí, ve kterém se hráč pohybuje. Může zahrnovat jak interiéry (budovy, jeskyně, bludiště, ...) tak exteriéry (města, lesy, louky, hory, ...).

Chunk (kus, část) je relativně malá (většinou přibližně několik metrů či desítek metrů) část herního světa. Některé hry využívají chunky k rozdělení světa na většinou pravidelně rozmístěné čtvercové oblasti, které umožňují např.:

- dynamicky načítat jen tu část, která je potřeba,
- rozdělit zátěž na procesor při generování či načítání světa,
- obejít omezení na velikost (detail) jednoho objektu v herním enginu,
- vytvářet svět předem neznámé velikosti.

Chunky se využívají většinou ve hrách s procedurálně generovaným terénem, pokud lze terén generovat (teoreticky) do nekonečna, pak je použití chunků téměř nutností. Ač se jedná o samostatné objekty, jsou vytvořeny a umístěny tak, aby tvořily souvislý jednotný terén bez zjevných přechodů mezi nimi.

Kamenný obchod je jakýkoliv fyzický obchod, kam si zákazník může přijít osobně něco koupit, tento výraz označuje opak internetových obchodů, které jsou zákazníkům přístupné pouze prostřednictvím Internetu.

Krabicová verze hry ve většině případů znamená hru na instalačním médiu (CD, DVD) zabalené buď v krabičce pro toto médium nebo ve větší krabici, kde bývá přiložen i návod či další bonusové materiály. Zvlášť v poslední době se lze ale setkat i s krabicovými verzemi bez instalačního média, pouze s kódem na stažení či aktivaci hry přes Internet.

Level of detail – LOD (úroveň detailu) je technika, která se používá pro optimalizaci běhu hry s trojrozměrnou perspektivní grafikou především na slabších počítačích. Využívá toho, že objekty, které jsou daleko od pozorovatele (hráče) jsou malé a nelze na nich pozorovat detaily tak, jako by byly blízko (což platí i v reálném světě). Hra ale tyto objekty počítá a vykresluje stejným způsobem, jako ty blízké. Z tohoto důvodu se tyto objekty při použití LOD nahrazují podobnými objekty s jednodušší strukturou, které z dálky vypadají podobně, jako jejich originální předlohy. Výjimkou nebývá ani nahrazení trojrozměrných objektů dvourozměrnými ekvivalenty zasazenými do trojrozměrného světa.

Level (úroveň) označuje zpravidla část s určitou úrovní obtížnosti ve hře. Některé hry bývají rozděleny na části, jimiž hráč prochází postupně od nejnižší směrem k vyšším. V každé části je obtížnost hry vyšší oproti té předchozí. Těmto částem se říká levely.

Level design (návrh úrovně) je část Game designu, která se zabývá tvorbou prostředí a scénářů jednotlivých levelů hry, případně určitých částí herního světa. Zahrnuje tedy návrh toho, jak

bude vybraná část hry vypadat a chovat se, zatímco např. řešení ukládání hry nebo síťové komunikace do level designu nepatří.

Mesh (síť, pletivo) nebo též „polygon mesh“ („polygonová síť“ či „mnohoúhelníková síť“) je výraz týkající se 3D grafiky a lze si jej představit jako síť tvořenou polygony neboli mnohoúhelníky (v praxi většinou trojúhelníky nebo čtyřúhelníky), která pak tvoří celý tvar 3D modelu.

Non-Player Character (NPC) neboli ne-hráčská postava je jakákoliv postava ve hře, která není ovládaná hráčem, ale počítačem (většinou pomocí umělé inteligence). V některých hrách mají NPC speciální určení oproti hráčským postavám, např. obchodníci nebo zadavatelé úkolů ve hrách typu RPG. V jiných mají stejné možnosti a určení jako hráč (virtuální protivník v šachách, soupeř ve střílečce apod.).

Polygon (mnohoúhelník) se podobně jako v geometrii užívá i v 3D grafice pro označení plochy ohraničené úsečkami spojující minimálně tři body, z nichž žádné tři sousední neleží na jedné přímce. Je tedy tvořen vrcholy a hranami, a spolu s dalšími polygony tvoří mesh.

Rigging (vytváření kostry) je v 3D grafice jedním ze základních kroků při tvorbě postav (lidí, zvířat, ale třeba i robotů) do her či filmů. Do 3D modelu je přidána kostra, což je soubor informací o tom, kde a jak se může postava ohýbat. Tedy především obsahuje informace o páteři a o kloubech jako jsou možné směry ohýbání a stupně volnosti pohybu v těchto jednotlivých směrech. Těmito informacemi se řídí algoritmy ovládající pohyb postav.

Sprite (sprajt) je termín typický pro 2D grafiku. Jedná se o malý dvojrozměrný obrázek, který může být i animovaný. Dříve se tento termín používal jako označení pro techniku sloužící k urychlení vykreslování dvojrozměrných objektů ve 2D i 3D hrách, dnes se tento termín v 2D grafice používá pro označení především pohybujících se objektů (např. postavy), ve 3D grafice pak bývá synonymem pro tzv. billboarding, což je technika pro vykreslování 2D objektů, které jsou vždy natočeny směrem ke kameře (tedy např. k hráči).

Status efekt (efekt stavu) znamená většinou dočasný, někdy i trvalý efekt pro herní postavu, který ji poskytuje určité zvýhodnění, např. regeneraci zdraví, zvýšenou odolnost nebo jiné vlastnosti (v tom případě jde o tzv. **buff**) nebo nevýhodu, např. znehybnění, snížení odolnosti či jiné vlastnosti (pak jde o tzv. **debuff**). Možná je také kombinace zvýhodnění i znevýhodnění v jednom. Status efekty jsou nejčastěji k vidění ve hrách RPG a často se využívají v boji, ale mívají využití i mimo boj. Význam status efektu spočívá v tom, že postava je v daném stavu,

ze kterého ji plynou určité bonusy či postihy, které by měly tento stav odrážet, např. pokud je postava zraněná, má postih k rychlosti, dokud se neuzdraví, nebo pokud vypije léčivý čaj, má na omezený čas bonus k regeneraci zdraví. Status efekt tedy postava získá za určitých podmínek a ztratí ho, pokud tyto podmínky přestanou platit, uplyne určitý čas nebo je tento efekt daným způsobem odstraněn. Efekt může vzniknout např. zkonzumováním předmětu (potravina, jed, lektvar, léčivá mast, kouzelný svitek), vyčarováním kouzla, útokem od protivníka (krvácení, otrava), aplikováním náhody, aktivací objektu v herním světě, na základě stavu postavy (hladovění), interakcí s NPC, vlivem okolního prostředí (nachlazení), vstupem do určité oblasti, nošením předmětů (síla kouzelného meče, přetížení) apod.

Vývoj hry v užším smyslu znamená samotnou implementaci hry, tj. především programování, tvorba grafiky a další tvůrčí práci, nezahrnuje však její návrh ani testování. Vývoj hry v širším smyslu zahrnuje celý proces tvorby hry od nápadu až po konečné vydání hry a případnou následnou podporu, jako např. vydávání opravných balíčků. V této diplomové práci je použit tento širší význam nebo jeho synonymum „tvorba hry“.

1.2 Obory ve vývoji her

V herním průmyslu najde uplatnění mnoho oborů. Čtyři nejdůležitější z nich jsou popsány níže.

1.2.1 Programátor

„Po programátorech je v herním průmyslu největší poptávka a řada dílčích specializací – od skriptérů misí (na pomezí programování a level designu) přes junior a senior programátory k různým specializacím – networking, umělá inteligence, grafický rendering, nástroje a další.“ (Jirkovský, 2012, s. 18)

Programátor má na starosti především to, aby hra fungovala tak, jak má (jak byla navržena). Jeho hlavní pracovní náplní je navrhování struktury programu a psaní kódu, případně programování pomocí vizuálních nástrojů. Využívá různé nástroje jako IDE, různé editory apod., postupuje zpravidla podle design dokumentu a přebírá od dalších členů týmu (např. od grafiků či zvukařů) jejich práci a zařazuje ji do výsledného funkčního produktu. Alternativně lze naprogramovat funkční celek, do kterého se konečná podoba grafiky či zvuků zařadí až poté.

1.2.2 Grafik

„Co by byly hry v dnešní době bez grafiky? Trendy ukazují, že zejména AAA tituly se čím dál více blíží úrovni filmového zpracování. Po schopných graficích je tedy zákonitě poptávka.

Velkou výhodou tohoto oboru je jeho přenositelnost do jiných průmyslů. Šikovný 2D grafik může dělat GUI (grafické uživatelské rozhraní do her) i celou herní grafiku, případně může přesedlat k typografii a grafice pro weby, reklamě apod. 3D grafik se vedle her uplatní i ve filmu, reklamě a případně architektonických studiích a v technických firmách (vizualizace interiérů, modelování 3D prototypů vozidel atp.).“ (Jirkovský, 2012, s. 19)

Grafik zodpovídá zejména za vizuální podobu hry. V závislosti na tom, zda se jedná o 2D nebo 3D hru, se jeho způsob práce v určitých ohledech liší. 2D grafika zahrnuje tvorbu obrázků, dvojrozměrných scén a tzv. spritů. Pro tvorbu 3D grafiky je typické modelování trojrozměrných objektů, texturování či tzv. rigging. V obou případech do oblasti grafiky spadá i animace, která se ale u 2D a 3D grafiky značně liší. Animaci samozřejmě může provádět jiný člověk na již hotových spritech či modelech. Je nutno říci, že součástí 3D hry je v naprosté většině případů i 2D grafika – například se její pomocí vytváří menu, různé dialogy a další grafické prvky.

1.2.3 Tester

„Kariéra herního testera nemusí být v rámci celoživotní perspektivy až tak zajímavá jako cesty jiné, nicméně je skvělým startovním bodem a jste-li dostatečně cílevědomí, otevírá dveře k celé řadě dalších profesních specializací – programování, level designu, game designu, managementu. Obrovskou výhodou zde totiž je, že na nástupní pozici (Junior Tester) nepotřebujete výraznější předchozí zkušenosti, vzdělání a kvalifikaci. Samozřejmě, jsou zde různé vhodné předpoklady či řekněme preference zaměstnavatele (orientace v oblasti IT, přehled o programování, SŠ či VŠ technického oboru, pečlivost, analytické myšlení), ale i mezi testery je určitá hierarchie – třeba Lead Tester už je de facto manažerská pozice a nemusí se za ni u většího studia stydět ani čtyřicetiletý veterán, neboť má pod sebou desítky zaměstnanců a brigádníků, komunikuje přímo s vedením firmy a zodpovídá za funkčnost a bezchybnost výsledného produktu.“ (Jirkovský, 2012, s. 20)

Úkolem testera je zkoušet funkčnost hry a zajistit odhalení co nejvíce existujících chyb a nedostatků ve hře. Jelikož se většinou chyby neprojeví hned a některé mohou nastat jen ve specifických případech, práce testera často spočívá v mnoha opakováních hraní testované části hry a pozorování chyb či odchylek od návrhu (game design dokumentu). Pokud najde nějaký nedostatek, nález zdokumentuje včetně postupu, který řešiteli dovolí problém zreplikovat. Následně popis předá k opravě jiným členům týmu, zpravidla programátorům, případně je sám opraví.

1.2.4 Level designér a Game designér

„Podobně jako testing je level design dobrý začátek kariéry v herním vývoji. Zatímco opravdový game designér je v týmu jeden či je jich jen pár (z prostého důvodu, že by si lezli do zelí a lepší jeden člověk s konzistentní tvůrčí vizí, nežli komunismus a vláda lidu), zejména na rozsáhlejších projektech pracuje často i několik desítek level designérů (též skriptérů). Level designér má na starosti navrhování jednotlivých levelů, resp. jejich „donavrhování“, neboť návrh samotný pochází od hlavního designéra. Aby hra měla všude obdobný styl a jednotlivé mise či mapy byly propojené, nemůže mít každý level designér úplnou tvůrčí svobodu, nicméně přesto je zde hodně prostoru pro seberealizaci.“ (Jirkovský, 2012, s. 21)

Game designér navrhuje hru jako takovou, od podoby menu, přes různé herní principy až po podobu hlavního hrdiny či detailní chování umělé inteligence NPC. Pochopitelně je též autorem Game design dokumentu, který všechny tyto principy popisuje. Level designér má oproti Game designérovi na starosti pouze jeden či více levelů, map, nebo jiných dílčích součástí hry. Také platí, že se při navrhování takové dílčí části musí řídit Game design dokumentem, jelikož tvorba Game designéra je level designu nadřazená. Z čehož vyplývá, samozřejmě v závislosti na konkrétní hře, že např. nepřátelé jsou stejní pro celou hru a level designér navrhne pouze tvar bludiště a umístění pokladů.

1.3 Další přidružené obory a dovednosti

Vývoj počítačových her je komplexní disciplína, ve které najde uplatnění řada dalších oborů či dovedností. Jirkovský (2011, s. 35) uvádí jejich přehled, který je uveden níže vyjma oborů, které již byly popsány a s některými přidanými obory navíc. Je nutno říci, že ne ve všech hrách se uplatní všechny tyto obory. Hry typu Tetris či Pacman si vystačí krom výše zmíněných pouze s několika málo dalšími obory, zatímco historická 3D MMORPG hra jich využije daleko více.

Angličtina: I při vývoji českých her má angličtina využití. Především je nutné brát v potaz, že většina literatury zabývající se tvorbou her je v angličtině. Své využití najde při využití zahraničních investorů, vydavatelů, outsourcingových studií či třeba zahraničních zaměstnanců. Z tohoto důvodu se často dokumentace ke hře (design document) píše anglicky. Zajímavostí je také fakt, že i některá česká herní studia vydávají hry v angličtině (v některých případech pouze v angličtině), jelikož anglicky mluvících hráčů je mnohem více než česky mluvících, což znamená větší trh pro hru v angličtině.

Animace: Při tvorbě her je třeba herní svět „rozhýbat“, k čemuž slouží právě animace. Cílem animace je nejen, aby se postavy a libovolné jiné objekty hýbaly, ale aby tyto pohyby vypadaly přirozeně, což je pro moderní hru důležité.

Antropologie: Hráči pocházející z různých koutů světa, různých pohlaví a věku mohou mít různé preference. Je důležité studovat cílovou skupinu hráčů, pro kterou je hra určena, protože pocit štěstí či motivace u nich zapříčiňují různé věci.

Architektura: Při návrhu herních světů (map) se často navrhují také města a v nich budovy. Pokud se ve hře budovy objevují, je vhodné porozumět světu architektury a tyto znalosti při návrhu uplatnit.

Brainstorming: Především pro návrh her se hodí nejen spousta nápadů, ale hlavně technika, jak tyto nápady třídit, prioritizovat, domýšlet a kritizovat.

Dabing: V některých hrách jsou dialogy či monology postav a např. monology vypravěče namluveny dabéry, což hře značně přidává na kvalitě a zábavnosti pro hráče. Slyšet postavy či vypravěče mluvit je mnohem zábavnější než jen číst texty na obrazovce.

Ekonomika: V některých hrách se principy ekonomiky (nebo její části) uplatňují, např. fungování státu, města, obchodování. Proto je pochopení principů ekonomiky pro návrh některých her důležité.

Fotografie: Zajímavostí je, že v některých případech najde uplatnění při tvorbě hry i fotografie, přičemž výsledky práce fotografa jsou následně v grafickém programu odpovídajícím způsobem upraveny a mohou být použity jak pro 2D grafické prvky, tak pro texturování 3D modelů. V obou případech se použitá technika fotografování liší.

Historie: Mnoho her bývá zasazeno do historie. Dokonce i fantasy hry často čerpají z historie lidstva. Proto je dobré se při návrhu takové hry orientovat v dějepisu, což poslouží jak k inspiraci, tak pro samotný návrh a lepší přehled o zdrojích dalších informací.

Hudba: Ke hrám, podobně jako k filmům, hudba téměř neodmyslitelně patří. Hudba má za úkol především umocnit zážitek, který má hra uživateli poskytnout. Proto se typicky volí vhodná hudba v závislosti na situaci nebo lokaci, ve které se postava nachází – jiná hudba bude hrát při klidné procházce krajinou, jiná při boji.

Kinematografie: Ve hrách se také často vyskytují filmové sekvence. Ty jsou většinou přehrávány na začátku hry, na konci nebo v jejím průběhu a mohou být buď předrenderované

(již „natočené“), anebo předskriptované a přehrávané herním enginem. Ty druhé mají výhodu v tom, že nemusí být pokaždé stejné, ale mohou se mírně lišit dle aktuálního stavu hry. V obou případech je ale třeba rozumět tomu, jak se filmy dělají, např. co se týče kompozice scény, velikosti záběru, pohybu kamery, světel a stínů či střihu.

Komunikace: Vývoj her je většinou týmová záležitost. Vzniklo sice mnoho dobrých her vytvořených jedním člověkem, ale převážná většina jich je tvořená v týmech. Členové týmu, často lidé několika různých profesí, mezi sebou musí umět komunikovat a umět řešit případné problémy v komunikaci a spory.

Kreativní psaní: Podobně jako filmy mají scénář a často i knižní předlohu, hry se v tomto ohledu příliš neliší. Příběh, který se hra vypráví, někdo předtím napsal. I to, jak vypadá herní svět a co se v něm děje musel někdo před tím popsat. Dialogy, které spolu postavy vedou i monology vypravěče se píší podobně, jako u filmu. U některých her navíc bývají i další příběhy napsané např. formou knih, které může postava během putování světem najít.

Matematika: Matematika má při tvorbě her široké uplatnění. Je důležitá jak pro samotné fungování hry, tj. pro výpočet fyziky, pohybu, generování dynamického obsahu atd., tak pro návrh herních mechanik jako je obchod, různé pravděpodobnosti a náhody apod. Zde najdou též uplatnění podobory matematiky jako je statistika či teorie pravděpodobnosti.

Mluvený projev: I pro malé herní studio může přijít chvíle, kdy bude potřeba hru prezentovat na veřejnosti ústní formou, což je vcelku efektivní způsob propagace. Takový mluvčí (což může být i vedoucí studia či třeba herní designer) se musí umět elegantně vypořádat s připomínkami ostatních, přijímat jejich argumenty, hledat jejich původ, volit řešení a stanovit jasný závěr.

Obchod: Pomineme-li tvorbu freeware (zdarma) her, tak je herní průmysl obchod. Většina her je vydávána proto, aby vydělala peníze. Znalosti obchodu jsou tedy pro úspěšný prodej hry důležité.

Psychologie: Účelem hry je hráče především bavit. Game designér by měl tedy pochopit, co hráče baví, jak myslí a proč hraje. Pro tento účel se hodí právě psychologie.

Technické psaní: Game designér musí být schopný vyhotovit Game design dokument tak, aby byl stručný a jasný, a přitom obsahoval vše potřebné k tomu, aby se podle něj dalo pracovat.

Zvuk: Kvalitní zvuková kulisa je pro kvalitní hru důležitá. Dalo by se říci, že zvuk je po dobrém návrhu, grafice a samozřejmě programové části pravděpodobně nejdůležitější prvek her. Zvuk

je důležitý jak jako odezva na hráčovy akce, tak jako upozornění hráče na nějakou situaci, a v neposlední řadě spolu s hudbou pro navození té správné atmosféry v konkrétní situaci ve hře.

1.4 Nástroje a vývojová prostředí

Vývoj her může probíhat v celé řadě nástrojů počínaje jednoduchým textovým editorem, přes různá obecná vývojová prostředí až po specializované enginy pro vývoj her. Příkladem obecného vývojového prostředí jsou:

- Netbeans,
- Eclipse,
- Idea,
- Visual Studio,
- Monodevelop,
- JDeveloper,
- a jiné.

Příkladem specializovaných enginů pro vývoj her jsou:

- Unity,
- Cry engine,
- Unreal engine,
- Game maker,
- RPG maker,
- a další.

Jednotlivá vývojová prostředí a enginy se liší především používaným programovacím jazykem a také způsobem tvorby v těchto prostředích či enginech. Herní engine má oproti obecnému vývojovému prostředí výhodu v tom, že je zaměřen na tvorbu her a obsahuje velké množství různých funkcí, prvků či mechanik používaných právě ve hrách, čímž činí vývoj her jednodušším a rychlejším. Pro praktickou část této diplomové práce je použit herní engine Unity, který je blíže popsán v následujícím pododdílu.

1.4.1 Herní engine Unity

Herní engine Unity slouží pro tvorbu jak 2D tak 3D her. Disponuje editorem scén, což si lze představit jako jednoduchý 3D editor s omezenými možnostmi modelování, určený spíše pro skládání scén z již hotových modelů vytvořených ve 3D grafických programech. Tento editor umí pracovat i ve 2D režimu.

Unity je v základní verzi dostupný zdarma s několika málo omezeními. Tato verze je určena především pro začátečníky, studenty a fanoušky. Dále je dostupný ve dvou variantách plné verze, a to verze Plus a verze Pro, která je určená pro profesionály a herní studia. Plnou verzi Unity má povinnost zakoupit autor či studio, jehož příjmy z vývoje v Unity za rok přesáhly určitou částku (pro obě varianty plné verze jsou částky odlišné), případně je lze samozřejmě zakoupit dobrovolně. V současné době Unity nepodporuje jednorázové zakoupení plné verze, namísto toho mají uživatelé možnost používat plnou verzi na základě měsíčního či ročního předplatného.

Nejen díky existenci verze zdarma, ale i díky existenci značného množství kvalitních tutoriálů (které jsou k dispozici především v angličtině, stejně jako samotný editor, ale existují i české tutoriály) jak textových, tak audiovizuálních a poměrně intuitivním ovládání samotného editoru je Unity vhodný pro začátečníky, ale díky velikému množství možností je vhodný také pro profesionály a herní studia pro tvorbu téměř libovolného typu hry.

V čem je Unity jistě zajímavý je široká podpora platforem, na které lze vytvořenou hru zkompilovat. Zde je několik příkladů: Windows, Mac OS X, Linux, Android, iOS, tvOS, Xbox One, PS Vita, PS4, Universal Windows Platform, WebGL, Facebook a další. Samotný editor Unity lze spustit na platformách Windows a Mac OS X.

Programování skriptů pro hry je v Unity možné v jazycích C# a UnityScript (který vychází z JavaScriptu). Skriptování probíhá v Unity tím způsobem, že autor píše tzv. komponenty, tj. skripty, které umisťuje na objekty v herní scéně. Engine Unity se pak sám stará o spouštění skriptů v ten správný okamžik, včetně jejich opakovaného spouštění při aktualizaci herní scény.

Zajímavostí ve skriptování v Unity jsou tzv. scriptable objekty. Jedná se v podstatě o uložený objekt (instanci třídy), kterou lze vytvářet a měnit přímo v editoru. Podmínkou pro vznik scriptable objektu je třída naprogramovaná ve skriptu, na základě které je pak scriptable objekt vytvářen.

2 ANALÝZA VÝVOJE POČÍTAČOVÉ HRY

Vývoj počítačové hry je oproti obecnému vývoji softwaru v určitých ohledech specifický. V této kapitole budou nejprve zmíněny standardní metodiky vývoje software, které jsou využívány i při vývoji her. Dále se text zaměří na oblasti vývoje software, které jsou pro počítačové hry specifické.

2.1 Standardní metodiky vývoje software

Metodika je rámec či postup pro tvorbu softwaru. Volba správné metodiky a její dodržování pomáhá při organizaci vývoje softwaru ve všech jeho fázích. Metodiky vývoje se v čase postupně vyvíjely dle potřeb vývojářů. Gunderloy (2007, s. 33) ve své knize Z kodéra vývojářem uvádí jejich základní přehled, který je popsaný v následujících pododdílech.

2.1.1 Vodopádový přístup

Jedná se pravděpodobně o nejstarší přístup z metodik probíraných v této práci. Vodopádový přístup zahrnuje tyto aktivity: analýza požadavků, předběžný návrh, detailní návrh, implementace, systémové testy a akceptační testy. Aktivity „tečou“ jedna za druhou a nikdy se neopakují, na konci tohoto procesu je produkt hotov.

Vodopádový přístup je vhodný spíše pro velké projekty s požadavkem na vysokou spolehlivost a byl využívaný hlavně v dřívějších dobách, kdy vývoj softwaru probíhal poněkud odlišně od dnešní doby, za což částečně mohly méně propracované nástroje a také mnohonásobně delší proces kompilace zdrojového kódu.

2.1.2 Iterativní přístupy

Když postupem času přešel proces kompilace od zpracovávání balíčků děrných štítků v sálových počítačích přes noc k editorům a překladačům na stolních počítačích a proces kompilace se zkrátil ze dnů na minuty, nastala doba průzkumného kódování. Začalo být jednodušší nápady zkusit rovnou realizovat, než je detailně popisovat a navrhovat a rozdílné fáze návrhu, implementace a testování se zahajovaly ve zmatku společně.

Jako odpověď na tyto změny se dostaly do popředí zájmu softwarového průmyslu některé iterativní metodiky, jako například Rational Unified Process (RUP) či Microsoft Solution Framework (MSF).

Iterativní metodiky jsou charakteristické tím, že projektové fáze jako návrh, implementace či testování mohou být v průběhu projektu několikrát opakovány, což je jejich klíčovou výhodou. Jakmile jsou všechny fáze hotovy, začíná nová tzv. „iterace“ a celý proces začíná znovu.

Takovýchto iterací může před vydáním projektu proběhnout libovolné množství, dle potřeb vývojového týmu.

2.1.3 Agilní přístupy

Nová skupina metodik, která začala upoutávat stále větší pozornost ve druhé polovině 90. let zahrnuje takové procesy jako Extrémní programování (XP), Scrum a Crystal. V některých sektorech byly tyto vývojové přístupy uvítány jako ulehčení, v jiných nikoliv.

„Agilní metodiky se dívají na přechod od vodopádového přístupu k iterativním metodám a pokračují v započatém trendu. Jestliže jsou rychlejší cykly užitečné, proč je neredukovat na týdny nebo dokonce dny? Jestliže bude redukována byrokracie, proč ji neodstranit úplně? To jsou atraktivní představy, obzvláště pro vývojáře, kteří mají rádi rychlý vývoj a nesnáší byrokracii.“ (Gunderloy, 2007, s. 33)

Extrémní programování (XP) je jednou z nejznámějších agilních metodik. Jedná se o kolekci praktik, z nichž některé jsou poněkud kontroverzní (např. párové programování, jež zahrnuje dva vývojáře pracující společně při psaní kódu). Je to ale také místo mnoha zajímavých výzkumů v oblasti softwarového vývoje.

2.2 Techniky specifické pro vývoj počítačových her

Vývoj počítačové hry může být přirovnán k výrobě filmu, speciálně 3D animovaného filmu, s prvky obecného vývoje software. Hry mají především oproti obecnému softwaru odlišný účel využití. Zatímco běžný software je vyvíjen proto, aby lidem usnadnil život nebo zvýšil zisk, hry jsou vyvíjeny především za účelem poskytnout lidem zážitek, podobně jako zmíněné filmy.

Jak již bylo naznačeno v oddílech 1.2 „Obory ve vývoji her“ a 1.3 „Další přidružené obory a dovednosti“, vývoj her zahrnuje velmi široké spektrum lidské činnosti. V tomto oddílu budou zmíněny ucelené techniky, které se běžně využívají při tvorbě her, avšak nikoliv při tvorbě obecného softwaru – nemusí to být ale pravidlem, existují výjimky. Je možno pozorovat spojitost mezi těmito technikami a zmíněnými obory, typicky například kinematografií, kreativním psaním či dabingem.

2.2.1 Psaní příběhu

Ne každá hra je založena na nějakém příběhu, avšak drtivá většina her ano, což se o obecném softwaru většinou říci nedá. Účelem příběhu je hráče především zaujmout a pobavit, případně v něm vyvolat jiné pocity, podobně jako to dělají knihy nebo filmy. Příběh je podstatnou součástí většiny her, které by bez něj postrádaly smysl.

Zajímavé ovšem je, že ne každá hra má příběhovou linii pevně danou, existují hry s rozvětveným příběhem, které mají zpravidla jeden začátek a více možných konců, kde postup příběhem a konečný výsledek závisí na rozhodnutí hráče. Taková počítačová hra by se dala přirovnat ke gamebooku nebo k interaktivnímu filmu.

Existují ale také hry, ve kterých příběh mohou hráči tvořit sami. Jedná se především o sandboxové hry nebo o hry, které umožňují hráčům tvořit si svůj vlastní obsah. Tento druh počítačových her by se dal přirovnat ke stolním hrám na hrdiny jako je např. Dračí doupě nebo Dungeons & Dragons.

2.2.2 Scénář

Scénář počítačové hry může být velmi komplexní a obsáhlý dokument, v závislosti na složitosti samotné hry. Jak uvádí Gajdůšek (2015), dnešní náročné a propracované hry obsahují příběh, který se odehrává na základě činů hráčem ovládané postavy, tudíž má více dějových linií, které na sebe předem daným způsobem navazují. Krom hlavní dějové linie, která se v určitých okamžicích dělí a pak zase spojuje, obsahují hry také vedlejší dějové linie, které zpravidla nijak zásadně neovlivňují vývoj hlavního děje, avšak dávají hráči pocit, že je hra mnohem větší.

„Scénář filmu si můžete představit jako rovný hladký kmen. Příběh začíná dole a končí nahoře – nic složitého. Ale scénář hry je jako strom – tedy kmen s mnoha a mnoha větvemi. Příběh také začíná dole – na hladkém kmeni. Ale dál už se rozvětvuje a rozvětvuje. Každá hra může být jiná – jedna může mít kmen bez větví jako u filmu, u jiné se její kmen může v polovině rozvětvit na dvě hlavní větve, některá zase může mít jen spoustu vedlejších větviček, z nichž některé mohou končit, jiné vás dovedou opět k hlavnímu kmeni.“ (Gajdůšek, 2015)

Obsahem scénáře ale není jen popis děje, naopak celkově popisuje hru po obsahové (nikoliv technické) stránce. Scénář bývá založen na již zmíněném příběhu (námětu) a v závislosti na typu hry může obsahovat následující informace:

- Herní klíč neboli princip či pravidla hry. Definuje, co je cílem hry, co bude a nebude možné ve hře dělat a dále např. kolik toho postava unese, jestli se bude měnit počasí, zda bude možné mluvit s dalšími postavami apod.
- Dějové linie, tedy popis průběhu možných variant hlavního děje a určení, kde se děj větví, případně spojuje. Dále také popis vedlejších dějových linií.
- Postavy, jejich charakter a vlastnosti, přičemž se nemusí jednat pouze o lidské postavy, ale i např. o zvířata. Sem mohou být zahrnuty také případné frakce, tj. skupiny postav, které mezi sebou mají určité vztahy, např. rod, kmen či národ. Mělo

by zde být také určeno, zda je tato postava ovládaná hráčem nebo počítačem (pomocí umělé inteligence), přičemž tato skutečnost nemusí být pěvně daná, ale určená na základě pravidel – existují hry, kde si může hráč zvolit, kterou postavu bude v daném okamžiku ovládat (přičemž ostatní postavy ovládá počítač), nebo také hry, kde v jednom okamžiku existuje více postav ovládaných hráčem (hry pro více hráčů).

- Lokace neboli mapy herního světa, kam se hráč může dostat. V některých hrách se lokace shodují s úrovněmi hry, které postava postupně prochází, v jiných se hráč se svojí postavou může pohybovat mezi lokacemi více či méně libovolně.
- Rekvizity, výzbroj a výstroj, jejich vlastnosti a účinek. Jedná se v podstatě o předměty, se kterými mohou herní postavy manipulovat. Zbraně mají předem daný účinek, výzbroj danou úroveň ochrany, batohy určitou kapacitu a celkově každý předmět má vlastnosti v závislosti na svém druhu a účelu použití.
- Filmová videa, videosekvence či cut scény – tyto názvy jsou synonymy pro videa, která se spustí v určitých fázích hry. Každé takové video má svůj vlastní scénář, který se může podobat filmovému scénáři. Blíže budou popsána v následujícím pododdílu 2.2.3 „Videosekvence“.
- Dialogy hovořících postav. Ty bývají v moderních počítačových hrách rozvětvené. Hráč má typicky na výběr z několika variant odpovědi a podle této volby se konverzace dále ubírá. Je tedy třeba napsat všechny možné varianty dialogů a určit případné následné reakce postavy v rámci herního světa. Dialogy se mohou také lišit v závislosti na denní době, náladě postavy nebo také podle toho, jaký mají dané postavy nebo jejich frakce v danou chvíli vztah.
- Akce neboli události, často označované anglickým výrazem eventy, se kterými se hráč může ve hře setkat a reagovat na ně. Může se jednat o různé náhodně se vyskytující úkoly, přepadení, pohromy či příležitosti např. k obchodu apod.

Dílem scénáristy neboli herního designéra mohou být též doprovodné brožurky, vysvětlivky či nápovědy ke hře.

Jak je zřejmé, herní scénář je oproti tomu filmovému velmi obsáhlý a složitý, v případě, že se nejedná o jednoduchou hru. Na takovém scénáři mnohdy spolupracuje více designérů. Scénář bývá součástí game design dokumentu a může mít podobu textového dokumentu, vhodnější je ale nelineární hypertextový dokument, kde jsou na sebe jednotlivé prvky logicky navázány pomocí odkazů tak, jak spolu souvisí, a celý dokument je tak mnohem přehlednější.

2.2.3 Videosekvence

Jak již bylo zmíněno v předchozím pododdílu, ve hrách se vyskytují videosekvence, někdy též nazývané filmové sekvence či videa, častěji též cut scény. Jedná se o krátká videa, která se spouští v určitých fázích hry, typicky na jejím začátku, konci, či také uprostřed hry ve chvíli, kdy např. hráč se svou postavou vejde do nějaké lokace, něco se mu opravdu dobře povede, nastane předem daná událost nebo při rozhovorech postav. Samozřejmě to, kdy se videosekvence spustí a jestli se vůbec spustí, záleží na konkrétní hře, není to žádné obecné pravidlo. Během přehrávání videosekvence zpravidla nelze hru ovládat, většinou ale lze videosekvenci přeskočit, ovšem záleží na konkrétní hře a také na konkrétní videosekvenci v rámci dané hry.

Videosekvence by se daly rozdělit do následujících kategorií:

- Předem vytvořená videa, která mohou být, stejně jako filmy, buď hraná, animovaná, nebo „předrenderovaná“ (typicky pomocí kvalitnějších variant modelů postav a herního světa). Tato videa mohou dosahovat nejvyšší kvality, jelikož jejich přehrávání je na hardware počítače relativně nenáročné. Bývají proto, především u starších her, vzhledově odlišná od zbytku hry.
- „In-game“ videosekvence, které se odehrávají uvnitř herního světa typicky takovým způsobem, že se změní úhel pohledu kamery uvnitř scény (např. může scénou „prolétat“), což může být doplněno komentářem vypravěče nebo předem danými („předskriptovanými“) pohyby postav apod. Jsou typické především tím, že odrážejí stav hry (až na výjimky, kdy je omezen zobrazovaný obsah pouze na statické objekty), což umožňuje v rámci videosekvence spatřit změny, které hráč ve hře provedl (např. postavené stavby), či v případě her pro více hráčů pohybující se postavy ostatních hráčů. Tyto videosekvence na rozdíl od předem vytvořených videí zpravidla nedosahují takové kvality, jelikož, přestože může být při přehrávání takové videosekvence pozměněn způsob renderování scény, se stále jedná o renderování, a to je na rozdíl od přehrávání videa na hardware počítače poměrně náročné.
- Scény se skriptovaným dějem přímo ve hře, které patří mezi videosekvence spíše napůl, ale je důležité se o nich zmínit. Princip se příliš neliší od „in-game“ videosekvencí, zpravidla se zde vyskytují postavy s předem daným scénářem, tj. pohyby, akcemi a dialogem, přičemž tento děj se odehraje podobně jako v předchozích dvou kategoriích videosekvencí. Obrovský rozdíl je ale v tom, že scénu vidí hráč stále z pohledu své postavy, kterou ve většině případů může během

videosekvence ovládat a v některých případech má dokonce možnost i zasáhnout do děje.

Videosekvence mohou sloužit k různým účelům, například lze jejich prostřednictvím vyprávět příběh, často se používají pro rozhovory postav, pro upozornění hráče na určitou důležitou skutečnost či jen „popostrčení“ hráče správným směrem v rámci dějové linie.

2.2.4 Grafika

Pokud se nejedná o software pro práci s grafikou, tj. grafické a video editory, nebo např. architektonický, geografický, návrhový či simulační software, zřídkakdy má grafika v softwaru větší využití než pro ovládací prvky, loga a další elementy často jen malého rozsahu. U her je tomu přesně naopak, s výjimkou poměrně malé kategorie textových her jsou hry v podstatě založené na grafice.

Zatímco v začátcích herního průmyslu vznikaly spíše 2D hry, v současné době se trend ubírá spíše ke 3D grafice, avšak i v současnosti existují úspěšné 2D hry a uplatnění tak v herním světě najdou obě varianty, které se v určitých ohledech vcelku liší, co se týče způsobu práce.

Ve 2D grafice se mohou uplatnit následující techniky používané při tvorbě počítačových her:

- Kreslení obrázků, např. pozadí scény, grafické prvky menu apod.
- Vytváření spritů, např. postav, předmětů apod.
- Tvorba 2D scén, typicky jejich skládání z hotových obrázků a spritů.
- Tvorba GUI.

Mezi techniky herní 3D grafiky patří:

- Modelování 3D objektů, kdy grafik dává objektům vyskytujícím se ve hře požadovaný tvar. Mezi objekty paří např. postavy, předměty a další prvky herního světa.
- Texturování 3D objektů, kdy grafik dává objektům jejich vzhled, co se týče barevnosti povrchu.
- Rigging, tedy vytvoření kostry typicky pro postavy, ať už lidské nebo zvířecí, která slouží jako informace o tom, jakým způsobem se postavy mohou hýbat.
- Tvorba 3D scén, která probíhá typicky skládáním z již hotových modelů.
- Programování shaderů, což je technika na pomezí grafiky a programování. Shadery jsou algoritmy, pomocí kterých probíhá renderování scéna na grafické kartě počítače.

Do technik grafiky je možno zařadit i techniky animace, jelikož s grafikou úzce souvisí. Mezi ně patří např. klíčování, dopředná kinematika, zpětná kinematika nebo motion capture.

2.2.5 Hudba, zvuk a dabing

S technikami týkajícími se zvuku je situace podobná, jako v případě grafiky: krom výjimek jako je software na tvorbu a úpravu zvuku či hudby se obecně v softwaru zvuk používá spíše sporadicky jako akustická odezva na akce vykonané uživateli, jako připomenutí nebo upoutání pozornosti uživatele. Přestože je zvuk v rámci software celkově důležitý, hry mají pro zvuk mnohem širší využití, a to v následujících oblastech:

- Zvuk uvnitř herního světa, který je velmi důležitý, jelikož dobrá hra se neobejde bez ozvučení výstřelů, kroků či např. cvaknutí kliky při otevírání dveří. Počítačová hra by měla obsahovat alespoň zvuky z této oblasti, jelikož jsou podstatným základem celého ozvučení hry.
- Hudba, která má za úkol především umocnit zážitek, který má hra uživateli poskytnout. Proto se typicky volí hudba vhodná pro danou lokaci, ve které se postava hráče nachází, přičemž se může také měnit v závislosti na situaci, do níž se hráč dostane, zejména v boji. Právě technika plynulé změny hudebního podkresu v závislosti na lokaci, situaci a případně také stavu hráčovy postavy se v moderních propracovaných hrách hojně využívá, podobně jako např. ve filmech.
- Dabing řeči postav, bez kterého se dobrá hra, ve které postavy mluví, neobejde. Je ovšem nutno říci, že existuje řada úspěšných her, kde se nevyskytuje řeč postav nebo tato řeč pro děj hry není podstatná. Krom originálního dabingu mívají některé hry lokalizovaný dabing, většinou tvořený místními distributory, což vytváří podstatnou přidanou hodnotu výsledného produktu.
- Ozvučení menu a ovládacích prvků, které hry využívají jako akustickou odezvu na akce, které hráč vykoná především v menu hry. Patří sem i ukázka hlasitosti či varianty zvuků při jejich nastavování uživatelem. Tato oblast je spíše příjemnou přidanou hodnotou pro hráče a nevyskytuje se zdaleka ve všech hrách.

2.2.6 Vývoj pro virtuální realitu

Přestože se virtuální realita v několika oborech jako je např. vojenství, lékařství či letectví používá a existuje pochopitelně i příslušný software, v oblasti počítačových her dochází v posledních letech k rozmachu této technologie, která se mimo zmíněné obory v běžném softwaru většinou nevyužívá. Je třeba podotknout, že navzdory tomu že technologie VR existuje již několik desítek let, většího rozšíření se dočkala až během cca. posledního desetiletí, a to právě díky počítačovým hrám. Jako technika specifická pro vývoj her by tedy rozhodně neměla být opomíjena.

Vývoj her pro VR je specifický především způsobem ovládání, které se od klávesnice a myši, případně od herních zařízení jako jsou gamepady, joysticky či volanty přenáší k ovládání pohybem těla, kdy namísto ovládání pohybu celé postavy hráč ovládá jednotlivé snímané části těla (typicky ruce a hlavu), což hra poté přepočítává na pohyb celé postavy. Je tedy nutno brát postavu jako entitu, jejíž části se mohou pohybovat nezávisle na sobě (např. hráč může střílet do strany nebo za sebe, přičemž se dívá dopředu) oproti pojetí postavy v klasických počítačových hrách, kde je zpravidla ovládaná jako celek (např. hráč střílí vždy tím samým směrem, kterým se dívá, tj. dopředu).

Zbývá ještě zmínit pravděpodobně zřejmou skutečnost, že pro VR v podstatě nemá smysl vyvíjet 2D hry, ale pouze 3D s tím, že se scéna pro navození trojrozměrného vjemu prakticky renderuje ze dvou úhlů (kamer), pro každé oko hráče zvlášť (což je jeden z důvodů, proč hry pro VR bývají poměrně náročné na výkon hardware počítače).

3 HERNÍ TRH

Tato kapitola se zabývá především prodejem her a jeho úskalími. Obchod s hrami má svá specifika, přestože se jinak podobá např. obchodu se softwarem obecně.

3.1 Vydavatelství

Herní vydavatel je společnost, která investuje do vývoje her a stará se o jejich propagaci na veřejnosti. Jak zmiňuje Jirkovský (2011, s. 100), vývoj kvalitní hry je v dnešní době velmi drahá záležitost a zaměstná mnohdy i několik stovek lidí, proto působí vydavatel také jako investor. Proces vydávání hry může probíhat jak interně, tak externě. Při interním procesu má vydavatel sám tým vývojářů, který hru vyvíjí. V případě externího procesu vydává hru jiného vývojářského týmu a jelikož působí zároveň jako investor, má i zde možnost zasahovat do procesu tvorby hry. Investice probíhají nejčastěji průběžně, po dosažení stanovených milníků.

3.1.1 Self-Publishing

Tento pojem by se dal přeložit jako „samovydavatelství“ a představuje takový způsob vydávání her, kdy vývojářský tým vydává a propaguje hru sám, na svoje náklady. Poslední dobou nabývá tento směr na popularitě především díky rozšíření rychlého připojení k Internetu a moderním technologiím jako jsou online platební systémy, mikrotransakce, sociální sítě či reklamní služby. Díky těmto technologiím je totiž jednodušší a méně finančně nákladné dopravit hru od vývojářského týmu ke koncovému zákazníkovi i hru propagovat. Velkým přínosem pro self-publishing je také rozvoj kanálů digitální distribuce. Pro počítače, konzole i mobilní zařízení existují on-line obchody, což vývojářům vydávání her svépomocí značně usnadňuje.

3.2 Distribuce

Distribuce je proces, pomyslná cesta, kterou musí projít hra, než se dostane od vydavatele ke hráči. Existuje více možných cest a tyto cesty se nazývají distribuční kanály. Dnes existují dva hlavní modely, jak se hry dostávají k cílovým zákazníkům (Jirkovský, 2011, s. 102).

3.2.1 Fyzická distribuce

model: vydavatel → distributor / distribuční kanál → obchodník → zákazník

Jedná se o klasický model prodeje krabicových verzí her v kamenných obchodech. Distributor má na starosti především cestu hry od vydavatele na poličky obchodů a (hlavně lokální) propagaci dané hry. Stará se také o lokální úpravy titulů, jako je například překlad obalu, manuálu a někdy i celé hry do místního jazyka.

3.2.2 Digitální distribuce

model: vydavatel → distributor / distribuční kanál → zákazník

Především díky rozšíření připojení k Internetu v poslední době se vyvinul nový model distribuce her, a to distribuce právě výhradně přes Internet. Takový model nepotřebuje obchodníka (prodávajícího), distribuční kanály směřují přímo do domova kupujícího a distributor s obchodníkem splývají v jedno a to samé.

4 PROCEDURÁLNÍ GENEROVÁNÍ

Procedurální generování nebo též procedurální modelování je technika používaná při tvorbě počítačových her. Tato technika slouží pro vytvoření většinou rozsáhlého souboru náhodných dat podle určitých pravidel z malého množství dat pomocí algoritmu – procedurálního generátoru. Počáteční data většinou tvoří

- tzv. seed (semínko či násada), který generátor následně použije pro generování náhodných čísel, na základě kterých pak tvoří výsledná data,
- nastavení procedurálního generátoru.

Pomocí techniky procedurálního generování se z pohledu tvorby počítačových her dají tvořit např.

- předměty (jejich vzhled a vlastnosti),
- postavy jak lidské, tak zvířecí (opět jejich vzhled – tvar těla či barva a charakterové vlastnosti),
- herní mapy – světy (tvar terénu, jeho textury a další prvky jako voda apod.),
- počasí,
- události (např. útoky zvířat, přistání mimozemšťanů, ...),
- úkoly (které má hráč plnit).

Oproti procedurálnímu generování bývá druhou možností využití „tradiční“ ruční tvorby, kdy herní obsah netvoří algoritmus, ale přímo vývojáři hry – obsah je tak statický. V porovnání s ruční tvorbou má procedurální generování několik výhod a několik nevýhod.

Jednou z výhod je datová velikost generovaného obsahu. Přesněji řečeno, velikost toho, co se musí skutečně ukládat. Samotný generovaný obsah je sice přibližně stejně velký, jako při ruční tvorbě, ale pro generování stačí pouze již zmíněné malé množství dat (použití semínka zajistí to, že se pokaždé ze stejných vstupních dat vygeneruje stejný obsah) a ukládat stačí jen změny, které hráč provedl.

Další výhodou je možnost nekonečných herních světů. Přesněji řečeno, velikost herního světa je vždy limitována, přinejmenším hardwarovými možnostmi. Přesto lze ale dosáhnout zdánlivě nekonečných herních světů, k čemuž je ale procedurální generování téměř nezbytné. Teoretickou možností by bylo nekonečné opakování šikovně ručně vymodelovaného světa, což se ale v praxi dle zkušeností autorky nepoužívá.

Výhodou může být též variabilita generovaného obsahu. Díky procedurálnímu generování lze vytvořit velké množství velmi různorodého obsahu. Při ruční tvorbě většinou vznikne méně navzájem odlišných objektů než při procedurálním generování.

Nespornou výhodou je také to, že procedurální generování šetří čas vývojáře hry. Vymyslet a naprogramovat algoritmus generátoru není sice jednoduché, ale následně generátor ušetří vývojáři čas strávený ruční tvorbou herního obsahu.

Procedurální generování má ale také své nevýhody. Jak už bylo naznačeno, algoritmus generátoru je mnohdy složitější implementovat – přestože ve výsledku šetří čas.

Především pro hráče je nevýhodou náročnost na procesor počítače, což se projevuje tím, že se hra déle načítá, případně může pomaleji reagovat i ve hře, pokud se obsah generuje za běhu hry.

Pro vývojáře přináší procedurální generování nevýhodu také v tom, že se těžko odhaduje výsledek a také se generátor špatně testuje. Je tedy třeba často a důkladně generátor testovat, bohužel ani tímto způsobem se nemusí odhalit chyby, které se projeví např. jednou za 1000 různých generování nebo jen při určitém nastavení generátoru.

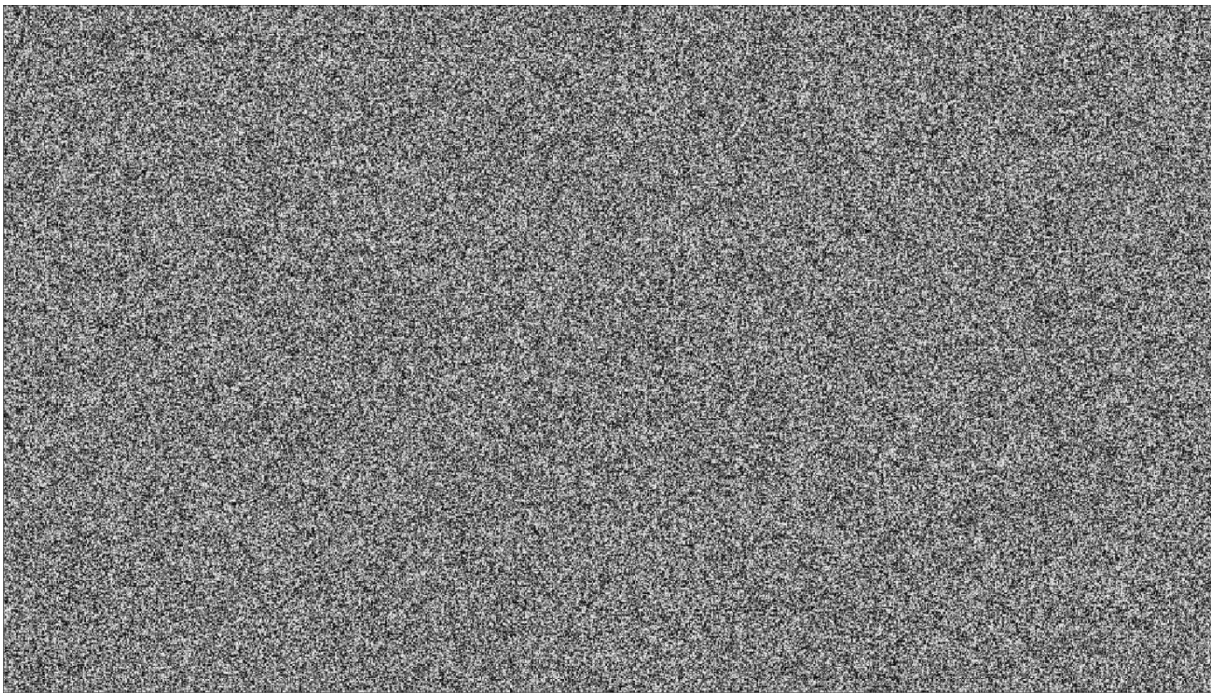
Občas vývojáři potřebují do generovaného světa přidat „statický“ (ručně vytvořený) obsah, což může být problematické na implementaci do algoritmu generátoru. To je též jeho nevýhoda, pokud potřebujeme např. přidat ručně vytvořené město do procedurálně generovaného herního světa.

V každém případě hodně záleží na kvalitě generátoru. Při nízké kvalitě (narychlo psaný algoritmus a minimum testování) se mohou často vyskytnout problémy, často jen vzhledové, někdy ale mohou tyto problémy vést až k nehratelnosti hry nebo k nutnosti vygenerovat nový svět.

V praktické části diplomové práce je využito procedurální generování pro tvorbu terénu. Následuje popis funkce tohoto konkrétního generátoru. Generátor je vytvořen podle tutoriálu (Lague, 2016a) a následně upraven pro potřeby této práce.

4.1 Šumová mapa

V první fázi práce procedurálního generátoru je vytvořena tzv. šumová mapa. Bílý šum si lze představit jako obrázek, kde jsou náhodně vygenerovány černé a bílé pixely. Obrázek 1[1] je příkladem bílého šumu.



Obrázek 1: Bílý šum

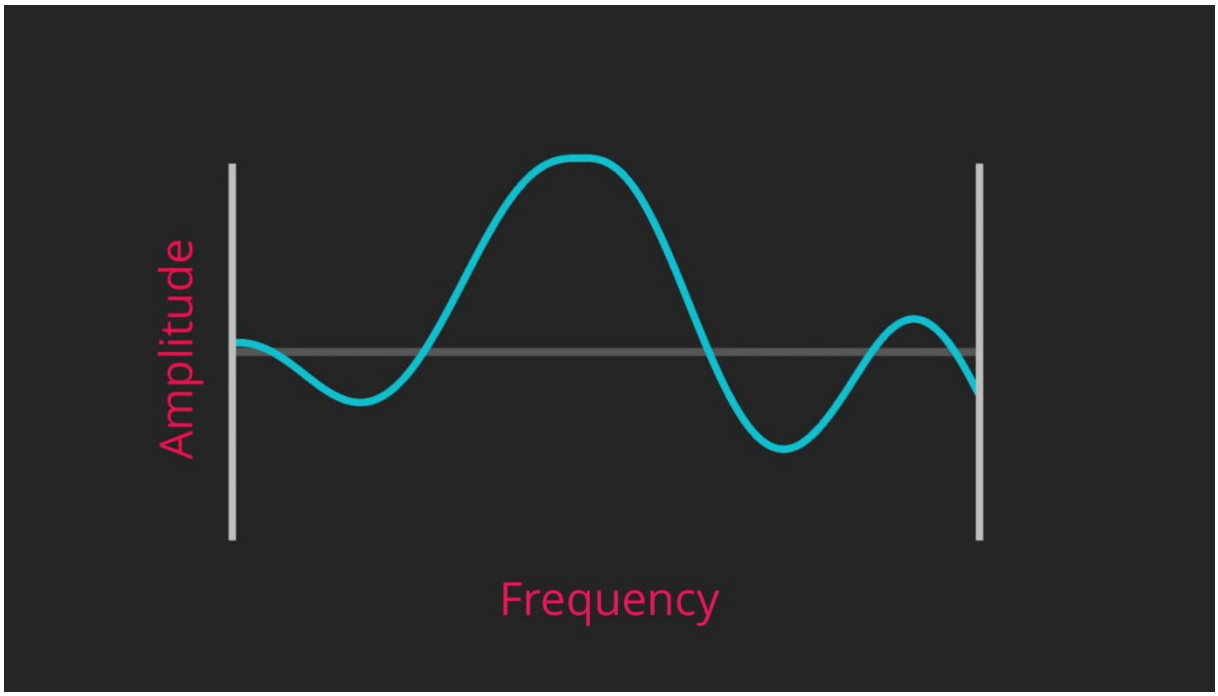
Pro potřeby generování terénu je v této práci použit takzvaný Perlinův šum, který je souvislý a lze si jej představit jako obrázek ve stupních šedi s náhodně umístěnými světlejšími a tmavšími oblastmi, které se navzájem plynule prolínají. Příkladem Perlinova šumu je Obrázek 2[2].

[1] Obrázek převzat z (Lague, 2016a).
[2] Obrázek převzat z (Lague, 2016a).

Obrázek 2: Perlinův šum

Pokud bychom udělali „řez“ Perlinovým šumem, získáme křivku připomínající část hornatého terénu sledovaného z profilu. Svislá osa *y* této křivky se nazývá amplituda (amplitude), Vodorovná osa *x* se nazývá frekvence (frequency). Příklad takové křivky znázorňuje Obrázek 3[3].

[3] Obrázek převzat z (Lague, 2016a).



Obrázek 3: Profil Perlinova šumu s označením amplitudy a frekvence

Po vygenerování terénu pomocí tohoto šumu by vznikl příliš hladký terén, což by nevypadalo přirozeně. Z tohoto důvodu je třeba přidat více detailů při zachování celkového tvaru terénu. Toho je docíleno přidáním více vrstev (úrovní) šumu. Tyto vrstvy se nazývají oktávy (octaves).

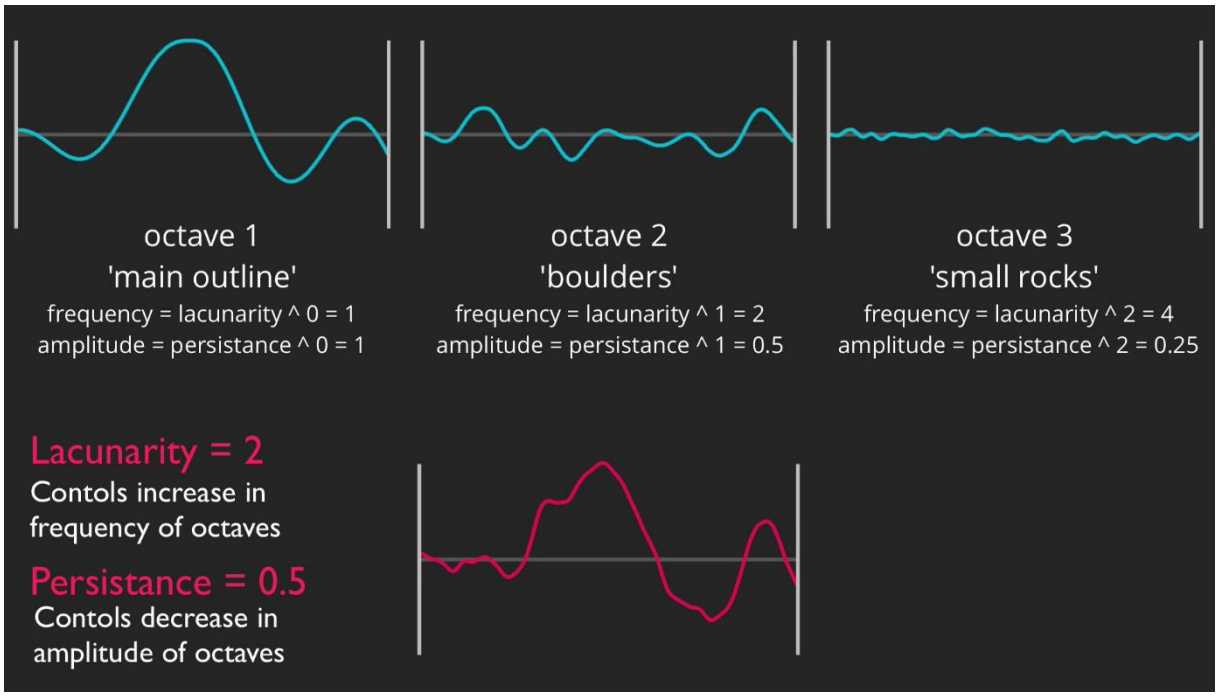
Jednotlivé vrstvy se pak sčítají do výsledného tvaru terénu. Aby ovlivňovaly terén správným způsobem, je na nich třeba provést určité úpravy, jinak by jejich přidání nemělo smysl. V tomto generátoru jsou pro tento účel vytvořeny dva artibuty, a to lakunarita (lacunarity, jinak také „mezerovitost“) a perzistence (persistance, jinak také „vytrvalost“). Z těchto atributů se pomocí následujících vzorců počítá amplituda a frekvence každé oktávy.

$$frekvence = lakunarita^{\check{c}\acute{i}slo\ okt\acute{a}vy}$$

$$amplituda = perzistence^{\check{c}\acute{i}slo\ okt\acute{a}vy}$$

Přičemž oktávy jsou číslovány po jedné od nuly, lakunarita je číslo větší než 1 a perzistence číslo mezi 0 a 1. To způsobí, že každá další oktáva má větší frekvenci a menší amplitudu, tedy i menší vliv na tvar terénu. Vliv na výše uvedenou křivku při počtu oktáv 3, lakunaritě rovné 2 a perzistenci 0,5 znázorňuje Obrázek 4[4].

[4] Obrázek převzat z (Lague, 2016a)



Obrázek 4: Vylepšení Perlinova šumu

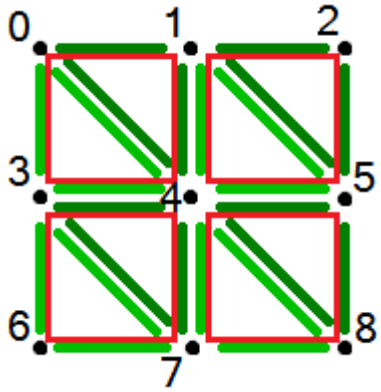
V této práci je pro generování terénu použita vestavěná metoda Mathf.PerlinNoise pro generování Perlinova šumu, která je takto vylepšená. Tato metoda se volá pro každou souřadnici šumové mapy (což si lze představit jako každý pixel nebo následně každý vrchol v terénu) zvlášť. Metoda vrací pro celočíselný vstup vždy stejné hodnoty, což není problém, ale je třeba s tím počítat. Proto se v metodě, která slouží pro generování šumové mapy, nachází parametr škála (scale), kterou je vstup vydělen, díky čemuž vstup přestanou tvořit pouze celočíselné hodnoty. Z pochopitelných důvodů škála nemůže být 0 (nastala by chyba dělení nulou). Zároveň tento parametr slouží pro nastavení tvaru výsledného terénu – obecně řečeno, čím vyšší škála, tím plošší a méně hornatý terén.

4.2 Mesh terénu

Výraz „mesh“ lze z angličtiny přeložit jako „síť“ nebo „pletivo“. V této práci je ale použit anglický výraz, jelikož se v rámci počítačové grafiky používá běžně i v češtině více než české ekvivalenty. Mesh, nebo též „polygon mesh“ („polygonová síť“ či „mnohoúhelníková síť“) si lze představit jako síť tvořenou vrcholy, mezi nimiž jsou umístěny hrany, které ohraničují plochy, které pak tvoří celý tvar.

V této konkrétní implementaci tvorby meshe je použit velice obvyklý způsob, kdy jsou vrcholy rozestavěny do pravidelné mřížky v osách *x* a *z* (horizontální osy v trojrozměrném prostoru). Tři sousední body vždy tvoří trojúhelník a dva protilehlé trojúhelníky tvoří čtverec. Tyto čtverce

jsou opět umístěny na mapě pravidelně. Lepší představu poskytne Obrázek 5. Pozice vrcholů je pak upravena vertikálně (podle osy *y*), čímž vytvoří požadovaný tvar terénu.



Obrázek 5: Mesh

Na ukázkovém obrázku meshe 3 × 3 jsou vidět černě vrcholy, které jsou spojeny zelenými hranami. Hrany tvoří trojúhelníky, na obrázku rozlišené světlejší a tmavší zelenou barvou. Trojúhelníky jsou složeny tak, aby představovaly červeně vyznačené čtverce. Pro algoritmus nejsou čtverce důležité, ale dávají nám lepší představu o rozložení mapy.

Trojúhelníky je třeba do algoritmu zanést. Pro tento účel je použito jednorozměrné pole, které definuje body, jenž tvoří trojúhelníky. Pole obsahuje body všech trojúhelníků postupně za sebou, přičemž je třeba zapisovat body jednotlivých trojúhelníků po směru hodinových ručiček. Příklad pro výše uvedený obrázek meshe: 0, 4, 3, 4, 0, 1, 1, 5, 4, 5, 1, 2, ...

Je nutno předem spočítat, kolik elementů budou obsahovat jednotlivá (číselná) pole pro mesh. Počet vrcholů spočítáme jednoduše:

$$počet\ vrcholů = šířka \cdot výška$$

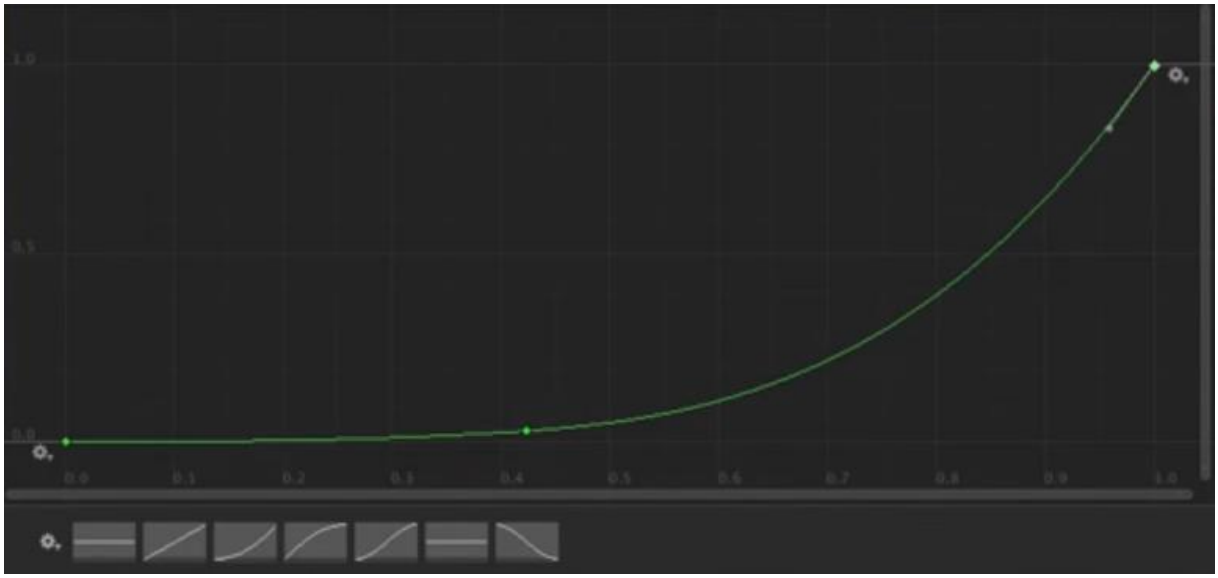
Velikost pole pro data o trojúhelnících má jen mírně složitější výpočet. Nejprve potřebujeme znát počet čtverců mezi jednotlivými vrcholy. Z výše uvedeného obrázku by mělo být patrné, že jich je v řadě i sloupci o jeden méně než vrcholů. Každý čtverec tvoří dva trojúhelníky, které mají každý 3 body. Proto:

$$velikost\ pole\ trojúhelníků = (šířka - 1) \cdot (výška - 1) \cdot 6$$

Jak již bylo zmíněno, je třeba ještě nastavit pozici vrcholů vertikálně, jinak by výsledkem byla pouze rovná plocha. K tomuto účelu použijeme právě výše uvedenou šumovou nebo též výškovou mapu. Souřadnice meshe jsou trojrozměrné a souřadnice výškové mapy dvojrozměrné, přičemž souřadnice *x* meshe (x_m) odpovídá souřadnici *x* výškové mapy (x_v), ale

souřadnice *z* meshe (z_m) odpovídá souřadnici *y* výškové mapy (y_v), protože souřadnice *y* meshe (y_m) je výška terénu (ta odpovídá hodnotě výškové mapy v tomto bodě). Pro vrchol meshe na souřadnicích x_m, y_m, z_m nastavíme souřadnici y_m na hodnotu výškové mapy v bodě $x_v = x_m$ a $y_v = z_m$.

V nastavení výškové mapy existují ještě dva parametry, násobič výšky terénu (height multiplier) a výšková křivka (height curve). Násobič určuje, jak vysoký celý terén bude – bez násobiče by byl terén velmi plochý (což odpovídá nastavení násobiče na 1). Výškova křivka určuje, jak moc by měly být různé hodnoty výšky terénu ovlivněny násobičem. Hodnoty na ose *x* od 0 do 1 reprezentují skutečné hodnoty výšky, které jsou uložené ve výškové mapě, na ose *y* je pak možno definovat, jak se tyto hodnoty projeví v meshi (ve výsledné výšce terénu). Příklad výškové křivky názorně zobrazuje Obrázek 6[5].



Obrázek 6: Výšková křivka

4.3 Skládání terénu

Celý terén není jeden velký objekt (mesh), nýbrž je složen z několika menších, a to hned ze dvou důvodů, avšak v této práci ve skutečnosti jen z jediného důvodu. Editor Unity totiž nepodporuje meshe o velikosti více, než 65535 vrcholů, což odpovídá šířce i výšce[6] terénu 255 bodů ($255 \cdot 255 = 65\,025$, ale $256 \cdot 256 = 65\,536$, což už přesahuje povolenou mez).

Druhý důvod je to, že se při generování mapy po částech rozkládá zátěž na procesor, která by jinak vznikla při startu hry, do delšího časového úseku, a provádí se jen tehdy, když je potřeba

[5] Obrázek převzat z (Lague, 2016b)
[6] Je použit čtvercový rozměr (výška i šířka jsou stejné), jelikož práce s ním je jednodušší a dochází k podstatně méně komplikacím při různých výpočtech.

(hráč se přiblíží do oblasti s nevygenerovanou mapou). Při generování nekonečného terénu je načítání po částech nutnost. Avšak toto neplatí pro tento projekt, jelikož používá relativně malou omezenou velikost mapy a celá mapa je načtena ihned, protože se na ní pohybuje jak kamera hráče, tak jednotlivé postavy, a je tedy třeba ji mít načtenou celou najednou.

Jeden mesh, tj. jedna část terénu, se nazývá chunk (v překladu „kus“, zde ale bude opět používán i v češtině v herní terminologii obvyklý anglický výraz). Tyto chunky jsou umístěny těsně vedle sebe tak, aby tvořily souvislý terén. Pochopitelně jsou generovány tak, aby na sebe tvarem terénu plynule navazovaly.

5 EVOLUČNÍ SCHÉMATA

Ve vývoji her evoluční schéma představuje prvek hry, který se na základě určitých skutečností vyvíjí. Nejčastěji to bývá na základě akcí provedených hráčem nebo času. Technicky vzato si lze evoluční schéma představit jako graf stavů, mezi kterými lze předem danými směry přecházet.

Pojem evoluční schéma není v oblasti herního průmyslu dle dostupných zdrojů příliš známý. Pravděpodobně nejznámější příklad evolučního schématu ve hrách je technologický strom (tech tree).

5.1 Obecná herní evoluční schémata

Evoluční schéma ve většině případů nebývá ve hře vizuálně zachyceno. Typickou výjimkou bývají stromy výzkumu, což je zpravidla zobrazení pokroku hráče ve výzkumech v dané hře. Přestože ostatní evoluční schémata zpravidla nejsou ve hře přímo zobrazena, bývají součástí Game design dokumentu.

Je zřejmé, že ne všechny druhy evolučních schémat jsou součástí všech her. Jednoduchá malá hra s jednostránkovým menu pravděpodobně nebude mít evoluční schéma popisující menu podobně jako typická jednoduchá střílečka nebude obsahovat technologický strom, strom výzkumu, úkolovou linii apod.

5.1.1 Stromy výzkumu

Stromy výzkumu jsou jako jedny z mála evolučních schémat většinou viditelné i ve hře. Jak napovídá slovo „strom“ v názvu, zpravidla mají tyto grafy podobu stromů. Obsahují výzkumy, které hráč může ve hře objevit nebo vynalézt. V různých hrách se systém výzkumů velmi liší a nejenže graf nemusí mít nutně podobu stromu, ale nemusí být zobrazen vůbec a výzkumy probíhají například náhodně. Existují dva typické způsoby zadávání úkolů: buď hráč zadá výzkum, na jehož dokončení pak čeká danou dobu, nebo v průběhu hry sbírá výzkumné body a za ty si pak výzkum koupí. Navíc dokončení výzkumu může hráče stát i nějaké suroviny. Výzkumy na sebe určitým způsobem navazují a nelze spustit následující výzkum bez dokončení předchozího. Tyto návaznosti jsou dané návrhem hry.

5.1.2 Technologické stromy

Technologické stromy bývají často zaměňovány se stromy výzkumu. Tyto jsou prakticky v technologických stromech zahrnuty, ale technologické stromy poskytují více informací o tom, které technologie jsou hráči skutečně dostupné. Krom hotového výzkumu musí mít často např.

k dispozici ještě postavenou budovu nebo vyrobený předmět, aby mohl danou technologii použít.

5.1.3 Úkolová linie

Hry, ve kterých hráči plní různé úkoly, mohou mít systém úkolů více či méně propracovaný. Typicky se úkoly řadí do úkolových linií, v rámci kterých úkoly navazují lineárně za sebou, tj. vždy je potřeba dokončit předchozí úkol, než je možno plnit úkol následující. Tyto úkolové linie na sebe zpravidla též navazují, ale ne vždy lineárně. A právě zde najde využití evoluční schéma, ve kterém jsou zaneseny úkolové linie a je tak zřejmé, jak na sebe navazují, které lze plnit paralelně a které je naopak třeba plnit v sérii.

5.1.4 Příběh

Hry obsahující příběh se mohou na základě vývoje děje měnit (typicky se změní něco přímo v herním světě, otevře se nová lokace apod.). Tyto změny mohou být zachyceny opět v evolučním schématu. Nabízí se jistá podobnost se schématem stavu hry, avšak toto schéma se zpravidla vyznačuje tím, že se mezi stavy dá přecházet obousměrně, zatímco v evolučním schématu příběhu se stav mění nejčastěji jednosměrně. Příběhová evoluční schémata mají využití zejména u her s rozvětveným příběhem, kde se hráč dostane jen do omezeného počtu uzlů grafu evolučního schématu příběhu, projde pomyslnou linii grafem od začátku do konce, přičemž konců může být i více, v závislosti na konkrétní hře.

5.1.5 Dialogy

Rozhovory postav s hráčem bývají především v příběhově založených hrách jako např. adventurách či RPG poměrně rozvětvené. Výběr dialogů může ovlivnit příběh (viz výše), ale samotné dialogy bývají různě provázané bez ohledu na to, zda děj ovlivní nebo ne. Typickým příkladem je to, když má hráč v rozhovoru s postavou na výběr z několika otázek a pořadí závisí na něm, tak se po zodpovězení otázky postavou vrací do dialogu s výběrem otázek, aby se mohl zeptat na něco dalšího.

5.1.6 Vývoj postavy (talentové stromy)

Talentové stromy bývají běžnou součástí především her RPG a zpravidla se jedná o evoluční schéma viditelné hráčem, podobně jako stromy výzkumu. Je to jeden ze způsobů, kterými se herní postava rozvíjí. Hráč má možnost si v rámci daných možností určit, v čem bude jeho postava vynikat, volbou (aktivací) tzv. talentů z talentového stromu postavy (což jsou v podstatě uzly grafu tohoto evolučního schématu). Talenty mohou být v závislosti na návrhu hry vybrány i vícekrát – v tomto případě se většinou využívá tzv. talentových bodů, které hráč

získává v průběhu hry a poté je umisťuje (utrácí) do jednotlivých talentů až do jejich maximální kapacity. Talentové stromy mohou mít různou podobu, např. se mohou podobat stromům výzkumu. Běžnější podoba návaznosti jednotlivých uzlů (talentů) spočívá v rozdělení do úrovní, které lze připodobnit k úrovním kořenového stromu, nejedná se však o klasický strom, spíše si jej lze přestavit jako orientovaný graf, kde ze všech uzlů na jedné úrovni můžeme přejít do všech uzlů na další úrovni a takto přecházet postupně mezi jednotlivými úrovněmi. Tento přechod na další úroveň je v herním návrhu často podmíněn, např. výběrem jednoho z talentů v současné úrovni nebo utracením daného počtu talentových bodů v současné úrovni.

5.2 Stavové diagramy

Evoluční schémata jsou snadno zaměnitelné se stavovými diagramy, ačkoliv se jedná o mírně odlišné druhy grafů. Hlavním rozdílem je to, že v evolučním schématu se ve většině případů pohybujeme jedním směrem, zatímco pohyb opačným směrem je spíše výjimečný jev. Ve stavovém diagramu je běžný pohyb všemi možnými směry a tento diagram je tedy zpravidla neorientovaným grafem. Níže jsou uvedeny druhy stavových diagramů, které se ve vývoji her mohou uplatnit.

5.2.1 Stav hry

Každá hra má své stavy, mezi kterými lze přecházet a tyto stavy se u různých her velmi liší. U bojových her, ať už se jedná o strategie či střílečky, může hra přecházet mezi stavem boje a stavem klidu, od čehož se odvíjí ve většině případů přehrávaná hudba, ale může se změnit i styl ovládání a u her s tahovým ovládáním boje se případně pozastaví čas, aby se hráč mohl věnovat vymýšlení strategie pro boj. U šachů by se mohl stav hry měnit podle toho, který hráč zrovna hraje a v daném okamžiku přejít do stavu „šach“, pro který existují mírně odlišná pravidla. Graf stavového diagramu pro stavy hry může být jak neorientovaný, tak orientovaný, typicky pokud do takového grafu zahrneme stavy jako např. „šach-mat“, ze kterých není návratu zpět.

5.2.2 Menu

Hlavní menu i vnitřní („in-game“) menu ve hře mívají podmenu, typicky s různými nastaveními hry. To, jak na sebe menu navazují, lze zachytit právě pomocí stavového diagramu, kde jedna stránka menu je jeden stav, potažmo jeden uzel grafu. Mezi těmito stavy hráč přepíná pomocí pohybu v menu, tj. vstupováním do podmenu a vystupováním z nich. V tomto případě se bude typicky jednat o neorientovaný graf, jelikož se lze v menu pohybovat většinou libovolným směrem.

6 IMPLEMENTACE POČÍTAČOVÉ HRY TRIBE SURVIVAL

Praktickou částí této diplomové práce je proof-of-concept počítačové hry nazvané Tribe survival (v překladu „Přežití kmene“). Jedná se o multi RPG survival simulaci. Děj hry se odehrává v pravěku v době kamenné, kde hráč je v roli náčelníka kmene pravěkých lidí. Cílem hry je přežití tohoto kmene a jeho rozvoj.

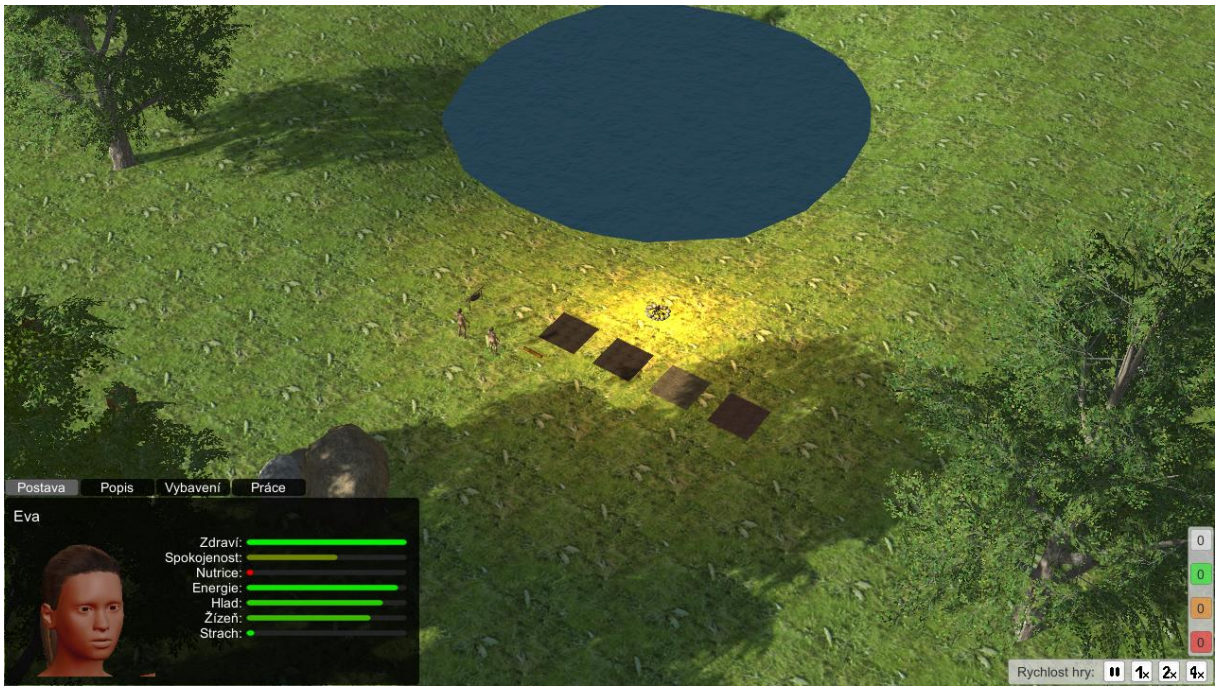
Po spuštění hry hráč uvidí pohledem shora herní svět, ve kterém se nachází zdroje surovin, postavy a stavby. Herní svět je situován do letního období mírného až subtropického podnebného pásu, hráč tedy nemusí brát v potaz teplotu prostředí. Úkolem hráče je v první řadě postavám zajistit naplnění jejich potřeb tím, že jim bude zadávat úkoly. Postavy pak tyto úkoly plní a tím např. získávají suroviny, které mohou dále využít. Dalším úkolem hráče je rozvíjet znalosti postav, což probíhá prostřednictvím výzkumů. Zajímavostí je, že tyto výzkumy neprobíhají výběrem konkrétního výzkumu, jako tomu běžně v počítačových hrách je, ale zkoušením různých kombinací předmětů, z čehož poté může, ale nemusí vzejít úspěšný výzkum. Postavy se snaží hráči pomoci tím, že mají vlastní vůli a plní své základní potřeby tak, že se samy chodí napít, pokud mají jídlo tak se samy nají a v případě potřeby se vyspí. Úkolem hráče je zajistit jim další potřeby a dostatek zásob k přežití.

Implementace navazuje na diplomovou práci Herní design (Černík, 2018), jejíž koncept je podstatně rozšířen dle game design dokumentu, který je její součástí, a dále je rozšířen o procedurální generátor modelu herního světa. Právě procedurální modelování jakožto jedno z nosných témat této diplomové práce je realizováno jako procedurální generátor modelu herního světa. Druhé nosné téma, evoluční schémata, je realizováno prostřednictvím technologických stromu výzkumů a výroby. Tato témata byla teoreticky popsána v předchozích kapitolách, samotná praktická realizace bude popsána v následujícím textu.

6.1 Menu a varianty hry

Po spuštění aplikace, ještě před vstupem do samotné hry, se uživateli objeví hlavní menu, kde má možnost vstoupit do hry nebo celou aplikaci ukončit. Na výběr jsou dvě možné varianty herního světa: Ručně připravený svět a procedurálně generovaný svět.

Ručně připravený svět slouží jako ukázka všech herních principů implementovaných v této práci s výjimkou procedurálního generátoru světa. Ten je z důvodů značné implementační náročnosti a náhodnosti výsledného vygenerovaného světa implementován odděleně, jelikož plná hratelnost není z těchto důvodů zaručena. Podobu ručně vytvořeného světa znázorňuje Obrázek 7.



Obrázek 7: Ukázka ručně vytvořeného světa hry Tribe survival

Procedurálně generovaný svět obsahuje stejné herní principy jako ručně připravený svět s tím rozdílem, že mapa je generovaná na základě principů uvedených v kapitole 4 „Procedurální generování“ a na ni jsou následně za pomocí procedurálního generátoru objektů rozmístěny náhodně zdroje surovin a pevně ostatní objekty (např. postavy či stavby). Jednu z možných podob procedurálně generovaného světa zobrazuje Obrázek 8.

Obrázek 8: Ukázka procedurálně generovaného světa hry Tribe survival

6.2 Typický průběh hry

Pro lepší představu o hře a postupu hráče hrou od začátku do konce je dále popsán typický scénář, jak hraní může probíhat. Hráč se samozřejmě nemusí držet tohoto postupu, je to ale zamýšlený způsob, jak hru dokončit a na tento postup je hra testovaná.

Na začátku hry je dobré vesničanům zajistit dostatek potravy. Hráč dá vesničanům za úkol natrhat jablka z jabloně, což provede pomocí kontextového menu jabloně (obdobně provádí i získávání dalších surovin). Pokud je při zadávání úkolu označen vesničan, zadá mu tím úkol přímo a vesničan jde úkol provést. Pokud nemá označeného vesničana, úkol je zadán nepřímo a jeden z vesničanů se ho posléze ujme (více o zadávání úkolů viz oddíl 6.6 „Úkoly a umělá inteligence postav“). Jedna jabloň obsahuje 50 jablek. Vesničané natrhaná jablka odnosí do skladiště, odkud si je v případě potřeby mohou sami brát.

Následně je třeba zajistit dostatek materiálu na výzkumy. Základní potřebné materiály jsou kameny a větve. Větve lze získat otrháváním ze stromů, kameny jak těžbou, tak sběrem oblázků, z hroudy kamenů. V každém surovinovém ložisku je 10 nebo 20 kusů dané suroviny pro každý typ sběru. Podobně jako při sběru jablek vesničané získané suroviny uskladní. V případě potřeby se lze kdykoliv k získávání surovin vrátit a doplnit tak zásoby.

Dalším krokem je výzkum. Hráč pomocí kontextového menu výzkumného stanoviště postupně zadává předměty, které se mají pro výzkum použít. Po jejich zadání hráč výzkum odstartuje, opět pomocí kontextového menu. Pro možnost zahájení výzkumu je třeba zvolit 2–5 předmětů. Po zahájení výzkumu vesničan nanosí potřebné předměty do výzkumného stanoviště a začne zkoumat. V závislosti na několika různých faktorech se výzkum může, ale nemusí povést. Všechny použité předměty se ale v každém případě spotřebují. Více informací o výzkumech, jejich podmínkách i surovinách potřebných pro jednotlivé výzkumy je popsáno v oddílu 6.10 „Výzkumy“ níže.

Po úspěšném dokončení výzkumu se zpřístupní výroba nového vynálezu ve výrobním stanovišti, kde lze od okamžiku, kdy je výzkum hotov, zadat úkol na jeho výrobu přes kontextové menu. Ihned po výběru předmětu začne vesničan nosit potřebné předměty do výrobního stanoviště obdobně jako při výzkumu, načež započne samotnou výrobu. Pro výrobu jsou potřeba stejné suroviny, jaké byly potřeba na výzkum, bez ohledu na to, jaké suroviny byly ve skutečny při výzkumu spotřebovány. Při výrobě jsou tyto suroviny spotřebovány a vesničan hotový výrobek odnese do skladu.

Jelikož hra obsahuje několik výzkumů a některé z nich vyžadují nejen znalost předchozích výzkumů, ale také předměty, které lze vyrobit pouze se znalostmi z předchozích výzkumů, je třeba dále systematicky střídat výzkumy a výrobu (přičemž lze díky více herním postavám tyto činnosti provádět současně). Přitom by se měl hráč též starat o potřeby jednotlivých postav a o případné doplňování potřebných surovin do skladiště.

Pomyslný konec hry nastává v okamžiku, kdy hráč vyzkoumá všechny výzkumy, které hra obsahuje. Nic mu ale nebrání hrát dále, hra se po dokončení posledního výzkumu neukončí.

6.3 Ovládání hry

Interakce hráče se hrou probíhá primárně pomocí myši jak v herním světě, tak pomocí GUI, které je popsáno v následujícím oddílu. Některé akce lze též ovládat klávesnicí.

V rámci herního světa se lze pohybovat pomocí kláves W, A, S a D, kurzorovými šipkami, případně stisknutím a podržením pravého tlačítka myši a současně pohybem myši proti směru, kam se má pohled posunout (hráč „posouvá herní mapu“). Lze se otáčet kolem bodu, na který je pohled zaměřen (střed obrazovky) přidržením prostředního tlačítka myši a současným pohybem myši do strany. Stejným způsobem lze naklápět pohled také vertikálním směrem při pohybu myši nahoru nebo dolů.

Kliknutí levým tlačítkem myši pak slouží pro označování postav a dalších objektů ve scéně, po kliknutí pravým tlačítkem myši se zobrazí kontextové menu, přes které se provádí podstatná část ovládání celé hry od zadávání úkolů, přes nastavování výzkumů až po manipulaci s inventářem.

Klávesy 1, 2 a 3 slouží pro ovládání rychlosti plynutí času ve hře, přičemž lze čas nastavit na normální rychlost, případně jeho plynutí zrychlit 2× nebo 4×. Klávesou 0 nebo P lze hru pozastavit. Klávesa Escape pozastaví hru a zároveň zobrazí herní menu.

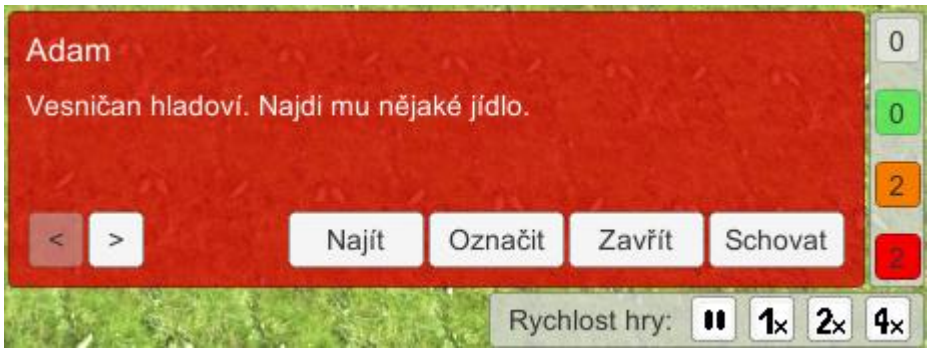
Zpracování naprosté většiny událostí souvisejících s ovládáním zajišťuje třída *InputController*.

6.4 GUI

Grafické uživatelské rozhraní hry je rozděleno do několika částí rozmístěných v předem daných pozicích na obrazovce. V pravém dolním rohu se nachází horizontální panel, který slouží pro ovládání rychlosti plynutí času stejně, jako výše zmíněné klávesy 0, 1, 2 a 3.

Dále se též v pravém dolním rohu nachází vertikální panel notifikací, na kterém se nachází barevná tlačítka, kde barva označuje úroveň důležitosti notifikace a číslo na tlačítku vyjadřuje

aktuální počet aktivních notifikací dané úrovně. V případě vzniku nové notifikace se číslo na tlačítku navýší o 1, tlačítko zabliká a po kliknutí na něj se zobrazí text notifikace a tlačítka pro práci s touto notifikací. Podobu panelu notifikací včetně zobrazení obsahu samotné notifikace a příslušných tlačítek spolu s panelem pro ovládání rychlosti plynutí času zobrazuje Obrázek 9.



Obrázek 9: Podoba GUI notifikací a ovládání plynutí času

Za pomocí tlačítek se šipkami vlevo a vpravo je možno listovat mezi notifikacemi v rámci jedné úrovně důležitosti. V případě, že notifikace pochází od postavy, tlačítko „Najít“ zaměří pohled na tuto postavu, zatímco tlačítko „Označit“ tuto postavu označí tak, jako by na ni hráč klikl ve scéně levým tlačítkem myši. Tlačítko „Zavřít“ notifikaci zavře a odebere ze seznamu aktivních notifikací, Tlačítko „Schovat“ slouží pro její minimalizaci zpět do panelu notifikací.

Správu notifikací zajišťuje třída *NotificationGuiController*, datovou složku samotné notifikace pak představuje třída *Notification*.

6.4.1 Kontextová nabídka pro vybraný objekt

V případě, že hráč vybere postavu nebo jiný objekt, v levém dolním rohu obrazovky se objeví kontextová nabídka, jejíž obsah je závislý na tom, jaký objekt je vybrán. Pokud se jedná o postavu, zobrazí se následující záložky s dále popsaným obsahem:

- Postava – zde se nachází informace o aktuálním stavu potřeb postavy. Po najetí myší na jednotlivé zobrazení linky se objeví textová informace o úrovni této potřeby, v případě strachu pak informace o úrovních jeho jednotlivých složek.
- Popis – zde hráč nalezne jaké má postava charakterové rysy a její případné status efekty.
- Vybavení – jinými slovy inventář postavy, kde je v levé části zobrazeno, co má postava na sobě nebo v rukách, v pravé části se pak objeví batoh a jeho obsah, pokud postava na sobě batoh má.

- Práce – na této záložce lze nastavit priority práce přetahováním jednotlivých ikonek pomocí levého tlačítka myši. Pořadí priorit je takové, jako při čtení knihy, tzn. úkoly, které patří do kategorie úplně vlevo nahoře se provedou jako první a úkoly patřící do kategorie dole vpravo dole jako poslední. Dále je zde možnost postavě deaktivovat činnosti, které vykonává z vlastní vůle, pomocí checkboxů „Pít“, „Jíst“ a „Spát“.

Více o principech a významu jednotlivých prvků je uvedeno v následujících oddílech, zejména v oddílu 6.5 „Postavy“.

V případě, že je vybráno skladiště nebo jiná budova ve které lze uskladňovat předměty, jsou zobrazeny následující záložky:

- Popis – popis budovy, především její účel. Může obsahovat též informace o jejím stavu, např. u výzkumného stanoviště se zobrazují informace o průběhu výzkumu.
- Skladiště (popř. Výzkum, v závislosti na druhu budovy) – zde je seznam předmětů, které jsou uvnitř této budovy uskladněné. V případě výzkumného stanoviště se jedná o předměty, které jsou již fyzicky připravené na výzkum.

Pokud je vybrán jiný objekt než výše změněné, zobrazí se kontextová nabídka pouze s popisem, nenachází se zde tedy žádné záložky.

Pro obsluhu kontextové nabídky slouží třída *SelectedGuiController*, obsah některých záložek se řídí svou vlastní třídou: obsah záložky „Postava“ se tvoří dle *VillagerStats* aktuálně vybraného vesničana, obsah záložky „Vybavení“ zajišťuje *EquipGuiController*, který využívá *VillagerInventoryManager* vybraného vesničana, záložka „Práce“ se řídí dle *VillagerJobs* zvoleného vesničana. Pokud je vybráno skladiště nebo výzkumné či výrobní stanoviště, třída *StorageGuiController* spravuje obsah záložky „Skladiště“, „Výzkum“ či „Výroba“, k čemuž využívá třídy *StorageBehaviour* nebo jejího potomka příslušící k vybrané budově. Obecné informace o jménu, obrázku či popisu zvolené entity se získávají ze *SelectableEntity*, což má přiřazené každá entita, kterou lze vybrat.

6.5 Postavy

Herními postavami jsou pravěcí lidé, kteří tvoří kmen a spolupracují spolu. Hráč jim zadává úkoly, které pak postavy vykonávají. Postavy samy o sobě také vykonávají činnosti potřebné k jejich přežití: spát, jíst a pít. Úkoly a činnosti postav jsou blíže popsány v samostatném oddílu 6.6 „Úkoly a umělá inteligence postav“ níže.

6.5.1 Potřeby postav

Podobně jako v reálném životě mají postavy určité potřeby, které je potřeba naplňovat. Postavy v Tribe Survival mají následující potřeby:

- zdraví,
- energie,
- hlad,
- žízeň,
- spokojenost,
- nutrice,
- strach.

Strach je tvořen třemi složkami, a to:

- strach ze tmy,
- strach ze samoty,
- strach ze vzdálení se od domova.

Každá potřeba je reprezentována body od 0 do 100, přičemž více bodů znamená lepší naplnění potřeby, krom strachu a jeho složek, u kterých je tomu přesně naopak.

Správa potřeb postav využívá ke své činnosti třídu *VillagerStats*, kde se nachází hodnoty všech potřeb.

6.5.2 Charakterové rysy

Postavy mají určité charakterové rysy, které slouží k vytvoření různorodých charakterů postav, díky čemuž nejsou všechny postavy ve hře stejné, a to pobízí hráče jednat více strategicky. Každá postava má vždy dva charakterové rysy (jeden kladný a jeden záporný, např. Silák a Jedlík), jsou „vrozené“ a nedají se měnit. Zároveň postava nesmí mít dva odpovídající opačné rysy (např. když je lenoch, nemůže být zároveň pracant). Charakterové rysy ovlivňují dané vlastnosti postavy, což spolu s výše zmíněnými informacemi popisuje Tabulka 1 níže.

Tabulka 1: Seznam charakterových rysů

Název	Popis	Ovlivňuje	Množství	Polarita	Opačný rys
Lenoch	Rychlost práce -2 WU/GH	Rychlost práce	-2 WU/GH	Záporný	Pracant
Pracant	Rychlost práce +2 WU/GH	Rychlost práce	+2 WU/GH	Kladný	Lenoch

Název	Popis	Ovlivňuje	Množství	Polarita	Opačný rys
Silák	Nosnost +10 kapacita INV	Nosnost (kapacita INV)	+10	Kladný	Slaboch
Slaboch	Nosnost -10 kapacita INV	Nosnost (kapacita INV)	-10	Záporný	Silák
Jedlík	Modifikátor úbytku hladu 1,25	Modifikátor úbytku hladu	× 1,25	Záporný	Asketa
Asketa	Modifikátor úbytku hladu 0,75	Modifikátor úbytku hladu	× 0,75	Kladný	Jedlík
Ranní ptáče	Modifikátor úbytku spánku 0,75	Modifikátor úbytku spánku	× 0,75	Kladný	Sedmispáč
Sedmispáč	Modifikátor úbytku spánku 1,25	Modifikátor úbytku spánku	× 1,25	Záporný	Ranní ptáče
Optimista	Spokojenost má trvalý bonus +10 bodů	Spokojenost	+10 bodů	Kladný	Pesimista
Pesimista	Spokojenost má trvalý postih -10 bodů	Spokojenost	-10 bodů	Záporný	Optimista

Charakterové rysy jsou u postavy uloženy ve *VillagerStats* v podobě status efektů (scriptable object třídy *StatusEffect*, viz níže).

6.5.3 Status efekty

Jak již bylo zmíněno v oddílu 1.1 „Základní pojmy“, status efekty představují určité zvýhodnění či znevýhodnění pro postavu. V této hře jsou dva hlavní typy status efektů, a to efekty charakterových rysů a efekty ovlivňující spokojenost. Prvně zmíněné jsou dány charakterovými rysy, které postava má. Ve hře se ukazují jako charakterové rysy, ale vnitřně fungují jako status efekty a chovají se v principu stejně, jako druhý zmíněný typ. Efekty ovlivňující spokojenost

vznikají za předem daných podmínek na základě různých událostí nebo úrovní hodnot potřeb, případně na základě náhody. Tento druh efektu ovlivňuje pouze spokojenost postavy. Seznam všech status efektů ve hře uvádí Tabulka 2.

Tabulka 2: Seznam všech status efektů a příslušných souborů

Soubor	Jméno	Typ
Happiness_Fears	Bojí se	Efekt na spokojenost
Happiness_FeelingSafe	Cítí se v bezpečí	Efekt na spokojenost
Happiness_FriendBanished	Vyhnání přítele	Efekt na spokojenost
Happiness_FriendLeft	Odchod přítele	Efekt na spokojenost
Happiness_Frightened	Vystrašený	Efekt na spokojenost
Happiness_Happy	Šťastný	Efekt na spokojenost
Happiness_Hungry	Hladový	Efekt na spokojenost
Happiness_Injured	Zraněný	Efekt na spokojenost
Happiness_NewDiscovery	Nový objev	Efekt na spokojenost
Happiness_NewInvention	Nový vynález	Efekt na spokojenost
Happiness_ScaredToDeath	Vyděšený k smrti	Efekt na spokojenost
Happiness_Sorrow	Zármutek	Efekt na spokojenost
Happiness_Unhappy	Nespokojený	Efekt na spokojenost
Happiness_WellFed	Dobře najedený	Efekt na spokojenost
Trait_Ascetic	Asketa	Charakterový rys
Trait_Dormouse	Sedmispáč	Charakterový rys
Trait_Drudger	Pracant	Charakterový rys
Trait_EarlyBird	Ranní ptáče	Charakterový rys
Trait_Eater	Jedlík	Charakterový rys
Trait_Feebling	Slaboch	Charakterový rys
Trait_Lazybones	Lenoch	Charakterový rys
Trait_Optimist	Optimista	Charakterový rys
Trait_Pessimist	Pesimista	Charakterový rys
Trait_Strongman	Silák	Charakterový rys

Status efekty pro charakterové rysy jsou aktivní stále, po celou dobu života postavy. Status efekty ovlivňující spokojenost jsou dočasné, přičemž existují dva druhy:

- časově omezené, které mají uvedený čas trvání,
- stavově omezené, které zmizí po dosažení určitého stavu (ten je většinou opačný ke stavu, který efekt vyvolal).

Status efekty ovlivňující spokojenost působí na spokojenost okamžitě po svém vzniku, přičemž se úroveň spokojenosti mění skokově o určený počet bodů. Vznik tohoto druhu status efektu je určen předem danými podmínkami na základě události ve hře, hodnoty úrovně potřeby či na základě náhody. Dále tyto efekty mohou mít své protiefekty. Pokud je na dané postavě aktivní konkrétní protiefekt, nelze již tento efekt aplikovat, a to ani v případě, že byly splněny podmínky pro jeho vznik. Přehled status efektů ovlivňujících spokojenost uvádí Tabulka 3.

Tabulka 3: Status efekty ovlivňující spokojenost postav

Jméno	Popis	Body	Doba	Protiefekt
Šťastný	Vesničan se cítí šťastný. Zvyšuje spokojenost o 5 bodů na 8 hodin. Může se objevit s pravděpodobností 5 % každé dvě hodiny. Pokud je efekt již přítomen na vesničanovi, efekt se prodlouží na max. dobu 8 h. Pokud je osadník nespokojen, nemůže být spokojen.	+5	8h	Nespokojený
Nespokojený	Stejné jako spokojený, pouze opačný efekt na spokojenost. Pokud je vesničan spokojen, nemůže být nespokojený.	-5	8h	Šťastný
Hladový	Vesničanův status hladu klesl pod hodnotu 15 bodů. Spokojenost klesne o 5 bodů. Pokud se vesničan nají a hodnota hladu přeroste 15 bodů, status zmizí.	-5	-	-
Dobře najedený	Vesničanův status hladu překročil 90 bodů. Spokojenost se zvyšuje o 5 bodů. Pokud hlad klesne pod 90 bodů, status zmizí.	+5	-	-
Zraněný	Vesničan cítí bolest, a to se mu vůbec nelíbí. Spokojenost se snižuje o 10 bodů. Efekt je aplikován, pokud má vesničan méně než 70 bodů zdraví. Pokud hodnota zdraví stoupne nad 70, efekt mizí.	-10	-	-

Jméno	Popis	Body	Doba	Protiefekt
Cítí se v bezpečí	Aktivuje se, pokud je globální hodnota strachu rovna nebo menší než 10 bodů. Přidává 5 bodů spokojenosti.	+5	-	-
Bojí se	Aktivuje se, pokud je globální hodnota strachu rovna nebo vyšší než 30. Odebírá 5 bodů spokojenosti.	-5	-	-
Vystrašený	Aktivuje se, pokud je globální hodnota strachu rovna nebo vyšší než 50. Odebírá 10 bodů spokojenosti. Nahrazuje status efekt Bojí se.	-10	-	-
Vyděšený k smrti	Aktivuje se, pokud je globální hodnota strachu rovna nebo vyšší než 80. Odebírá 17 bodů spokojenosti. Nahrazuje efekt Vystrašený.	-17	-	-
Nový objev	Některý z vesničanů učinil zajímavý objev. Celá vesnice oslavuje. Spokojenost se zvyšuje o 15 bodů na 2 dny.	+15	2d	-
Nový vynález	Některý z vesničanů vynalezl novou technologii. Celá vesnice oslavuje. Spokojenost se zvyšuje o 10 bodů na 1 den.	+10	1d	-
Zármutek	Některý z vesničanů zemřel. Celá vesnice truchlí. Spokojenost se snižuje o 20 bodů na 2 dny.	-20	2d	-
Odchod přítele	Nespokojený vesničan odešel. Vesnice truchlí nad jeho odchodem. Spokojenost se snižuje o 10 bodů na 1 den. Aplikuje se při odchodu vesničana z kmenu kvůli nespokojenosti.	-10	1d	-

Jméno	Popis	Body	Doba	Protiefekt
Vyhnání přítele	Bůh rozhodl a vesničané vyhověli jeho přání. Bohem určený vesničan byl vyhnán z kmene. Spokojenost se snižuje o 15 bodů na 1 den. Aplikuje se při vyhnání vesničana z kmene.	-15	1d	-

Pro spouštění (vznik, aplikaci) status efektů slouží tzv. spouštěče, jejichž popis z důvodu přehlednosti pro implementaci zobrazuje oddělená Tabulka 4. Spouštěč je prakticky metoda obsahující podmínku, volaná v místě, kde nastává daná událost či změna, která aktivuje daný status efekt, pokud je podmínka splněna. Tato metoda pak volá přímo metodu pro přidání či odebrání status efektu na konkrétní postavě nebo na všech postavách, dle charakteru status efektu. Odebírání slouží především pro odstranění stavově omezených status efektů.

Tabulka 4: Spouštěče pro status efekty ovlivňující spokojenost postav

Událost	Status efekt	Akce	Podmínka	Cíl
Časovač (každé 2 hodiny)	Šťastný	Přidat	Náhoda (5 %)	Tento vesničan
	Nespokojený	Přidat	Náhoda (5 %)	Tento vesničan
Hlad (snížení)	Hladový	Přidat	Hlad < 15	Tento vesničan
	Dobře najedený	Odebrat	Hlad < 90	Tento vesničan
Hlad (zvýšení)	Dobře najedený	Přidat	Hlad > 90	Tento vesničan
	Hladový	Odebrat	Hlad > 15	Tento vesničan
Zdraví (snížení)	Zraněný	Přidat	Zdraví < 70	Tento vesničan
Zdraví (zvýšení)	Zraněný	Odebrat	Zdraví > 70	Tento vesničan
Strach (snížení)	Cítí se v bezpečí	Přidat	Strach =< 10	Tento vesničan
	Bojí se	Přidat	Strach < 50	Tento vesničan
		Odebrat	Strach < 30	Tento vesničan
	Vystrašený	Přidat	Strach < 80	Tento vesničan
		Odebrat	Strach < 50	Tento vesničan
	Vyděšený k smrti	Odebrat	Strach < 80	Tento vesničan
Strach (zvýšení)	Cítí se v bezpečí	Odebrat	Strach > 10	Tento vesničan
	Bojí se	Přidat	Strach >= 30	Tento vesničan
		Odebrat	Strach >= 50	Tento vesničan
	Vystrašený	Přidat	Strach >= 50	Tento vesničan
		Odebrat	Strach >= 80	Tento vesničan

Událost	Status efekt	Akce	Podmínka	Cíl
	Vyděšený k smrti	Přidat	Strach >= 80	Tento vesničan
Nový objev	Nový objev	Přidat	-	Všichni vesničané
Nový vynález	Nový vynález	Přidat	-	Všichni vesničané
Smrt	Zármutek	Přidat	-	Všichni vesničané
Odchod	Odchod přítele	Přidat	-	Všichni vesničané
Vyhnání hráčem	Vyhnání přítele	Přidat	-	Všichni vesničané

Status efekty jsou scriptable objekty třídy *StatusEffect*, které jsou uložené ve *VillagerStats* konkrétního vesničana.

6.5.4 Inventáře

Každá postava má svůj inventář, což je v podstatě systém pro správu předmětů, které má postava u sebe. Více o předmětech je uvedeno v následujícím oddílu 6.7 „Předměty“. Inventář se skládá ze tří hlavních částí:

- Ruce: zde může postava nést libovolný předmět, a to v každé ruce nezávisle na sobě.
- Equip sloty: mohou obsahovat speciální druhy předmětů, kdy každý z těchto předmětů má určený equip slot, do kterého patří. V překladu equip slot znamená přihrádka na vybavení, je však použit tento běžnější anglický termín.
- Batoh: předmět, který se umisťuje do equip slotu pro batoh a sám může obsahovat libovolné předměty.

Grafická reprezentace inventáře postavy je ve hře dostupná po výběru konkrétní postavy a následném zvolení záložky Vybavení na informačním panelu vybrané postavy. Její podobu znázorňuje Obrázek 10.

Obrázek 10: GUI inventáře postavy bez batohu

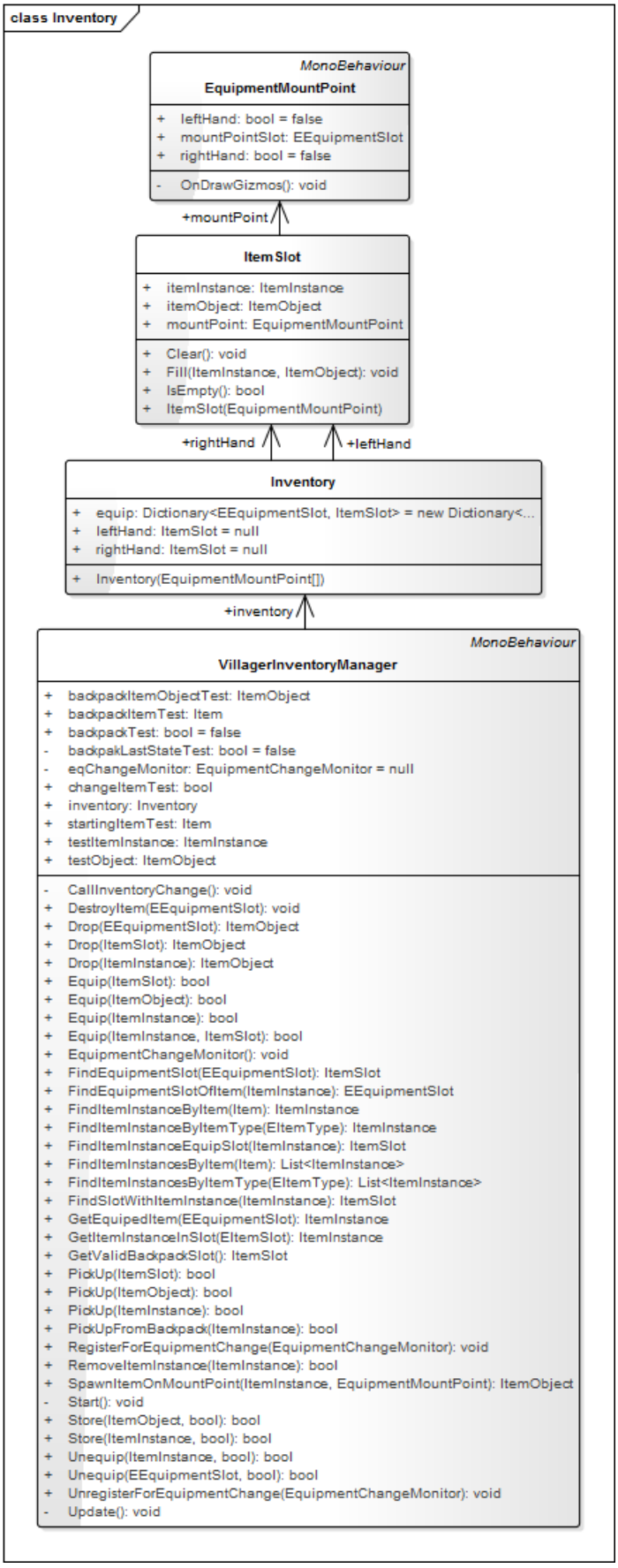
Ikony v inventáři reprezentují tzv. sloty, tedy „přihrádky“ pro předměty. Dále v textu je opět použit běžnější anglický termín. Horní levý a horní pravý slot reprezentuje levou a pravou ruku. Dále jsou tu již zmíněné equip sloty. Uprostřed nahoře je equip slot pro oblečení či zbroj, vlevo dole pro nádobu na vodu a vpravo dole pro batoh.

Jelikož je batoh volitelná součást inventáře, kterou postava může, ale nemusí mít, není na obrázku výše znázorněný. Pokud se postava vybaví batohem, objeví se v GUI sloty batohu, jak znázorňuje Obrázek 11.

Obrázek 11: GUI inventáře postavy s batohem

Ovládání inventáře je řešeno primárně přes kontextové menu vyvolané kliknutím pravým tlačítkem myši nad konkrétním předmětem. Za daných podmínek jsou k dispozici následující akce:

- Vybavit se (pouze v batohu a jen na vybavení) – postava si tento předmět oblékne do equip slotu, pro který je předmět určen. Pokud v tomto slotu už nějaký předmět je, postava ho svlékne.
- Odložit do batohu (pouze v equip slotech kromě slotu batohu) – postava svlékne daný předmět a uloží ho do batohu.
- Odložit na zem – postava položí daný předmět na zem (pokud je oblečený, svlékne ho).
- Sníst (pouze u jídla) – postava sní dané jídlo.
- Napít se (pouze u nádoby na vodu) – postava se napije z dané nádoby.



Obrázek 12: UML diagram tříd inventáře

UML diagram hlavních tříd pro inventář znázorňuje Obrázek 12. Hlavní třída pro správu inventáře postavy je *VillagerInventoryManager*, každá postava má svoji vlastní instanci obsahující následující veřejné metody.

bool PickUpFromBackpack(ItemInstance itemInstance): Vzít předmět z batohu do ruky.

bool PickUp(ItemSlot itemSlot): Vzít předmět ze slotu (typicky equip) do ruky.

bool PickUp(ItemObject itemObject): Vzít předmět z objektu (typicky položený na zemi) do ruky.

bool PickUp(ItemInstance itemInstance): Vzít předmět do ruky. Pozor, neovlivní instanci předmětu na původním místě, předmět následně odkazuje na stejnou instanci, hodí se proto pro nové předměty nebo v případě, že je vyřešeno odebrání instance předmětu z původního umístění.

bool Equip (ItemSlot itemSlot): Obléct si předmět ze slotu do equip slotu. Equip slot je určen automaticky podle typu předmětu. Lze např. obléct předmět z konkrétní ruky.

bool Equip (ItemObject itemObject): Obléct si předmět z objektu (typicky položený na zemi) do equip slotu. Slot je určen automaticky podle typu předmětu.

bool Equip (ItemInstance itemInstance): Obléct si předmět do equip slotu. Slot je určen automaticky podle typu předmětu. Pozor, neovlivní instanci předmětu na původním místě, předmět následně odkazuje na stejnou instanci, hodí se proto pro nové předměty nebo v případě, že je vyřešeno odebrání instance předmětu z původního umístění.

bool Equip (ItemInstance itemInstance, ItemSlot itemSlot): Obléct si předmět do konkrétně určeného slotu. Lze použít i na sebrání předmětu do konkrétní ruky. Pozor, neovlivní instanci předmětu na původním místě, předmět následně odkazuje na stejnou instanci, hodí se proto pro nové předměty nebo v případě, že je vyřešeno odebrání instance předmětu z původního umístění.

bool Unequip (ItemInstance itemInstance, bool noPackDrop = true): Svlékne určený předmět a umístí ho do batohu. Pokud postava batoh nemá nebo je plný a je to povoleno (noPackDrop == true), postava položí předmět na zem.

bool Unequip (EEquipmentSlot slot, bool noPackDrop = true): Svlékne předmět z určeného equip slotu a umístí ho do batohu. Pokud postava batoh nemá nebo je plný a je to povoleno (noPackDrop == true), postava položí předmět na zem.

ItemSlot GetValidBackpackSlot(): Zkontroluje, zda má postava batoh a pokud ano, vrátí slot s batohem. Kontroluje, zda existuje slot na batoh (postava na sobě musí mít také mount point pro batoh), tento slot není null, je v něm předmět a tento předmět má modul storage.

ItemObject Drop (EEquipmentSlot slot): Svlékne a položí předmět z určeného equip slotu na zem.

ItemObject Drop (ItemSlot itemHolder): Svlékne a položí předmět z určeného slotu na zem.

ItemObject Drop (ItemInstance itemInstance): Svlékne a položí konkrétní předmět na zem. Předmět se musí nacházet v některém z equip slotů.

bool Store(ItemObject itemObject, bool equip = true): Uloží předmět z objektu do batohu. Pokud postava batoh nemá nebo je plný a je to povoleno (equip == true), postava si předmět vezme do ruky.

bool Store (ItemInstance itemInstance, bool equip = true): Uloží předmět do batohu. Pokud postava batoh nemá nebo je plný a je to povoleno (equip == true), postava si předmět vezme do ruky. Pozor, neovlivní instanci předmětu na původním místě, předmět následně odkazuje na stejnou instanci, hodí se proto pro nové předměty nebo v případě, že je vyřešeno odebrání instance předmětu z původního umístění.

void DestroyItem (EEquipmentSlot slot): Zničí předmět v daném equip slotu.

EEquipmentSlot FindSlotOfItem (ItemInstance itemInstance): Najde equip slot, ve kterém je daný předmět.

ItemInstance GetEquipedItem (EEquipmentSlot slot): Vrátí předmět z konkrétního equip slotu.

ItemObject SpawnItemOnMountPoint (ItemInstance itemInstance, EquipmentMountPoint mountPoint): Spawne předmět ve světě na konkrétním mount pointu.

ItemSlot FindItemInstanceSlot(ItemInstance itemInstance): Najde v equip slotech daný předmět a vrátí jeho slot.

ItemSlot FindEquipmentSlot(EEquipmentSlot equipmentSlot): Vrátí slot na základě equip slotu.

void RegisterForEquipmentChange (EquipmentChangeMonitor callback): Registrace delegátu pro sledování změn (vhodné pro GUI).

void UnregisterForEquipmentChange (EquipmentChangeMonitor callback): Odregistrace delegátu pro sledování změn (vhodné pro GUI).

6.6 Úkoly a umělá inteligence postav

Hráč může vesničanům zadávat různé úkoly dvěma způsoby: přímo a nepřímo. Zadávání úkolů probíhá zpravidla přes kontextové menu vyvolané kliknutím pravým tlačítkem myši na cílový objekt (např. zdroj suroviny). V menu si hráč zvolí akci, kterou chce s daným objektem provést (ne všechny položky menu jsou zadáním úkolu, např. přidávání předmětů na výzkumném stanovišti tyto předměty pouze „nastaví“ pro výzkum). Pokud při tom má označeného vesničana, zadává mu takto úkol přímo a vesničan ho půjde splnit přednostně, přičemž přeruší právě vykonávanou činnost. Pokud není označen žádný vesničan, zadávání probíhá nepřímo – globálně, kdy se úkolu ujme vesničan, který v danou chvíli nevykonává žádnou činnost.

Výjimku tvoří příkaz pro přesunutí se na zadanou pozici, který lze zadat pouze přímo konkrétnímu vesničanovi (nelze ho tedy zadat globálně) pomocí kliknutí pravým tlačítkem myši na zem na určené místo ve chvíli, kdy je vesničan označen. Druhou výjimku tvoří činnosti jezení, pití a spánek, které vesničané vykonávají sami, pokud to nemají zakázané, nelze jim je tedy zadat ručně.

Nepřímo zadané úkoly se řadí do globální fronty úkolů, ze které si je postavy odebírají ve chvíli, kdy nevykonávají žádnou činnost, a to nezávisle na ostatních postavách, tj. ve smyslu „kdo dřív přijde, ten dřív bere“. Odebírání z této fronty probíhá na základě již zmíněných priorit prací, které hráč může u každého vesničana nastavit rozdílně, tím způsobem, že se odebírají postupně nejprve úkoly s vyšší prioritou a pokud již žádný s touto prioritou ve frontě není, přesune se výběr na úkoly s nižší prioritou a takto postupně až se vyčerpají všechny úkoly.

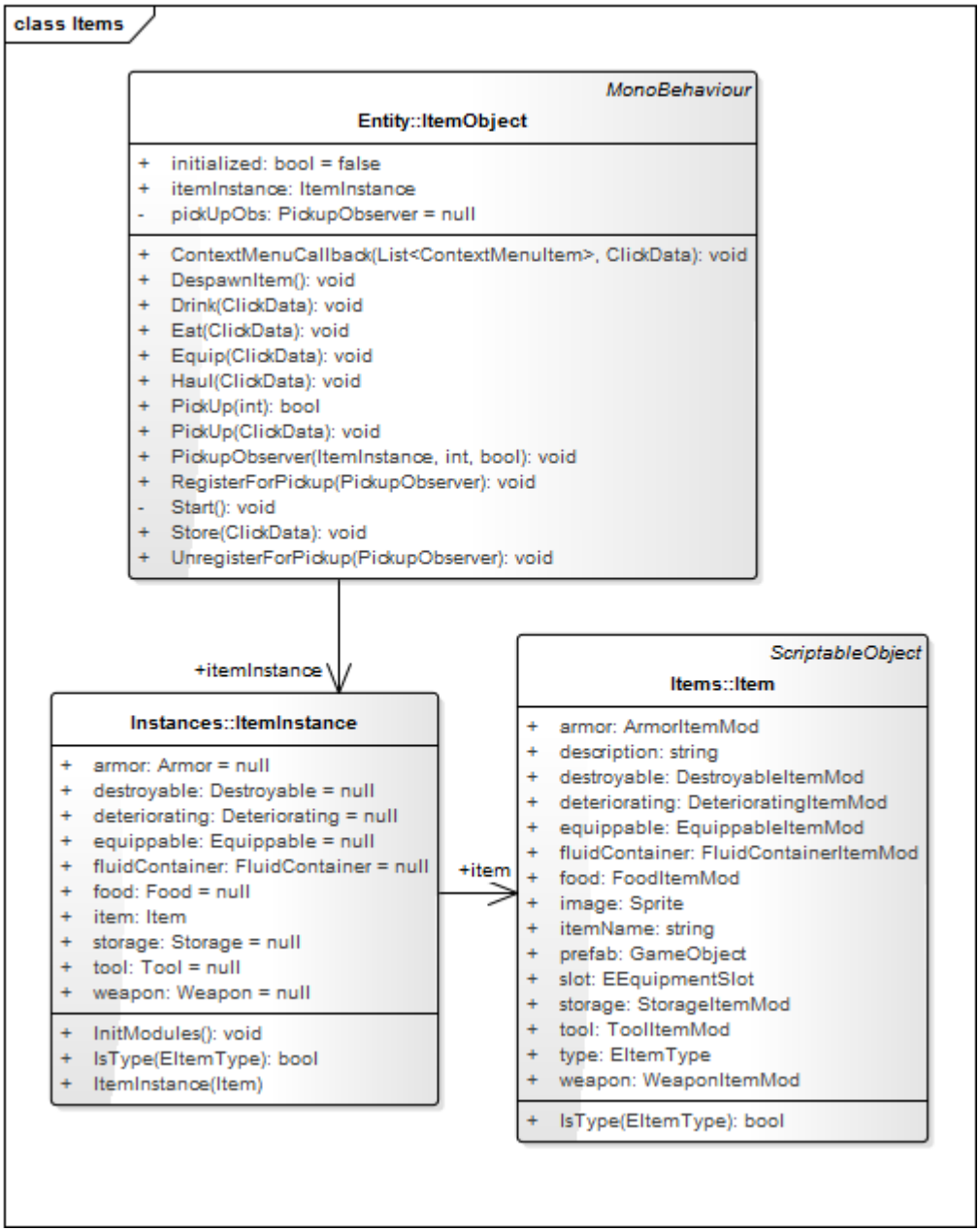
6.7 Předměty

Ve hře se vyskytuje několik předmětů, se kterými postavy mohou manipulovat. Předměty postavy mohou nosit u sebe v inventáři (viz pododdíl 6.5.4 výše), používat je (např. jíst jídlo) nebo je např. skladovat. Seznam předmětů dostupných ve hře popisuje Tabulka 5.

Tabulka 5: Seznam předmětů ve hře

Soubor	Jméno	Popis	Typ	Moduly
Stone	Kámen	Obyčejný kámen	Materiál	-
Apple	Jablko	Vypadá velmi chutně.	Jídlo	Food
Waterskin	Vak na vodu		Nádoba na tekutiny	FluidContainer, Equippable

Soubor	Jméno	Popis	Typ	Moduly
StoneAxe	Kamenná sekera	Celkem šikovná sekerka. A kdyby jen sekerka.	Nástroj	Tool
Branch	Větev	Větev ulomená ze stromu	Materiál	-
WoodLog	Poleno		Materiál	-
Backpack	Batoh	Lze v něm nosit věci.	Ostatní	Storage, Equippable
FistWedge	Pěstní klín		Nástroj	Tool
PointedStick	Špičatý klacek		Zbraň	Weapon
Club	Kyj		Zbraň	Weapon
StoneKnife	Kamenný nůž		Nástroj	Tool
Spear	Oštěp		Zbraň	Weapon
FistAxe	Pěstní sekera		Nástroj	Tool
SpearThrower	Vrhač oštěpů		Zbraň	Weapon



Obrázek 13: UML diagram tříd pro předměty

UML diagram hlavních tříd pro inventář znázorňuje Obrázek 13. Hlavními třídami pro správu předmětů jsou Item a ItemInstance. Třída Item představuje šablonu předmětu (např. obecné jablko), jedná se o scriptable object (což je funkce herního engine Unity zmíněná v pododdílu 1.4.1 „Herní engine Unity“), tudíž její instance lze tvořit v grafickém rozhraní editoru Unity. Třída ItemInstance pak reprezentuje konkrétní instanci předmětu ve hře (např. konkrétní jablko, které má vesničan v batohu). Pokud by bylo např. ve skladu pět kamenů, existovalo by pět instancí třídy ItemInstance navázaných na jednu instanci třídy Item – šablonu kamene. Pokud se předmět dostane na místo viditelné ve hře, např. je položen na zem nebo ho vesničan nese

v ruce či ho má oblečený, vytvoří se pro takový předmět instance třídy ItemObject, který reprezentuje objekt předmětu fyzicky umístěný do scény.

6.7.1 Moduly předmětů

Modul předmětu si lze v této hře představit jako druh předmětu či funkci, kterou tento předmět zastává. Jelikož jeden předmět může mít více různých funkcí, používá tato hra takový systém modulů, kdy jeden předmět může mít libovolný počet modulů (nemusí tedy mít žádný, pokud není ničím speciální). Seznam všech modulů, které mohou předměty mít, uvádí Tabulka 6. Rozdíl mezi atributy a atributy instance je ten, že atributy jsou u daného druhu předmětu napevno daná nastavení (např. pro všechna jablka stejná), zatímco atributy instance jsou pro každý jednotlivý předmět individuální, a navíc je možno je v průběhu času měnit.

Tabulka 6: Moduly předmětů ve hře

Jméno	Popis	Atributy	Atributy instance
Food	Předmět lze sníst	saturation, nutritions	
Storage	Do předmětu lze uložit jiné předměty	capacity, itemTypes[]	items[]
Tool	Předmět lze využít jako nástroj (např. pro získávání surovin)	level, type	
Armor	Předmět slouží jako zbroj, pokud si ho postava obleče nebo ho nese v ruce	protection	
Destroyable	Předmět je zničitelný	durabilityMax	durability
Weapon	Předmět lze použít jako zbraň	damageMin, damageMax, range	
Deteriorating	Předmět se může kazit	deteriorPerGH	deteriorating, freshness
FluidContainer	Předmět lze využít jako nádobu na vodu	maxCapacity	fluidType, fluidLevel
Equippable	Předmět lze obléci na postavu	equipSlot	

6.8 Zdroje surovin

Postavy mohou získávat surovinové předměty ze zdrojů surovin, které zobrazuje Tabulka 7. Ve hře se vyskytují tři druhy surovin, a to skála, strom a jabloň.

Tabulka 7: Zdroje surovin dostupné ve hře

Jméno	Akce	Získaný předmět
Skála	Vytěžit kámen	Kámen
	Sbírat oblázky	Kámen
Strom	Pokácet strom	Poleno
	Otrhat větve	Větev
Jabloň	Pokácet strom	Poleno
	Otrhat větve	Větev
	Sklidit jabloň	Jablko

Zdroj surovin představuje třída *ResourceEntity*, kde lze zároveň nastavovat parametry zdroje surovin.

6.9 Stavby

V herním světě se krom zdrojů surovin nachází také stavby, které se liší především tím, že jejich primárním účelem není poskytování surovin, ale slouží k různým účelům, které spolu se seznamem staveb ve hře popisuje Tabulka 8.

Tabulka 8: Stavby dostupné ve hře

Jméno	Popis	Účel
Sklad	Zde tvoji vesničané uskladňují své věci	Skladování
Výzkumné stanoviště	Zde tvoji vesničané zkoumají nové vynálezy	Výzkum
Výrobní stanoviště	Zde tvoji vesničané vyrábějí nové předměty	Výroba
Ohniště	V kruhu kamenů plápolá příjemný oheň. Jeho praskání uklidňuje mysli tvých vesničanů.	Okrasa
Stojan s vakem na vodu		Okrasa

6.10 Výzkumy

Důležitou součástí hry jsou výzkumy. Způsob, jakým výzkumy probíhají, je v této hře poměrně netradiční. Běžný přístup k výzkumům v počítačových hrách je, že si hráč vybere výzkum a poté na tento výzkum shání zdroje, nebo naopak shromažďuje zdroje a když jich má dostatek, vybere si výzkum, kam tyto zdroje vloží. V této hře nejsou výzkumy předem známy a hráč má za úkol pomocí zkoušení různých kombinací předmětů výzkum objevit a provést. Krom správné kombinace je též třeba mít pro některé výzkumy předchozí znalosti, tzn. hotové výzkumy, a také trochu štěstí – výzkum se nemusí vždy povést, nýbrž má jen určitou šanci na úspěch. Aby to hráč neměl příliš těžké, má možnost se orientovat podle zpráv, které se objeví po skočení každého výzkumu:

- *<jméno> objevil následující věc: <název výzkumu> <popis výzkumu>* znamená úspěšný výzkum.
- *Vesničan nic nevyzkoumal. <jméno>:"No tak tohle se nepovedlo… Ale stejně si myslím, že by z toho něco mohlo být…"* znamená, že se zkrátka výzkum nepovedl, jelikož má pouze určitou šanci na úspěch.
- *Vesničan nic nevyzkoumal. <jméno>:"Zatím tomu tak úplně nerozumím, ale až budu chytřejší, určitě to půjde!"* znamená, že k danému výzkumu nejsou splněné některé předešlé výzkumy.
- *Vesničan nic nevyzkoumal. <jméno>:"Tohle je na nic. Jak to vůbec mohlo někoho napadnout?"* znamená, že k dané kombinaci předmětů neexistuje žádný výzkum, který není hotový.

Ve hře se nachází několik výzkumů, jejichž přehled uvádí Tabulka 9 včetně surovin, které jsou na výzkum potřeba, potřebných předchozích znalostí a šance, zda se výzkum povede. Každý výzkum trvá 5 minut reálného času.

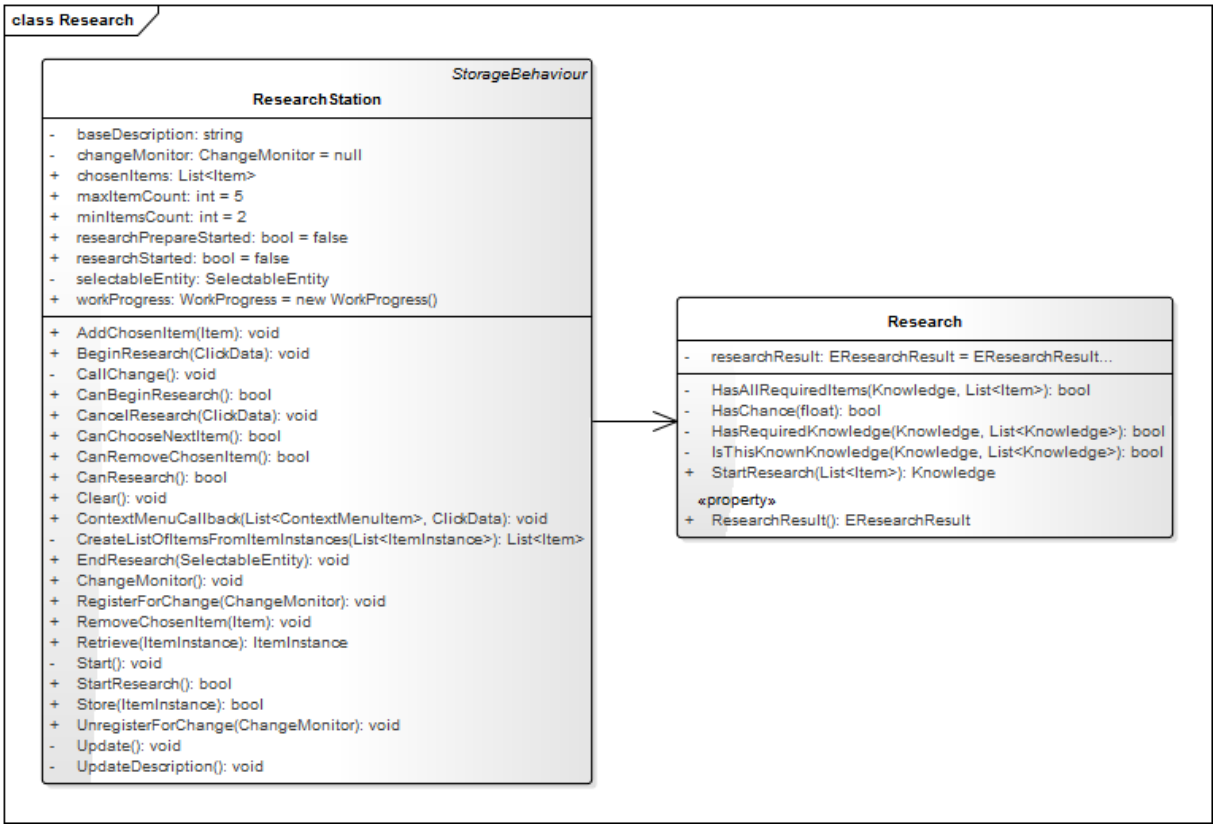
Tabulka 9: Přehled výzkumů dostupných ve hře

Soubor	Jméno	Popis	Potřebné předchozí znalosti	Suroviny	Šance
FistWedge	Pěstní klín	Je čas do toho praštit!	-	kámen, kámen	90 %

Soubor	Jméno	Popis	Potřebné předchozí znalosti	Suroviny	Šance
PointedStick	Špičatý klacek	Naostřený klacek. Není sice o mnoho lepší než pěst ale lepší než klackem do oka no ne?	-	kámen, větev	90 %
Club	Kyj	Au! Au! … Už nefunguje...	-	větev, větev	90 %
StoneKnife	Kamenný nůž	… a když s tím správně praštíš, tak se to rozpadne. A když to vyjde, bude to mnohem víc au au než to předtím.	Pěstní klín, Špičatý klacek	kámen, pěstní klín	50 %
Spear	Oštěp	… hele a když to jde s kamením… nešlo by to i s tímhle?	Špičatý klacek	špičatý klacek, kámen	70 %
FistAxe	Pěstní sekera	Hele, a když hořej ty malý větve, myslíš že by hořel i strom? Já jen jestli by to nešlo nějak… porazit. Já že bysme toho nemuseli tahat tolik.	Pěstní klín, Kamenný nůž	kámen, pěstní klín	35 %
SpearThrower	Vrhač oštěpů	A víš že takhle to hodíš dál?... Jo a nebo někomu vypíchneš oko.	Oštěp	oštěp, větev, kamenný nůž	20 %
StoneAxe	Kamenná sekyra		Pěstní klín, Pěstní sekera	pěstní sekera, větev, kamenný nůž	15 %

UML diagram tříd zajišťujících zpracování výzkumů znázorňuje Obrázek 14. Třída *ResearchStation*, potomek třídy *StorageBehaviour*, využívá metod skladiště pro uchovávání a

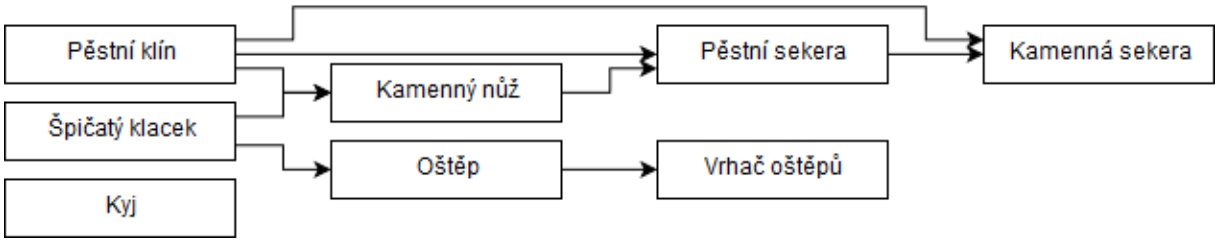
správu předmětů připravených k výzkumu, přičemž přidává vlastní metody pro zpracování výzkumu. Důležitými metodami jsou *AddChosenItem(Item)* a *RemoveChosenItem(Item)*, které slouží pro přidání, resp. odebrání předmětu pro budoucí výzkum. Dále *BeginResearch(ClickData)*, která slouží pro spuštění výzkumu a předání úkolu AI. Tyto metody jsou určeny pro volání z kontextového menu výzkumného stanoviště. AI pak volá metodu *StartResearch()* v okamžiku, kdy jsou ve výzkumném stanovišti připravené všechny předměty zvolené pro výzkum. Následně, po uplynutí času potřebného na výzkum, zavolá AI metodu *EndResearch(SelectableEntity)*, která provede vyhodnocení výzkumu a výzkum zakončí. Pro vyhodnocení výzkumu používá samostatnou třídu *Research*, resp. její metodu *StartResearch(List<Item>)*, kde probíhá vyhodnocení výzkumu dle algoritmu popsaného v pododdílu 6.10.2 „Popis algoritmu zpracování výzkumu“.



Obrázek 14: UML diagram tříd pro výzkumy

6.10.1 Evoluční schéma výzkumů

Evoluční schéma výzkumů neboli strom výzkumů znázorňuje Obrázek 15. Evoluce, jinými slovy postup či vývoj probíhá ve směru šipek, z diagramu lze tedy jednoduše vyčíst, které výzkumy musí přecházet danému výzkumu – např. pro úspěšný výzkum pěstní sekery je třeba mít hotové výzkumy pro pěstní klín a kamenný nůž. Evoluční schéma tak poskytuje ucelený přehled o návaznostech jednotlivých výzkumů ve hře.



Obrázek 15: Strom výzkumů

6.10.2 Popis algoritmu zpracování výzkumu

Jelikož výzkum probíhá pouze na základě vložených předmětů (2–5 kusů), přičemž je možno vložit i více stejných předmětů, byl vytvořen následující algoritmus, který výzkumy vyhodnocuje. Nutno podotknout, že existuje možnost, že z vložených předmětů půjde provést více výzkumů.

Algoritmus vezme seznam všech dostupných výzkumů ve hře a pro každý z nich vykoná následující kontroly v tomto pořadí:

1. Jsou ve výzkumném stanovišti všechny předměty potřebné pro tento výzkum?
2. Není tento výzkum již hotový?
3. Jsou hotové všechny předchozí výzkumy potřebné pro tento výzkum?
4. Povedl se výzkum (dle šance na úspěch)?

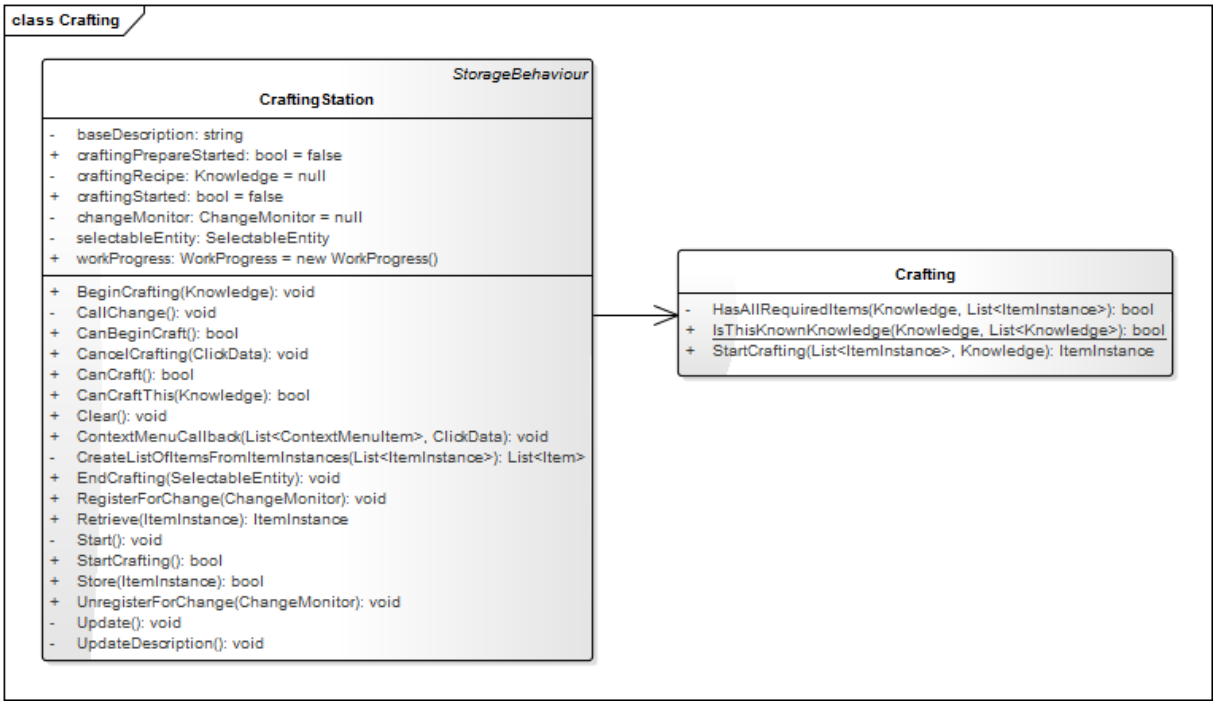
Pokud není splněn některý z bodů 1–3, pokračuje algoritmus dalším výzkumem ze seznamu, aniž by pokračoval na další bod kontroly stávajícího výzkumu. Pokud algoritmus dojde až do bodu 4, nepokračuje již na další výzkum bez ohledu na to, jak dopadne kontrola v tomto bodě.

Algoritmus si pamatuje nejvyšší bod, kterého dosáhl, na základě čehož se pak uživateli zobrazí příslušná hláška. Tudíž pokud lze z daných předmětů provést více výzkumů a např. jeden z nich už je hotový a na druhý nejsou potřebné znalosti, zobrazí se hráči informace o tom, že na výzkum nejsou potřebné znalosti. V případě úspěchu se krom zprávy též přidá hotový výzkum mezi získané znalosti kmene.

6.11 Výroba

Výroba tvoří další důležitou součást hry, bez které by výzkumy nesplnily svůj účel a také by nebylo možné realizovat pokročilé výzkumy, které vyžadují předměty z předchozích výzkumů. Pro výrobu každého předmětu jsou potřebné stejné suroviny, jako byly potřeba na jeho výzkum, bez ohledu na to, jaké suroviny byly při výzkumu skutečně spotřebovány. Tyto suroviny jsou následně spotřebovány a je vyroben nový předmět. Výroba jakéhokoliv předmětu trvá 1 minutu a 40 sekund reálného času.

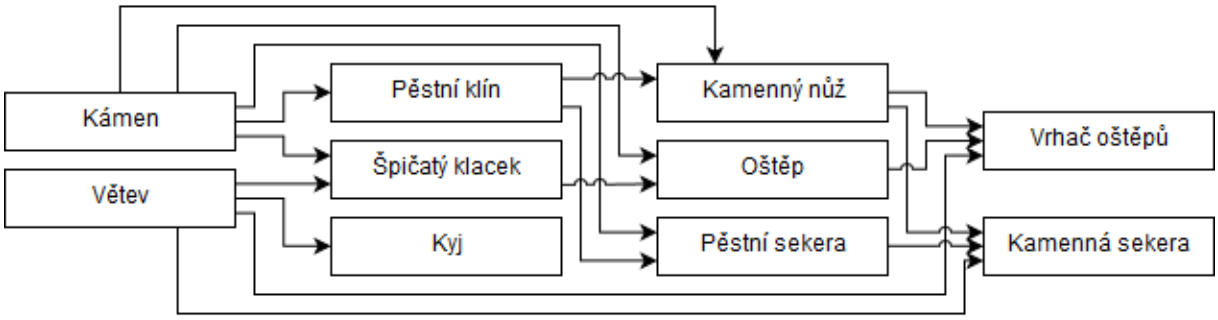
UML schéma tříd zajišťujících zpracování výroby znázorňuje Obrázek 16. Třídy jsou velmi podobné třídám pro výzkum, stejně tak práce s nimi je obdobná. Třída *CraftingStation* je potomkem třídy *StorageBehaviour*, čehož využívá pro správu předmětů, se kterými pracuje, a to jak surovin, tak výsledného výrobku. Mezi důležité metody patří *BeginCrafting(Knowledge)*, která je určena pro volání z kontextového menu výrobního stanoviště a slouží pro spuštění výroby konkrétního předmětu a předání úkolu AI. Metoda *StartCrafting()* je pak určena pro volání prostřednictvím AI v okamžiku, kdy jsou ve výrobním stanovišti připravené všechny potřebné předměty. Následně, po uplynutí času potřebného na výrobu, zavolá AI metodu *EndCrafting(SelectableEntity)*, která provede zakončení výroby tím, že nahradí v prostoru výzkumného stanoviště potřebné suroviny za výsledný výrobek a ten ze stanoviště vyhodí ven, načež zadá úkol pro odnesení tohoto výrobku do skladu. Pro vyhodnocení přítomnosti potřebných surovin a jejich nahrazení za výsledný výrobek používá samostatnou třídu *Crafting*, resp. její metodu *StartCrafting(List<Item>)*.



Obrázek 16: UML diagram tříd pro výrobu

6.11.1 Evoluční schéma výroby

Evoluční schéma výroby neboli technologický strom výroby znázorňuje Obrázek 17. Toto evoluční schéma poskytuje informaci o návaznostech výroby předmětů, tj. který výrobek slouží jako surovina pro jiný výrobek – např. špičatý klacek a kámen slouží k výrobě oštěpu. Lze si pochopitelně všimnout určitých podobností se stromem výzkumů, avšak mezi oběma stromy jsou zřejmé rozdíly.



Obrázek 17: Technologický strom výroby

6.12 Scény

Hra obsahuje tři scény, z nichž každá má svůj účel, jak uvádí Tabulka 10. Při spuštění hry se načte scéna „Menu“, následně si uživatel vybírá mezi scénami „Main“ a „Game“.

Tabulka 10: Dostupné scény ve hře

Soubor	Jméno	Popis scény
Menu	Hlavní menu	Hlavní menu, které se zobrazí po spuštění hry.
Main	Ručně připravený svět	Hlavní ručně vytvořená scéna sloužící jako ukázka hry.
Game	Procedurálně generovaný svět	Generovaná scéna, sloužící jako ukázka procedurálního generátoru. Není zaručena plná hratelnost tohoto světa.

7 VALIDACE SE ZADÁVACÍ DOKUMENTACÍ

Zadávací dokumentací k praktické části této diplomové práce je game design dokument, který je přílohou k diplomové práci Herní design (Černík, 2018). Tato kapitola shrnuje stav implementace a rozdíly oproti zadávací dokumentaci.

7.1 Implementace klíčových mechanik

Co se týče klíčových mechanik, je implementováno téměř celé uživatelské rozhraní, aby bylo možno hru plnohodnotně ovládat. Co se týče úkolů, je implementováno jádro umělé inteligence, které se stará o přiřazování úkolů jednotlivým postavám, rušit je možno pouze některé úkoly. Je možno globálně zadávat úkoly, které jsou poté přiděleny jednotlivým vesničanům, případně lze přiřadit vybrané úkoly přímo konkrétnímu vesničanovi a lze mu přikázat, aby šel na zadané místo na mapě. Je implementováno i získávání surovin, konkrétně kopání kamene, získávání dřeva ze stromů a trhání jablek. Přehled o stavu implementace jednotlivých klíčových mechanik poskytuje Tabulka 11.

Tabulka 11: Stav implementace klíčových mechanik

ID mechaniky	Název	Implementováno	Komentář
CTR-PLR-000	Uživatelské rozhraní	Téměř plně	Viz 7.1.1 níže
ATR-VILL-001	Potřeby	Téměř plně	Viz 7.1.2 níže
CTR-PLR-001	Ovládání vesničanů hráčem	Ano	Viz 7.1.4 níže
CTR-PLR-002	Přiřazování úkolů	Ano	
CTR-PLR-003	Prioritizace úkolů	Ano	
CTR-PLR-004	Rušení úkolů a činností	Ne	
CTR-RES-001	Získávání surovin	Ano	
REC-DIR-001	Výzkum	Ano	

7.1.1 Uživatelské rozhraní (CTR-PLR-000)

Uživatelské rozhraní je v praktické části téměř celé implementováno, přehled o stavu implementace znázorňuje Tabulka 12.

Tabulka 12: Stav implementace uživatelského rozhraní

Prvek	Stav implementace
Ovládání klávesnicí a myší	Plně implementováno.

Prvek	Stav implementace
Hlavní menu	Implementováno mimo použití funkcí, které implementace vůbec neobsahuje (načtení uložené hry, nastavení).
Hlavní nabídka (menu ve hře)	Implementováno mimo použití funkcí, které implementace vůbec neobsahuje (uložení hry, načtení uložené hry, nastavení).
Ovládání rychlosti hry	Plně implementováno.
Informační zprávy (FEA-MSG-001)	Plně implementováno.
Kontextová nabídka: Vesničan	Částečně implementováno, podrobnosti viz další položky. Omezená kontextová nabídka je implementována i pro některé další objekty ve hře.
Kontextová nabídka: Vesničan (Postava)	Plně implementováno, u strachu se nadále zobrazuje číselná hodnota od 0 do 100, jelikož slovní hodnoty pro strach nejsou v dokumentaci definovány.
Kontextová nabídka: Vesničan (Popis)	Plně implementováno.
Kontextová nabídka: Vesničan (Vybavení)	Plně implementováno.
Kontextová nabídka: Vesničan (Práce)	Plně implementováno s výjimkou ikon na tlačítkách.

7.1.2 Potřeby (ATR-VILL-001)

Byly implementovány všechny potřeby, co se týče jejich zobrazování a změn, dále pak vybrané mechaniky potřeb. Stav implementace jednotlivých potřeb zobrazuje Tabulka 13 níže.

Tabulka 13: Stav implementace potřeb postav

Potřeba	Stav implementace
Zdraví	Implementováno jako ukazatel, změna však nikdy nenastává (postavy jsou stále zdravé).
Energie	Implementováno včetně úbytku i doplňování při spánku podle vypočtených hodnot ze vzorců.
Hlad	Plně implementováno.

Potřeba	Stav implementace
Žízeň	Plně implementováno s výjimkou pití z vaku / nádoby.
Spokojenost	Implementována s tím, že je ovlivňována pouze status efekty ovlivňujícími spokojenost a neovlivňuje nic dalšího (odejití od kmene není implementováno). Status efekty jsou implementovány všechny s tím, že některé nelze aktivovat (viz pododdíl 7.1.3).
Nutrice	Implementováno doplňování pomocí jídla.
Strach	Strach není sice v tabulce potřeb zahrnut, v dokumentaci se s ním ale pracuje a je plně implementovaný s výjimkou toho, že strach ze tmy neovlivňují předměty jako ohniště, jen střídání dne a noci.

7.1.3 Status efekty

Status efekty byly všechny implementovány (jak status efekty charakterových rysů, tak status efekty ovlivňující spokojenost) s tím, že následující status efekty nemohou postavy nikdy získat:

- zraněný, jelikož ubývání zdraví není implementováno,
- nový objev, jelikož objevy nejsou implementovány,
- zármutek, jelikož ubývání zdraví není implementováno a postavy tedy nemohou umřít,
- odchod přítele a vyhnání přítele, jelikož odchody ani vyhnání vesničanů nejsou implementovány.

7.1.4 Ovládání vesničanů hráčem (CTR-PLR-001)

Aktivní ovládání vesničana (možnost mu přikázat, kam má jít) je implementováno na kliknutí pravým tlačítkem na zem. Primárně se zadávají globální úkoly, které jádro AI poté rozděluje. Je také možno vesničanům vybrané úkoly zadávat přímo. Implementovány jsou úkoly pro těžbu, výzkum, výrobu, nošení, sbírání předmětů a přesun na danou pozici (ten jako jediný nelze zadat globálně).

Samotní vesničané se umí (pouze z vlastní vůle) napít, najíst a vyspat.

7.2 Implementace vedlejších mechanik

Z vedlejších mechanik byly implementovány především charakterové rysy postav a informační zprávy. Kažení vody a jídla, náhodné objevy ani odchod a příchod vesničana do kmene implementovány nebyly.

Tabulka 14: Stav implementace vedlejších mechanik

ID mechaniky	Název	Implementováno	Komentář
ATR-VILL-002	Charakterové rysy postav	Ano	Postavy mají dva určité pevně dané charakterové rysy, se kterými žijí po celou hru
FEA-ITE-001	Kažení vody a jídla	Ne	
FEA-VILL-001	Odchod a příchod vesničana do kmene	Ne	
FEA-MSG-001	Informační zprávy	Ano	
REC-IND-001	Náhodné objevy	Ne	

7.3 Implementace assetů

Ve hře bylo použito několik assetů, převážně modelů různých objektů v herním světě. Stav jejich implementace zobrazuje Tabulka 15.

Tabulka 15: Stav implementace assetů

Asset	Stav implementace
Déšť a vítr	Implementováno, ve hře se náhodně mění počasí.
Behavior Designer	Implementováno.
Kamenná sekera	Implementováno.
Vlk	Není implementováno.
Držák na vak s vodou	Implementováno jako okrasný předmět.
Sbírka kamenů 1	Implementováno jako zdroj kamene.
Jablko	Implementováno jako hlavní zdroj potravy.
Stromy	Implementováno jako zdroje surovin.
Oheň – zvuk	Není implementováno.

ZÁVĚR

Cílem teoretické části diplomové práce bylo vysvětlení technik vývoje software s důrazem na vývoj počítačových her. Popis technik měl být zaměřen zejména na techniky tvorby evolučních schémat a procedurálního modelování. Cílem praktické části práce byla implementace software podle design dokumentu s využitím evolučních schémat a procedurálně generovaných modelů. Konkrétně se jedná o proof-of-concept počítačové hry typu multi RPG survival simulace.

V první kapitole je čtenář seznámen s problematikou vývoje počítačových her včetně základních pojmů využívaných v této oblasti. Dále jsou zde popsány obory lidské činnosti využívané při vývoji počítačových her. Na závěr kapitoly je uveden stručný přehled vývojových prostředí a nástrojů využívaných při vývoji her, přičemž je popsán herní engine Unity, který je využit pro implementaci počítačové hry v rámci praktické části této práce.

Ve druhé kapitole jsou popsány metodiky vývoje software, které se používají obecně, tedy i při vývoji her. Následuje popis technik specifických pro vývoj počítačových her. Třetí kapitola krátce popisuje problematiku vydavatelství a distribuce počítačových her. Čtvrtá kapitola je věnována podrobnějšímu popisu procedurálního modelování, pátá kapitola pak popisuje evoluční schémata. Šestá kapitola je věnována podrobnějšímu popisu software implementovaného v rámci praktické části práce, který je následně v sedmé kapitole validován oproti zadávací dokumentaci – game design dokumentu.

Diplomová práce pro mě znamenala, krom zjišťování značného množství nových informací na internetu a v knihách, především výzvu v podobě implementace složitějšího systému počítačové hry spolu s využitím pro mě zcela nových technik procedurálního generování a evolučních schémat. Důležitým poznatkem je pro mě skutečnost, že vývoj na první pohled jednoduché počítačové hry je poměrně složitou, a především časově náročnou záležitostí. Přestože zadání bylo splněno, neobsahuje hra z tohoto důvodu některé původně plánované funkcionality jako lov nebo přání vesničanů. Další zkušeností je, že při vývoji software je lepší vyvinout funkční produkt s minimálním množstvím funkcí a následně přidávat novou funkcionalitu, než vyvinout funkční části a poté se je snažit spojit do výsledného, do té doby nepoužitelného, produktu.

Praktickým přínosem této práce je možnost čtenáře seznámit se s algoritmy ve hře použitými, např. s procedurálním generátorem či jinými funkčními celky, a získané znalosti následně využít při implementaci vlastního software.

POUŽITÁ LITERATURA

ČERNÍK, Jiří, 2018. *Herní design.* Pardubice. Diplomová práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky, Katedra softwarových technologií.

GAJDŮŠEK, Michal, 2015. *Jak napsat scénář: OSTATNÍ* [online]. 16. 4. 2015 [cit. 2018-08-18]. Dostupné z: http://www.scenar.filmovani.cz/jaknapsat/ostatni-scenare-1.html#E_Videohra_hry_interaktivn%C3%AD_programy_

GUNDERLOY, Mike, 2007. *Z kodéra vývojářem: nástroje a techniky pro opravdové programátory.* Brno: Computer Press. 287s. ISBN 978-80-251-1517-6.

JIRKOVSKÝ, Jan, 2011. *Game industry: vývoj počítačových her a kapitoly z herního průmyslu*. Praha: D.A.M.O. 135s. ISBN 978-80-904387-1-2.

JIRKOVSKÝ, Jan, 2012. *Game industry 2.* Praha: D.A.M.O. 240s. ISBN 978-80-904387-3-6.

LAGUE, Sebastian, 2016a. *Procedural Landmass Generation (E01: Introduction)* [online video]. 31. 1. 2016 [cit. 2017-07-29]. Dostupné z: https://youtu.be/wbpMiKiSKm8.

LAGUE, Sebastian, 2016b. *Procedural Landmass Generation (E06: LOD)* [online video]. 28. 2. 2016 [cit. 2017-08-01]. Dostupné z: https://youtu.be/417kJGPKwDg.