**Univerzita Hradec Králové**

**Fakulta informatiky a managementu**

**Katedra informatiky a kvantitativních metod**

# Objektový přístup v návrhu RPG hry

Bakalářská práce

Autor: Lukáš Valnoha

Studijní obor: ai3

Vedoucí práce: doc. Ing. Hana Tomášková, Ph.D.

Hradec Králové Listopad 2014

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

podpis

V Hradci Králové dne Lukáš Valnoha

**Anotace**

Lukáš Valnoha, Objektový přístup v návrhu RPG hry, Fakulta Informatiky a Managementu Univerzity Hradec Králové, 2014, bakalářská práce.

Tato práce se zabývá problematikou tvorby počítačových RPG (Role Playing Game) her. Hlavní část práce je věnována vymyšlení ekonomicky zaměřené hry s RPG prvky a následnému vypracování jejího objektového modelu. Další část práce se zabývá návrhem implementace několika vybraných částí hry.

Klíčová slova: RPG hra, objektové modelování, analytický model, návrhový model, java.

**Annotation**

Lukáš Valnoha, Object-oriented approach in design of the RPG game, Faculty of Informatics and management, University of Hradec Králové, 2014, bachelor thesis

This thesis deals with the issues of creation of PC Role Playing Game. The main part is devoted to inventing economic-oriented game with RPG features along with subsequent creation of its object-oriented model. Another part of the thesis deals with the implementation of several selected parts of the game.

Key words: RPG, object-oriented modeling, object-oriented analysis, object-oriented design, java language

# Obsah

# 1 Úvod

Počítačové hry jsou v dnešní době neodmyslitelnou součástí zábavy většiny mládeže. Zároveň přiživují počítačový průmysl, neboť s novějšími hrami vzrůstají hardwarové nároky na počítač. Žánrů počítačových her je nespočet, proto bude oblast zájmu zaměřena pouze na RPG hry.

RPG je zkratka pro Role-Playing Game, v překladu hra rolí, nebo lépe, hra na hrdiny. RPG hry existují v „ne-elektronickém“ provedení, například stolní hry, nebo hry typu Dungeons & Dragons. Tyto hry mají většinou bohatší obsah, protože nejsou ničím omezeny, pouze lidskou fantazií. Také existuje tzv. Larp (live action role playing), což je hra na hrdiny v reálném světě – fiktivní svět i postavy jsou reprezentovány skutečným světem a živými postavami.

Cílem této práce je navrhnout první alpha verzi ekonomické počítačové indie hry s RPG prvky pro jednoho hráče. Jelikož se jedná pouze o jádro hry, bude návrh zaměřen tak, aby případné rozšiřování hry bylo pokud možno co nejpohodlnější. Herní mechanismy budou co nejvíce zjednodušené, aby se snadno vytvořilo stabilní jádro hry. Herní obsah a doba hraní nebudou s největší pravděpodobností nijak uspokojivé, protože se jedná o první alpha verzi hry. Ta slouží hlavně k vyjádření hlavní myšlenky hry, jde pouze o základ, na kterém lze dál stavět. Dalším cílem je vytvoření objektového modelu hry, konkrétně diagramu tříd, a to analytického i návrhového. Po vytvoření diagramů tříd bude navržena implementace vybraných herních mechanismů, například generování herní mapy.

## 1.1 Charakteristika RPG her

### 1.1.1 Vývoj postavy

Základem počítačové RPG hry je možnost vývoje herní postavy, hlavním cílem každého autora RPG hry by tedy mělo být vymyšlení propracovaného systému vylepšování postavy, získávání zkušeností, případně obohacení hry kvalitním příběhem. Právě v herním příběhu je největší rozdíl oproti stolním RPG hrám:

Počítačové RPG má příběh předem pevně daný, u stolních RPG je dán začátek příběhu a zbytek si vytváří sami hráči až v průběhu hraní.

Pod pojmem „vývoj postavy“ si lze představit několik věcí: Jednou z nejdůležitějších je zvyšování úrovně postavy. Ta se zvyšuje při splnění určitých podmínek, například při dosažení požadovaného počtu zkušeností. Dále může mít herní postava své postavové atributy, například síla, vitalita, atd. Atributy pozitivně ovlivňují postavu, kupříkladu síla může zvyšovat míru poškození zbraněmi na blízko, výdrž, nosnost, či zdraví. Zvyšují se většinou při postupu na novou úroveň, buď automaticky, nebo hráč dostane určité množství bodů a rozdělí si je mezi atributy sám. Hráč může mít také možnost zvyšovat postavové dovednosti, například dovednost meče, řečnictví nebo obchodování. Dovednosti mohou být závislé na atributech: Jsou efektivnější, když je jejich řídící atribut větší (například dovednost meče má jako řídící atribut sílu).

Možností získávání zkušeností je několik. Nejstarší a nejzákladnější princip „expení“ (český slang, odvozený od anglické zkratky XP – experience, zkušenost) je ničení nepřátel - za zabití je získána zkušenost. Tento princip je k vidění například ve hře Diablo od společnosti Blizzard Entertainment. Další způsob získávání zkušeností je plnění úkolů, tzv. „questů“ (odvozeno od anglického slova quest, úkol). Questy pomáhají hráči vžít se do RPG světa, nebo mohou prohlubovat příběh hry. Na tomto principu je postavena nejznámější MMORPG (Massively-Multiplayer Online RPG) hra World of Warcraft, také od Blizzard Entertainment. Zajímavý princip zvyšování úrovně postavy je k vidění u hry The Elder Scrolls 3: Morrowind od společnosti Bethesda Softworks. Tam je potřeba k postupu na další úroveň zvýšit dovednosti postavy dohromady o deset bodů. Dovednosti se zvyšují jejich praktikováním. Existují samozřejmě i další unikátní principy získávání zkušeností, které ale většinou bývají omezené pouze pro danou hru.

### 1.1.2 Herní svět

Prostředí, kde se herní postava pohybuje, se nazývá herní svět. Může být „ručně“ vymodelovaný, či náhodně vygenerovaný. Obě tyto metody mají své výhody a nevýhody:

V herním světě, který je vymodelovaný autorem, je vše na svém místě. Každý strom, kámen, rybník, nebo jeskyně, vše je tam, kam to autor umístil. Díky tomu může autor pracovat s konkrétními objekty a hra je pak barvitější a pestřejší. Manuálním modelováním lze lépe dosáhnout „vizuální dokonalosti“ než s algoritmem pro náhodné generování. Hlavními nevýhodami modelovaného světa je velká časová náročnost na tvorbu a také to, že takový svět je i při opakovaném hraní stále stejný. Tento nedostatek musí být vykompenzován bohatým herním obsahem či kvalitním příběhem, popř. obojím. Mezi nevýhody se dá zařadit i fakt, že velikost modelovaného světa je konečná. Jako příklad lze uvést hru The Elder Scrolls 5: Skyrim od Bethesda Softworks.

Náhodně generované světy se kvalitou nemůžou rovnat modelovaným, mají ale oproti nim několik předností: Velikost generovaných světů nemusí být nutně omezena – teoreticky může takový svět být nekonečný. Časová náročnost na tvorbu je oproti modelovaným světům výrazně menší. Také se může svět generovat dynamicky – na začátku je vygenerována pouze hlavní oblast kolem hráče a v závislosti na jeho poloze se dále generují další a další oblasti. Při nedokonalém algoritmu generování se ale mohou vyskytnout různé povrchové anomálie, levitující objekty a podobně. Velká výhoda generovaných map je ta, že jsou pokaždé jiné. Dobrý příklad generovaného světa je hra Minecraft od společnosti Mojang.

Herní světy se dále mohou dělit na otevřené a uzavřené (lineární). V otevřeném světě má hráč naprostou volnost, může jít, kamkoli se mu zachce a kdy se mu zachce. *Na druhou stranu, hry postavené na volnosti pohybu zvládají zaujmout hráče zhruba napůl a vždycky jsou jakýmsi způsobem nedokončené, respektive poloprázdné* (Kremser, 2013). V uzavřených světech je volnost pohybu hráče omezena, více nebo méně. Většinou takový svět bývá rozdělen do tzv. lokací, kterými hráč postupně prochází, od začátku až na konec.

### 1.1.3 Umělá inteligence

*Herní umělá inteligence (dále AI – Artificial Intelligence) je v současné době jedním z nejbouřlivěji se rozvíjejících odvětví interaktivního zábavního průmyslu. Až do druhé poloviny 90. let byla umělá inteligence v počítačových hrách považována za okrajovou záležitost; herní vývojáři měli v tu dobu jinou prioritu – dokonalejší a realističtější grafiku. To se projevilo nejen ve velmi malém podílu na výkonu procesoru, který měla herní AI k dispozici, ale také ve spěchu a nekoncepčnosti, se kterými byla AI do her na poslední chvíli implementována* (Jakob, 2012). Umělá inteligence „oživuje" herní svět, respektive dává do pohybu živé entity ve hře, např. NPC postavy. NPC (Non-player character, v překladu „nehráčská postava") postavy jsou počítačem ovládané živé entity, které ve hře zastávají různě důležité role. Označení NPC je používáno většinou pouze pro postavy, které jsou vůči hráči přátelské, nebo alespoň neutrální. Pro nepřátelské neherní postaví je mezi MMORPG hráči nejrozšířenější výraz „mobové". Složitost umělé inteligence mobů je většinou menší, neboť se u ní neřeší interakce s hráčem, pouze reakce na hráče (většinou útok), tudíž je u nich nejdůležitější hlavně hledání cesty (pathfinding) a útok, jehož složitost se pohybuje od primitivního až po složitě naskriptovaný (sekvence různých útoků a krytí, popř. reakce na akce hráče, snímání okolí, úskoky, léčení apod.).

### 1.1.4 Uživatelské rozhraní

Uživatelské rozhraní, neboli UI (ang. user interface), zpracovává informace ze hry a zobrazuje je hráči v přehledné formě na obrazovku. Také reaguje na požadavky hráče, např. zobrazení okna s informacemi o postavě atd. V jiných žánrech počítačových her nepotřebuje hráč zobrazit tolik dat a uživatelské rozhraní nemusí být interaktivní. Takové rozhraní se nazývá HUD (heads-up display). To zobrazuje hráči jen nezbytné informace, potřebné k plnohodnotnému hraní, například počet životů, popř. many, aktuální stav munice, mini-mapu nebo směr (Wilson, 2006).

Trendem dnešní doby je minimalizace zobrazovaných informací a tím zvýšení viditelnosti okolního světa. Např. ukazatele života, resp. many zmizí, když hráč není v boji, resp. delší dobu nepoužil kouzlo (k vidění ve hře The Elder Scrolls 5: Skyrim),

nebo ukazatel munice není zobrazen na obrazovce, ale přímo na zbrani (ve hře Doom 3 od společnosti id Software). Existují i hry, které žádné výstupní informace na obrazovce neobsahují, tzv. HUD-less hry. V RPG hrách také existuje část UI, která by se dala považovat za HUD, ale informací, potřebných k hraní je mnohem více, proto je nutné rozhodnout, které informace budou viditelné permanentně a které se budou zobrazovat pouze na požadavek hráče. Také lze rozhodnutí částečně lze nechat na hráči. Tvorba uživatelského rozhraní by se neměla podceňovat, neboť nepřehledné a těžce ovladatelné UI může hráče od hraní velice snadno odradit. Dobré UI by mělo být tzv. user-friendly – hráč se v něm vyzná a informace jsou snadno dostupné. Také by nemělo hráče přehlcovat informacemi. Je vhodné, aby bylo uživatelské rozhraní nejdříve stručné, všechny informace není třeba mít zobrazené permanentně. Například při najetí myší na předmět se zobrazí jen základní informace, teprve až po kliknutí se zobrazí veškerá data o předmětu. Nesmí chybět možnost UI kompletně skrýt, pro lepší pořizování snímků obrazovky, tzv. screenshotů.

## 1.2 Seznámení s předními, či zajímavými RPG tituly

### 1.2.1 Série Diablo


Obrázek 1 - Diablo 1 (Zdroj: http://www.diablowiki.net/Diablo_I)

První díl Diabla vydala společnost Blizzard Entertainment v roce 1996. Je to pravděpodobně nejznámější představitel žánru akčního RPG, kombinující první klasického RPG s akčním herním stylem. Tento koncept inspiroval mnoho dalších her. Někdy se takovéto hry nazývají „Diablo klony“. Hraje se ve 2D izometrickém pohledu. Na svoji dobu měla hra kvalitní nejen grafiku v rozlišení 640x480 pixelů, ale i zvukovou stránku a to jak hudbu, tak i zvuky a dabing postav. Dále nabízí vysokou výdrž hry, spolu s vysokou znovuhratelností, díky náhodnému generování obsahu. To zahrnuje náhodné generování levelů, výběr monster, které se v daném levelu objeví, generování předmětů a také se vybírá z množství questů, které budou dostupné a které ne. Zvláště unikátní je systém generování magických předmětů: Ve hře jsou tři druhy předmětů – obyčejné, magické a unikátní. Obyčejné předměty jsou stejné (v rámci druhu) a nemají žádné zvláštní schopnosti. Unikátní mají pevně dané speciální vlastnosti a jméno. Magické předměty mají svojí předponu, nebo příponu, popřípadě obojí, podle kterých se odvíjí speciální vlastnosti. Například magický dlouhý meč s názvem Jagged Longsword of The Ages bude nezničitelný a bude mít zvýšené poškození. Předpony i přípony můžou být i negativní, tzn., mají záporné vlastnosti. Ve hře je 16 levelů, rozdělených do čtyř částí (klášter, katakomby, jeskyně a peklo), z nichž každá má svojí vlastní architekturu. Dohromady se tyto části nazývají labyrint (ang. dungeon). Hráč ovládá jednu ze tří postav (válečník, zlodějka nebo kouzelník), postupně prochází levely a získává zkušenosti zabíjením monster. Při postupu na další úroveň získá pět bodů, které si dle vlastního uvážení rozdělí do vlastností (síla, obratnost, vitalita, magie). Hlavním cílem hry je zničit Pána teroru, Diabla, jakožto zdroj zla, ohrožující vesničku Tristram. Hra má dva herní módy – pro jednoho hráče (singleplayer) a pro více hráčů (multiplayer). Kromě počtu hráčů se módy liší tím, že v multiplayeru se neukládá postup ve hře, ukládá se pouze postava, na rozdíl od singleplayeru. Ve vesnici jsou vchody do všech čtyř částí labyrintu, v multiplayeru je pro vstup potřeba určitá úroveň postavy, kdežto v singleplayeru se vchod odemkne, až když tam postava dojde skrz labyrint. V rámci zajištění vyváženosti obou módů jsou monstra v multiplayeru silnější. Hra také obsahuje tři herní obtížnosti, po projití první se odemkne druhá atd. V roce 1998 vyšel datadisk Hellfire, ale hráčskou komunitou nebyl přijat příznivě, hlavně kvůli tomu, že ho nevydala společnost Blizzard Entertainment, nýbrž Sierra Entertainment (DiabloWiki, 2014).


Obrázek 2 - Diablo 2 (Vlastní zdroj)

V roce 2000 vyšel druhý díl série, Diablo 2, který zachovává koncept akčního RPG a nabízí více herních postav, herních prvků, jako jsou například stromy dovedností. Při postupu na novou úroveň hráč kromě pěti atributových bodů dostane navíc jeden dovedností bod, který si dle vlastního uvážení rozdělí do jednoho ze tří stromů dovedností, což hráči dává ještě větší volnost při vývoji herní postavy. Dále hra nabízí více výbavových slotů postavy (v původní hře bylo dohromady jen 7 slotů, zde je 10), více předmětů, možnost kombinace a vylepšování předmětů, lepší podporu multiplayeru (obchod mezi hráči, vytváření skupin). Multiplayer se od singleplayeru už v ničem neliší. Síla monster (a s ní i výše získaných zkušeností) se odvíjí od počtu hráčů ve hře. Lze také simulovat hru více hráčů, než je skutečně ve hře, což umožňuje individuálně přizpůsobovat obtížnost hry. Oproti Diablu 1 už hra probíhá převážně v exteriérech a hráč postupně prochází čtyřmi částmi hry – akty. O rok později vyšel datadisk Lord of Destruction, který do hry přidává jeden celý akt, dvě nové hrací postavy (celkem 7), nové předměty, questy a monstra (James).


**Obrázek 3 - Diablo 3 (Zdroj: http://www.gamer.nl/achtergrond/454098/diablo-3-meer-plezier-zonder-auction-house)**

V roce 2012 společnost vydala dlouho očekávaný třetí díl Diabla, který dále rozšiřuje a vylepšuje herní mechanismy. Hra je kompletně ve 3D a klade větší důraz na plnění questů, kterých je výrazně větší množství oproti předchozím dílům. V březnu 2014 vyšel datadisk Reaper of Souls, který do hry přidává další akt, novou postavu, vylepšené herní systémy, nové mechanismy jako sdružování hráčů do gild (ang. guild) nebo novou profesi řemeslníka (DiabloWiki, 2014).

### 1.2.2 The Elder Scrolls 5: Skyrim

Obrázek 4 - TES4 Skyrim (Zdroj: http://hdwallsource.com/tag/skyrim)

The Elder Scrolls V: Skyrim je akční RPG s otevřeným světem pro jednoho hráče vytvořené společností Bethesda Game Studios a vydané společností Bethesda Softworks. Skyrim vyšel 11. 11. 2011 (přesně rok od oznámení hry). Hlavní příběhová linie Skyrimu vypráví o boji hlavní postavy (hráče) proti dračímu bohu Alduinovi, který má podle pradávného proroctví zničit svět. Hra nabízí nelineární herní systém, který je v sérii The Elder Scrolls (dále jen TES) tradiční. Hráč může prozkoumávat svět buď pěšky, nebo jízdou na koni. Hra také obsahuje možnost rychlého cestování na již prozkoumaná místa. Plnění úkolů je důležitou částí hry, ale zároveň hra nenutí hráče, aby je plnil, což mu dává větší volnost. Úkolů je ve hře obrovské množství, některé typy questů hra náhodně generuje, tudíž lze tvrdit, že jich je nekonečně mnoho. Questy často mívají víc způsobů jak je dokončit. To pak ovlivňuje další faktory hry, jako třeba questy, které dále hráč dostane po splnění jiného questu, nebo postoj NPC postavy, či celé vesnice vůči hráči. To dává hráči do ruky možnost ovlivňovat okolní svět svými činy. Ve hře jsou různé frakce, kam se hráč může připojit, mající vlastní příběhové linie, které hráč může plnit vedle hlavní příběhové linie. Chování NPC postav působí velmi realisticky, každá postava má svůj denní harmonogram. Někdo celý den vysedává po krčmách, někdo pracuje na poli, kněží chodí po poledni na hřbitov pronášet modlitby atd. Postavy mezi sebou často vedou dialogy, když se

potkají například na ulici, opilci křičí na krčmářku „o další rundu“ a krčmářka jim barvitě odpovídá. V noci jsou obchody zavřené a lidé jsou ve svých domovech, kde spí. Specifickou částí hry jsou draci a dračí řev, neboli „Thu'um“. Hráč také ovládá umění dračího řevu a jakožto Drakorozený do sebe umí absorbovat dračí duše, které mu umožňují zlepšit dračí řevy, kterých je ve hře několik desítek. Každý řev se skládá ze tří slov, které hráč objevuje u tzv. „Dračích zdí“, které jsou rozesety po celém světě.

TES má svůj originální systém vývoje postavy. Na začátku si hráč zvolí rasu a vzhled postavy. Skyrim obsahuje 18 dovedností, které jsou rozděleny do tří skupin: boj, magie a zlodějské dovednosti. Hráč postupuje na další úrovně tím, že používá a trénuje dovednosti, čímž získává dostatečné množství zkušeností, které se následně promítnou v postupu na novou úroveň. Předchozí díly Elder Scrolls využívaly systém povolání, který určoval, jaké dovednosti se budou podílet na postupu na vyšší úroveň (a které naopak nikoliv). Odstranění tohoto systému ve Skyrimu umožnilo hráčům, aby se jejich postava sama přirozeně vyvinula podle toho, jaké dovednosti skutečně používají. Jakmile hráčova postava dosáhne nové úrovně, hráč dostane tzv. „perk“ (Sillmen, 2011). Každá dovednost má svůj vlastní perkový strom a je čistě na hráči kam každý perk vloží. Každý stupeň stromu, vyžaduje určitou úroveň dovednosti, takže si hráč může perk uschovat na později, až dosáhne v dané dovednosti požadované úrovně.

Ke hře byly vydány tři oficiální dodatky:

Dawnguard byl vydán 26. Června 2012. Přidává do hry dvě frakce – Upíří rod Volkihar a frakci lovců upírů Dawnguard. Hráč se může rozhodnout, jestli se přidá na stranu upírů nebo lovců upírů. Obě frakce mají vlastní příběhovou linii, které se často prolínají.

Heartfire byl ohlášen 28. Srpna 2012 a na PC vyšel 4. Října. Dodatek do hry přidává možnost koupit si vlastní pozemek, na kterém si lze postavit dům. Dům se skládá z několika částí a každou část si lze jinak zařídit, například v levém křídle si hráč může postavit alchymistickou dílnu, nebo zbrojnici, v suterénu vinný sklípek, cvičiště, či pokoj pro upíry s rakví místo postele. Dále si může hráč adoptovat sirotka, najmout služebníky, či osobního barda. Nově jsou hráči dispozici aktivity jako farmaření,

včelařství, pečení a rybaření. Čas od času musí svůj dům hráč chránit před nájezdy banditů, či obrů.

Dragonborn přidává do hry unikátní dračí řev, pomocí kterého lze zkrotit a osedlat draka. Také přidává novou příběhovou linii, spolu s ostrovem Solsteim, který byl dějištěm datadisku Bloodmoon ve třetím dílu série TES: Morrowind.

### 1.2.3 Mount & Blade

Obrázek 5 - Mount & Blade (Zdroj: http://knights.calltoreason.org/?p=8505)

Mount & Blade je akční RPG hra, dobově posazená do středověku, vytvořená tureckým nezávislým studiem TaleWorlds a vydaná 20. Listopadu 2008. Někdy se označuje jako simulátor středověku. Hra neobsahuje žádný příběh, hráč je vržen do země zvané Calradia, o kterou soupeří 5 frakcí. Ve hře je propracovaný systém diplomacie, frakce mezi sebou válčí, uzavírají příměří a mírové smlouvy, které dřív nebo později jedna frakce poruší a opět mezi sebou válčí. Každá frakce začíná se svým územím, které rozšiřuje, popř. ztrácí válkou. V jejich čele je vůdce, většinou král, nebo chán. Pod sebou mají vůdci své vazaly, kteří mají každý určitý stupeň loajality, tzn., jsou více, nebo méně věrní svému vůdci. Při velmi nízké oddanosti může vazal dezertovat k jiné frakci. Ve hře je také funkční systém cti, jak pro hráče, tak pro počítačem ovládané vazaly. Čest lze získat různými způsoby, například odmítnutím

odměny za quest od vesničanů, či výhrou v turnaji. Naopak lze čest ztratit například okrádáním vesničanů či karavan, nebo pleněním vesnic. Kladná míra cti zlepšuje vztah u čestných lordů a zhoršuje vztah s nečestnými lordy. U negativní míry cti je tomu naopak. Dále je ve hře systém ekonomie a obchodu. Každé město a vesnice má míru prosperity, která je ovlivněna mnoha faktory. Od prosperity se odvíjí výše daní pro lorda (nebo hráče), který město, či vesnici vlastní.

Hráč si může vybrat, jestli se přidá k nějaké frakci, nebo si založí vlastní, popřípadě zda bude žoldákem na volné noze, či obchodníkem. Největší důraz ve hře je kladen na boj, ať už na otevřeném prostranství, či obléhání hradů. Každý lord i hráč má vlastní družinu vojáků, kteří se zdokonalují bojem, od branců z vesnic až po elitní veterány. Velikost družiny je dána schopností velení, ctí a mírou známosti v zemi. Velmi zajímavý aspekt hry je systém boje na koni.

Hra se hraje ze dvou pohledů: Strategická mapa slouží k cestování, neobsahuje žádné detaily, pouze pozice vesnic, hradů, měst a družin. Když dojde k boji, nebo hráč navštíví například vesnici, hra se přepne do pohledu třetí nebo první osoby.


**Obrázek 6 - Mount & Blade - strategická mapa (Zdroj: http://vygame.net/mount-blade-medieval-game/4mountblade-play/)**

Hráčova postava zvyšuje svojí úroveň účastí v bitvách nebo plněním úkolů. Za každou úroveň dostane jeden bod do vlastností, jeden dovednostní bod a několik zbraňových bodů. Dovednosti zlepšují pohyb po mapě (například rychlost, dohled, stopování) i bitevní dovednosti (např. jezdectví, jízdní lukostřelba, krytí, vrhací zbraně).

Vývojáři implementovali vlastní grafický a fyzikální engine, podporující shadery a HDR osvětlení. Ačkoli hra oplývá (na svou dobu) vysoce kvalitní grafikou, hra samotná na disku zabírá překvapivě málo místa a doba nahrávání mezi pohledy trvá sotva pár sekund, je to díky tomu, že při spuštění hry se načtou do paměti všechny důležité části, např. textury (Sillmen, 2008).

Ke hře vyšlo několik datadisků:

Warband rozšiřuje původní svět, přidává novou frakci a zvyšuje hratelnost hry novými funkcemi a mechanismy, jako třeba hru více hráčů.

With Fire and Sword posouvá dobu o 200 let později (1650), obsahuje kompletně novou mapu, nové frakce, herní mechanismy a střelný prach.

Napoleonic Wars nabízí pouze multiplayer hraní z období posledních let vlády Napoleona Bonaparte. Simuluje skutečné bitvy z dob Napoleonských válek, například bitva u Waterloo. Rozsah bitvy je až 250 hráčů (Taleworlds).

### 1.2.4 World of Warcraft


Obrázek 7 - World of Warcraft (Vlastní zdroj)

World of Warcraft je fantasy představitel žánru MMORPG, vyvinutá společností Blizzard Entertainment v roce 2004. Hra drží rekord v Guinnessově knize rekordů za nejpopulárnější MMORPG. Před vydáním třetího datadisku hru hrálo 12 milionů hráčů. Hlavní myšlenkou hry je nekončící boj mezi dvěma frakcemi – Aliancí a Hordou. Příslušníci Aliance bojují za světlo a spravedlnost, odmítajíce temnotu. Příslušníci Hordy bojují za své místo ve světě a ctí tradice. Nelze říci, že je Aliance dobrá a Horda špatná, obě frakce mají od každého trochu, z obou vzešlo mnoho zla i dobra. Hra rozšiřuje příběh světa Warcraftu, který má počátek až v roce 1994, kdy vyšel první díl série Warcraft: Orcs and Humans. Za tu dobu vyšlo několik knížek, které dále prohlubují děj příběhu. Tvorba postavy je následující: Hráč si na začátku hry zvolí, za jakou rasu bude hrát, tím zvolí i frakci, dále si vybere povolání a definuje vzhled a jméno postavy. Hráč začíná ve startovní lokaci své rasy, kde ho hra v průběhu prvních úrovní postavy seznámí se základními principy hry. Zkušenosti hráč získává především plněním questů, popřípadě zabíjením „mobů“ (výraz pro NPC postavy, které jsou vůči hráči nepřátelské).

Ve hře existuje mnoho mechanismů pro interakci hráčů mezi sebou, které zajišťují obrovskou hratelnost, aby si každý ve hře našel část, která ho baví. Hráči se sdružují do gild, které vytváří menší komunity, například za účelem dosažení společného cíle.

Herní mechanismy se dají rozčlenit do tří částí:

Obecná část – Sem lze zařadit například profese. Ty se člení na „farming" a „crafting" profese. Farming profesí se získává materiál, který se zpracovává crafting profesí. Například profese Herbářství a Alchymie. Hráč díky herbalismu může sbírat rostliny, které jsou rozeseté po celém světě (farming) a pomocí alchymie z nich dělá lektvary (crafting). Dále sem patří obchodování. K tomu slouží systém aukce. Hráč může nalezené nebo vyrobené předměty vložit do aukce, kde ho jiní hráči mohou koupit, buď okamžitě za zvýšenou cenu, nebo postupným přihazováním peněz k dané částce. Popřípadě je možné obchodovat přímo mezi dvěma hráči. Také hráč může plnit tzv. Achievementy (počeštěno z anglického slova „achievement"), což je ocenění s určitou podmínkou pro jeho získání. Některé jsou lehce získatelné, jiné naopak velice náročné, ať už časově nebo jinak. Existují typy hráčů, kteří se snaží získat extrémní množství ocenění, takzvaní „Achievement hunteři". Do této časti lze také zařadit systém pošty, nebo možnost uložení předmětů do banky.

PvE (Player versus Environment) – Do této části spadají hlavně dungeony a raidy. To jsou oblasti, kde jsou silnější monstra a tzv. bossové. Ke vstupu do oblasti je potřeba určité množství hráčů, na dungeon je to 5 hráčů, na raid 10-40 hráčů. Výpravy do dungeonů a raidů se podnikají hlavně za účelem získání lepšího vybavení pro svou postavu.

PvP (Player versus Player) – Hlavní představitelé této části jsou Battlegroundy a Arény. V Battlegroundu (dále jen BG) proti sobě bojují obě frakce v různě velkých skupinách (10, 15, nebo 40 hráčů), podle typu BG. Za vítězství v těchto bitvách hráč získává tzv. Honor pointy, což je jeden z typů měny ve hře. V arénách bojuje tým proti týmu (zde se nerozlišuje jakou frakci tým má) o velikosti 2, 3 nebo 5 hráčů. Jako podpora soupeřivosti mezi hráči slouží žebříčky (jeden na každou velikost týmu). Za výhru tým získá body, za prohru naopak body ztratí. Za účast v arénách je hráč

odměňován Arena Pointy, což je další herní měna. Další možnost soupeření je v podobě PvP zón. Také v několika normálních zónách jsou tzv. PvP úkoly, například obsadit několik věží. Za splnění takových úkolu jsou všichni hráči dané frakce, v rámci zóny odměněni kouzlem (ve hře označován jako „buff"), který jim dává výhodu v boji (např. větší udělené poškození).

Protože má hra tak velikou hráčskou základu, jsou oficiální servery rozděleny na tzv. realmy, které se liší typem a jazykem. Jsou čtyři typy realmů:

Normal – Tento typ je především pro PvE hráče. S výjimkou Battlegroundů, Arén a PvP zón je možnost soubojů mezi hráči implicitně vypnutá, avšak hráč si jí může kdykoli zapnout a tím dát příležitost hráčů z opačné frakce na něj zaútočit.

PvP – Zaměřeno na podporu PvP. Až na startovní lokace a svatyně (herní výraz pro město, které je společné pro obě frakce) lze na ostatní hráče kdykoli zaútočit.

RP – Typ realmu, zaměřený na podporu speciálního typu hraní – Roleplayingu (hraní rolí). To znamená, že hráč se ve hře chová a mluví tak, jak by se chovala a mluvila postava, za kterou hraje. Vytvoří si vlastní minulost postavy, používá dobové výrazy a společně s ostatními hráči utvářejí vlastní příběhy.

RPPvP – Realm, kombinující typy RP a PvP.

*Datadisky*

The Burning Crusade – Vyšel v roce 2007. Maximální dosažitelnou úroveň postavy navyšuje ze šedesáti na sedmdesát, přináší nový kontinent, dungeony a raidy, jeden nový Battleground, novou profesi, dvě nové rasy a mnoho předmětů. Nové dungeony mají kromě normální obtížnosti i tzv. Heroic (HC) obtížnost. HC dungeony jsou speciálně navržené pro postavy s maximální úrovní – jsou těžší, ale odměny jsou vyšší.

Wrath of the Lich King (2008) – Maximální úroveň, které lze dosáhnout zvyšuje na osmdesát, umožňuje prozkoumat legendární Northrend, kontinent daleko na severu. Hra je obohacena o nové povolání, novou profesi, dungeony a raidy. Také přináší tzv. Looking for Group systém, do kterého se hráč přihlásí, systém sám vytvoří skupinu z

přihlášených lidí a automaticky jí přemístí do dungeonu. Datadisk také rozšiřuje hru o výše zmíněné Achievementy.

Cataclysm (2010) – Maximální dosažitelnou úroveň navyšuje o dalších 5 úrovní (85). Kompletně předělává kontinenty z původní hry jako následek běsnění dračího Aspekta Smrti, Deathwinga. Přináší do hry dvě nové rasy, Gobliny (Horda) a prokleté Worgeny (Aliance). Tradičně nabízí nové raidy a dungeony, dále mnoho nových achievementů, předmětů, novou profesi, či větší nebo menší úpravy stávajících herních mechanismů. Také umožňuje tzv. transmogrifikaci, což je možnost změny vzhledu předmětu.

Mists of Pandaria (2012) – Přidává do hry nově objevený kontinent Pandaria. Spolu s ním přichází i nová rasa, pandaren a nové povolání, mnich. Datadisk umožňuje postoupit hráčům až na devadesátou úroveň. Nabízí nové herní mechanismy, jako například souboje ochočených zvířátek. Obohacuje hru o nové dungeony, raidy, achievementy, nově také tzv. „Scénáře“, což je forma dungeonu pro tři hráče, silně založeného na příběhu.

Warlords of Draenor – jeho vydání je naplánováno v tomto roce (WoWWiki, 2014).

## 1.3 Objektově orientovaný přístup

### 1.3.1 Objektově orientované modelování

*Jde o způsob nazírání na SW pomocí abstrakce reálného světa. Základním stavebním prvkem je objekt (entita), která v sobě zahrnuje jak datovou strukturu popisující určitou entitu, tak i pravidla chování této entity. Tím se objektový přístup liší od konvenčního programování, kde datová struktura a chování jsou spojeny jen velmi volně* (Šmíd, 2003).

Pro každý objekt jsou zavedeny čtyři charakteristické aspekty:

- **Jedinečnost** – každý objekt je rozlišitelnou entitou, i když má stejné chování a strukturu jako jiný objekt.
- **Zatřiditelnost** – objekty se stejnými atributy a operacemi jsou seskupovány do tříd.

- **Mnohotvárnost (polymorfismus)** – stejně označená operace se může chovat jiným způsobem a dávat různé výsledky pro rozdílné třídy objektů.
- **Dědičnost** – je mechanismus, umožňující převzít (zdědit) vlastnosti třídy z předka do potomka. Od předka jsou zděděny atributy, třídy, relace (vztahy) a omezení. V nové třídě lze přidat nové atributy a metoda a tím obohatit vlastnosti předka. Zákon nahraditelnosti říká, že všude tam, kde je možné použít předka, musí být možné použít i potomka.

Objektově orientované modelování je založeno na čtyřech principech, které vyjadřují podstatu objektového přístupu:

- **Abstrakce (zobecnění)** – objekty obsahují pouze podstatné vlastnosti objektů z reality, nikoli všechny.
- **Zapouzdření** – „Představuje uzavřenost vnitřní struktury objektů vůči jeho okolí. Atributy objektu by měly být přístupny pouze prostřednictvím metod objektu. Je nejdůležitější vlastností objektů, jelikož snižuje vnější závislosti objektu a tím usnadňuje vývoj, testování, údržbu a rozvoj. Přispívá ke znovupoužitelnosti“ (Svoboda, 2013).
- **Znovupoužitelnost** – je možná díky dědičnosti datových struktur a chováním mezi několika podobnými třídami bez redundance. To umožňuje opakované použití již funkčních programů v jiných aplikacích.
- **Spolupůsobení** – jednoznačnost, zatřiditelnost, mnohotvárnost a dědičnost charakterizují hlavní proud objektově orientovaných jazyků. Každý z těchto aspektů lze izolovat, ale dohromady působí symetricky.

### 1.3.2 Srovnání objektového a procedurálního přístupu

Procedurální (algoritmický) přístup se někdy nazývá hierarchickým rozkladem, neboť je spojen s funkční dekompozicí. Tento přístup úzce souvisí se strukturovaným programováním. Analyzovaný a navrhovaný systém je rozdělen na dvě oblasti: data a funkce. Každá z těchto oblastí se analyzuje, navrhuje a implementuje zvlášť.

*Data se modelují pomocí datových modelů, které potom mohou vést k návrhu databází. Funkce se analyzují samostatně s využitím funkční dekompozice. V programu se funkce spojují s určitými daty, která jsou však zpravidla globální. Takovéto dělení na*

*data a funkce je však umělé, protože v běžném světě existují objekty reality, které vlastní jak data, tak funkce* (Administrator, 2005) *.*

Jinými slovy, procedurální přístup je transformace objektů reality na funkční a datové struktury.

V objektově orientovaném přístupu člověk může informační systém představit mnohem snáz, neboť místo prvků dat a funkcí zavádí prvek **objekt**. Tento objekt je vyjádřením reálného objektu v běžném světě. Obsahuje jak data (atributy), tak funkce (metody) relevantní pro daný objekt (Administrator, 2005).

## 1.4 UML

*Jazyk UML (Unified Modeling Language) je univerzální jazyk pro vizuální modelování systémů. Přestože je nejčastěji spojován s modelováním objektově orientovaných softwarových systémů, má mnohem širší využití, což vyplývá z jeho zabudovaných rozšiřovacích mechanismů* (Arlow, a další, 2011)*.* Nenabízí žádný druh metodiky modelování, poskytuje pouze vizuální syntaxi, kterou lze využít při sestavování modelů.

### 1.4.1 Historie UML

*Do roka 1994 existovalo několik soupeřících jazyků pro vizuální modelování a také několik metodik. Více než polovinu tehdejšího trhu si mezi sebe rozdělily metody Booch a OMT (Object Modeling Technique, Rumbaugh). Mezi metodikami si nejlépe vedla metodika Objectory (Jacobson). Jedním z prvních pokusů o sjednocení byla metodika Fusion, do které ale nebyli zapojeni autoři metod, které tvořily podstatnou část tehdejšího trhu – skončila neúspěchem, když se Booch a Rumbaugh spojili ve firmě Rational Corporation, která pracovala na tvorbě jazyka UML. V roce 1996 navrhlo sdružení OMG (Object Management Group) specifikaci RFP (Request for Proposal) pro objektově orientovaný jazyk pro vizuální modelování, v němž jako standard navrhlo jazyk UML. V roce 1997 OMG jazyk UML přijalo a stal se otevřeným průmyslovým standardem. V roce 2000 byl jazyk UML významně rozšířen o sémantiku akcí. Ta slouží k popisu chování množiny primitivních akcí, jež lze implementovat pomocí specifických*

*akčních jazyků. V roce 2005 byla dokončena finalizace specifikace jazyka UML 2.0* (Arlow, a další, 2011).

### 1.4.2 Struktura jazyka UML

Model UML obsahuje dva aspekty:

- Statická struktura – popisuje důležité typy objektů systému a jejich vztahy.
- Dynamické chování – Znázorňuje životní cyklus objektů a způsob jejich vzájemné spolupráce za účelem dosažení požadované funkce navrhovaného systému.

Oba aspekty spolu tvoří nedělitelný celek – žádný z nich není bez druhého úplný.

Struktura jazyka UML se skládá z těchto tří součástí:

- Stavební bloky – základní prvky modelu, diagramy a relace.
- Společné mechanismy – obecné způsoby, kterými lze dosáhnout specifických cílů.
- Architektura – pohled na architekturu navrhovaného systému.

Jazyk je sestaven ze tří stavebních bloků:

- Předměty – prvky modelu
- Vztahy – vazby mezi předměty. Určují, jak spolu předměty významově souvisí.
- Diagramy – pohledy na modely UML. Ukazují, **co** bude systém dělat a **jak**.

Předměty se dále dělí na:

- Strukturní abstrakce – podstatná jména modelu – třídy, komponenta, spolupráce, uzel, případ užití, rozhraní atd.
- Chování – slovesa modelu – např. stav. interakce.
- Seskupení – balíčky, používané k seskupování souvisejících prvků do celků.
- Poznámky – anotace, kterou lze k modelu připojit. Používají se k vyjádření dodatečných informací, např. zvýraznění nějaké důležité skutečnosti.

Diagramy jsou pohledy na model. Existuje celkem třináct různých typů diagramů.

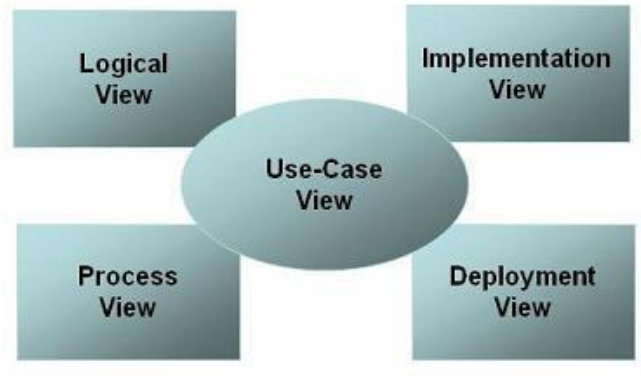
Diagramy struktury:

- Diagram tříd
- Diagram složené struktury
- Diagram komponent
- Diagram nasazení
- Objektový diagram
- Diagram balíčku

Diagramy chování:

- Diagram aktivit
- Diagram interakce
    - Diagram posloupnosti
    - Diagram komunikace
    - Stručný diagram interakce
    - Diagram časování
- Diagram případu užití
- Diagram stavového automatu

Architektura je zachycením strategických aspektů vyšší struktury systému. Pohlíží se na ní nejčastěji „pohledem 4+1“:

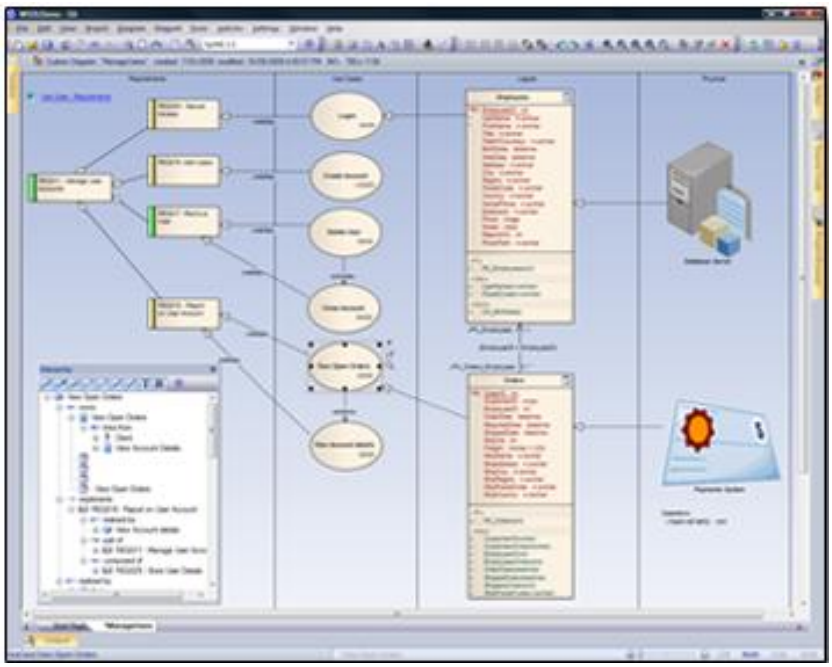



**Obrázek 8 - Pohled 4+1 (Zdroj: http://epf.eclipse.org/wikis/openup/core.tech.common.extend_supp/guidances/examples/four_plus_one_view_of_arch_9A93ACE5.html)**

- Logický pohled – znázorňuje slovník oblasti problému jako množinu tříd a objektů.
- Pohled procesů – vyjadřuje spustitelná vlákna a procesy jako aktivní třídy
- Pohled implementace – modeluje soubory a komponenty, utvářející kód systému.
- Pohled nasazení – modeluje fyzické nasazení komponent na množinu fyzických výpočetních uzlů.
- Pohled případů užití – tvoří základ tvorby všech dalších pohledů, zachycuje základní požadavky na příslušný systém (Arlow, a další, 2011).

### 1.4.3 Enterprise Architect

Enterprise Architect je vysoce kvalitní nástroj pro tvorbu modelů. Jeho poslední verze je založena na syntaxi UML 2.4.1. Nabízí výkonné vizuální prostředí pro řízení požadavků, strategické a business modelování, návrh enterprise architektury a systémovou analýzu. Obsahuje podporu vývoje - tvorba softwarového designu, generování kódu, testování, deployment atd. Obsahuje i funkčnosti, které jsou výhodné pro nasazení v rozsáhlých týmech, jako například podpora životního cyklu vývoje software, verzování modelů, audit a další.



Obrázek 9 - EA (Zdroj: http://www.sparxsystems.com/products/ea/)

Enteprise Architect poskytuje plnou podporu životního cyklu modelů pro:

- Business a IT systémy
- Softwarové a systémové inženýrství
- Vývoj realtime systémů

Díky podpoře správy požadavků Enterprise Architect umožňuje sledovat jednotlivá konkrétní zadání k jejich analytickým, designovým, implementačním modelům a modelům popisujícím testy a údržbu. Využívá UML, BPMN, SysML a další jazyky (DataProjekt).

## 1.5 Objektově orientované jazyky

### 1.5.1 C++

C++ vyvinul Bjarne Stroustrup na základě programovacího jazyka C, který je až na pár, jasně definovaných výjimek podmnožinou C++. Jedná se multiparadigmatický programovací jazyk. Podporuje jak objektově orientované, tak procedurální, či generické programování. Jazyk byl vyvíjen od počátku osmdesátých let a v průběhu devadesátých let se stal dominujícím programovacím jazykem v mnoha oborech, například v bankovnictví, telekomunikacích, či CAD systémech. Proto se mnoho národních organizací pro standardy, spolu s International Standards Organization (ISO) v roce 1989 začali snažit o standardizaci C++ a po osmi letech práce byl schválen první standard C++. Standardizace jazyka výrazně usnadnila jeho učení (zavedly se například jednotné kurzy pro střední školy ve Spojených Státech), použití C++ aplikací, či přenositelnost aplikací mezi počítači (Stroustrup, 1997).

C++ obsahuje čtyři typy přetypování:

- Statické přetypování – **static_cast<>** – používá se ke konverzím mezi primitivními datovými typy, pro přetypování z potomka na předka, pro přetypování referencí i ukazatelů z potomka na předka, atd. Tento operátor není vhodný pro přetypování z předka na potomka.

- Dynamické přetypování – **dynamic_cast<>** – slouží pouze k přetypovávání, referencí nebo ukazatelů. Z reference je možné vytvořit jen reference, z ukazatele lze vytvořit jen ukazatel. Je vhodné ho použít pro přetypování ukazatele (nebo reference) na předka na ukazatel (referenci) na potomka. Bezpečně přetypovává polymorfní třídy, neboť k přetypování dojde až za běhu programu díky dynamické identifikaci typů. Výsledný ukazatel ukazatele, který nelze přetypovat má hodnotu NULL. Pokud selže přetypování reference, je vržena výjimka typu bad_cast.
- Obecné přetypování – **reinterpret_cast<> –** slouží k přetypování datových typů, které spolu nijak nesouvisí. Převádí ukazatele na instance nijak nesouvisejících tříd, nebo struktur (bez společného předka), dále převádí ukazatele na celá čísla a podobně (Builder.cz, 2001).
- Konstantní přetypování – **const_cast<>** – přetypovává konstantní proměnnou na nekonstantní a naopak.

Jazyk C++ podporuje použití šablon, tzn. lze napsat kód s neurčeným datovým typem, což vede k vyšší znovupoužitelnosti kódu. Také umožňuje přetěžování metod (lze deklarovat více metod se stejným názvem, překladač sám určí správnou metodu podle počtu a typu parametrů). Silnou stránkou jazyka je možnost přetěžování standardních operátorů (např. „+“, či „=“). Tím lze vytvářet abstraktní datové typy. C++ umožňuje vcelku unikátní možnost vícenásobné dědičnosti, tzn., že třída může dědit z více tříd (Fišer). Dále lze použit virtuální dědění, nebo jeden ze tří druhů klasické dědičnosti:

- **public (veřejná)** - Nejtypičtější způsob dědičnosti. Potomek získá všechny veřejné a chráněné členy rodičovské třídy se stejnými oprávněními, jako má rodičovská třída.
- **protected (chráněná)** - Potomek získá všechny veřejné a chráněné členy předka s novým oprávněním protected, tzn., všechny původně veřejné členy předka budou v potomkovi protected.
- **private (soukromá)** - Potomek získá všechny public a protected členy předka s novým oprávněním private. Veškeré původně veřejné a chráněné členy

předka budou v potomkovi private. Je to nejrestriktivnější způsob přenosu přístupových práv (Fabian, 2013).

Velké vývojářské společnosti počítačových her využívají pro své projekty právě C++.

### 1.5.2 Java

Java je objektově orientovaný programovací jazyk, vyvinutý firmou Sun Microsystems, kterou později koupila firma Oracle. Jazyk vychází z principů jazyka C++, ke kterému má syntakticky velmi blízko (v současné době se podobá více jazyku C#, ale ten vznikl až po nástupu Javy). Java vypouští některé konstrukce C++, například ukazatele. Ty jsou nahrazeny referencemi. Navíc přidává mnoho užitečných funkcí:

- automatická správa paměti pomocí garbage collectoru
- mechanismus vláken
- podpora tvorby dynamicky rozšiřitelných aplikací, např. plug-inů.
- vysoká bezpečnost
- možnost serializace objektů
- reflexe
- mechanismus výjimek
- velikost standardně dodávaných knihoven

Velkou výhodu Javy je hardwarová nezávislost, díky které lze aplikace vyvinuté pro Windows spustit bez problémů i v Linuxu nebo MAC OS, pokud je na nich nainstalován Java Runtime (Kubů).

### 1.5.3 C#

Jazyk C# je objektově orientovaný programovací jazyk od počítačové firmy Microsoft. Vyšel zároveň s platformou NET. Je to zjednodušená, vylepšená a čistě objektová verze programovacího jazyka C++. C# je vhodný zejména k tvorbě databázových programů, webových služeb a aplikací, formulářových aplikací ve Windows apod. V současné době existuje pět verzí C#:

- **verze 1.0**: vychází první verze jazyka C# s Frameworkem 1.0. Vznikají jednoduché programy a téměř žádné výhody pro jednoduchost kódu.
- **verze 2.0**: s touto verzí přichází nové doplňky jazyka jako je generika, částečné a statické třídy, iterátory a anonymní metody pro pohodlnější užívání delegátů.
- **verze 3.0**: tato verze vyšla společně s .NET Framework verzí 3.5. Jazyk C# byl touto verzí obohacen užitečnými doplňky. Jako příklad lze uvést dotazovací jazyk (LINQ), lambda výrazy, či anonymní třídy.
- **verze 4.0**: umožnila spolupráci s dynamickými aspekty programování a různými frameworky. Dále přinesla podporu dynamicky typovaných objektů, kovarianci a kontravarianci, či volitelné a pojmenované parametry.
- **Verze 5.0**: vydána společně s .NET Framework 4.5. Současně vyšlo i Visual Studio 2012. Přináší podporu asynchronního programování a *Caller Information* atributy (csharp.aspone.cz) (Janáček, 2013).

### 1.5.4 Vývojová prostředí

- Eclipse – v roce 2001 společnost IBM uvolnila Eclipse jako open-source IDE. Od té doby se stal jedním z nejpoužívanějších vývojových prostředí nejen pro jazyk Java, ale díky velké podpoře plug-inů i pro další programovací jazyky, jako jsou Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Perl, PHP, Python, R, Ruby (včetně frameworku Ruby on Rails), Scala, Clojure, Groovy, Scheme, či Erlang. Eclipse kompiluje kód na pozadí a nalezené chyby červeně značí. Vedle široké podpory pluginů je další jeho silnou stránkou systém debugování. Je velmi vhodný pro tvorbu velkých projektů (Bolton, 2013).
- NetBeans IDE – nezávisle ho začala v roce 1990 vyvíjet skupina pražských studentů Matematicko-fyzikální fakulty. V roce 1999 ho odkoupila společnost Sun a uvolnila jako open-source IDE o rok později (Bolton, 2013). *NetBeans nabízí vývojové prostředí pro Javu, Ruby, C/C++ a PHP aplikace. Součástí je i podpora pro XHTML, CSS, JavaScript, XML a SQL. NetBeans podporuje barevné označování syntaxe, vlastní verzovací systém pro snadný návrat k předchozím úpravám, práci v panelech, široké možnosti nastavení, podporu ladícího*

*nástroje, napovídání funkcí a metod tříd a další funkce* (Živě.cz). Také umožňuje vkládání plug-inů. Ve srovnání s vývojovým prostředím Eclipse je však méně výkonější a těžkopádný. Je doporučován začátečníkům.

- Intellij IDEA – byl vyvinut podobně jako Eclipse, v roce 2001. Je k dostání ve dvou verzích: placené „Ultimate" edici, nebo neplacené open-source verzi. Umožňuje vývoj aplikací ve více programovacích jazycích, včetně Javy, jazyka Scala, Groovy, Clojure nebo Kotlin. Obě verze podporují automatické doplňování kódu, analýzu kódu, či pokročilý refactoring. Komerční verze navíc obsahuje nástroje pro Code Coverage, návrh UML, či pro práci s databází. Také podporuje SQL, Ruby, ActionScript, Python and PHP. Podobně jako Eclipse, či NetBeans, i toto IDE umožňuje vkládání plug-inů (Bolton, 2013).
- Microsoft Visual Studio - *je balík nástrojů a služeb určený k vývoji softwarových aplikací pro desktopové i dotykové prostředí Windows, pro web/HTML 5, SharePoint, mobilní zařízení i cloud prostředí* (msdn.com, 2014) . Verzí MS Visual Studio je několik, od bezplatné Express verze, až po placenou Ultimate edici, která nabízí nejširší nabídku nástrojů pro vývoj aplikací (VisualStudio).

## 1.6 Dijkstrův algoritmus

Dijkstrův algoritmus hledá nejkratší cesty v kladně ohodnoceném grafu. Od zadaného uzlu zjišťuje nejkratší vzdálenosti do všech ostatních uzlů. Algoritmus byl navržen nizozemským informatikem Edsgerem Dijkstrou v roce 1959.

*Na Dijkstrův algoritmus lze pohlížet jako na zobecněné prohledávání grafu do šířky, při kterém se vlna nešíří na základě počtu hran od zdroje, ale vzdálenosti od zdroje (ve smyslu váhy hran). Tato vlna proto zpracovává jen ty uzly, k nimž již byla nalezena nejkratší cesta* (Mička).

Všechny uzly jsou umístěny v prioritní frontě a řazeny dle vzdálenosti od zdrojového uzlu. Na začátku má zdrojový uzel vzdálenost 0 a všechny ostatní uzly nekonečno. V každém kroku algoritmus vybere uzel s nejnižší vzdáleností od již zpracované části a zařadí jej mezi zpracované uzly. Poté projde všechny jeho nezpracované potomky, přidá je do fronty (pokud tam již nejsou) a zkontroluje, zda

nejsou blíže zdroji, než byli před zařazením právě vybraného uzlu mezi zpracované. Po průchodu přes všechny potomky se vybere z fronty nový uzel s nejvyšší prioritou a celý krok se opakuje. Když je prioritní fronta prázdná, algoritmus končí (Mička).¨

# 2 Metodika tvorby PC her

## 2.1 Otázka hratelnosti a herní aspekty

Tvorba PC her je odlišná od vývoje klasického IT systému, který často vzniká na míru zákazníkovi na základě jeho požadavků a analýzy firemního prostředí. Cílem IT systému je zefektivnit firemní procesy, cílem PC hry je zabavit co možná nejvíce hráčů, na co nejdelší dobu. Při návrhu PC hry, zejména co se hratelnosti týče, je vhodné použít dekompozici, tzn. rozdělit si hlavní problém na více menších, jednodušších problémů. Tím vzniknou jednotlivé herní aspekty. A čím více těchto aspektů hra má, tím větší má šanci na úspěch.

Dále je uveden výběr několika herních aspektů a jejich rozbor.

### 2.1.1 „Hloubka" hry a herních mechanismů

Hloubka hry odráží množství herních mechanismů. Ty mají každý svou vlastní hloubku. Zde je vhodné mít na paměti tzv. Bushnellův teorém: „Easy to Learn, Difficult to Master". První část této filozofie herního návrhu říká, že hra musí být jednoduchá na pochopení, tzn., hráč by měl vědět co dělat, jak to dělat, popř. proč to dělat. S tím jde ruku v ruce, tzv. „Pravidlo patnácti minut". Hráč poprvé spustí hru a během patnácti minut hraní už víceméně ví, jestli se mu hra líbí nebo ne. Proto je vhodné udělat dobrý první dojem. Druhá část teorému říká, že hra by měla hráči dát možnost se neustále zlepšovat, či objevovat něco nového. Dobrým příkladem této problematiky je MMORPG World of Warcraft. Samotná hra má ohromné množství herních mechanismů a nového hráče by pravděpodobně odradilo, kdyby měl co do činění se všemi hned od začátku hry, protože by nevěděl, co má dělat. Proto hra hráči servíruje herní obsah postupně. Začíná ve startovní lokaci, kde jsou jednoduché úkoly, slabá monstra a několik NPC postav. Hráč má k dispozici pouze základní útok a jedno kouzlo, či schopnost. V těchto lokacích se naučí ovládat svojí postavu, interagovat s NPC (přijmutí/odevzdání úkolu, obchodování a učení nových kouzel/schopností) a seznámí se základními mechanismy boje a vývoje postavy. Další herní obsah hra podkládá hráči až při dosažení určité úrovně postavy.

Hloubka herního mechanismu by měla být dostatečně velká, aby měl hráč pocit, že mu tato část hry má pořád co nabídnout. Například hraní za určitý typ postavy. Když už hráč nemá v herním světě konkurenci, porazí každého, na koho ukáže, tak je to známka toho, že tato část hry už nemá hráči co nabídnout. Také je potřeba co nejvíce eliminovat jednotvárnost. I sebezábavnější herní mechanismus po chvíli hráče přestane bavit, když je příliš jednotvárný.

### 2.1.2 Cíle hry

Cíle jsou důležitou částí hry. Dávají hráči důvod hru hrát a dodávají mu pocit pokroku. Je důležité zajistit, aby si hráč uvědomoval cíle hry. Cíle mohou mít různou formu, například dokončení příběhu, dosažení maximální úrovně, sjednotit království, či dosáhnout vrcholu v nějakém žebříčku. Dokončení cílů hry nemusí nutně znamenat konec hry. Hráč si může vytyčit vlastní cíle, například najít určitý předmět, nebo předměty, zničit hlavního padoucha za speciálních podmínek, či udělat si sbírku všech předmětů ve hře. Proto je vhodné dát hráči mnoho příležitostí k vytvoření vlastních cílů hry.

### 2.1.3 Křivka úrovní postavy

Tato křivka udává, jak dlouho hráči potrvá, než se dostane na maximální úroveň postavy (pokud nějaká je). Úrovně postavy jsou důležitou metodou v herním designu, jak hráči ukázat dosažený postup ve hře. Je nezbytné, aby měly ve hře nějaký význam, například zvýšení životů za každou úroveň, nová kouzla, vybavení závislé na úrovni postavy atd. Nejvhodnější je definovat tuto křivku pomocí nějaké geometrické funkce, v závislosti na tom, jakou chceme zajistit plynulost a délku hry.

- Lineární postup – na každou úroveň je potřeba stejné množství zkušeností. Hra je stejně svižná v každé fázi, což může vést k jednotvárnosti. Hry s lineárním postupem bývají krátké. To lze vykompenzovat množstvím úrovní, ale to bývá doprovázeno problémy s vybalancováním hry.
- Exponenciální postup – na začátku hry je získávání úrovní nejrychlejší, postupem času se ale tempo zvolňuje a postup na další úroveň se stává stále

větší výzvou pro hráče, což výrazně přispívá k době hraní. Tento způsob získávání úrovní by měl být více odměňován než lineární způsob, aby byl hráč dostatečně motivován k dalšímu postupu.

- Tangenciální postup – nejpomalejší způsob získávání zkušeností. Hráč už od začátku postupuje na další úroveň velmi pomalu a na vyšších úrovních je další postup už opravdová výzva. Tento způsob se využívá ve hrách, kde je neomezené množství úrovní nebo když postup na další úroveň nemá velký význam

### 2.1.4 Postup

Každý hráč potřebuje vidět, jak moc postoupil ve hře a to nejen formou úrovně jeho postavy. Lze například ukázat hráči tzv. XP bar. Ten hráči ukazuje, kolik ještě potřebuje zkušeností na postup na další úroveň. Také lze hráči zobrazit postup v hlavním příběhu hry v procentech. Nebo stačí, aby hráč viděl, o kolik se rozrostlo jeho území, kolik vydělal peněz od začátku hry apod. Jednoduše cokoli, co hráči ukáže, jakou cestu od začátku hry už urazil a co dokázal.

### 2.1.5 Vyváženost

Vyváženost hry je velice složitý herní aspekt, zvláště pak ve hrách více hráčů, kde je několik druhů postav, které proti sobě soupeří. Vyváženost ale zahrnuje i jiné mechanismy, například plnění úkolů a získávání odměn. Je třeba zajistit, aby odměna za úkol odpovídala jeho obtížnosti. Kdyby za obtížné úkoly byla malá odměna, hráči by se nevyplatilo je plnit. A naopak, kdyby za lehké úkoly byla velká odměna, tak by hráče brzo přestalo hraní bavit, protože by mu hra přišla příliš jednoduchá a nudná.

### 2.1.6 Soutěživost

Rivalita mezi hráči výrazně podporuje hraní. Nejvíce je viditelná ve hrách více hráčů, zejména MMORPG, kde se hráči snaží být lepší než ostatní, ať už třeba v počtu vydělaných peněz, v dovednostech v PvE, či PvP, nebo v rychlosti získávání zkušeností, či nacházení extrémně vzácných předmětů. Lze ovšem také podporovat

soutěživost ve hrách pro jednoho hráče. Například hráč zničí těžkého protivníka a pochlubí se svému kamarádovi. Ten ho bude chtít předčít a proto se bude snažit zničit stejně těžkého, nebo ještě těžšího protivníka.

### 2.1.7 Očekávání

Tento aspekt je důležitý, ale i riskantní. Hráč od hraní hry něco očekává, zajímá ho, jakou dostane odměnu za splněný úkol, co se stane po dokončení obtížné série úkolů, co bude ve zlaté truhle bráněné golemem, kterého právě skolil, co bude na konci hry atd. Očekávání hráče podporuje ve hraní, ovšem může s sebou nést i zklamání, když dosažený úspěch nepřinese kýžený výsledek. Při opakovaném zklamání může hráč od hraní snadno upustit.

### 2.1.8 Možnost rozhodování

Dát hráči možnost výběru zvyšuje jeho volnost ve hře a prohlubuje zážitek z hraní. Ať už jsou to maličkosti, jako třeba výběr z více odměn za splněný úkol, či rozhodování dle vlastního přesvědčení, například zabít padoucha, nebo ho ušetřit, nechat si odměnu pro sebe nebo ji rozdělit mezi chudinu. Také lze dát hráči naprostou volnost při plnění úkolů. Kupříkladu hráč dostane za úkol zorat pole. Ovšem jak svůj úkol splní je již čistě na něm, může ho zorat sám ručně, nebo použít tažnou sílu, popřípadě najmout lidi, kteří to udělají za něj. U příběhově založených her se může podle hráčových rozhodnutí různě odvíjet příběh, třeba podle toho, ke které frakci se hráč připojí, koho ušetří a koho ne, který úkol splní a který ne atd.

### 2.1.9 Explorace

Hráči rádi prozkoumávají herní svět, ať už otevřený, či uzavřený. Někdo se jen prochází a užívá si krásu prostředí, jiní chodí po světě a hledají výzvy, které by pokořili. Také lze prozkoumávat svět za účelem zisku, popřípadě hledat skryté místa a předměty, které ještě nebyly objeveny. Explorace je spíše odpočinkový herní aspekt, nabízí hráči odbočení od hlavních cílů hry, např. když je hráč vyčerpán po namáhavé sérii úkolů apod.

### 2.1.10 Příběh

Příběh je velmi silná stránka hry. Může provést hráče celým světem i herním obsahem. Herní příběh lze rozdělit na hlavní část a vedlejší části. Hlavní částí si hráč musí projít, vedlejší části jsou dobrovolné. Například ve hře můžou být frakce, které mají každá svou příběhovou linii. Příběh ve hře nemusí nutně znamenat hlavní účel hraní, stejně tak začátek hry nemusí znamenat začátek příběhu. Hernímu světu lze vytvořit vlastní minulost, tzv. „lore“. Hráči často rádi poznávají minulost světa, ve kterém hrají, a jeho obyvatel. Je to pro ně zajímavý pocit se v průběhu hraní například dozvědět, že ten rolník, kterému pomohl sklidit obilí je vysloužilý vojenský veterán, který kdysi v jedné bitvě svými činy rozhodl o výsledku celé války. Ve hrách více hráčů je také možnost, aby si hráči vymýšleli vlastní příběhy a prožívali je se svými přáteli.

### 2.1.11 Data

Hráči milují data. Rádi si pročítají statistiky a tabulky s informacemi ze svého hraní. Zajímá je každá drobnost, kterou ve hře dokázali. Počet zabitých monster, počet smrtí, množství splněných úkolů, celkové či nejvyšší udělené poškození, počet metrů ujetých na koni nebo pěšky, počet chycených ryb, nejpoužívanější zbraň atd. Proto je dobré zaznamenávat co nejvíce těchto informací a dát je hráči k dispozici.

### 2.1.12 Úkoly

Úkoly, neboli questy, jsou u příběhově založených her nezbytnou součástí. Hráč pomocí úkolů postupuje dále ve hře a odvíjí tak klubko příběhu. Questy můžou být hlavní nebo vedlejší. Splnění všech hlavních úkolů je nutnou podmínkou pro dokončení hry. Plnění vedlejších questů je volitelné, tzn. hru lze dokončit i bez nich. Ovšem vedlejší questy můžou nějakým způsobem ovlivňovat průběh hlavních questů. Například může být příběh rozdělen na kapitoly, z nichž každá má sérii volitelných úkolů a jejich plnění ulehčuje plnění hlavních questů dané kapitoly. To dává hráči možnost regulovat si poměr obtížnosti a doby hraní.

Quest, jako prvek, se dá definovat třemi faktory:

- Typ – nejdůležitější faktor. Je dobré, aby těchto typu bylo co nejvíce a to jak samostatných, tak i kombinací z několika typů. Základní typy můžou být například: zabít někoho, přinést něco, doručit zprávu někomu, dojít někam, najít něco, atd.
- Obtížnost - určuje míru výzvy pro hráče. Může být vyjádřena například doporučenou úrovní postavy hráče, či doporučeným počtem hráčů pro splnění úkolu.
- Doba plnění – určuje průměrnou dobu, kterou hráč stráví plněním questu.

### 2.1.13 Odměny

Odměny jsou důležitý motivační faktor pro hráče. Měly by ho podporovat v každé části hry, ať už to je plnění úkolů, zabíjení monster, či dělání nějaké profese. Pokud daná činnost nevede k žádné odměně, tak není žádný důvod, aby jí hráč konal. Nemusí jít pouze o hmotné odměny, jako odměnu lze brát i to, že hráč získá zkušenosti, nebo že se naučí něco nového vyrábět, zlepší se mu vztah s NPC postavou atd.

### 2.1.14 Risk

Mnoho hráčů rádo ve hře riskuje, hledají stále větší výzvy a objevují své herní limity. Proto by hra měla obsahovat příležitosti k otestování hráčových schopností. To lze realizovat více způsoby:

- Výběr obtížnosti
    - Na začátku hry - hráč si před začátkem hry zvolí obtížnost, na kterou se cítí. Každá obtížnost by měla obsahovat základní popis, aby hráč věděl, čím se od sebe liší a snáze si vybral.
    - Dynamicky – používá se, když obtížnost ovlivňuje pouze jednoduché faktory, jako například míru poškození a počet životů monster, nebo maximální hodnotu zdraví hráče, a nemá vliv na složitější mechanismy

jako třeba umělou inteligenci protivníků, nebo questy. Hráč si může obtížnost regulovat podle potřeby kdykoli během hry.

- „Danger zóny“ – herní svět obsahuje místa, která jsou pro hráče extrémně nebezpečná, obsahují velmi silné protivníky ve velkém množství, nejobtížnější questy, ale také největší odměny. Taková místa jsou pro hráče velikou výzvou.

### 2.1.15 Pravděpodobnost úspěchu

Tento aspekt přidává hře na dynamičnosti. Může ovlivňovat téměř každou část hry: Pravděpodobnost vypadnutí předmětu, šance na úspěšné vyrobení předmětu, možnost spuštění nějaké události, apod. Šance může být statická, nebo může být ovlivňována různými faktory. Také se lze rozhodnout, zda bude hráči velikost různých šanci zobrazitelná, či nikoli. V některých případech je to vhodné, v jiných nikoli. Například když jde o šanci na vypadnutí nějakého předmětu z monstra, je vhodné dát hráči vědět jak velká je šance na vypadnutí ze kterého monstra. Taková šance vypovídá o vzácnosti daného předmětu, což funguje zároveň i jako motivace pro hráče. Naopak jsou případy, kdy není vhodné hráče zpravovat o různých šancích. Například výše zmiňovaná PRG hra Mount & Blade je kompletně postavena na různých pravděpodobnostech a herní atmosféře by zcela jistě neprospívalo, kdyby se hráči zobrazovali hlášky typu „Pravděpodobnost vypuknutí války je 46%“ nebo „Pravděpodobnost, že nepřítel zaútočí na toto město, je 22%“. Pokud je nutné hráče informovat o takových věcech, je vhodnější použít sice méně přesné, ale konceptově vhodnější zprávy, například „Naše království se stalo obětí několika provokací a nájezdů ze strany jiného království. Hrozí vypuknutí války.“.

### 2.1.16 Čas

Herní aspekt času může hru učinit více realistickou a také nenásilnou cestou prodlužuje dobu hraní. Například po zadání stavby budovy se budova nepostaví okamžitě, stavba by měla trvat nějakou dobu. Také lze do hráčových schopností a kouzel zavést tzv. „cooldowny“, což je časový interval po použití kouzla, který znemožňuje opětovné užití kouzla po danou dobu. To přiměje hráče přemýšlet nad tím, v jakém pořadí kouzla sesílá.

Zároveň je ale nutné dodržovat dostatečnou svižnost hry, aby se hráč při hraní nenudil přílišným čekáním. Čas by měl být jeden z faktorů, které ovlivňují výši hráčovy odměny za provedenou akci (např. splnění úkolu).

### 2.1.17 Sebevyjádření

Tento aspekt vyjadřuje míru ztotožnění hráče s jeho hrací postavou. Větší sebevyjádření pomáhá hráči vžít se do role své hrací postavy. Nejzákladnější podpora sebevyjádření je možnost zvolit si vlastní jméno herní postavy. Další silný mechanismus jak podpořit sebevyjádření je možnost upravit si vzhled postavy dle vlastního uvážení. U dnešních her, které se nehrají z pohledu první osoby, je tento mechanismus běžným standardem. Většina takových her má svůj více, či méně detailní editor vzhledu postavy. Za zmínku stojí editory vzhledu u her The Sims 3, či TES 5: Skyrim. Také je možné podporovat tento aspekt formou možnosti rozhodování se podle vnitřního přesvědčení hráče. Dát mu možnost rozhodnout se, jestli bude s okolním světem interagovat pozitivně, či negativně. Například zda bude pracovitým farmářem, který si poctivě vydělává na živobytí, či bude loupežníkem, který v lese přepadává nevinné pocestné. Také lze hráči při dialozích nabídnout více odpovědí, aby si mezi nimi vybral tu, která mu vyhovuje nejvíce. Popřípadě může mít hráč možnost splnit úkol různými způsoby. Například hráč dostane mapu k pokladu od starého muže výměnou za slib, že se s ním rozdělí o polovinu kořisti. Na konci úkolu se může hráč rozhodnout, zda se se starcem rozdělí, nebo ho podvede a nechá si celou kořist. Dobrým příkladem tohoto mechanismu je hra Fable: The Lost Chapters. V této hře je přesvědčení jako aktivní prvek ve hře, který ovlivňuje postoj okolního světa vůči hráči. Ve hře jsou předměty a kouzla, ke kterým má přístup pouze zlá, nebo pouze dobrá postava. Také obsahuje mnoho momentů, kdy se hráč rozhoduje mezi tím, co je dobré a tím, co je zlé, co je spravedlivé a co není. Kupříkladu hráč může pomoci strážím při eskortování vůdce banditů na šibenici, nebo může pomoci banditům ho osvobodit.

### 2.1.18 Achievementy

Achievementy (ocenění) vyjadřují dosažené, či dosažitelné úspěchy ve hře. Jejich princip je prostý. Pro jeho splnění je potřeba splnit určitou podmínku. Druhů podmínek může být nesčetné množství a lze jimi pokrýt každou část hry. I přes svůj jednoduchý princip jsou achievementy velice oblíbeným a vyhledávaným herním aspektem, především mezi hráči sběrateli a perfekcionisty, kteří chtějí splnit každou část hry a tudíž i splnit všechny ocenění.

Můžou skládat z jedné nebo více podmínek, popřípadě můžou mít jako podmínku splnění více jiných ocenění. Měly by být rozmanité, více či méně obtížné, také by neměly chybět extrémně složité, či časově náročné achievementy, jako výzva pro hráče. Za dosažení ocenění může hráč dostat odměnu, ale není to pravidlem, jeho získání by pro hráče mělo být dostatečnou odměnou. V případě složitých, či časově náročných achievementů je ale vhodné dát hráči motivaci v podobě odměny. Takové odměny by neměly hráče nijak výrazně zvýhodňovat, vhodnou volbou jsou odměny, které obohacují hráče pouze vizuálně. Například tituly, vzhledově zajímavé části oblečení, či vtipné nebojové předměty a podobně.

### 2.1.19 Sběratelství

Sběratelství je jeden z možných volitelných cílů hry. Hráči sběratelé se snaží získat vše, co hra nabízí. Od všemožných předmětů až po achievementy. Zvláště zaujatí budou nejvzácnějšími předměty a achievementy, které se budou snažit získat za každou cenu, nehledě na to, jak obtížné to je. Ne každý hráč je extrémní sběratel, ale každého hráče do určité míry sběratelství baví. Proto je vhodné hráči nabídnout sběratelský obsah v co nejvyšší míře, s co nejširším intervalem obtížnosti, pro podporu soutěživosti mezi hráči.

*Zdroj:* (Gamification)

## 2.2 Indie hry

Název Indie hra vznikl z anglického výrazu „Independent video game“. Jedná se o nezávislé počítačové hry. Jsou to hry, stvořené jednotlivcem, popřípadě malým týmem, financované pouze ze strany vývojářů. Proto jsou nezávislé – nejsou omezeny žádnými požadavky vydavatele počítačových her, na rozdíl od mainstreamových titulů.

Indie hry bývají obvykle menší než tituly herních společností. Také kvůli nízkému rozpočtu je internet nejvýhodnější a často také jediná forma distribuce nezávislých počítačových her. Vývojáři většinou nemají dostatečné grafické prostředky, proto se spoléhají spíše na herní obsah. Vývoj indie her také zvyšuje možnost rozvoje jednotlivce. Protože nejsou omezeny vydavatelem, bývají tyto hry výsledkem tvůrčí kreativity a experimentování. Přinášejí do herního světa mnoho originálních herních mechanismů, popřípadě i nové herní žánry. Mezi nezávislé počítačové hry patří například legendární sandbox hra Minecraft, Braid, či 7 Days to Die.

Nezávislé hry se oproti titulům herních společností liší také tím, že bývají přístupné veřejnosti už v raných fázích vývoje hry jako tzv. „Alpha“ a „Beta“ verze. Autor hry má již třeba navrženou větší část hry, ale samotný vývoj směřuje tak, aby nejprve implementoval funkční základ (jádro) hry, který sice obsahuje třeba jen zlomek kompletního obsahu, nicméně je samostatně hratelné. Tuto první alpha verzi hry může autor zpřístupnit veřejnosti a buď přímo, nebo jako projekt na platformě Kickstarter (Techradar, 2010).

### 2.2.1 Kickstarter

Kickstarter je platforma pro financování kreativních projektů. Tato služba byla spuštěna 28. dubna 2009, zakladatelé této služby jsou Perry Chen, Yancey Strickler and Charles Adler. Účelem služby je přímé financování projektů lidmi, kteří svými dobrovolnými příspěvky rozhodují o úspěchu projektu. Úspěšnost služby zajišťují její striktní pravidla. Přispívat na projekty lze z celého světa, zakládat projekty lze pouze

ze Spojených Států, Kanady, Spojeného Království, Austrálie a Nového Zélandu. Typ projektu musí spadat pod některou z těchto kategorií: umění, komiksy, tanec, design, móda, film a video, jídlo, hry, hudba, fotografování, publikování, technologie, či divadlo. Autor projektu si při jeho založení zvolí cílovou částku a datum, do které se musí daná suma vybrat. Pokud se do zvoleného data podaří vybrat cílovou částku, je převedena na účet autora projektu. Když se nepodaří vybrat daný finanční obnos, jsou příspěvky vráceny zpět sponzorům a projekt je zrušen. V případě úspěchu je autor projektu povinen projekt realizovat a produkovat určité výstupy (Kickstarter).

Výše zmíněná indie hra 7 Days to Die zahájila svůj vývoj právě díky službě Kickstarter. Během jednoho měsíce lidé přispěli přes 500 000 dolarů na vývoj hry.

## 2.3 Použité technologie a nástroje

Pro objektový návrh bude použito prostředí Enterprise Architect verze 7.1. Hra bude vyvíjena v programovacím jazyce Java, pomocí vývojového prostředí Net Beans. Pro práci s textem je použit MS Word 2007.

# 3 Tvorba konceptu

## 3.1 Základní myšlenka

Děj hry bude zasazen do období venkovského středověku, co se technologické úrovně týče. Pro snadnou orientaci bude herní mapa vyobrazena formou pravidelné mřížky. Hráč bude ovládat pouze jedinou postavu, kterou si vytvoří, a jejím prostřednictvím bude ovlivňovat herní prostředí. Herní svět bude sestávat z vesnic a okolní krajiny (lesy, říčky, louky, atd.). Ve vesnicích žijí osadníci. Na začátku hry je každá vesnice velmi malá, s nízkou populací, silně závislá na obchodu s ostatními vesnicemi, či potulnými obchodníky. Postupem času se vesnice rozrůstají a stávají se více soběstačnými. Primární cíl hry je dovést vybranou vesnici (či vesnice) na vrchol prosperity.

## 3.2 Herní principy

### 3.2.1 Herní svět

Herní svět je definován jako dvojrozměrné pole, v němž každá buňka obsahuje herní políčko. Většina políček je samostatně fungujících, např. strom, cesta, plot. Ve hře je však několik vícepolíčkových objektů, například budovy. Prvotní myšlenka vykreslení herního světa je, že každé herní políčko bude vyjádřeno obrázkem, který bude vyplňovat buňku pole. Postava hráče a neherní postavy se budou pro jednoduchost pohybovat po buňkách, ovšem v budoucím vývoji bude vhodné, aby byl pohyb postav mezi buňkami plynulý.

### 3.2.2 Postava hráče

Hráč si zvolí jméno a pohlaví postavy. Disponuje herními atributy a dovednostmi (rozepsáno níže). Po herní mapě se pohybuje pomocí tlačítka myši. Hráč potřebuje zajišťovat základní životní potřeby (jídlo, pití, spánek). Postupuje na další úrovně díky plnění úkolů, za každý postup dostane několik bodů, kterými si zvýší

některý z atributů. Dovednosti se zvyšují individuálně, a úroveň hráče zatím nijak neovlivňují.

### 3.2.3 Atributy

Ve hře je šest základních atributů.

- **Síla** – zvyšuje maximální výdrž a nosnost.
- **Odolnost** – menší penalizace obnovy výdrže při spaní venku.
- **Charisma** – zvyšuje slevu při obchodování
- **Zručnost** – zvyšuje rychlost výroby předmětů
- **Inteligence** – zvyšuje šanci na naučení bonusového receptu při postupu na vyšší úroveň dovednosti.

### 3.2.4 Dovednosti

- Zkušenosti potřebné na další úroveň se zvyšují exponenciálně.
- Každá dovednost začíná na úrovni 0
- U výrobních dovedností (truhlářství, tkalcovství a vaření) se hráč za každou úroveň dovednosti naučí jeden recept.

#### *3.2.4.1 Dřevorubectví*

Zkušenosti v této dovednosti se získávají za sázení, kácení a ořezávání stromů. Nejprve se strom pokácí sekerou a poté se pilou ořeže. Vyšší dovednost umožňuje rychlejší pokácení a rozřezání stromu a z rozřezání se získá více dřeva a sazeniček.

- 10% rychlejší zpracování stromu/1 bod dřevorubectví
- +0,33 dřeva/1 bod dřevorubectví
- +0,2 sazeničky/1 bod dřevorubectví

*Potřebné předměty:* Sekera, pila

#### *3.2.4.2 Truhlářství*

- Práce se dřevem. Výroba nábytku, stavebního materiálu (trámy, rámy, ploty, atd.) a nástrojů.

Na začátku umí hráč tři základní recepty – obyčejná židle, prkna, dřevěné tyčky. K výrobě je potřeba truhlářský stůl. Recepty jsou rozděleny do dvou kategorií: Materiál a předměty. Do materiálu patří např. prkna, tyčky, trámy, atd. K výrobě předmětů je potřeba materiál, ne surové dřevo. Při výrobním procesu je základní 30% šance, že se jedna výrobní část zničí, tudíž se předmět nepodaří vyrobit. Navíc, pokud se jedna část zničí, u další ze zbylých částí je poloviční šance na zničení a tak dále, dokud se nepřeruší řetězec neúspěchů, nebo nebudou všechny části zničeny.

- -3% základní šance na zničení předmětu/1 bod truhlářství
- +5% kvalita vyrobeného předmětu/1 bod truhlářství

*Potřebné předměty:* pilka, kladivo, pilník

#### *3.2.4.3 Kamenictví*

Schopnost těžit a zpracovávat kámen. Vesnický kameník hráče naučí základy dovednosti, tj. těžit surový kámen a opracovat ho na stavební kostku. Později se hráč může naučit z kamene vyrábět různé stavební materiály, například dlažby. Od úrovně 5 může hráč těžit jíl a vypalovat cihly a tašky. Při výrobě a vypalování je 30% šance na zničení předmětu.

- -5% šance na zničení/1 bod dovednosti kamenictví
- +1 kámen/každý třetí bod dovednosti kamenictví
- +1 jíl/každý třetí bod dovednosti kamenictví (od 5. úrovně)

*Potřebné předměty:* krumpáč, dláto, kladivo

#### *3.2.4.4 Farmářství*

Umění chovat zvířata a pěstovat zeleninu a ovocné stromy. Na začátku umí hráč ručně orat pole a sázet mrkev. Každá zasazená rostlina má základní 60% šanci na uchycení. Zoraná půda ovlivňuje šanci na uchycení rostliny. Dovednost také ovlivňuje šanci na uchycení. Od úrovně 5 se může hráč učit sázet ovocné stromy a od desáté úrovně může hráč začít chovat zvířata. Každé zvíře potřebuje jíst a pít. Ve hře je 5 kvalit půdy. S rostoucí dovedností hráč půdu orá kvalitněji.

- -10%/-5%/+0%/+5%/+10% šance na uchycení za kvalitu půdy
- Zvýšení kvality zorané půdy/2 body dovednosti farmářství
- +5% šance na uchycení/1 bod dovednosti farmářství
- +1 košík ovoce/každý třetí bod dovednosti farmářství (od 5. úrovně)
- +1 vejce, kyblík mléka a maso/každý třetí bod dovednosti farmářství (od 10. úrovně)

*Potřebné předměty:* motyka

#### *3.2.4.5 Vaření*

Schopnost přípravy pokrmů. Na začátku umí hráč tři základní recepty. K vaření je potřeba pec a kamna. Při vaření je základní 20% šance na spálení jídla, tzn. ztrátu všech surovin. Zvyšování dovednosti snižuje šanci na spálení jídla a zvyšuje výslednou kvalitu jídla. Čím větší kvalita jídla, tím více zasytí a také jeho cena je vyšší.

- -2% na spálení jídla/1 bod dovednosti vaření
- +4% kvalita jídla/1 bod dovednosti vaření

#### *3.2.4.6 Tkalcovství*

Umožňuje vyrábět a zpracovávat látky a kůže. Základním výrobním materiálem je vlna a usně. Z usní se vyrábí kůže a z vlny látka a nitě. Základní množství z jedné vlny jsou tři látky, nebo deset nití. Z jedné usně jsou to dvě kůže. Na začátku umí hráč vyrábět kůže, nitě a látky. K výrobě je potřeba tkalcovský stav. Při

výrobním procesu je základní 30% šance, že se jedna výrobní část zničí, tudíž se předmět nepodaří vyrobit. Navíc, pokud se jedna část zničí, u další ze zbylých částí je poloviční šance na zničení a tak dále, dokud se nepřeruší řetězec neúspěchů, nebo nebudou všechny části zničeny.

- -3% základní šance na zničení předmětu/1 bod tkalcovství
- +5% kvalita vyrobeného předmětu/1 bod tkalcovství
- +1 kůže, 1 látka, 3 nitě/1 bod tkalcovství

### 3.2.5 Neherní postavy (NPC)

Lidské neherní postavy vždy náleží k některé z vesnic. Někteří vesničané můžou zastávat funkci řemeslníka a vykonávat pro vesnici jednu z dovedností. Každý vesničan musí mít kde bydlet. Pro prvotní jednoduchost nemají neherní postavy aktivní potřebu jíst, pít, či spát, ovšem tyto činnosti jsou stále zahrnuty v jejich denním harmonogramu, pro zajištění věrohodnosti. Další neherní postavy jsou zvířata. Zatím jsou ve hře pouze chovná zvířata. Chovná zvířata mají potřebu jíst a pít a přítomnost vesničanů, či hráče ignorují. Chovají se pro maso (vepř, kráva, slepice), mléko (kráva), vejce (slepice), vlnu (ovce) a usně (vepř).

### 3.2.6 Vesnice a budovy

Vesnice sestává z budov a obyvatel a postupuje na další úrovně díky splněným úkolům pro danou vesnici (viz sekce Úkoly níže). Pro jednoduchost bude kromě krčmy ve hře pouze jeden typ budovy – obytný dům. Každá budova má svůj vlastní interiér, který lze vybavit podle potřeb řemeslníků. Na začátku hry každá vesnice začíná se čtyřmi budovami: Tři obytné domy a krčma. Krčma je nejdůležitější budova vesnice, vesničané jí často navštěvují, když něco potřebují. Také nabízí možnost stravování a ubytování. Při postupu na novou úroveň vesnice získá 3 vesničany a jednoho řemeslníka a maximální počet budov se zvýší o 2. Nové domy se musí nejprve postavit. Na to je potřeba stavební materiál, který je nutno donést přímo na „staveniště“.

### 3.2.7 Úkoly

Úkoly zadávají hráči sami vesničané. Jsou časově omezené. Za splnění hráč dostane hmotnou odměnu a zkušenosti a vesnice také získá množství zkušeností pro postup na další úroveň.

Hráč má k dispozici seznam úkolů, kde je o všech úkolech jasně vidět, co musí udělat, kdo úkol zadal, do kdy ho musí splnit a jaká bude odměna.

### 3.2.8 Inventář a předměty

Inventářem disponuje jak hráč a neherní postavy, tak i budovy a políčka. U budov a políček je nosnost omezena pouze prostorem, u hráče a neherních postav je omezena prostorem a váhou. Úložný prostor obsahuje 15 (3x5) slotů na předměty.

Každý předmět zabere v inventáři jeden slot. Předměty stejného typu se dají vrstvit na sebe do jednoho slotu. Zároveň má každý svojí váhu, kterou zatěžují postavu. Při přetížení ztrácí postava schopnost pohybu. Dále mají předměty svojí prodejní cenu, která je důležitá při obchodování. Vyrobené nástroje mají svojí kvalitu a životnost. Kvalita vyrobeného nástroje ovlivňuje jeho cenu i životnost. Životnost určuje počet použití, po kterých se daný předmět zničí.

Pro přesun těžkých předmětů slouží vozíky. Ty se dají propojit s budovou pomocí speciálních platforem pro vozíky.

### 3.2.9 Obchodování

Obchodování funguje na principu výměny předmětu za peníze (zlatky, stříbrňáky a měďáky). Hráč může obchodovat s každým řemeslníkem ve vesnici. Ve hře fungují i potulní obchodníci, kteří cestují mezi vesnicemi. Ti, kromě jiného, nabízí i potřebné předměty, které nelze jinou cestou získat (kvůli postupnému vývoji hry), například nerosty apod.

### 3.2.10 Herní čas

Jeden herní den trvá 20 skutečných minut. Když jde hráč spát, hra se automaticky posune na dobu, kdy obnova jeho výdrže dosáhne maxima.

# 4 Objektový návrh

## 4.1 Tvorba analytického modelu

Analytický model byl nejprve vytvářen následujícím postupem: Z výše uvedeného návrhu herního obsahu se vytvořily základní prázdné třídy, poté se mezi nimi vytvářely relevantní vztahy a nakonec se třídy vyplňovaly atributy a metodami. Tento postup se ukázal, alespoň v tomto případě, jako velmi neefektivní, neboť uvažuje hru jako celek a i při relativně malém herním obsahu, jako je výše navržený, vede k velkým dírám v návrhu. Hledání a vyplňování takových děr je časově velmi náročné a proto se od tohoto postupu upustilo a začalo se znovu jiným, organizovanějším postupem.

### 4.1.1 Postup vrstvení herního obsahu

Při tomto postupu se herní obsah rozdělí na více částí. Nejprve se kompletně namodeluje jedna část a potom se k ní přimodeluje další. Je vhodné vytvořit si logické pořadí částí. V tomto případě bylo použito následující pořadí:

#### *4.1.1.1 Herní mapa*

Je tvořena políčky, které jsou naskládané v pravidelné mřížce (dvojrozměrné pole).

#### *4.1.1.2 Políčko*

1. Má svoje id, název, rozměr, obrázek, nabídku interakcí (dynamickou).
    - Musí mít možnost nabídku interakcí měnit.
2. Každé konkrétní políčko má svou pozici v mřížce.
3. Každé políčko má k dispozici inventář.
    - Inventář je kolekce předmětů, které leží na daném políčku.
4. Políčka, která nemají speciální chování, jsou definovány jako statické atributy, zbylé políčka budou potomci třídy Políčko.

- Každé speciální políčko musí znát dobu trvání jednotlivých akcí.

5. Každé políčko má svůj časovač, pro obsluhu akcí.

#### *4.1.1.3 Budovy a interiéry*

1. Budova je složená z více políček.
2. Všechny budovy jsou pro jednoduchost jednopatrové.
3. Budova musí znát svůj interiér.
4. Hlavní část budovy jsou dveře.
    - Musí znát cílový interiér a souřadnice pro přemístění hráče (ven i dovnitř).
5. Zbylé části budovy jsou střecha, stěna a okno.
6. Každá budova má platformu pro vozík, na propojení inventářů.
7. Interiér se skládá z podlahy, stěn a dveří.
    - Podlaha je jediné místo, kam se dá umístit nábytek.

#### *4.1.1.4 Vesnice*

1. Skládá se z budov a obyvatelstva.
2. Zodpovídá za stavbu nových budov.
    - Umí najít vhodné místo pro novou budovu.
3. Má svou úroveň, xp a xp, potřebné na další úroveň.
4. Má omezený počet budov a vesničanů na úroveň.
    - S každou úrovní se limit zvýší.
5. Musí ovládat výši úrovně a xp.

#### *4.1.1.5 Neherní postava (NPC)*

1. Má své jméno a příjmení.
2. Nemají atributy ani dovednosti, namísto toho bude mít každé npc povolání, od kterého bude odvozeno jeho chování.
3. Musí vědět, kde bydlí.
4. Má svou pozici.

5. Má své AI (podle povolání).
6. Umí vygenerovat úkol pro hráče.
7. Zvíře je také neherní postava, ale není u ní potřeba znát tolik informací jako u vesničanů.
    - Pouze název, pozici a typ.

#### *4.1.1.6 Umělá inteligence (AI) a povolání*

1. Má na starost hledání cesty.
2. Definuje všechny možné akce NPC postav.
3. Abstraktní třída Akce, která obsahuje základní funkcionalitu akcí.
    - Každá specifická akce je potomkem třídy Akce.
4. Každá akce musí mít nastavitelnou prioritu.
5. Abstraktní třída AI drží kolekci akcí.
6. Třída AI musí umět najít akci s nejvyšší prioritou.
7. Každé povolání je potomkem třídy AI.
    - Obsahuje množinu akcí, dostupných pro dané povolání.

#### *4.1.1.7 Inventář*

1. Obsahuje kolekci předmětů.
2. Musí umět spočítat a udržet celkovou váhu předmětů v inventáři.
3. Musí umět pracovat s kolekcí (přidávat a odebírat předměty).

#### *4.1.1.8 Předmět*

1. Má svůj název, obrázek v inventáři, maximální množství, nabídku interakcí, kvalitu, cenu a váhu.
2. Třída Předmět má tři potomky, pro lepší odlišení předmětů:
    - Nábytek
    - Nástroj
    - Jídlo

3. Nábytek
    - Kromě obrázku v inventáři musí znát i obrázek na mapě.
    - Pro jednoduchost je prozatím nerozbitný.
    - Rozšiřuje nabídku interakcí políčka.
        i. Společná akce je odebrání nábytku z políčka.
    - Musí jít umístit.
4. Nástroj
    - Má svojí životnost (počet použití), která se odvíjí od kvality.
        i. Při použití se snižuje.
    - Jsou potřeba pro interakci s políčky.
5. Jídlo
    - Dá se konzumovat.
    - Musí vědět, kolik obnoví potřeby jíst/pít.
    - Některé jídlo se dá jíst ve stoje a některé se jí jen u stolu.

#### *4.1.1.9 Vozík*

1. Zná svou pozici.
2. Má svůj inventář.
3. Musí kontrolovat svou pozici, zda nestojí na platformě pro vozík.
4. Obsahuje metody pro připojení/odpojení vozíku k hráči/vesničanovi a při připojení vozíku k budově.

#### *4.1.1.10 Dovednost*

1. Má svůj název, úroveň, xp, xp na další úroveň.
2. Drží kolekci naučených receptů.
3. Zajišťuje učení nových receptů.
4. Musí vědět, kdy postoupit na další úroveň.
5. Každá dovednost má svůj koeficient úspěchu.

#### *4.1.1.11 Recept*

1. Zná svůj název, příslušnou dovednost a čas výroby.
2. Musí znát výsledný předmět.
3. Musí vědět, které nástroje jsou pro výrobu předmětu potřeba.
4. Drží kolekci předmětů, ze kterých se předmět vyrábí.
5. Každý recept má svojí úroveň, podle které se odvíjí množství xp získaných při výrobě předmětu.

#### *4.1.1.12 Hráč a atributy*

1. Má své jméno, pohlaví, pozici, zdraví, výdrž, potřebu jíst, pít.
    - Musí je umět obsluhovat.
2. Dále má svůj inventář.
3. Potřebuje držet přehled o aktivních úkolech.
4. Má atributy:
    - Síla
    - Odolnost
    - Charisma
    - Zručnost
    - Inteligence
5. Musí znát počet nerozdělených bodů do atributů.
6. Má svou rychlost pohybu (počet políček za sekundu), která je ovlivněna aktuální zátěží.
7. Musí umět přepočítat benefity z atributů.
8. Má dovednosti.
9. Musí znát svou úroveň, xp a kolik potřebuje xp na další úroveň.
    - Musí s nimi umět pracovat.
10. Umí se pohybovat.

#### *4.1.1.13 Quest (úkol)*

1. Má svůj název, deadline, cíl, startovní a koncové NPC a popis úkolu.
2. Musí držet odměnu a počet zkušeností za splnění.
3. Musí umět kontrolovat a upravovat cíl.
4. Při naplnění cíle se musí quest splnit aby se dál nekontroloval/neupravoval cíl.

#### *4.1.1.14 Generátor mapy*

1. Funguje na principu poměrů definovaných částí.
    - Musí znát celkový poměr.
2. Části:
    - Tráva
    - Les
    - Řeka
3. Dále si lze zvolit množství vesnic a kamenolomů.
4. Velikost mapy:
    - Malá (64x64 polí)
    - Střední (128x128 polí)
    - Velká (256x256 polí)
5. Hráč si může sám nadefinovat poměry částí, nebo si může nechat vygenerovat plnou náhodnou mapu.

## 4.2 Návrhový model

Návrhový model je zpřesnění analytického modelu do takové podoby, aby z něj bylo možné vygenerovat zdrojový kód pro konkrétní programovací jazyk. Je dobré nejprve zjistit, zda některé třídy nemají příliš mnoho zodpovědnosti. Takové třídy je vhodné rozdělit na více menších tříd. Třída PostavaHrace je případ právě takové třídy. Obsahuje mnoho atributů a mnoho metod pro jejich obsluhu. Proto byla vytvořena třída SpravceAtributu, která má na starost veškerou funkcionalitu, týkající se herních atributů (síla, zručnost, atd.). Drží informace o všech herních atributech, o

benefitech, plynoucích z atributů, množství volných atributových bodů a metody pro práci s nimi, jako například přepočítání benefitů z atributů.

Další třída, která potřebuje pozornost, je třída Ukol. Obsahuje atribut „cil“, který vyjadřuje cíl daného úkolu, což je akce, či množina akcí, které je potřeba vykonat, aby se úkol splnil. Pro takové chování nestačí žádný primitivní datový typ, ani nejsou k dispozici žádné třídy, které by tuto funkcionalitu poskytly. Proto je nutné vytvořit speciální třídu CilUkolu. Ve hře jsou zatím tři typy cíle úkolů – přinést nějaký předmět, či předměty, dále promluvit s někým, a nakonec „udělat něco“ (přesněji řečeno vykonat konkrétní interakci na konkrétním políčku). Proto bude třída CilUkolu abstraktní a bude mít tři potomky, každý bude vyjadřovat jeden typ cíle úkolu. Protože požadovaných předmětů, či vykonaných interakcí může být v úkolu vyžadováno víc, budou v abstraktní třídě zavedeny atributy, vyjadřující cílové a aktuální množství. Dále je třeba udržovat informaci, zda je cíl úkolu splněn. To se zajistí dalším atributem a metodou, která bude porovnávat cílové a aktuální množství. U typu cíle „promluvit s někým“ je potřeba znát NPC, se kterým je třeba promluvit, u typu cíle „přinést něco“ je potřeba znát daný předmět a u typu cíle „udělej něco“ je nutné znát cílové políčko a typ požadované interakce. Aby mohly být úkoly komplexní (obsahovat více typů cílů úkolu), je atribut cil ve třídě Ukol vyjádřen jako jednorozměrné pole typu CilUkolu.

Když už není potřeba rozdělovat, či vytvářet další třídy, může se začít zpřesňovat celý model. Nejprve se doplní správné datové typy u atributů a návratové hodnoty metod. Pak se vytvoří get/set metody ke všem potřebným atributům (ne u každého atributu jsou potřeba oba). Mohou se také doplnit parametry u metod. Dále je potřeba vytvořit konstruktory u tříd (i u těch je možné doplnit parametry). Nakonec je nutné zpřesnit vazby mezi třídami.

# 5 Návrh implementace

Kompletní implementace navržené hry je nad rámec této práce, pozornost bude věnována pouze několika vybraným částem hry.

## 5.1 Generátor náhodných map (bez vesnic)

U generátoru náhodných map je nutné určit pořadí generovaných částí. Následující návrh generátoru vytváří mapu, která je tvořena pěti druhy políček: Tráva, kámen, voda, les (bez stromu) a strom v lese. Nejprve se definují pravidla pro generování, například kámen lze umístit pouze na trávu, řeka teče vždy od okraje mapy apod.

Celé generování probíhá v následujícím pořadí:

1. **Vyplnění všech políček trávou**
2. **Umístění lesů –** Generování lesů je nesložitější část celého generátoru. Z poměru lesů a velikosti mapy se zjistí počet políček, které by měly obsahovat les. Podle velikosti mapy se také určí počet lesů (pro malou velikost mapy, tzn 64x64 políček jsou to 3-4 lesy). Každý les má různou velikost, počet použitých políček se následně odečte od celkového počtu vyhraněných políček. Počet vyhraněných políček je pouze orientační, neboť skrz les může protékat řeka, která nahradí množství lesa vodou.
3. **Umístění stromů do lesů** – zde se prohledají všechny políčka na herní mapě a u každého nalezeného políčka s lesem je 50% šance na umístění stromu.
4. **Tvorba řek** – Řeky z estetických důvodů vytékají pouze z okraje mapy. Podle počátečního okraje je určen směr toku, tzn., pokud řeka vytéká od horního okraje mapy, nemůže téct směrem nahoru. Při tvorbě mapy lze nadefinovat počet řek. Délka řek je ovlivněna velikostí mapy (pro 64x64 mapy je to 40-50 políček vody)
5. **Umístění kamenů –** kameny lze umístit pouze na trávu, takže se náhodně vybere políčko s trávou a na tu se umístí kámen. To se opakuje dokud nejsou umístěny všechny kameny. Počet kamenů se volí při tvorbě mapy.

Zdrojový kód navrženého generátoru je k vidění v přílohách.

## 5.2 Hledání nejkratší cesty

Herní mapa je v podobě dvourozměrného pole, na které se dá pohlížet jako na graf, tudíž lze pro hledání nejkratší cesty použít Dijkstrův algoritmus. Vzdálenost mezi uzly se dá nahradit tzv. průchodností terénu, což znamená, že například po písku se postavy pohybují pomaleji, po vybudovaných cestách naopak rychleji apod. V navržené hře se zatím počítá pouze se dvěma hodnotami průchodnosti – buď je políčko průchozí, či nikoli. Jinými slovy každé políčko, které je průchozí, zvýší vzdálenost od zdrojového políčka o 1. Proto bude algoritmus zjednodušen tak, aby nebylo nutné kontrolovat, zda je nově nalezená cesta do políčka kratší, než předchozí cesta, protože pokud algoritmus zpracovává nejdříve sousedy políček s nejkratší vzdáleností od zdroje, tak taková možnost nikdy nenastane. Algoritmus končí v okamžiku, kdy cílové políčko zná svou vzdálenost od zdrojového políčka.

Navržený algoritmus je v přílohách.

## 5.3 Ukládání a načítání hry

Pro ukládání a načítání instancí se používá rozhraní Serializable. Serializace instancí umožňuje rozložit instanci na posloupnost bajtů, kterou lze uložit do souboru a později načíst. Pro zapisování a čtení dat se používají třídy ObjectOutputStream a ObjectInputStream. Uložit lze jen instance implementující rozhraní Sezializable.

Ve hře je třeba ukládat stav všech políček, inventáře, veškeré informace o postavě hráče, o vesnicích a NPC postavách.

# 6 Výsledky

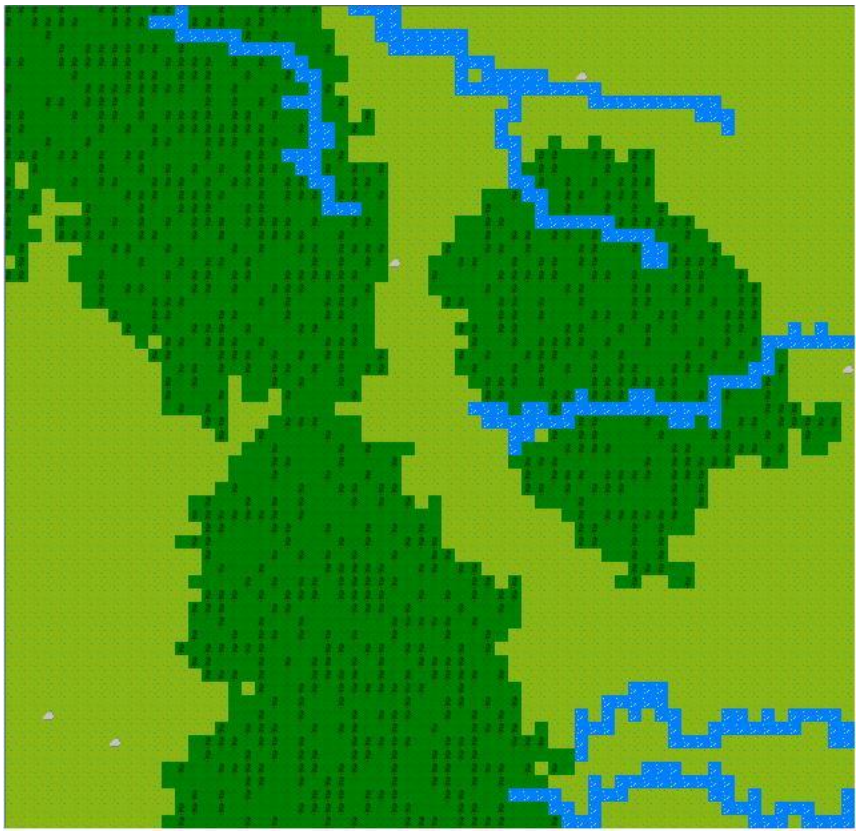
## 6.1 Objektový model

Hra bude nejvíce rozšiřovaná v oblasti políček, předmětů, receptů a umělé inteligence NPC postav. Ačkoli je vztah dědičnosti nejsilnější vazba mezi třídami a mělo by se s ní pracovat opatrně, stále je to velmi pohodlná metoda jak hru rozšiřovat. V případě nového políčka, pokud nevyžaduje žádné speciální chování, jednoduše se vytvoří nová instance třídy Policko, v opačném případě se navrhne nová třída, která zdědí základní chování políčka. Veškeré předměty a recepty mohou být uloženy v podobě statických atributů, popřípadě je lze načítat ze souboru. Každé povolání je potomkem třídy AI, s různými soubory akcí, které dané povolání může provádět. Taktéž každá akce je potomek třídy Akce.

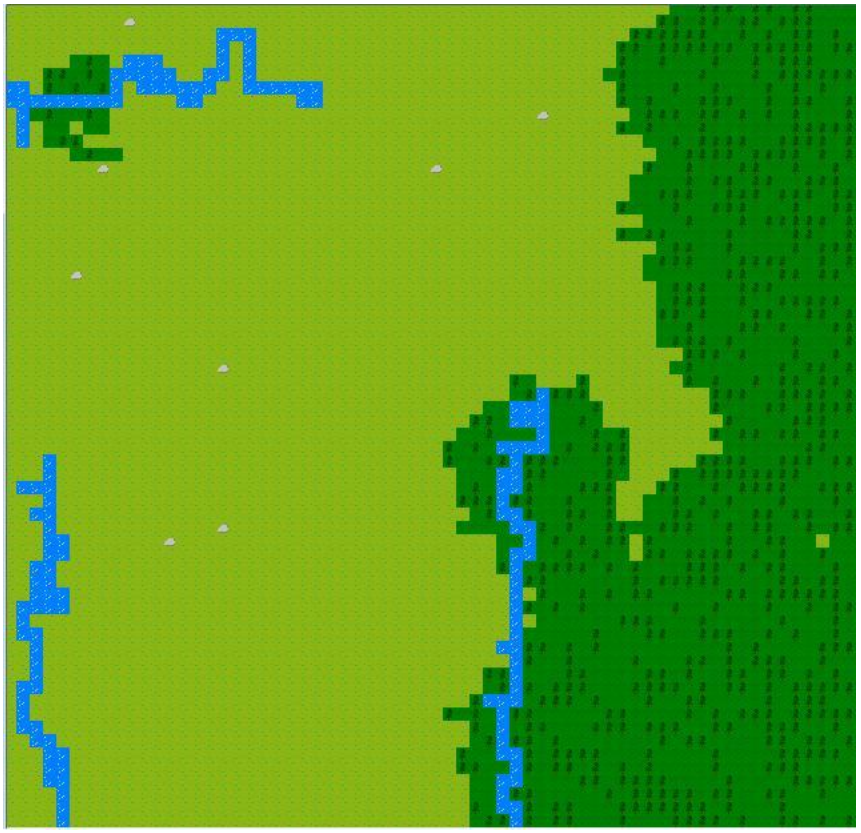
## 6.2 Generátor mapy

Pro lepší kontrolu navrženého generátoru mapy byl implementován jednoduchý zobrazovací algoritmus, který vykresluje vygenerované mapy do JFrame okna. Zde je několik ukázkových map:

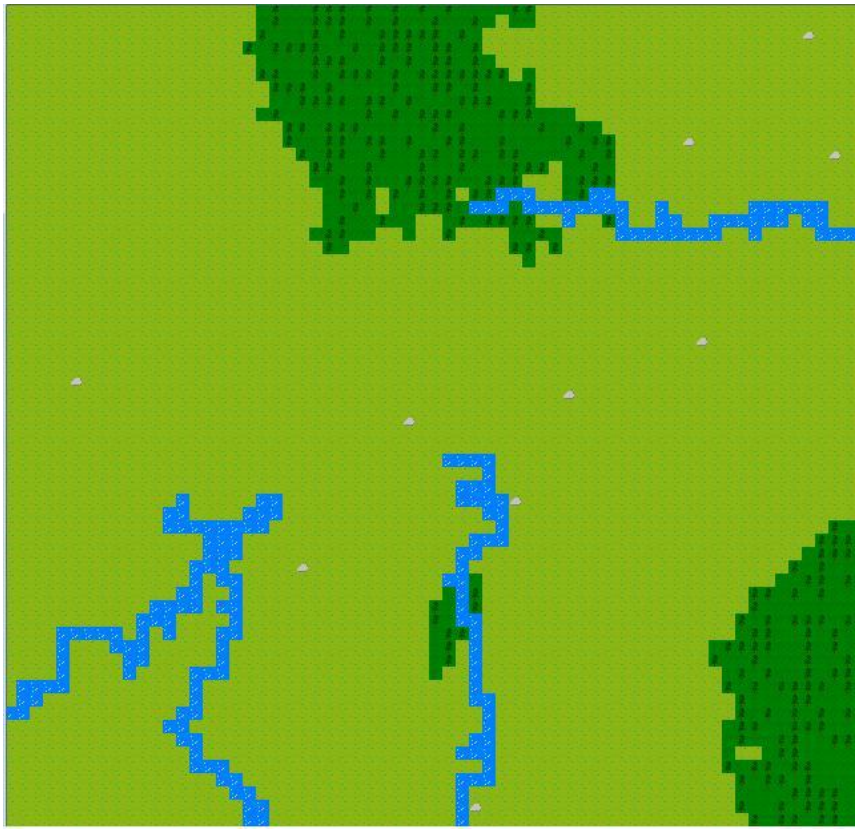
**Poměr lesů: 50%, řeky: 6, kameny: 5**



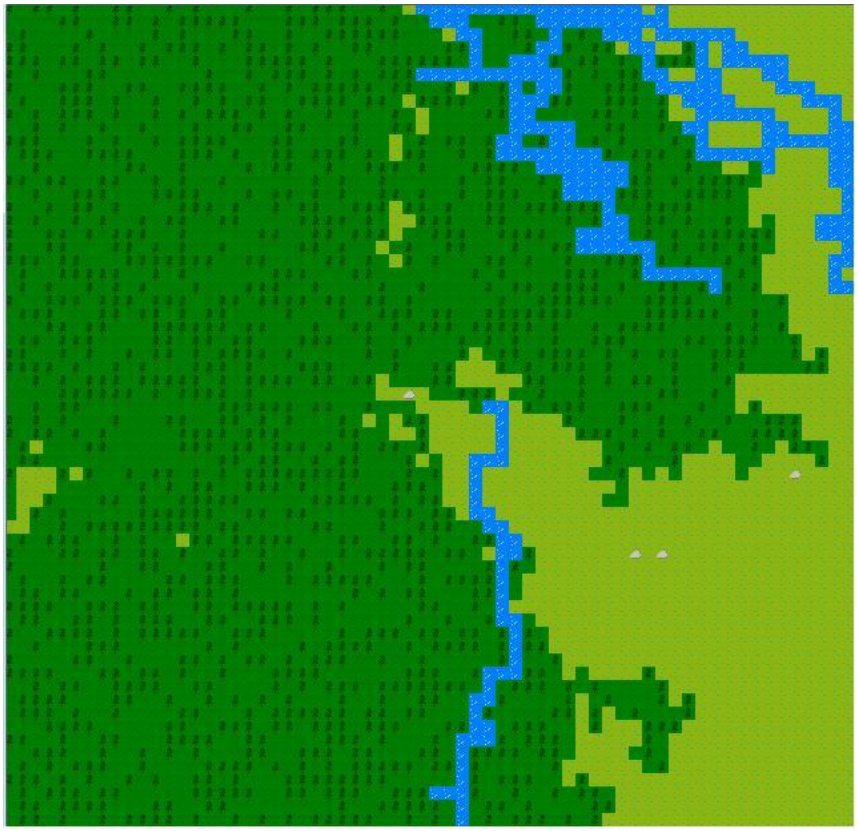
**Poměr lesů: 35%, řeky: 3, kameny: 8**



**Poměr lesů: 15%, řeky: 4, kameny: 10**

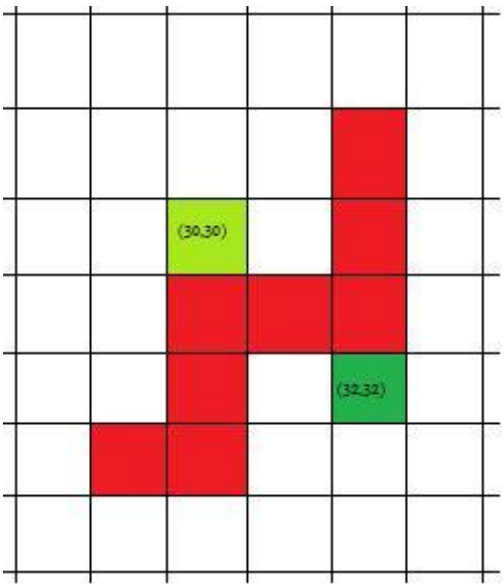


**Poměr lesů: 75%, řeky: 6, kameny: 5**



## 6.3 Algoritmus hledání cesty

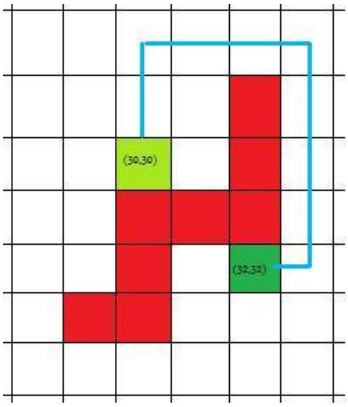
Pro demonstraci algoritmu byla použita následující situace:




Světle zelené políčko je startovní (souřadnice 30,30), tmavě zelené je cílové (souřadnice 32,32). Červeně zbarvená políčka nejsou průchozí. Algoritmus vrací pole souřadnic nalezené cesty:

| | | |
|---|---|---|
| pole | ArrayList<Point> | "size = 10" |
| [0] | Point | [30, 29] |
| [1] | Point | [30, 28] |
| [2] | Point | [31, 28] |
| [3] | Point | [32, 28] |
| [4] | Point | [33, 28] |
| [5] | Point | [33, 29] |
| [6] | Point | [33, 30] |
| [7] | Point | [33, 31] |
| [8] | Point | [33, 32] |
| [9] | Point | [32, 32] |

Na grafické reprezentaci vypadá cesta následovně:



# 7 Závěr

Cílem bakalářské práce bylo vymyslet a navrhnout jednoduchý základ počítačové hry s RPG prvky. Bylo zvoleno ekonomické zaměření hry. Hlavní důraz byl kladen na objektový návrh. Pro tento účel byl nejdůležitější diagram tříd, proto se nejprve vytvořil analytický model a poté podrobný návrhový model, implementačně závislý na programovacím jazyce Java. Následně byla pozornost věnována programování několika částem hry, konkrétně generátoru náhodných map, algoritmu hledání nejkratší cesty a nastínilo se i ukládání a načítání hry.

Možností rozšíření této práce je několik. V první řadě je to kompletní implementace navržené hry. K tomu je potřeba návrh uživatelského rozhraní (GUI), který není v práci obsažen. Také sekvenční diagramy by výrazně přispěly ke kvalitě implementace. Algoritmus pro nalezení nejkratší cesty byl navržen pouze pro demonstraci funkčnosti, hledání cest mezi vzdálenými políčky je příliš zdlouhavé na to, aby byl algoritmus použitelný ve hře. Generátor map je uspokojující, ovšem chybí v něm generování vesnic, které jsou pro hru klíčové. Nakonec je to samotné rozšiřování hry za účelem zvyšování hrací doby. Například více dovedností (rybářství, lovectví), dialogy u NPC postav, dynamicky generované mapy, roční období (a na něm závislé zemědělství), atd. V tomto směru jsou možnosti dalšího výzkumu téměř neomezené.

Téma bakalářské práce jsem si vybral, protože mě velmi zajímá oblast počítačových her, ať už v oblasti hraní, návrhu, či programování. Samotné programování je mi stejně sympatické jako objektový návrh, ale protože žádný větší projekt se bez návrhu neobejde, zaměřil jsem hlavní část práce právě na objektové modelování. Myslím, že mi tato práce poskytla vcelku jasnou představu o obtížnosti tvorby počítačových her.

# 8 Citovaná literatura

**Administrator. 2005.** ZaklSWIOOP. *Výuka programování a softwarového inženýrství na KIT VŠE.* [Online] 5. 6 2005. [Citace: 14. 11 2014.] http://java.vse.cz/pdf/ZaklSWIOOP.pdf.

**Arlow, Jim a Neustadt, Ila. 2011.** *UML 2 a unifikovaný proces vývoje aplikací.* Brno : Computer Press a.s., 2011.

**Bolton, David. 2013.** Three Java IDEs Compared. *Dice.com.* [Online] 24. Říjen 2013. [Citace: 13. Duben 2014.] http://news.dice.com/2013/10/24/three-java-ides-compared-147/.

**Builder.cz. 2001.** C/C++. *Builder.cz.* [Online] 27. Červenec 2001. [Citace: 2. Duben 2014.] http://www.builder.cz/rubriky/c/c--/pretypovani-v-c--155842cz.

**csharp.aspone.cz.** Něco málo o C#. *csharp.aspone.cz.* [Online] [Citace: 10. Duben 2014.] http://csharp.aspone.cz/.

**DataProjekt.** Popis aplikace Enterprise Architect. *DataProjekt.* [Online] [Citace: 26. Březen 2014.] http://www.dataprojekt.cz/content/popis-aplikace-enterprise-architect.

**DiabloWiki. 2014.** Diablo (Game). *Diablo Wiki.* [Online] 20. Leden 2014. [Citace: 20. Březen 2014.] http://diablo.wikia.com/wiki/Diablo_%28game%29.

**—. 2014.** Diablo III. *Diablo Wiki.* [Online] 5. Únor 2014. [Citace: 20. Březen 2014.] http://diablo.wikia.com/wiki/Diablo_III.

**Fabian, David. 2013.** Základy dědičnosti. *Informace, popis jazyka, užitečné rady :: Vše o jazyku C a C++.* [Online] 2013. [Citace: 3. Duben 2014.] http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/poznamky-k-jazyku-c_plus_plus/zaklady-dedicnosti.

**Fišer, Jiří.** Programovací jazyk C++. [Online] [Citace: 1. Duben 2014.] http://ithil.ujep.cz/workspace/cpp.pdf.

**Gamification, Wiki.** Game Design. *Gamification Wiki.* [Online] [Citace: 15. Květen 2014.] http://badgeville.com/wiki/Game_Design.

**Jakob, Michal. 2012.** Umělá inteligence v počítačových hrách (1). *Science World.* [Online] 17. Únor 2012. [Citace: 18. Březen 2014.] http://www.scienceworld.cz/technologie/umela-inteligence-v-pocitacovych-hrach-1-2333/.

**James.** Základní info - Obecně. *Diablo 2 Lord of Destruction | Diablo II.* [Online] [Citace: 20. Březen 2014.] http://www.diablofans.cz/diablo2lod/obecne.html.

**Janáček, Ondřej. 2013.** Verze jazyka C# . *DOTNETPORTAL.cz.* [Online] 28. Září 2013. [Citace: 8. Duben 2014.] http://www.dotnetportal.cz/blogy/12/Ondrej-Janacek/3967/Verze-jazyka-C-.

**Kickstarter.** Kickstarter Basics. *Kickstarter.* [Online] [Citace: 2. Květen 2014.] https://www.kickstarter.com/help/faq/kickstarter%20basics.

**Kremser, Daniel. 2013.** Za oponou Zaklínače 3 - otevřený svět se správnou příchutí . *Doupě.cz.* [Online] 22. Červenec 2013. [Citace: 18. Březen 2014.] http://doupe.zive.cz/clanek/za-oponou-zaklinace-3---otevreny-svet-se-spravnou-prichuti.

**Kubů, Petr.** Úvod do programovacího jazyka Java. *Petr Kubů Homepage.* [Online] [Citace: 6. Duben 2014.] http://kubu.wz.cz/help/java/java.html.

**Mička, Pavel.** Dijkstrův algoritmus. *Algoritmy.net.* [Online] [Citace: 29. 10 2014.] http://www.algoritmy.net/article/5108/Dijkstruv-algoritmus.

**msdn.com. 2014.** Visual Studio. *msdn.com.* [Online] 2014. [Citace: 14. Duben 2014.] http://blogs.msdn.com/b/vyvojari/p/visual-studio.aspx.

**Sillmen, David. 2008.** Mount and Blade - svěží středověký experiment. *Bonusweb.* [Online] 10. Listopad 2008. [Citace: 20. Březen 2014.] http://bonusweb.idnes.cz/mount-and-blade-svezi-stredoveky-experiment-f3g-/Recenze.aspx?c=A081021_132309_bw-pc-recenze_das.

**—. 2011.** The Elder Scrolls V: Skyrim překypuje možnostmi a velikostí. *Bonusweb.* [Online] 18. Listopad 2011. [Citace: 21. Březen 2014.] http://bonusweb.idnes.cz/the-elder-scrolls-v-skyrim-prekypuje-moznostmi-a-velikosti-pqt-/Recenze.aspx?c=A111114_172829_bw-pc-recenze_das.

**Stroustrup, Bjarne. 1997.** International standard for the C++ programming language approved! *Bjarne Stroustrup's homepage.* [Online] 17. Září 1997. [Citace: 1. Duben 2014.] http://www.stroustrup.com/iso_release.html.

**Svoboda, Kamil. 2013.** ..:: Objektové modelování I. ::.. . *..:: Objektové modelování ::.. .* [Online] 17. Září 2013. [Citace: 25. Březen 2014.] http://edu.uhk.cz/~svoboka1/omo1.php.

**Šmíd, Vladimír. 2003.** Objektově orientovaný přístup. *Fakulta Informatiky Masarykovy Univerzity.* [Online] 9. Leden 2003. [Citace: 25. Březen 2014.] http://www.fi.muni.cz/~smid/mis-objekt.htm.

**Taleworlds.** Home. *Taleworlds.* [Online] Taleworlds Entertainment. [Citace: 21. Březen 2014.] https://www.taleworlds.com/.

**Techradar. 2010.** Is indie gaming the future? *Techradar.* [Online] 10. Září 2010. [Citace: 2. Květen 2014.] http://www.techradar.com/news/gaming/is-indie-gaming-the-future--716500.

**VisualStudio.** Porovnání nabídek sady Visual Studio. *Visual Studio.* [Online] [Citace: 2014. Říjen 9.] http://www.visualstudio.com/products/compare-visual-studio-products-vs.

**Wilson, Greg. 2006.** Off With Their HUDs!: Rethinking the Heads-Up Display in Console Game Design. *Gamasutra - The Art & Bussiness of Making Games.* [Online] 3. Únor 2006. [Citace: 5. Květen 2014.] http://www.gamasutra.com/view/feature/2538/off_with_their_huds_rethinking_.php.

**WoWWiki. 2014.** World of Warcraft. *WoWWiki.* [Online] 31. Červenec 2014. [Citace: 22. Březen 2014.] http://www.wowwiki.com/Timeline_%28World_of_Warcraft%29.

**Živě.cz.** NetBeans IDE (verze All) 7.0 final. *Živě.cz.* [Online] [Citace: 11. Duben 2014.] http://www.zive.cz/programy/netbeans-ide-verze-all/pr-7057.

# 9 Přílohy

## 9.1 Příloha A - Generátor mapy

```
public class GeneratorMapy {

private int celkovyPomer = 0;
private int pocetKamenolomu = 0;
private short pocetVesnic = 0;
private int pomerLesu = 0;
private short pocetRek = 0;
private int velikostMapy = 64;
private Policko[][] mapa = null;
private final Random generatorCisel = new Random();

public GeneratorMapy(int pomerLesu, int pocetRek, int
pocetKamenolomu, short pocetVesnic) {

this.pomerLesu = pomerLesu;
this.pocetRek = pocetRek;
this.pocetKamenolomu = pocetKamenolomu;
this.pocetVesnic = pocetVesnic;
}

public Policko[][] vygenerujMapu() {
mapa = vytvorMapu();
vyplnMapuTravou();
umistiLes();
generujStromyVLese();
for (int i = 0; i < pocetRek; i++) {
generujReku();
}
umistiKamenolom();
for (int i = 0; i < pocetVesnic; i++) {
umistiVesnici();
}

return mapa;
}

private Policko[][] vytvorMapu() {
Policko[][] pole = new Policko[velikostMapy][velikostMapy];
for (int x = 0; x < velikostMapy; x++) {
```

```
for (int y = 0; y < velikostMapy; y++) {
pole[x][y] = new Policko();
} }
return pole;
}

public void vyplnMapuTravou() {

for (int x = 0; x < velikostMapy; x++) {
for (int y = 0; y < velikostMapy; y++) {
mapa[x][y] = new Trava();
} } }

public Point vyberNahodnePolicko() {

Point p = new Point();
p.x = generatorCisel.nextInt(velikostMapy);
p.y = generatorCisel.nextInt(velikostMapy);

return p;
}

public void umistiLes() {

Point p;
int velikostLesa = 0;
int maxVelikostLesa;
int pocetPolicekLesa = ((velikostMapy * velikostMapy) / 100)
* pomerLesu;
int maxPocetLesu = 0;
//Počet lesů na mapu
if (velikostMapy == 64) {
maxPocetLesu = 3 + generatorCisel.nextInt(2);
} else if (velikostMapy == 128) {
maxPocetLesu = 6 + generatorCisel.nextInt(5);
} else if (velikostMapy == 256) {
maxPocetLesu = 10 + generatorCisel.nextInt(8);
}
int pocetLesu = maxPocetLesu;
boolean[][] pouzityLes = new
boolean[velikostMapy][velikostMapy];
for (int i = 1; i <= pocetLesu; i++) {
p = vyberNahodnePolicko();
while (mapa[p.x][p.y] instanceof Les) {
```

```
p = vyberNahodnePolicko();
}
//Na vybrané políčko umístí les
mapa[p.x][p.y] = new Les();
//Určení velikosti generovaného lesa
if (i < pocetLesu) {
if (pocetPolicekLesa > pomerLesu) {
maxVelikostLesa = pomerLesu + generatorCisel.nextInt((int)
(pocetPolicekLesa * 0.75));
} else {
maxVelikostLesa = generatorCisel.nextInt((int)
(pocetPolicekLesa * 0.75));
}
pocetPolicekLesa -= maxVelikostLesa;
} else {
maxVelikostLesa = pocetPolicekLesa;
}

//Generování lesa
while (velikostLesa <= maxVelikostLesa) {
for (int x = 0; x < velikostMapy; x++) {
for (int y = 0; y < velikostMapy; y++) {
if (mapa[x][y] instanceof Les && velikostLesa <=
maxVelikostLesa && !pouzityLes[x][y]) {
int cislo = generatorCisel.nextInt(4) + 1;
switch (cislo) {
case 1:
if (x < velikostMapy - 1 && mapa[x + 1][y] instanceof Trava)
{
mapa[x + 1][y] = new Les();
velikostLesa++;
break;
}

case 2:
if (x > 0 && mapa[x - 1][y] instanceof Trava) {
mapa[x - 1][y] = new Les();
velikostLesa++;
break;
}

case 3:
if (y < velikostMapy - 1 && mapa[x][y + 1] instanceof Trava)
{
```

```
mapa[x][y + 1] = new Les();
velikostLesa++;
break;
}

case 4:
if (y > 0 && mapa[x][y - 1] instanceof Trava) {
mapa[x][y - 1] = new Les();
velikostLesa++;
break;
} } } } } }
//Označení vygenerovaného lesa, aby neovlivňoval generování
ostatních
for (int x = 0; x < velikostMapy; x++) {
for (int y = 0; y < velikostMapy; y++) {
if (mapa[x][y] instanceof Les) {
pouzityLes[x][y] = true;
     } } }
velikostLesa = 0;
} }

public void generujStromyVLese() {

for (int x = 0; x < velikostMapy; x++) {
for (int y = 0; y < velikostMapy; y++) {
if (mapa[x][y] instanceof Les) {
//50% šance na vygenerování stromu
int sanceNaStrom = generatorCisel.nextInt(2);
if (sanceNaStrom == 1) {
mapa[x][y] = new Strom();
} } } } }

public void generujReku() {
Point p = vyberNahodnePolicko();
int smer = generatorCisel.nextInt(4);
//Přiřazení správného okraje mapy, dle vybraného směru toku
switch (smer) {
case 0:
p.y = 0;
break;
case 1:
p.y = velikostMapy - 1;
break;
case 2:
```

```
p.x = 0;
break;
case 3:
p.x = velikostMapy - 1;
break;
}

mapa[p.x][p.y] = new Voda();
int poziceX = p.x, poziceY = p.y;
int voda = 0;
int maxDelkaVody = 30;
if (velikostMapy == 64) {
maxDelkaVody = 40 + generatorCisel.nextInt(11);
} else if (velikostMapy == 128) {
maxDelkaVody = 60 + generatorCisel.nextInt(21);
} else if (velikostMapy == 256) {
maxDelkaVody = 120 + generatorCisel.nextInt(31);
}
//generování řeky
while (voda < maxDelkaVody) {
p.x = generatorCisel.nextInt(4);
switch (p.x) {
case 0:
if (smer != 1) {
if (poziceY + 1 < velikostMapy) {
if (!(mapa[poziceX][poziceY + 1] instanceof Voda)) {
voda++;
}
mapa[poziceX][poziceY + 1] = new Voda();
poziceY++;
break;
}
}
case 1:
if (smer != 0) {
if (poziceY - 1 > 0) {
if (!(mapa[poziceX][poziceY - 1] instanceof Voda)) {
voda++;
}
mapa[poziceX][poziceY - 1] = new Voda();
poziceY--;
break;
}
}
```

```
case 2:
if (smer != 3) {
if (poziceX + 1 < velikostMapy) {
if (!(mapa[poziceX + 1][poziceY] instanceof Voda)) {
voda++;
}
mapa[poziceX + 1][poziceY] = new Voda();
poziceX++;
break;
}
}
case 3:
if (smer != 2) {
if (poziceX - 1 > 0 && !(mapa[poziceX - 1][poziceY]
instanceof Voda)) {
if (!(mapa[poziceX - 1][poziceY] instanceof Voda)) {
voda++;
}
mapa[poziceX - 1][poziceY] = new Voda();
poziceX--;
break;
} } } } }

public void umistiKamenolom() {

Point p;
int pocetKamenu = 0;
while (pocetKamenu < pocetKamenolomu) {
p = vyberNahodnePolicko();
if (mapa[p.x][p.y] instanceof Trava) {
mapa[p.x][p.y] = new Kamen();
pocetKamenu++;
} } } }
```

## 9.2 Příloha B – Algoritmus hledání nejkratší cesty

```
public class Dijkstra {

private int velikostMapy = 64;
private Policko[][] mapa;
boolean[][] pruchodnost;
int[][] vzdalenosti;
boolean[][] uzavreno = new
boolean[velikostMapy][velikostMapy];

public Dijkstra(Point start, Point cil) {
mapa = new Policko[velikostMapy][velikostMapy];
for (int i = 0; i < velikostMapy; i++) {
for (int j = 0; j < velikostMapy; j++) {
mapa[i][j] = new Policko();
uzavreno[i][j] = false;
}
}
pruchodnost = new boolean[velikostMapy][velikostMapy];
vzdalenosti = new int[velikostMapy][velikostMapy];
}

public ArrayList<Point> dijkstra(Point startovniPozice, Point
cilovaPozice) {

//Zjištění průchodnosti mapy
for (int i = 0; i < velikostMapy; i++) {
for (int j = 0; j < velikostMapy; j++) {
pruchodnost[i][j] = mapa[i][j].jePruchozi();
}
}

//Nastavení startovní pozice na 0, zbytek na nekonečno
for (int i = 0; i < velikostMapy; i++) {
for (int j = 0; j < velikostMapy; j++) {
if (i != startovniPozice.x && j != startovniPozice.y) {
vzdalenosti[i][j] = -1;
} else {
vzdalenosti[i][j] = 0;
} } }

ArrayList<Point> set = new ArrayList();
```

```
ArrayList<Point> pole = new ArrayList();
ArrayList<Point> kandidati;
set.add(startovniPozice);

//Dokud není ohodnoceno cílové políčko
while (vzdalenosti[cilovaPozice.x][cilovaPozice.y] == -1) {

for (Point p : set) {
if (!(uzavreno[p.x][p.y])) {
kandidati = zkontrolujSousedniPolicka(p);
uzavreno[p.x][p.y] = true;
//Přidání kandidátů do pomocného pole
for (Point o : kandidati) {
pole.add(o);
} } }
set = pole;
pole = new ArrayList();
}

return najdiNejkratsiCestu(cilovaPozice);
}

public ArrayList<Point> najdiNejkratsiCestu(Point cil) {
ArrayList<Point> pole = new ArrayList();
pole.add(0, cil);
int vzdalenost = vzdalenosti[cil.x][cil.y];
Point aktualniPolicko = cil.getLocation();
while (vzdalenost > 1) {
try {
if (vzdalenosti[aktualniPolicko.x + 1][aktualniPolicko.y] ==
vzdalenost - 1) {
pole.add(0, new Point(aktualniPolicko.x + 1,
aktualniPolicko.y));
aktualniPolicko.x++;
} else if (vzdalenosti[aktualniPolicko.x -
1][aktualniPolicko.y] == vzdalenost - 1) {
pole.add(0, new Point(aktualniPolicko.x - 1,
aktualniPolicko.y));
aktualniPolicko.x--;
} else if (vzdalenosti[aktualniPolicko.x][aktualniPolicko.y +
1] == vzdalenost - 1) {
pole.add(0, new Point(aktualniPolicko.x, aktualniPolicko.y +
1));
aktualniPolicko.y++;
```

```
} else if (vzdalenosti[aktualniPolicko.x][aktualniPolicko.y -
1] == vzdalenost - 1) {
pole.add(0, new Point(aktualniPolicko.x, aktualniPolicko.y -
1));
aktualniPolicko.y--;
}
vzdalenost--;
} catch (IndexOutOfBoundsException e) {
}
}
return pole;
}

//Zkontroluje sousední políčka a přidá do seznamu ty, které
jsou průchozí a nezkontrolované
public ArrayList<Point> zkontrolujSousedniPolicka(Point p) {
ArrayList<Point> policka = new ArrayList();
try {
if (pruchodnost[p.x + 1][p.y] && !uzavreno[p.x + 1][p.y]) {
vzdalenosti[p.x + 1][p.y] = vzdalenosti[p.x][p.y] + 1;
policka.add(new Point(p.x + 1, p.y));
}

if (pruchodnost[p.x - 1][p.y] && !uzavreno[p.x - 1][p.y]) {
vzdalenosti[p.x - 1][p.y] = vzdalenosti[p.x][p.y] + 1;
policka.add(new Point(p.x - 1, p.y));
}

if (pruchodnost[p.x][p.y + 1] && !uzavreno[p.x][p.y + 1]) {
vzdalenosti[p.x][p.y + 1] = vzdalenosti[p.x][p.y] + 1;
policka.add(new Point(p.x, p.y + 1));
}

if (pruchodnost[p.x][p.y - 1] && !uzavreno[p.x][p.y - 1]) {
vzdalenosti[p.x][p.y - 1] = vzdalenosti[p.x][p.y] + 1;
policka.add(new Point(p.x, p.y - 1));
}
} catch (IndexOutOfBoundsException e) {
}

return policka;
}
}
```

## 9.3 Příloha C – Analytický model

Kvůli velikosti modelu je obrázek na následujících slepených listech.