Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky

Tvorba príbehov v počítačových hrách

Bakalárska práca

2017 Tomáš Bakoš

Univerzita Komenského v Bratislave

Fakulta Matematiky, Fyziky a Informatiky

Tvorba príbehov v počítačových hrách

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: 2511 aplikovaná informatika

Školiace pracovisko: Katedra Aplikovanej Informatiky

Školiteľ: RNDr. Jozef Šiška, PhD.

Bratislava 2017 **Tomáš Bakoš**

Abstrakt

Táto bakalárska práca sa venuje najmä plánovaniu, ktorým sa zaoberá odvetvie informatiky, umelá inteligencia. Pojednáva o možnostiach vytvárania zaujímavých príbehov pomocou plánovacieho procesu. Práca navrhuje a vytvára systém, ktorý generuje zaujímavé príbehy do dobrodružnej textovej hry pomocou plánovania. Výsledkom je aplikácia, ktorá generuje vždy iný zaujímavý príbeh do nami implementovanej dobrodružej textovej hry.

Kľúčové slová: *umelá inteligencia, plánovanie, dobrodružná textová hra*

Abstract

This bachelor's thesis is mainly studying planning, which is a subject of a field of computer science, artificial intelligence. The thesis is exploring possibilities of generating interesting stories with the help of the planning process. We are designing and creating a system, which is generating interesting stories for adventure text games with the use of planning. The end product of this thesis is an application, which generates always changing interesting stories for adventure text game implemented by us.

Keywords: *artificial intelligence, planning, adventure text game*

Obsah

Úvod

Príbehy nás sprevádzajú celým životom, či už sú to naše vlastné, našich príbuzných, kamarátov alebo ľudí ktorých ani nepoznáme. Od malička ich čítame, stretávame sa s nim ai vyhľadávame ich. Z vlastnej skúsenosti viem, že človeka dokážu pohltiť, primeť ho k zamysľeniu alebo jednoducho si pri nich človek oddýchne. Čo ich ale robí tak zaujímavými?

Už starovekí gréci sa zaoberali písaním a štúdiom tvorby príbehov. Z naratológie, ktorej históriu datujeme až ku starovekým grékom vieme, že príbeh by mal mať nejakú dejovú štruktúru aby nás zaujal a príbeh by nemal mať veľa hlavných postáv, potom by sa mohlo stať že sa čitateľ v príbehu stratí. Navyše tvorba príbehov je komplexný proces, pri ktorom my ludia veľa vecí robíme tak nejak automaticky.

Zaujímavá problematika nastáva, keď túto tvorbu príbehov dáme za úlohu počítaču. Keďže problém je komplexný a navyše ťažko definovateľný (veľa procesov robíme v hlave automaticky), vzniká pre programátora netriviálna úloha ako sa s týmto problémom popasovať. Takýmito úlohami sa zaoberá vedná disciplnína výpočtová naratológia, ktorá študuje tvorbu príbehov z výpočtového hľadiska. Zameriava sa na algoritmické procesy ktoré vytváraju a interpretujú príbehy a tiež na modelovanie štruktúry príbehu z hľadiska vypočítateľných reprezentácií.

Počítačové hry sú mojou záľubou, ktorá ma sprevádza už odmalička a teda mi bolo prirodzené zasadiť problematiku tvorby príbehov do počítačovej hry. Hra príbeh zinterktívni a autori príbehu majú k dispozícii viacero nástrojov ako hráča viac vtiahnuť do deja. Môžu použiť rôzne obrázky, zvuky, animácie, ktoré príbeh dotvárajú a vytvárajú určitú atmosféru. Kombináciou tvorby príbehov a tvorby hier si myslím že vzniká zaujímavá téma na študovanie.

Cieľom tejto práce je teda vytvoriť systém, ktorého súčasťou bude jednoduchá textová hra, ktorý bude generovať zaujímavý príbeh. Tento príbeh sa zakaždým

vygeneruje nanovo, teda hru bude možné hrať znovu a znovu vždy s iným príbehom. Keďže ide o hru tak posun v príbehu bude záležať od interakcie hráča, ktorý hru hrá. Ďalším aspektom je zaujímavosť príbehu, to znamená, že budeme skúmať dynamiku daného príbehu, čiže či sa nebudú často opakovať tie isté akcie, či akcie budú na seba logicky nadväzovať alebo či budú zmysluplné z hľadiska deju atď.

V prvej kapitole práce si popíšeme základné pojmy týkajúce sa našej práce. Spomenieme tiež základy výpočtovej naratológie a jej systémy, ktoré sa používajú na generovanie príbehov. Budú tu popísané aj metódy, ktoré využijeme v našej práci.

Druhou kapitolou je špecifikácia nášeho systému, kde popíšeme čo bude naša aplikácia spĺňať.

Potom v tretej kapitole navrhneme riešenie špecifikácie, teda akým spôsobom budeme riešiť úlohy opísané v špecifikácii.

Štvrtou kapitolou bude implementácia, ktorá bude pojenávať o konkrétnych riešeniach navrhnutých úloh. Spomenieme tu aj rozdiely, ktoré nastali oproti návrhu systému.

Na koniec v piatej kapitole demoštruujeme výsledky našej aplikácie a ukážeme si aj nejaké štatistiky spojené s generovaním príbehov.

1. Východiská

Táto kapitola slúži na popísanie teórie, poznatkov a metód, ktorými sme sa inšpirovali alebo nám pomohli pri vypracovávaní témy zadania.

1.1 Inšpirácia

Už dlhší čas som mal chuť vytvoriť nejakú počítačovú hru. Rád som v pozícii hráča a tak som si chcel aj vyskúšať pohľad na hry z tej druhej strany a vyskúšať si aspoň v malom merítku s akými problémami sa vývojari stretávajú. V poslednej dobe ma začala zaujímať téma umelej inteligencie a tak téma tejto práce bola príjemným skĺbením mojich dvoch záujmov.

1.2 Plánovanie

Plánovanie je problém alebo oblasť, ktorou sa zaoberá odvetvie informatiky. Týmto odvetvím je umelá inteligencia. Je to proces hľadania postupnosti akcií, ktoré sa majú vykonať aby sa dosiahlo vytýčeného cieľa. Je základným stavebným kameňom v problematike zadanej témy a teda upresníme nejaké základné definicie z tejto oblasti.

1.2.1 Fluent

Fluent je vo všeobecnosti niečo, čo dokáže plynúť ako tekutina[11]. V našom ponímaní je to objekt, ktorý opisuje či niečo platí alebo nie a je schopný sa meniť časom. Je to niečo podobné ako predikáty z prvorádovej logiky.

1.2.2 Stav

Stav je definícia, opis sveta alebo objektu s ktorým pracujeme. Príbeh bude vlastne tvorený zmenou z jedného stavu herného sveta do druhého. Tieto zmeny sa uskutočňujú

pomocou akcií.

1.2.3 Akcia

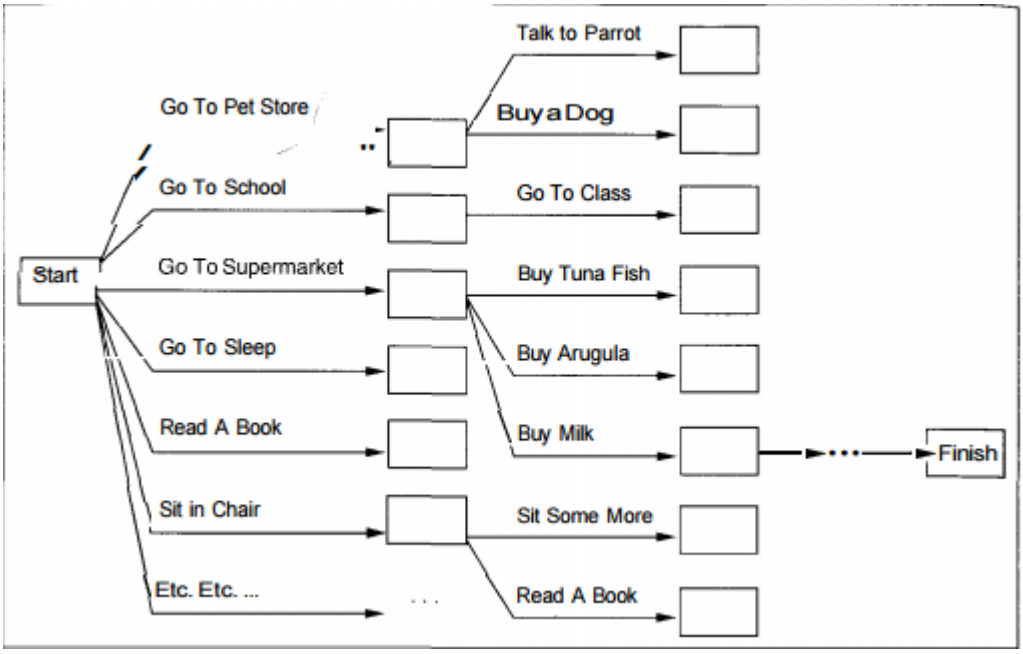
Akcia obsahuje predpoklady a dôsledky. Predpoklady sú nejaké podmienky alebo stav, ktorý musí byť splnený aby sa daná akcia mohla vykonať. Dôsledky na druhej strane sú efekty, ktoré ovplyvnia aktuálny stav sveta alebo iného objektu na ktorom sa akcia vykonáva. Pomocou akcií sa bude posúvať príbeh a vyvýjať postavy.

1.2.4 Riešenie

Riešenie je konečným vysledkom plánovania. Dostávame sa k nemu zadefinovaním počiatočného a koncového stavu (cieľa) a nasledným prechádzaním akcií, ktoré možno vykonať až kým nedosiahneme cieľa. Teda riešenie plánovania (plán) je neprerušená postupnosť akcií z počiatočného stavu do cieľa. V našom prípade budú tieto riešenia príbehmi.

1.2.5 Reprezentácia vzťahov

Jedna z reprezentácii ako definovať vzťahy medzi týmito pojmami sú grafové štruktúry. My na túto reprezentáciu využijeme strom.



Obr. 1.1: Ukážka reprezentácie stavov a akcií [11]

Na obrázku 1.1 môžeme vidieť stavy definované ako vrcholy grafu a akcie ako hrany medzi nimi. Máme počiatočný stav Start a koncový stav Finish. V tomto prípade je našim cieľom získanie mlieka. A teda v stave Start už máme definované fluenty a počiatočný stav sveta. Potom vhodným grafovým prehľadávacím algoritmom vieme hľadať postupnosť akcií a stavov pomocou, ktorých sa dostaneme z počiatočného stavu do koncového. Teda riešenie tohto plánovacieho problému je vlastne cesta v grafe začínajúca vo vrchole Start a končiaca vo vrchole Finish. Táto cesta nám teda hovorí, že na dosiahnutie nášho cieľa (získanie mlieka), musíme ísť do supermarketu a v ňom kúpiť mlieko. Riešení by mohlo byť aj viac ak by sme mali iný graf alebo doplnili do tohto nové akcie a stavy.

1.3 Výpočtová naratológia

Naratológiu ako pojem prvý krát zaviedol Tzvetan Todorov, bulharsko-francúzsky filozof, historik, sociológ a literárny kritik. Je to humanitná disciplína, ktorá sa zaoberá štúdiom narácie, teda príbehu alebo postupnosti udalostí [1]. Ďalej skúma štruktúru, logiku, princípy a tiež aj reprezentáciu narácie. Zo začiatku dominovali štrukturálne prístupy štúdia z ktorých sa vyvynuli rôzne teórie, koncepty a analytické procedúry. Tieto koncepty a modely sú používané ako heuristické nástroje a naratologické teórie hrajú kľúčovú roľu v našej schopnosti vytvárať a spracovávať narácie vo všetkých možných formách. Čo nás privádza k umelej inteligencii a výpočtovej naratológii.

Výpočtová naratológia študuje tvorbu príbehov z pohľadu vypočítateľnosti a spracovávania informácií. Zameriava sa na algoritmické procesy, ktoré vytvárajú a interpretujú príbehy a tiež na modelovanie štruktúry príbehu z hľadiska vypočítateľných reprezentácií. Sem patria aj spôsoby automatickej interpretácie a tvorby príbehov, daľej aj prístupy k rozprávaniu príbehov pomocou umelej inteligencie v hrách. Teda výskumníci sa snažili vytvoriť systémy umelej inteligencie, ktoré by rozprávali príbeh ako ľudia a tiež sa pokúšali vytvoriť inteligentné počítačové prostredie na interakciu s naráciami. V rámci vývoja týchto systémov, výskumníci využili princípy z naratológie na vytvorenie výpočtových princípov a vysvetlili prepojenia medzi nimi. Jeden z príncípov bolo využitie naratologického rozdelenia fabuly a sujetu. Kde fabula je zvyčajne charakterizovaná ako prirodzený sled udalostí celého príbehu v

chronologickom poradí. Sujet na druhú stranu je umelecky realizovaná fabula, teda je to konštrukcia epickej alebo dramatickej fabuly. Výpočtová naratológia bola významne ovlplyvnená aj lingvistikou napríklad gramatikami pribehov. Je to rýchlo rozvíjajúce sa odvetvie, hľavne vďaka zvýšenému záujmu o interaktívne hry a príbehy, ktoré sa javia ako živé.

1.4 Systémy generujúce príbehy

Systémy generujúce príbehy vznikali ako odpoveďe na otázky výpočtovej naratológie. Hľadaním všeobecných vypočtových metód, ktoré by sa dali použiť na rôzne druhy narácie sa v 70-tych rokoch 20. storočia upriamila pozornosť na plánovanie. Odvtedy sa toto zameranie veľmi nezmenilo, ale plánovacie techniky sa vylepšili aby mohli poňať obsiahle problémy naratológie.

V problematike plánovania, na pochopenie príbehu je potrebné vyvodenie založené na Aristotelovom ponímaní mýtu, kde príčiny udalostí príbehu a cieľe zapojených postáv su známe. V podstate zrekonštruovať z viet v sujete plán, ktorý reprezentuje súslednosť udalostí, ktoré dokážu počiatočný stav transformovať na cieľový. Takéto systémy, ktoré sa snažia pochopiť príbeh sa nedostali veľmi daleko z troch dôvodov. Po prvé, vyvodenie cieľov zainteresovaných postáv si vyžaduje obrovský priestor na prehľadávanie. Daľej, ľudia využívaju obrovské množstvo znalostí na pochopenie aj tých najjednoduchších príbehov. Príkladom nám môže byť veta od anglického spisovateľa Edwarda Morgana Fostera: "Kráľ zomrel a kráľovná zomrela od žiaľu.", z ktorej nám ľudom je jasné prečo bola kráľovná smutná, avšak definovať takúto dávku zdravého rozumu počítaču je náročné. Po tretie, niektoré aspekty jazyka, ktoré sú pre pochopenie príbehu dôležité, sa tažko formalizujú ako napríklad humor, irónia a iné nepatrné lexikálne prostriedky. Na druhú stranu algoritmy, ktoré využívajú na generovanie príbehu plánovanie pomocou fabuli sa osvedčili oveľa viac aj preto že si autor môže výrazne obmedziť systém. O takýchto algoritmoch si v tejto sekcii povieme viac.

1.4.1 Novel Writer system

Prvý storytellingový systém, ktorý bol zaznamenaný je Novel Writer system[4] od Sheldona Kleina. Vytváral príbehy o vražde na víkendovej oslave. Údajne vedel vygenerovať 2100 slovné príbehy o vražde, s hlbokou myšlienkou za menej ako 19 sekúnd. Ako vstup sa zadal opis sveta v ktorom sa mal príbeh odohrať a aj charakteristika postáv, ktoré vystupovali v príbehu. Vrah a obeť závyseli od charakterových čŕt tiež špecifikovaných ako vstup. Všetky možné motívy na vraždu boli obmedzené iba na chamtivosť, zlosť, žiarlivosť a strach. Príbeh bol vygenerovaný pomocou dvoch algoritmov. Prvým bola množina pravidiel, ktorá obsahovala všetky možné zmeny stavu sveta z akutuálneho stavu do nasledujúceho. Druhým bola postupnosť scén ktorá korešpondovala s typom príbehu, ktorý mal byť prerozprávaný. Nevýhodou bolo, že množina pravidiel vysoko obmedzovala typ príbehu, ktorý sa vedel generovať. Rozdiely v príbehu spočívali len v tom, kto koho zabil s čím a prečo a v tom kto objavil mrtvolu.

1.4.2 Talespin

Neskôr bol vyvinutý systém TALESPIN[6], ktorý generoval príbehy o lesných tvoroch. Na tvorbu príbehu sa postave zadal cieľ alebo zámer a následne sa zkonštruoval plán ktorým bol dosiahnutý daný cieľ. TALESPIN ako prvý využil charakterové cieľe ako spúšťače akcií. Taktiež tu bola po prvý krát využitá možnosť viacerých takýchto postáv, kde každá mala svoj zoznam cieľov, ktoré chce dosiahnuť. Čo dalo možnosť vymodelovať zložitejšie vzťahy medzi postavami, napr. sútaživosť, dominancia alebo vernosť. Tieto vzťahy potom slúžili ako predpoklady pre niektoré akcie a ako dôsledky pre ostatné akcie. Toto predstavovalo jednoduchý model motivácie postáv. Nakoniec sa vygenerovali charakterové črty postáv pomocou rôznych hodôt láskavosti, samoľúbosti, úprimnosti a inteligencie.

1.4.3 Author

Ďalej si povieme o programe AUTHOR[3], ktorý mal simulovať myseľ autora ako vymýšľa príbeh. Premisa tohto programu bola, že prostredie príbehu je vytvorené zdôvodnením udalostí pre ktoré sa autor už rozhodol, že budú súčasťou príbehu. Autor

sám môže mať, pri vytváraní príbehu, nejaké cieľe, ktoré priíbehom chce dosiahnuť. A aj ak nemá tak je všeobecne známe, že pri vytváraní príbehu existujú nejaké podvedomé obmedzenia a cieľe tvorivého procesu. Pod týmto sa má na mysli napríklad, či je príbeh konzistentný, či je vierohodný, či postavy v príbehu sú uveriteľné alebo aj či pozornosť čitateľa je udržiavaná počas celého príbehu. Toto sa môže preniesť na nejaké podcieľe situácií v príbehu do ktorých chce autor zaviesť niektoré postavy z príbehu alebo do rolí ktoré jednotlivé postavy majú zahrať v príbehu. Môžme teda povedať, že príbeh sa dá chápať ako nejaký výsledok zložitej spleti autorových zámerov. Tieto zámery potom prispievajú ku štrukturovaniu príbehu a nejako riadia proces budovania príbehu. Avšak vo výslednom diele už niesu vidno.

1.4.4 Universe

O 2 roky neskôr prišiel systém UNIVERSE[5], ktorý generoval scenáre pre radu epízód do telenovely v ktorej hralo rolu veľa postáv a plynulo viaceo súčastne prebiehajúcich a prekrývajúcich sa dejov. UNIVERSE bol prvým storytelingovým systémom, ktorý venoval špecialnu pozornosť na proces tvorby postáv. Komplexné dátové štruktúry boli použité na reprezentáciu postáv. Spolu s nimi bol použitý aj jednoduchý algoritmus, ktorý mal tieto štruktúry naplnit. Vačšina však ostala na používateľovi. Tento systém bol skôr prostriedkom na prieskum sériového generovania príbehu ako na generovanie príbehu so začiatkom a koncom. Pôvodne bol zamýšľaný ako pomôcka spisovateľom s nádejou na postupnú tranformáciu na samostaný systém generovania príbehu. UNIVERESE priblížil otázku týkajúcu sa vytvárania fiktívneho sveta. Či by sa najprv mal vybudovať svet a potom do sveta vložit príbeh alebo či by príbeh mal riadiť konštruovanie sveta, kde by sa postavy, lokality a rôzne objeky vytvárali podľa potreby. Tu nastáva rozdiel medzi UNIVERSE-om a AUTHOR-om. Tvorca systému UNIVERSE sa priklonil ku prvej moznosti a implementoval vytváranie postáv nezávisle od deja a na druhej strane bol tvorca AUTHOR-u ktorý sa priklánal k druhej možnosti.

1.4.5 Minstrel

Ďalší v chronologickom poradí, ktorý si spomenieme je počítačový program MINSTREL[12]. Tento program vytváral príbehy o Kráľovi Artušovi a jeho rytieroch okrúhleho stola. MINSTREL používal stavebné jednotky pozostávajúce z cieľov a

plánov ako dosiahnuť dané cieľe. Tieto fungovali na dvoch rôznych úrovniach: autorské cieľe a cieľe vystupujúcich postáv. Vytváranie príbehu fungovalo na základe procesu, ktorý mal dve štádiá, ktoré zahrnovali najprv plánovacie štádium a potom štádium riešenia problémov, ktoré navyše opätovne využívalo vedomosti z predošlých príbehov. Každý beh programu bol založený na nejakej vete, ktorá slúžila ako jadro okolo, ktorého sa píbeh staval. MINSTREL vytváral príbehy dlhé pól strany až jedna strana.

1.4.6 Mexica

MEXICA[9] je počítačový model, ktorý bol vytvorený na skúmanie kreatívneho procesu tvorby príbehov. Jeho úlohou bolo vytvárať krátke príbehy o pôvodných obyvateľoch Mexica. Autor tohto modelu sa riadil hypotézou tvorby príbehu na základe dvoch stavov. Prvý je stav, kedy spisovateľ je silno angažovaný do produkcie materiálu pre dané dielo. V tomto stave sa spisovateľ spolieha na predošlé skúsenosti iné mentálne schémy a zároveň sa vyhýba úmyselnému plánovaniu a rôznym dejovým štruktúram na vývoj diela. V druhom stave spisovateľ analyzuje a hodnotí to, čo doteraz napísal a uvážlivo to skúma a pozmieňa. A teda táto hypotéza hovorí, že cyklus medzi tymito dvoma stavmi tvorí dôležitú časť tvorby príbehu.

Podobne teda bol aj zkonštruovaný program tohto modelu. Pozostáva z dvoch hlavých súčastí. Prvá je založená na množine predchádzajúcich príbehov, ktoré sú definované používateľom pomocou textového dokumentu. Vytvorí sa skupina dátových štruktúr, ktoré reprezentujú abstraktné schémy príbehu. V druhej časti sa program odkazuje na štruktúry vytvorené v prvej časti a program prechádza dvoma stavmi z udanej hypotézi na vytvorenie príbehu. Tento model bol prevratný v tom, že využíval na pohánanie a hodnotenie príbehu aj emocionálne prepojenia a napätia medzi postavami.

1.4.7 Brutus

Ďalším zaujímavým programom, ktorý si spomenieme je BRUTUS[2]. Tento program vytváral krátke príbehy o zrade. Zaujímavý bol v tom, že na tvorbu príbehu využíval logický model zrady. Tento model bol dosť rozsiahly na to, aby sa z neho dali vyvodzovať informácie, ktoré umožnovali tvorbu kvetnatých príbehov. Bol navrhnutý tak, aby vedel brať do úvahy aj veľké množstvo vedomostí o literatúre a gramatike.

Príbehy, ktoré tento program vytváral obsahovali vlastnosti, aké by sme mohli nájsť v príbehoch vytvorených ľudmi napr. rôzne literárne trópy a dialógy. Tvorcovia programu ale podotkli, že BRUTUS vôbec nie je tvorivý program je to skôr výsledok reverzného inžinierstva príbehu, na ktorom potom sledovali, či dokáže vybudovať daný príbeh. Dalo by sa povedať, že to bolo skĺbenie kreativity a logiky.

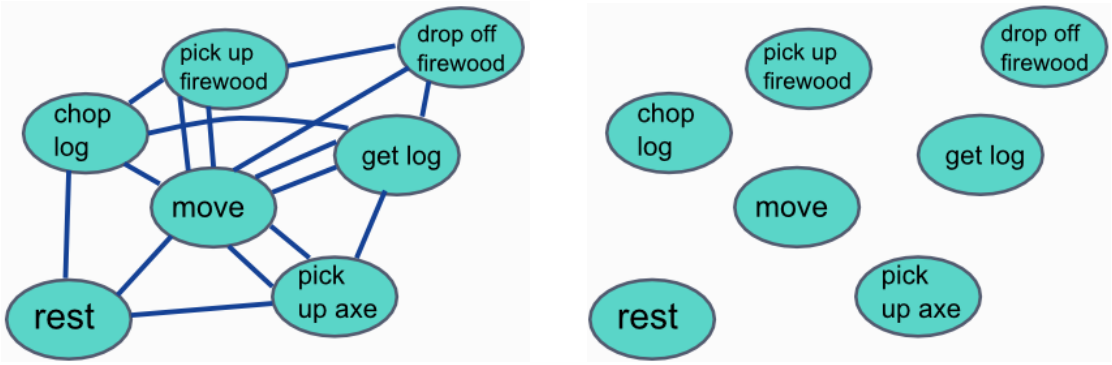
1.4.8 Fabulist

Najnovším spomedzi spomínaných algoritmov je architektúra FABULIST[10], ktorá automatizuje generovanie príbehu a jeho prezentáciu. Využíva techniku takzvaného upresňujúceho vyhľadávania na riešenie problému ohľadom generovania príbehu a na nájdenie zvučnej a uveriteľnej postupnosti akcií postáv, ktorá teda počiatočný stav pretransformuje na konečný. Túto techniku využíva IPOCL (Intent-based Partial Order Causal Link) plánovač, ktorý priniesli v tejto architektúre, ktorý okrem vytvorenia bežne znejúceho deja aj pojednáva o zámeroch postáv v príbehu identifikovaním možných cieľov pre danú postavu, ktoré vysvetľujú spravanie postavy a následne vytvára plánovacie štruktúry, ktoré vysvetľujú prečo sa tieto postavy zameriavajú na ich cieľ.

1.5 Goal Oriented Action Planning

Goal Oriented Action Planning (GOAP) teda cieľovo orientované plánovanie akcii, je systém umelej inteligencie, ktorý ľahko zabezpečí možnosti a nástroje na rozumné rozhodovanie sa pre agentov bez toho, aby sme museli udržiavať veľký a zložitý stavový automat. Umožňuje agentom naplánovať postupnosť akcií, ktoré dosiahnu ich cieľ. Táto postupnosť nezávisí len od cieľa ale aj od aktuálneho stavu sveta a stavu agenta. To znamená, že ak by sme dali rovnaký cieľ dvom rôznym agentom alebo dvom agentom v rôznom stave sveta, môžeme dostať úplne odlišné postupnosti akcií, čo robí umelú inteligenciu viac dynamickú a realistickú.

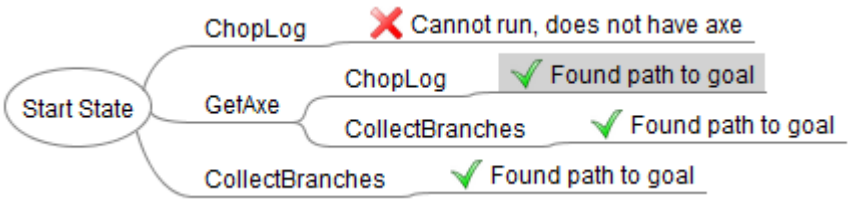
Tento systém vyniká v situáciach, kedy by nám vznikali pri riešení problému veľké a komplikované stavové automaty, kde sa často stáva, že so dôjde do bodu, kedy už nechceme pridávať nové akcie lebo by to vyvolalo priveľa vedľajších efektov a medzier v umelej inteligencii.



Obr. 1.2: Rozdiel medzi reprezentáciou pomocou stavového automatu a GOAP-u[8]

GOAP teda umožnuje z ľavej časti obrázka 1.2 spraviť pravú, kde sú akcie samostatnou jednotkou a nezaoberajú sa vedľajšími efektami svojej činnosti. Tým, že sme jednotlivé akcie odpojili, tak sa môžme venovať každej zvlášť. Toto robí kód viac modulárnym a ľahšie sa testuje a udržiava. Ak by sme chceli pridať novú akciu, tak ju nám stačí pridať do našej reprezentácie a nijako to neovplyvní ostatné akcie, čo sa o reprezentácii stavovým automatom povedať nedá. Navyše rôzne akcie môžeme agentom pridávať a odoberať za behu. Aby sme pomohli GOAP-u zistiť aké akcie chceme vykonať, každá akcia dostane svoju cenu. Akciu s vysokou cenou nevyberieme namiesto akcie s cenou nižšou. Keď akcie usporiadame do postupnosti, sčítame hodnoty jednotlivých akcii, aby sme nakoniec mohli vybrať postupnosť s najnižšou cenou.

O toto sa stará GOAP plánovač, ktorý sa pozerá na predpoklady a dôsledky akcii a vytvára frontu akcii, ktoré spĺnajú náš cieľ. Cieľ je poskytnutý agentom spolu so stavom sveta v ktorom sa agent vyskytuje a zoznamom akcií, ktoré agent môže vykonať. S týmito informáciami potom plánovač môže skúšať vykonávať akcie, uvidí, ktoré sa dajú vykonať a ktoré nie a potom sa rozhodne, ktoré akcie je najlepšie vykonať. Plánovač neskončí hneď ako nájde prvú postupnosť, ktorá nás privedie do cieľa ale prejde aj ostatnými akciami, pretože sa môže stať že existuje iná postupnosť, ktorá má nižšiu cenu ako tá prvá. Teda prejde cez všetky možnosti aby našiel tú najlepšiu. Pri plánovaní sa buduje strom. Vždy keď plánovač použije nejakú akciu, tak ju vyhodí zo zoznamu akcií aby sa nestalo, že sa zacyklíme a chceme vykonávať tú istú akciu dookola. Následne sa zmení stav podľa dôsledku akcie. Keby sme mali za úlohu napr. získať drevo na oheň, tak by plánovací strom mohol vyzerať, tak ako popisuje obrázok 1.3.



Obr. 1.3: Plánovací strom GOAP plánovača [8]

1.6 Predošlé bakalárske práce

Pri uvedení do problematiky a tvorbe bakalárskej práce mi boli nápomocné dve predošlé bakalárske práce. A to Tvorba interaktívnych príbehov ako plánovanie od Martina Zvaru a Tvorba zápletiek v dobrodružných hrách od Dominika Dobiáša.

Prvá spomenutá bakalárska práca je podobná mojej v tom, že tiež využíva plánovanie na generovanie príbehov. Táto práca používa ako základ zformalizovaný Proppov model ktorý je zapísaný ako plánovací problém a naprogramovaný ako logický program pomocou ASP. A na samotné generovanie príbehov bol použitý solver Clingo, ktorý bol spúštaný pomocou Javy. Jeden z rozdielov v našej práci je skutočnosť, že tu sa bude využívať len programovací jazyk Java a nie logické programovanie. Ďalší z rozdielov je, že na riešenie plánovania sa použije vlastný solver. Taktiež v spomenutej práci sa nehľadí na zaujímavosť generovaného príbehu. Touto problematikou sa naša práca zaujímať bude. A nakoniec tvorba príbehov v tejto práci bude zasadená do jednoduchej dobrodružnej textovej hry, kdežto v práci od pána Zvaru išlo čisto o program, ktorý generuje príbehy.

Druhá bakalárska práca, ktorú spomíname je našej viac podobná ako prvá spomenutá. Práca od pána Dobiáša taktiež rieši problematiku zaujímavosti príbehu a podobne ako naša práca tiež implementovala vlastný solver na generovanie príbehov. K rozdieľom patrí nepoužité GOAP plánovanie v práci pána Dobiáša a tiež vygenerovaný príbeh nebol vyjadrený vo forme hry.

2. Špecifikácia

V nasledovnej kapitole si popíšeme, čo by mala naša aplikácia robiť aby spĺňala cieľ zadania našej bakalárskej práce. Z cieľa našej práce vyplýva, že chceme vytvoriť systém, ktorý bude vytvárať zaujímavé a interaktívne príbehy. Tieto príbehy následne budú hratelné v našej dobrodružnej textovej hre.

2.1 Dobrodružná textová hra

Dobrodružná textová hra je dobrý nástroj ako zinteraktívniť nejaký príbeh. Používateľ sa pohybuje vo svete vlastným tempom a interaguje s daným vymysleným svetom podľa vlastného uváženia. Naša aplikácia bude implementovať aj jednoduchú dobrodružnú textovú hru. Bude teda bude ponúkať používateľovi možnosti ako prechádzať daným herným svetom. Keďže ide o textovú hru tak, všetká komunikácia bude v textovej forme. Opis sveta a ostatných herných prvkov ako napríklad inventár hráča a aj ovládanie sa bude vykonávať cez konzolu. Ovládanie našej aplikácie bude formou príkazov do konzoli. Tieto príkazy budú viditeľné hráčovi na konzole a hráč si bude môcť vybrať, ktorý chce použit.

2.2 Herné objekty

Herné objekty, ktoré budú použité v hernom svete, budú popísané v preddefinovaných súboroch. Na začiatku sa tieto súbory spracujú a hodnoty sa zaznamenajú do našej aplikácie. Z týchto hodnôt sa následne vygenerujú všetky možné akcie, ktoré budú neskôr tvoriť príbeh hry. Súbory by mali byť ľahko upravovateľné. Toto zaručí jednoduché modifikovanie herného sveta a príbehov v našej hre.

2.3 Generovanie sveta a príbehu

Herný svet bude náhodne generovaný podľa učitých pravidiel. Pravidlá budú definované zápletkami, ktoré sa do našej hry vymyslia. Použité náhodné generovanie sveta zaistí znovuhrateľnosť príbehov, teda že príbeh bude zakaždým rozdielny. Po náhodnom vygenerovaní sveta sa následne naplánuje príbeh v rámci tohto sveta. Naplánovaný príbeh zaistí hrateľnosť vygenerovaného sveta. Príbeh vlastne zistí, či je daný svet vhodným kandidátom do našej hry. Keďže sa v danom svete našiel príbeh, tak vieme, že sa tento svet dá dohrať v našej hre. Následne sa ohodnotí zaujímavosť naplánovaného príbehu. Toto sa bude diať skúmaním štruktúry a rôznych vlastností naplánovaného príbehu. Ako napríklad, či sa neopakuje nejaká akcia v príbehu veľakrát za sebou. Po vygenerovaní určitého počtu svetov sa zistí, ktorý svet má najlepšie hodnotenie jeho príbehu. Po tomto zistení sa hráčovi ponúkne najlepší vygenerovaný príbeh a môže sa ho zahrať.

2.4 Generovanie do súboru

Ďalšou funkcionalitou našej aplikácie bude možnosť generovať tieto svety do súboru. Súbor by mal byť vo formáte, ktorý je ľahko čitateľný človeku. Používateľ si bude môcť vybrať počet náhodne generovaných svetov. Svety sa budú generovať do preddefinovaného priečinka a používateľ si následne bude môcť prezerať tieto vygenerované svety.

2.5 Generovanie zo súboru

Treťou hlavnou funkcionalitou aplikácie bude schopnosť nášho programu načítať používateľom zadaný súbor vo formáte, ktorý bude použitý pri generovaní do súboru. Tento súbor sa spracuje a vytvorí sa reprezentácia sveta podľa tohto súboru. Následne sa náš program pokúsi naplánovať príbeh nad načítaným svetom. Ak sa príbeh podaril naplánovať tak ho nechá hrať používateľovi, ak nie tak program upozorní používateľa, že daný svet je nevhodný. Takto si používateľ môže jednoducho vytvoriť vlastný svet a skúsiť si ho zahrať.

2.6 Nastavenie generátora hodnôt

Náhodné generovanie sveta sa bude generovať podľa pseudo-náhodného generátora hodnôt. Tieto generátory používajú takzvané semiačka na generovanie hodnôt. Používateľ bude mocť pri generovaní sveta v našej aplikácii zadať hodnotu semiačka aby sa mu podľa ňej vygeneroval svet. Bude to možné pri generovaní do súboru aj pri generovaní sveta rovno na hranie. Taktiež sa hodnota semiačka bude zobrazovať pri náhodnom generovaní sveta pred začiatkom hry. Týmto spôsobom si potom používateľ môže znovu zahrať nejaký príbeh, ktorý už hral. A môže si ho aj nechať vygenerovať do súboru a prezrieť si daný svet.

3. Návrh

Táto kapitola priblíži, akým spôsobom budeme riešiť cieľe aplikácie definované v špecifikácii. Spomenie sa tu architektúra aplikácie, vzťahy medzi jednotlivými častami aplikácie, reprezentácia údajov a návrh riešenia konkrétnych problémov.

3.1 Jazyk

Našim programovacím jazykom pre ktorý sme sa rozhodli je jazyk Java. Jednou z najväčších výhod tohto jazyka je jeho univerzálnosť. Je prístupný na všetkých možných platformách a programátor sa nemusí zaoberať rodzielmi medzi platformami a kód bude funkčný. Ďalšou výhodou je štrukturovateľnosť aplikácie v tomto jazyku. Túto výhodu využijeme naplno a povieme si o architektúre aplikácie v ďalšej sekcii. Tento jazyk je objektovo orientovaný a teda pri architektúre si spomenieme aj aké objeky sa budú vyskytovať v našej aplikácii.

3.2 Architektúra

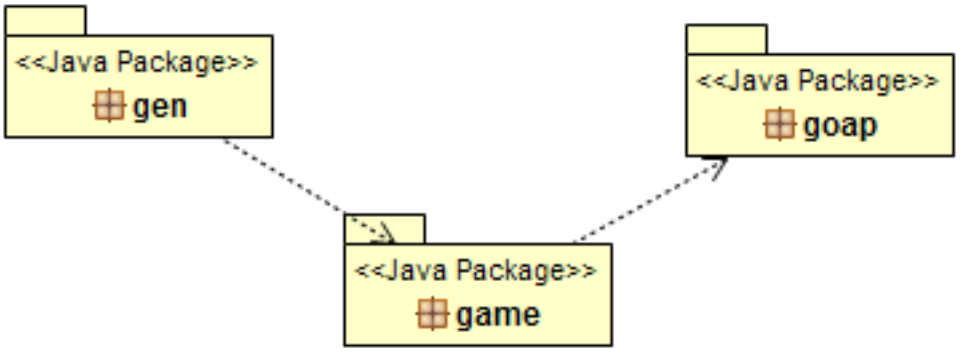
Pred implementáciou aplikácie je vhodné si rozvrhnúť architektúru aplikácie a definovať, čo bude každá z častí obsluhovať. Aplikáciu sme si rozdelili na tri časti a každá z týchto častí bude reprezentovaná balíčkom v Jave. Táto sekcia pojednáva čisto o architektúre aplikácie návrh riešenia jednotlivých problémov bude popísaný neskôr.

Prvá časť aplikácie bude obsluhovať načítavanie herných objektov z textových súborov. Bude sa venovať aj generovniu herného sveta pomocou načítaných objektov. Ďalej bude sprosdredkúvať generovanie akcií, ktoré sa budú využívať na plánovanie príbehu a samotné hranie hry. Taktiež zaistí zapisovanie reprezentácie herného sveta do súboru a aj jeho načítavanie zo súboru.

Druhá časť sa bude zaoberať definíciou reprezentácie akcie. Bude sa tu

odohrávať aj plánovanie príbehu a následné ohodnocovanie zaujímavosti tohto príbehu.

Posledná časť bude obsluhovať samostatné hranie vygenerovaného príbehu. Tiež sa bude venovať komunikácii medzi používateľom a aplikáciou a prepojí ostatné dve časti aplikácie. Grafickú reprezentáciu vzťahov medzi spomínanými častami môžeme vidieť na obrázku 3.1.



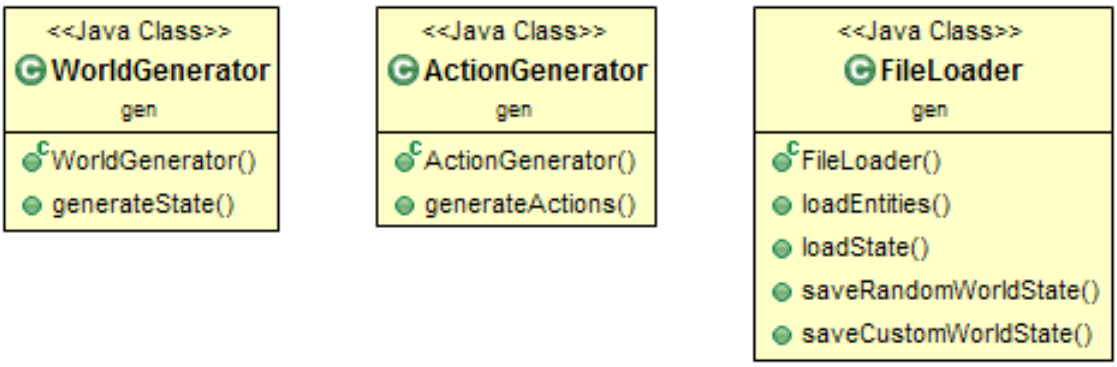
Obr. 3.1: Grafické reprezentácia Java balíčkov našej aplikácie.

Na tomto obrázku balíček gen zahrňuje funkcionalitu generovania spolu s načítavaním a zapisovaním do súboru. Balíček goap v sebe skrýva funkcionalitu plánovania a reprezentáciu akcií. A posledný balíček game obsahuje samotnú hru a ovládanie aplikácie. Šípky naznačujú vzťahy medzi balíčkami, kde balíček gen pomocou jeho funkcií vygeneruje objekty do balíčka game a ten následne použije funkcie z balíčka goap na naplánovanie a ohodnotenie jeho príbehu a výsledok zobrazí používateľovi.

3.2.1 Balíček gen

Tento balíček bude obsahovať triedy zabezpečujúce načítavanie herných objektov, generovanie herného sveta a akcií a aj zapisovanie herného sveta do súboru. Mal by teda obsahovať tri triedy, z toho každá bude obsluhovať vyššie uvedené funkcionality.

Prvá trieda bude FileLoader, ktorá zabezpečí prácu so súborom. V tejto triede by sa mali nachádzať štyri funkcie, ktoré obslúžia všetkú interakciu so súbormi v našej aplikácii. Ďalšou v poradí bude trieda WorldGenerator, ktorá bude mať na starosti generovanie herného sveta za pomoci objektov, ktoré načítal FileLoader. Na koniec v tomto balíčku budeme mať triedu ActionGenerator, ktorá nám z načítaných objektov vytvorí všetky možné herné akcie.



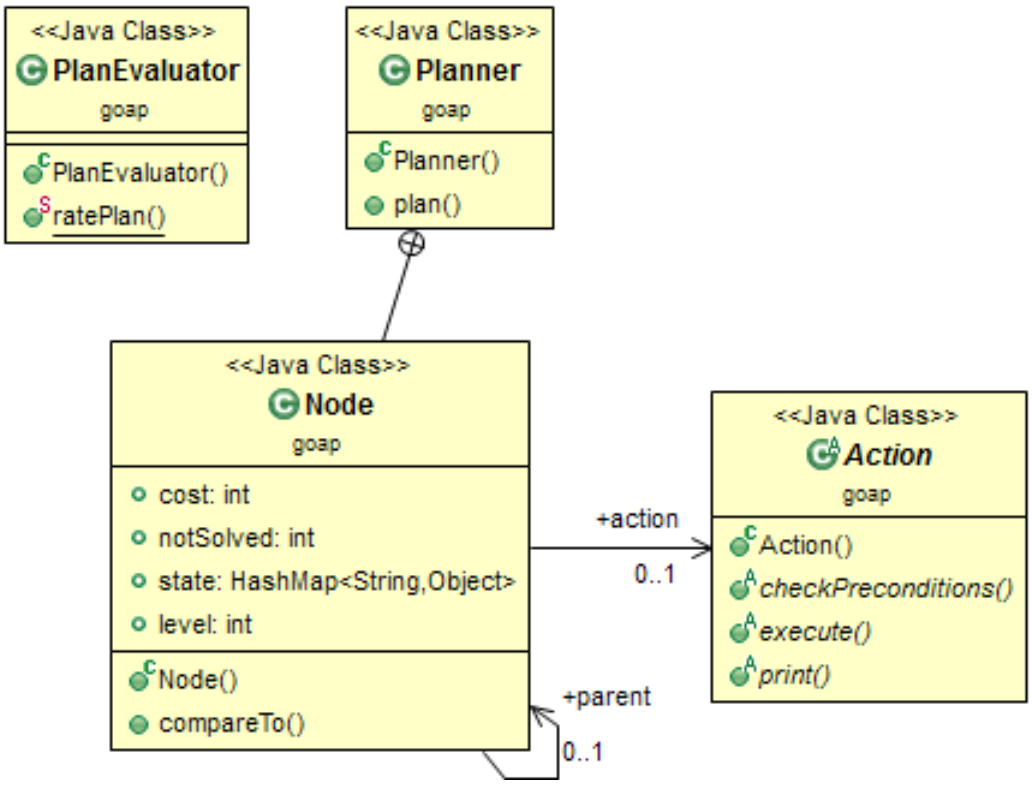
Obr. 3.2: Grafická reprezentácia balíčka gen.

Grafickú reprezentáciu môžeme sledovať na obrázku 3.2. Tento diagram nie je veľmi zložitý, keďže tieto triedy sú od seba nezávislé. Ale majú spoločnú funkcionalitu a preto sú zaradené do tohto balíčka.

3.2.2 Balíček goap

Balíček goap bude pozostávať z tried, ktoré budú mať na starosti plánovanie akcií, teda príbehu a tiež aj hodnotenie naplánovaného príbehu.

Prvou triedou tohto balíčka bude trieda reprezentujúca akciu. Bude to abstraktná trieda a bude slúžiť ako predloha jednotlivým, konkrétnym akciám. Ďalej sa tu bude nachádzať trieda Planner. Táto trieda bude mať za úlohu naplánovať pomocou akcií príbeh, ktorý sa bude neskôr skúmať kvôli ohodnoteniu a následne sa bude dať zahrať. Keďže plánovanie by malo byť implementované pomocou grafového algoritmu, tak táto trieda bude obsahovať aj vnorenú triedu pre reprezentáciu vrcholu v grafe. Tento vrchol bude uchovávať štyri atribúty. Bude si uchovávať cenu, ktorou sme sa dostali do daného vrcholu, počet zatiaľ nevyriešených cieľov v tomto vrchole, stav, ktorý tento vrchol reprezentuje a nakoniec akciu, ktorou sme sa dostali do tohto vrcholu. Taktiež si bude uchovávať referenciu na svojho rodiča, aby sme po skončení plánovacieho procesu vedeli, nájsť cestu od počiatočného stavu k cieľovému. Posledná trieda nachádzajúca sa v tomto balíčku bude PlanEvaluator, ktorý po naplánovaní príbehu plánovačom, ohodnotí daný príbeh podľa určených kritérii. Grafický model môžeme vidieť na obrázku 3.3.

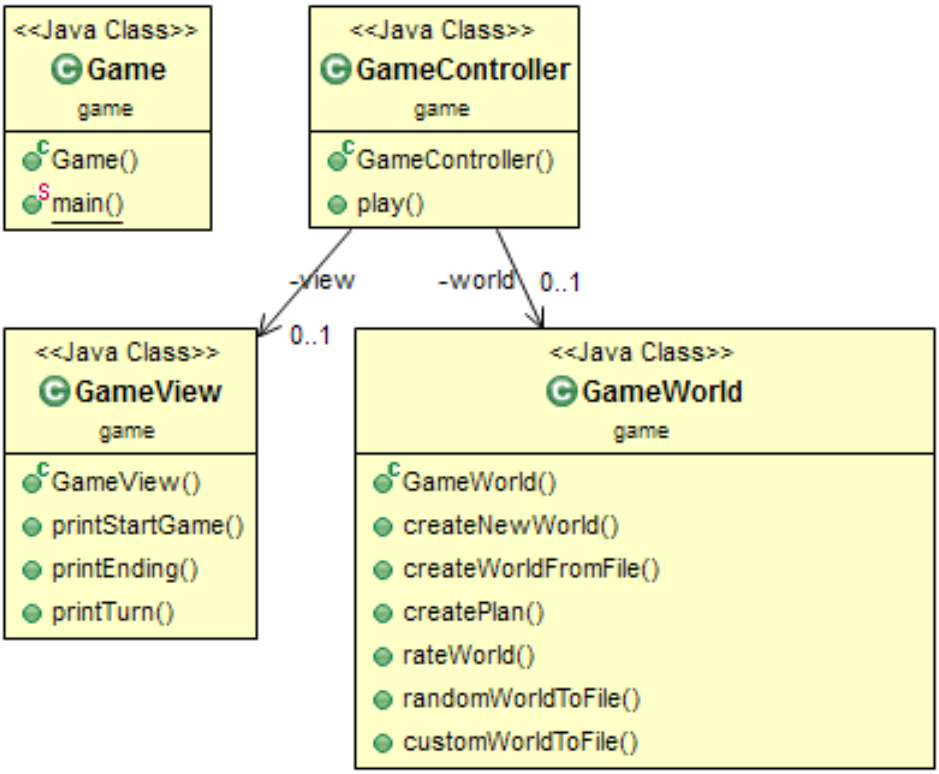


Obr. 3.3: Grafická reprezentácia balíčka goap.

3.2.3 Balíček game

V tomto balíčku budeme uchovávať triedy, súvisiace priamo s hracím prostredím aplikácie. Prvú ktorú spomenime je trieda GameWorld. Trieda GameWorld bude uchovávať všetky možné informácie o hernom svete. Bude si pamätať všetky načítané objeky daného sveta aj akcie vygenerované tomuto svetu. Ďalej bude obsahovať reprezentáciu pre svoj svet a aj cieľ, ktorý v tomto svete chceme dosiahnuť. Navyše, bude uchovávať naplánovanú postupnosť akcií, hodnotenie príbehu pre daný svet a aj hodnotu semiačka, ktorá bola použitá na generovanie tohto sveta. Taktiež, bude obsahovať referencie na triedy z balíčka gen, pomocou, ktorých sa svet, ktorý táto trieda reprezentuje, vygeneruje. Trieda si bude uchovávať aj referenciu na plánovač z balíčka goap, ktorý využie na vytvorenie príbehu pre daný svet. Druhou triedou bude trieda GameView. Táto trieda bude sporsdredkúvať odozvu používateľovi počas hrania niektorého z príbehov. Bude využívaná na opis herného sveta a vlastnosti hrdinu, ktorého bude používateľ stvárnovať vo svete. Tiež ju použijeme na uvedenie používateľa do deja a na zobrazenie každého ťahu, ktorý sa vykoná. Ak sa používateľ dostane do cieľového stavu hracieho sveta, tak táto trieda mu sprosdredkuje koniec

príbehu. Ďalšou triedou tohto balíčka by mala byt trieda GameController. Táto sa využije na ovládanie aplikácie. Bude teda poskytovať prepojenie medzi našimi troma hlavnými funkcionalitami. Umožní používateľovi vybrať si z týchto troch hlavných funkcionalít a spustí vybranú časť aplikácie. Navyše bude fungovať aj ako hrací engine. Posledná trieda v tomto balíčku bude trieda Game, ktorá iba spustí GameController a prípadne odchytí výnikmy a poskytne spätnú väzbu používateľovi ak sa vyskytne nejaký problém. Na obrázku 3.4 uvádzame grafickú reprezentáciu týchto tried.



Obr. 3.4: Grafická reprezentácia balíčka game.

3.3 Reprezentácia herného sveta

Kým sa dostaneme ku generovaniu sveta musíme najprv navrhnúť reprezentaciu herného sveta. Herný svet by mal byť reprezentovaný, niečim ako interpretáciou predikátov. Teda v aplikácii by sme mali definovavé rôzne predikáty a ich hodnoty by mali tvoriť herný svet. Napríklad by sme mali predikát Place(X), kde X reprezentuje niektorý z herných objektov a tento predikát by nám vrátil hodnotu, kde sa v hernom svete tento objekt nachádza. Podobne by sme mohli mať predikát CanTravelFromTo(X,Y). Tento predikát by hovoril o tom, či existuje cesta z miesta X

do miesta Y. A takýmito rôznymi ďalšími predikátmi, vytvoríme celý náš herný svet.

Ako je spomenuté v špecifikácii, mali by sme vedieť túto reprezentáciu zapísať do súboru vo formáte ľahko čitateľnom človeku. Čiže naša aplikácia prejde postupne cez všetky hodnoty predikátov, ktoré sú pravdivé v danej reprezentácii sveta a zapíše ich do textového súboru. Formát súboru by mal vyzerať podobne ako popisuje obrázok 3.5.

```
Place(Prince) = castle
Place(Dragon) = cave
Place(Princess) = cave
Place(Sword) = forest
CanTravelFromTo(castle,forest)
CanTravelFromTo(forest,castle)
CanTravelFromTo(forest,cave)
CanTravelFromTo(cave,forest)
```

Obr. 3.5: Ukážka formátovania reprezentácie herného sveta v súbore.

Opačným postupom by sme mali vedieť získať reprezentáciu sveta zo súboru do našej aplikácie. Teda rozparsujeme daný textový súbor a jednotlivými hodnotami naplníme hodnoty predikátov v aplikácii s ktorými potom budeme ďalej pracovať.

3.4 Reprezentácia akcií

Akcie budú reprezentované ako jednotlivé triedy v aplikácii. Budú mať za úlohu meniť hodnoty predikátov a tým teda budú môcť ovplyvnovať herný svet. Pre vhodnú reprezentáciu akcií bude potrebné aby obsahovali dve funkcie.

Prvá, ktorá zistí, či sa daná akcia môže vykonať v konkrétnom stave herného sveta. Druhou bude funkcia, ktorá zmení hodnoty v reprezentácii sveta podľa konkrétnej akcie. Túto reprezentáciu akcií budeme využívať pri plánovaní príbehu aj pri samotnom hraní.

Taká najjednoduchšia by mala byť akcia Move(Who, Where), ktorá teda zistí kde sa nachádza objekt Who pomocou hodnoty z predikátu Place(Who), potom sa pozrie či hodnota predikátu CanTravelFromTo(Place(Who),Where) je nastavená ako pravdivá. Toto je súšasťou prvej spomínanej funkcie, teda akcia Move zistí či daný objekt Who je možné presunúť na miesto Where. Ak áno tak sa použije druhá zo spomínaných funkcií akcie, ktorá zmení hodnotu predikátu Place(Who) na hodnotu Where.

3.5 Generovanie sveta a akcií

Už sme popísali reprezentáciu herného sveta aj akcií, tak v tejto časti podrobnejšie uvedieme ako budeme generovať herný svet a akcie.

Na generovanie herného sveta využijeme pseudo-náhodný generátor hodnôt z Javy. Za pomoci načítaných herných objektov zo súborov a podľa hodnoty, ktorú dostaneme z generátora hodnôt, postupne popriradzujeme hodnoty predikátom, ktoré tvoria herný svet. Teda ak by sme zo súboru mali načítané entity Prince, Princess, Dragon, Sword, tak za pomoci generátora náhodných priradíme predikátom Place(Prince), Place(Princess), Place(Dragon), Place(Sword) hodnotu prislúchajúcu jednému z načítaných herných miest. Týmto spôsobom naplníme hodnoty všetkým predikátom našej aplikácie pričom generátor náhodných hodôt nám vždy vyberie konkrétnu hodnotu predikátu.

Ako už bolo spomenuté, pri generovaní využívame pseudo-náhodný generátor hodnôt, ktorý generuje hodnoty podľa nejakej inicializačnej hodnoty. Táto hodnota sa po vygenerovaní sveta a pred začiatkom hry zobrazí používateľovi. Používateľ bude mať možnosť pred generovaním sveta nastaviť túto hodnotu a tým zreprodukovať predošlé vygenerované svety.

Generovanie akcií bude spravené jednoduchým spôsobom. Každá akcia sa vygeneruje pre všetky možné kombinácie načítaných objektov zo súboru. Napríklad teda budeme mať, toľko inšatncií akcie Move(Who, Where) koľko kombinácii dvojíc Who a Where bude načítané zo súboru. Dôvodom je potencionálna možnoť presunúť, každý objekt Who na hociktoré z miest Where.

3.6 Plánovanie príbehu

Pri plánovaní budeme využívať našu reprezentáciu sveta a akcií na vygenerovanie postupnosti akcií, ktoré nás dostanú z počiatočného stavu do cieľového. Na plánovanie využijeme algoritmus, ktorý je popísany vo východiskovej kapitole v časti o cieľovo orientovanom plánovaní akcii.

Aby bolo toto možné tak, každá akcia bude mať svoju hodnotu založenú na zaujímavosti danej akcie. Túto hodnotu použije algoritmus na ocenenie hrán grafu. Hodnotiť budeme spôsobom, čím menšia hodnota tým lepšie. Napríklad akcia na

premiestňovanie postáv bude mať horšiu hodnotu ako akcia na zdvihnutie predmetu a tá bude mať zase horšiu hodnotu ako akcia na súboj dvoch postáv, keďže toto nám príde zaujímavejšie. Táto hodnota sa bude kumulovať v triede ktorá reprezentuje vrchol grafu. Vrchol grafu bude teda obsahovať súčet zaujímavostných hodôt svojich rodičov a seba.

Po zistení, že sme v prehľadávaní dostali do cieľového stavu, teda že sme našli konkrétny plán, si zapamäťáme vrchol, ktorý reprezentuje tento cieľový stav a pokračujeme v prehľadávaní až kým nenájdeme všetký možné plány daného sveta. Keď prehľadávanie skončí, tak prejdeme všetkými zapamätanými vrcholmi, ktoré sme našli a vyberieme ten s najmenšou cenou, pretože reprezentuje najzaujímavejší príbeh. V tomto vrchole využijeme jeho referenciu na rodiča a príbeh zrekonštruujeme tak ako má byť od začiatku do konca.

3.7 Hodnotenie príbehu

Hodotiť príbeh budeme formou analyzovania dynamiky príbehu. Podľa našej hypotézi by mal príbeh mať rovnomerne rozložené zaujímavé akcie a menej zaujímave akcie, tak aby sa v príbehu nevyskytovala nejaká jednotvárna časť. Budeme teda sledovať štruktúru príbehu a aj poradie akcií v príbehu.

Každý príbeh bude mať svoje hodnotenie v číselnej forme a keď sa vyskytne nejaká vlastnosť v príbehu, ktorú onzačíme ako nezaujímavú tak z daného hodnotenia odpočítame príslušnú hodnotu. Budeme sledovať napríklad či sa daná akcia v príbehu neopakuje veľa kráť za sebou, či sú v príbehu využité všetky možné akcie, následnosť jednotlivých zaujímavejších akcií, náväznoť akcií teda či sú akcie, ktoré na seba naväzujú, prerušené inou akciou alebo nie, akou akciou začína príbeh a pod. Po tomto ohodnotení sa hodnota priradí k danému svetu, aby sa neskôr mohol vybrať ten najzaujímavejší, ktorý sa poskytne používateľovi.

3.8 Ovládanie aplikácie

Aplikácia sa bude ovládať pomocou príkazov do konzoli. Používateľ vždy dostane na výber zoznam príkazov, ktoré môže v danom štádiu aplikácie použiť. V niektorých štádiách bude aplikácia očakávať aj vstupné údaje od používateľa. Po zadaní príkazu

aplikácia vykoná daný príkaz a poskytne používateľovi ďalšiu sadu príkazov alebo skončí.

3.9 Herná časť aplikácie

Samotná hra bude taktiež reprezentovná pomocou konzoli, koniec koncov ide o textovú hru. Po vygenerovaní alebo načítaní sveta aplikácia vypíše na konzolu semiačko podľa, ktorého bol svet vygenerovaný a uvedie používateľa do deja. Po úvode sa už vypíše samotný hrací ťah. Aplikácia poskytne používateľovi na konzolu lokáciu v ktorej sa hrdina nachádza. Budú poskytnuté aj informácie čo sa nachádza v danej lokalite, teda či sú tam nejaké objekty alebo iné herné entity s ktorými by hrdina mohol interagovať. Daľším výpisom budú pozície na ktoré hráč môže posunúť svojho hrdinu. Tiež sa vypíše aj hrdinov inventár. Na koniec sa používateľovi vypíše zoznam akcií, ktoré môže hrdina v danej lokácii vykonať. Potom ako používateľ zadá nejakú akciu na vykonanie, tak sa tento proces zopakuje s tým, že sa údaje aktualizujú vzhľadom na akciu, ktorú používateľ vykonal. Ak sa vykoná cieľová akcia, tak hra vypíše koniec príbehu a aplikácia skončí.

4. Implementácia

Táto kapitola poslúži na opis niektorých problémov s ktorými sme sa stretli pri implementácii, popíše niektoré detialy implementácie a tiež aj rozdiely mezdi návrhom aplikácie a reálnou implementáciou aplikácie.

4.1 Reprezentovanie herného sveta

Keďže na implementáciu používame jazyk Java, tak sme sa na reprezentáciu herného sveta rozhodli použiť už implementovaný objekt v Jave. Využili sme na to, objekt HashMap, ktorý obsahuje zoznam kľúčov, na ktoré mapuje zadefinované objekty. My sme sa rozhodli klúče reprezentovať typom String a hodnoty, ktoré sú namapované na tieto Stringy sú typu Object.

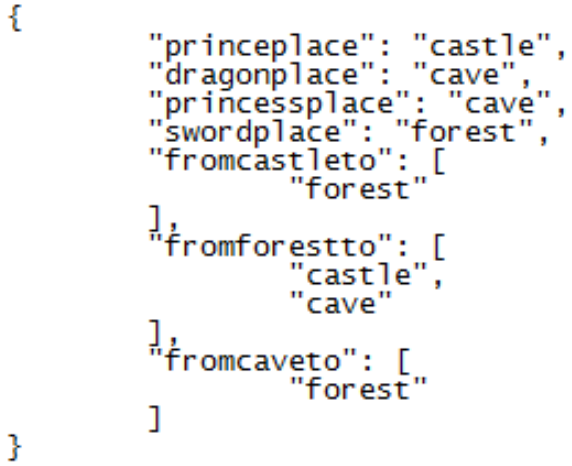
Touto HashMapou simulujeme správanie predikátu. Vieme z nej dostať hodnotu podobne ako z predikátu, napríklad vieme si vypýtať namapovanú hodnotu na String "princeplace", čo nám povie kde sa princ v hernom svete nachádza. Takisto vieme zistiť hodnotu namapovanú na String "fromcastleto", ktorou je pole hodnôt, ktoré reprezentujú miesta do ktorých sa vieme presunúť z lokácie castle. Čiže napriíklad môžeme dostať hodnoty swamp,garden, jail a z tohto potom vieme povedať, že z lokácie castle sa môžeme presunúť do močiara, záhrady alebo väznice.

Mapovným typom je Object z dôvodu univerzálnosti hodnôt, ktoré naše simulované predikáty môžu poskytovať. Typ Object dedí v Jave každá trieda a teda hodnotou nášho simulovaného predikátu môže byť hocičo. Na našich príkladoch môžme vidieť využitie tohto faktu, jeden klúč vracia hodnotu String a druhý pole Stringov. Toto nám umožnuje mať jeden jediný objekt na popísanie celého herného sveta.

4.1.1 Reprezentácia v súbore

S reprezentáciou herného sveta súvisí aj jej zápis do súboru. Keďže máme svet reprezentovaný ako jediný objekt, tak sme sa rozhodli využiť knižnicu na skonvertovanie Javovského objektu do textového súboru.

Knižnicu, ktorú sme použili, je od spoločnosti Google a nazýva sa Gson. Táto knižnica umožňuje konvertovať objekty z Javy do textového súboru vo formáte JSON. JSON formát je celkom rozšíreným textovým formátom a teda jedna z výhod je aj že sú ľudia naň zvyknutý. Ak použijeme mód pekného vypisovania, tak je tento formát relatívne jednoduchý na čitanie pre človeka, tak ako sa požadovalo v návrhu. Na obrázku 4.1 môžeme vidieť čitateľnosť tohto formátu v móde pekného vypisovania.



Obr. 4.1: Ukážka formátu zápisu reprezentácie herného sveta do súboru.

Ďalšou výhodou je, že sme nemuseli implementovať vlastný parser pre načítavanie zo súboru, pretože knižnica obsahuje aj spätnú konverziu, teda zo súboru vo formáte JSON do daného objektu v Jave.

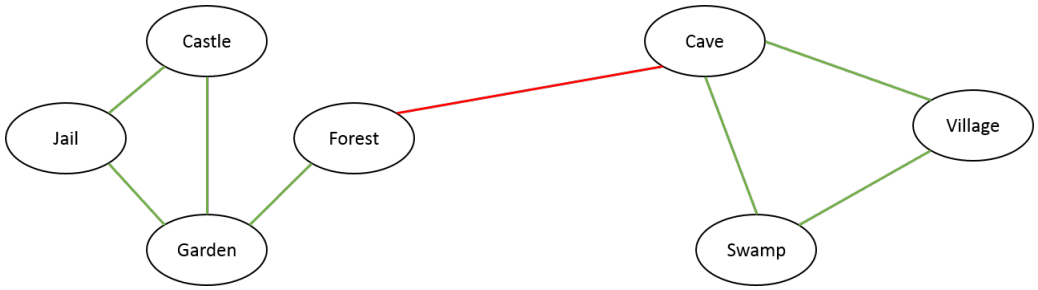
Jedinou nevýhodou je nemožnosť načítania objektu zo súboru ak je súbor v móde pekného vypisovania. Na načítanie objektu zo súboru musí byť súbor zapísaný v kompaktnom móde, čo znamená, že údaje sú zapísané v jednom riadku. Túto prekážku sme prekonali tak, že súbor v kompaktnom móde generujeme vždy a používateľovi poskytujeme pri zápise do súboru výber, či chce vygenerovať aj súbor v móde pekného vypisovania. Takto si používateľ vie vybrať, či bude chcieť daný súbor aj čítať a teda si vie vygenerovať aj pekne vypísaný súbor. Na druhú stranu ak používateľ reprezentáciu nebude chcieť čítať vlastnými očami tak mu nezahltíme systém zbytočnými súbormi. Použitie tejto knižnice je pre nás oveľa jednoduchšie ako implementovanie vlastného

zapisovania do súboru a následného parsovania súboru pri načítavaní herného sveta zo súboru.

4.2 Generovanie herného sveta

Jednou z našich vymyslených zápletiek, je premôcť príšeru v hernom svete, ktorá blokuje jedu z ciest, ktrou sa váš hrdina môže vydať. Keď dané monštrum používateľ premôže tak sa mu uvoľní cesta, ktorú príšera blokovala. Táto udalosť je podľa nás zaujímavejšia ako väčšina ostatných akcií, ktoré hrdina môže vykonať a teda sme ju pre účely vyhľadávania ohodnotili nízkou cenou. A keďže naše plánovanie v podstate hľadá najzaujímavejší príbeh, lebo hľadá plán z najnižšou cenou, tak mohla nastať situácia, že plánovač našiel príbeh, ktorý zahŕňa aj boj s nejakým monštrom ale existovala aj nejaká dlhšia cesta k cieľu bez tejto udalosti a teda hráč mohol obísť našu vymyslenú zápletku a mohlo by sa mu zdať že daný vygenerovaný príbeh je mdlý. Kvôli tomuto fenoménu sme sa rozhodli trochu upraviť generovanie herného sveta.

Toto generovanie si zachováva vlastnosť náhodnosti, len je upravené tak aby bolo zaručené, že hráč bude musieť niekoľko krát bojovať s nejakým monštrom v rámci jedného príbehu. Generovanie sme upravili tak, že načítané lokácie rozdelíme do skupín. Medzi lokáciami v rámci jednej skupiny vygenerujeme náhodne hodnoty odkiaľ kam sa vieme presunúť a následne skupiny prepojíme príšerou, ktorá sa nachádza na jednej z lokácii v prvej skupine a blokuje cestu do niektorej lokácie z druhej skupiny. Týmto spôsobom vieme zreťaziť ľubovolný počet skupín. Na priblíženie tohto spôsobu generovania sveta použijeme konkrétny príklad znázornený aj na obrázku 4.2.



Obr. 4.2: Grafická reprezentácia vygenerovaného herného sveta.

V tomto prípade sme načítali sedem lokácii zo súboru, ktoré to sú vidieť na obrázku 4.2 reprezentované ako elipsy. Následne sme si lokácie rozdelili na dve skupiny

v tomto prípade ľavú a pravú. V každej skupine zvlášť sme si vygenerovali cesty, tieto sú označené zelenou farbou. Potom sme vygenerovali náhodnú pozíciu príšery z ľavej skupiny lokácii, v tomto prípade je to Forest a zároveň sme vygenerovali náhodnú lokáciu, ktorú príšera blokuje z pravej skupiny, tou je miesto Cave. Teda vytvorili sme cestu z ľavej skupiny lokácii do pravej pomocou monštra, ktoré musí byť najprv porazené aby sa ňou dalo prejsť. Táto cesta je na obrázku 4.2 znázornená červenou farbou

V takto vygenerovanom svete, keď umiestnime cieľ hry do poslednej skupiny lokácii, tak máme zaručené, že ak náš plánovač nájde plán v ktorom sa nachádza akcia na boj s príšerou, tak sa pri hraní tohto sveta nedá táto akcia nijako obísť.

4.3 Akcie a ich generovanie

Akcie reprezentujeme podľa návrhu ako samostatné triedy. Jedna zmena oproti návrhu však nastala a to tá, že sme tieto triedy akcii vložili do samostatného Java balíčka, aby bola aplikácia prehľadnejšia.

Pri generovaní akcií sme chceli čo najviac zefektívniť plánovanie a teda pri generovaní herného sveta zisťujeme, ktoré objekty sa v ňom budú nachádzať a ktoré nie. Túto informáciu potom použijeme pri generovaní akcii aby sme limitovali počet akcií pre objekty, ktoré sa v hernom svete nenachádzajú. Čiže ak z piatich načítaných príšer pri generovaní herného sveta použijeme iba dve tak pri generovaní akcii vygenerujeme akcie len tým dvom príšerám, ktoré sa vo svete nachádzajú.

Ďalšou takouto optimalizáciou je posienanie iba tých akcií, ktoré sa týkajú hrdinu príbehu. Keďže hrdina je hlavným poháňačom deja a tiež používateľ ovláda len tohto hrdinu, tak dáva zmyslel plánovať len tieto akcie.

Tieto dve optimalizácie znížia počet akcií používaných pri plánovaní, čo rapídne zníži počet potencionálnych stavov, ktoré treba prejsť pri plánovaní príbehu.

4.4 Plánovací algoritmus

Pri implementácii sme sa stretli s jedným závažným problémom a to, že náš plánovací algoritmus bol príliš neefektívny. Prehľadávať všetky možné cesty v grafe je zbytočné a časovo veľmi náročné. Rozhodli sme sa teda použiť ako plánovací algoritmus

vyľadávanie A*, ktoré hľadá najkratšiu cestu v grafe. Tento prístup bol v podobnom kontexte využitý v článku[7]. Toto si nevyžadovalo veľké zmeny, keďže tento algoritmus sa vykonáva taktiež na grafoch. Zmeny, ktoré sme vykonali si popíšeme v tejto časti.

Jedným z aspektov, ktoré si vyhľadávanie A* vyžaduje je akási heuristika alebo niečo čo nám v priebehu vyhľadávania povie ako ďaľeko sme od cieľa. V našom prípade sme si vytvorili čiastkové cieľe príbehu a počet doteraz nevyriešených cieľov v aktuálnom stave sme použili ako túto heuristiku.

A* vyhľadávanie využíva prioritnú frontu na výber ďalšieho vrcholu na spracovávanie. Aby sme mohli vytvoriť v Jave prioritnú frontu s triedou Node, čo je naša reprezentácia vrcholu, tak táto trieda musí implementovať funkciu compareTo(), ktorá sa používa na utriedenie fronty podľa priority. Túto funkciu sme implementovali tak, že spočítava zaujímavostnú hodnotu vrchola a počet nevyriešených čiastových cieľov v danom vrchole. Prioritná fronta sa usporadúva podľa tejto hodnoty vzostupne, aby sme vyberali stavy najbližšie k cieľu ako prvé.

Keď toto prehľadávanie nájde cestu do cieľového stavu, tak máme zaručené že sme našli najzaujímavejší príbeh daného sveta, pretože tento algoritmus hľadá najlacnejšiu cestu v grafe. Podobne ako v pôvodnom algoritme využijeme vrchol, ktorý reprezentuje cieľový stav na zrekonštruovanie celého nájdeného príbehu.

Táto zmena nám umožnila generovať oveľa dlhšie príbehy a za zlomok času oproti pôvodnému algoritmu.

4.5 Ovládanie a hra

Tak ako popisuje návrh ovládanie aplikácie je implementované cez konzolu. Používateľovi sa vypíšu do konzoli možnosti, ako je ukázané na obrázku 4.3, ktoré môže v danej chvíli vykonať. Tieto možnosti sú očíslované a používateľ zadaním príslušného čísla spustí daný príkaz.

```
1: Generate and Play, 2: Generate to File, 3: Play from File
```

Obr. 4.3: Základné príkazy aplikácie.

Po týchto prvých príkazoch, prichádza špecifikovanie vykonania vybraného príkazu. Napríklad pri vybraní prvého príkazu z obrázku 4.3 aplikácia potrebuje vedieť,

či chceme hrať náhodný svet alebo svet vygenerovaný podľa nejakého semiačka, ako je zobrazené na obrázku 4.4.

```
1: Generate and Play, 2: Generate to File, 3: Play from File
1
1: Generate random world and play, 2: Generate world from seed and play
```

Obr. 4.4: Špecifikujúce príkazy aplikácie.

Niektoré z týchto špecifikačných príkazov zahŕňajú aj zadávanie rôznych vstupov používateľom. Ak si vyberieme v aplikácii generovanie herného sveta do súboru, tak tak aplikácia očakáva od používateľa boolovskú hodnotu, teda true alebo false, či chce vygenerovať aj súbor v móde pekného vypisovania (obr 4.5). Podobne pri náhodnom generovaní herného sveta do súboru si aplikácia vyžiada od používateľa aj počet svetov, ktoré chce používateľ vygenerovať (obr. 4.6).

```
1: Generate and Play, 2: Generate to File, 3: Play from File
2
1: Generate random world(s) to file(s), 2: Generate world from seed to file
1
Generate pretty print file? (true / false)
true
```

Obr. 4.5: Zadávanie boolovských vstupov do aplikácie.

```
1: Generate and Play, 2: Generate to File, 3: Play from File
2
1: Generate random world(s) to file(s), 2: Generate world from seed to file
1
Generate pretty print file? (true / false)
true
Put in number of worlds to generate:
6
```

Obr. 4.6: Zadávanie čisených vstupov do aplikácie.

V prípade, keď generujeme herný svet zo súboru, tak sa od používateľa očakáva textový vstup, ktorým popíše relatívnu cestu ku textovému súboru formátu JSON, ktorý sa použije na generovanie herného sveta (obr. 4.7).

```
1: Generate and Play, 2: Generate to File, 3: Play from File
3
Put in relative address to file:
worlds/world1.json
```

Obr. 4.7: Zadávanie textových vstupov do aplikácie.

Čo sa týka samotnej hry, tak každý herný ťah je opísaný pár riadkami, ako môžme vidieť na obrázku 4.8.

```
Your location is: bridge
From where you are standing you can see path(s) leading to: beach,
Additionally you see path to island, but it is blocked by dracula
The monster seems to be vulnerable to suspicious_coin
You can also see a friendly king, who is looking for magic_sword
You will surely get a reward if you bring magic_sword
Your inventory: [magic_sword, fairy_dust, suspicious_coin]
You have 0 coins
What do you do?
1: prince kills dracula, 2: prince trades with king, 3: Move prince To beach,
```

Obr. 4.8: Opis ťahu v aplikácii.

Vidíme, že hra opisuje lokáciu v ktorej sa nachádzame a poskytuje informácie o ďaľších entitách nachádzajúcich sa na tejto lokácii. Tieto entity majú svoje vlatnosti, ktoré výpis ťahu taktiež opisuje napríklad moštrum je zraniteľné na niektorý z načítaných objektov alebo nejaká priateľská postava hľadá niektorý z načítaných objektov. Tento výpis tiež opisuje inventár hlavnej postavy, teda aké objekty má pri sebe a koľko mincí vlastní. Na konci máme možnosti, ktoré používateľ môže vykonať takisto očíslované a zadaným čislom si zvolií niektorú z možností.

5. Výsledky

V tejto kapitole, uvedieme niektoré zo štatistýk týkajúcich sa generovania herných svetov, zanalyzujeme ich a ukážeme si vygenerovaný príbeh našou aplikáciou.

5.1 Štatistiky generovania svetov

Pri rozhodovaní koľko náhodných herných svetov naraz je vhodné generovať, sme museli počitať s tým, aby boli vyvážené hodnoty času generovania a dostatočného ohodnotenia príbehu. Nasledovné obrázky 5.1 a 5.2 znázorňujú tabuľky štatistík, pri rôznych počtoch vygenerovaných herných svetov naraz. Štatistiky boli namerané na počítači s procesorom Intel(R) Core(TM) i7-4790K s výkonom 4GHz a s pamäťou o velkosti 16GB.

hodnotenie	Počet svetov: 10			Počet svetov: 100			Počet svetov: 200			Počet svetov: 1000			Počet svetov: 10000		
skupiny	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg
2	82	100	91	92	99	96	95	99	96	97	99	98	99	99	99
3	75	92	82	78	93	86	76	96	87	84	94	89	n/a	n/a	n/a
4	n/a	n/a	n/a	n/a	n/a	n/a	68	76	72	n/a	n/a	n/a	n/a	n/a	n/a

Obr. 5.1: Tabulka najlepších ohodnotení pri rôznych generovaniach herného sveta.

čas	Počet svetov: 10			Počet svetov: 100			Počet svetov: 200			Počet svetov: 1000			Počet svetov: 10000		
skupiny	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg
2	0,01	0,26	0,05	0,29	1,03	0,49	0,73	1,42	0,91	3,61	5,16	4,49	36,09	47,90	41,99
3	0,01	4,10	2,95	2,03	4,64	3,44	4,63	8,06	5,67	23,47	34,00	28,73	n/a	n/a	n/a
4	0,11	13,70	1,60	4,19	23,18	14,32	84,09	85,23	84,66	n/a	n/a	n/a	n/a	n/a	n/a

Obr. 5.2: Časy trvania rôznych generovaní herného sveta.

Vygenerované svety následne slúžia ako množina z ktorej vyberieme svet s najlepšie ohodnotením príbehom. Na ľavej strane tabuľky máme zobrazené koľko skupín lokácii sme použili pri generovaní a zbytok tabuľky je rozdelený na časti v ktorých opisujeme hodnoty, ktoré sa vyskytli pri generovaní rôznych množstiev svetov naraz.

Z týchto hodnôt sme museli vydedukovať, koľko svetov sa nám oplatí generovať pred začiatkom hry, tak aby používateľ nečakal príliš dlho ale zároveň aby sa mu dostal obstojný príbeh. Vidíme, že pri dvoch skupinách lokácii nám stačí generovať sto svetov, pretože vyšším počtom už veľa na ohodnotení príbehu nezískame.

Ďalší faktor, ktorý sme museli zohľadniť pri rozhodovaní je spolahlivosť. Čím viac svetov vygenerujeme tým je väčšia šanca, že vôbec nájdeme nejaký vygenerovaný svet, ktorý sa dá prejsť. Toto zohralo úlohu pri rozhodovaní sa pri troch skupinách, kde medzi generovaním sto alebo dvesto svetov nieje až taký rozdiel v štatistikách, ktoré sa týkajú ohodnotení vygenerovaných svetov ale ak vygenerujeme dvesto svetov tak máme v podstate dvojnásobne vyššiu pravdepodobnosť, že vygenerujeme hrateľný príbeh. Používateľ si síce o trochu viac počká ale šanca, že sa nevygeneruje hrateľný svet sa dvojnásobne zníži.

Pri štyroch skupinách lokácii sa veľmi neodporúča generovať herné svety, pretože generovanie je buď priveľmi nespolahlivé (malé množstvá generovaných svetov) alebo je nedostatočne rýchle aby bolo použiteľné. Vidíme, že pri nízkych počtoch vygenerovaných svetov sú hodnoty v tabuľke ohľadom hodnotenia prázdne a to preto, že sa pri týchto generovaniach nepodarilo vygenerovať nejaký hrateľný svet.

Čo sa časov týka, tak vidíme, že pri štyroch skupinách je generovanie dosť nepredvídateľné. Ak sa vygenerujú svety v ktorých plánovanie zistí celkom skoro, že svet nie je hrateľný tak sa generovanie skončí rýchlo. Komplikovanejšie svety však trvajú zase priveľmi dlho. Toto môžeme sledovať v tabuľke pri generovaní desiatich svetov, kde najkratšie trvalo 0,11 sekundy ale najdlhšie generovanie trvalo až 13,70 sekúnd. Pri generovaní sto svetov je čas uvedený v tabuľke za ktorý sa nepodarilo vygenerovať ani jeden hrateľný svet pretože pri tomto počte je toto generovanie ešte stále nespolahlivé. Pri dvesto svetoch generovanie začína byť ako tak spolahlivé ale čas generovania je neúnosný. Vyššie počty generovaných svetov sme ani neskúšali, pretože čas za ktorý by generovanie skončilo je nepriateľný na používanie aplikácie. To platí aj pri troch skupinách a desaťtisíc svetoch.

V tabuľke časov tiež môžeme vidieť, že zvýšením počtu skupín lokácii sa rapídne zvyšuje čas, ktorý je potrebný na generovanie takýchto svetov. Čas, ktorý tu narastá je čas plánovania príbehu. Ako môžeme vidieť na obrázku 5.3, tak dĺžky príbehov sa zvyšujú so zvýšeným počtom skupín lokácii.

	dĺžka príbehu		
skupiny	min	max	avg
2	14	33	20
3	20	39	28
4	43	44	43

Obr. 5.3: Dĺžky príbehov pri rôžnych veľkostiach skupín lokácii.

Toto je dôvodom zvyšujúceho sa času na generovanie herných svetov, keďže plánovanie príbehu je jeho súčasťou. A* vyhľadávanie má časovú zložitosť závislú od hĺbky vnorenia, ktorá je exponentom v tomto vzťahu, takže je zrejmé, že čas plánovania bude rásť tak rapídne ako nám ukazuje naša štatistika.

Dĺžka naplánovaného príbehu súvisí tiež aj s ohodnocovaním príbehu. Čím je príbeh dlhší, tým sa zvyšuje šanca na vyskytnutie sa niektorej z neželaných situácii v príbehu, ktoré my sledujeme a podľa ktorých príbeh hodnotíme. Túto skutočnosť môžeme tiež sledovať v tabuľke ohodnotení 5.1, kde vidieť, že čím viac skupín lokácii pri generovaní máme tým je priemerné ohodnotenie príbehu menšie.

5.2 Výsledný príbeh

Naša aplikácia generuje príbehy o princovi, ktorý sa snaží zachrániť princeznú, ktorú uniesli banditi nikto nevie kam. Títo banditi požadujú výkupné za unesenú princeznú. Princ bude musieť prekonávať prekážky na ceste za princeznou ako napríklad riešiť rôzne hádanky alebo bojovať s príšerami. Navyše musí nejako získať mince na výkupné za princeznú.

Najprv si ukážeme jeden z horšie ohodnotených príbehov podľa našej hypotézi. Môžeme ho vidieť na obrázku 5.4 a popíšeme si prečo ho tak naša aplikácia ohodnotila.

```
prince solves genie's riddle,
Move prince To volcano,
Move prince To village,
Move prince To castle,
prince picks up magic_sword,
prince trades with goblin,
Move prince To village,
Move prince To volcano,
Move prince To island,
Move prince To marketplace,
Move prince To garden,
prince picks up rusty_axe,
Move prince To marketplace,
Move prince To island,
Move prince To volcano,
prince trades with king,
Move prince To forest,
Move prince To beach,
prince saved princess,
```

Obr. 5.4: Ukážka menej zaujímavého príbehu.

Hneď prvá akcia príbehu je jednou z chýb prečo má tento príbeh zlé ohodnotenie. Podľa nášeho prístupu by hráč nemal hned riešiť úlohy, ktoré posúvajú príbeh a táto akcia je jednou z nich. Hráč by sa mal na začiatku trochu spoznať s prostredím a objavovať.

Druhou chybou je zbytočná dĺžka príbehu. Príbeh má zbytočne veľa akcii a veľa krát sa v ňom opakuje akcia Move. Viac ako polovicu príbehu zaberá len premiestňovanie sa, čo je podľa nás nezaujímavé a aj kvôli tomuto má príbeh znížené hodnotenie.

Posledná chyba v tomto príbehu sa týka akcii PickUp a Trade, ktoré sú v príbehu hneď po sebe, čo znamená že obchodník aj vec, ktorú tento obchodník chce kúpiť sa nachádzajú na tej istej lokácii. Toto je nežiadúca situácia, pretože obchodník týmto stráca význam, keďže hráč danú vec nemusí hľadať.

Ďalej si ukážeme príbeh, ktorý naša aplikácia ohodnotila vyskoým hodnotením a popíšeme, prečo si myslíme, že je hodný vysokého hodnotenia. Tento príbeh je zobrazený na obrázku 5.5.

```
Move prince To village,
prince picks up suspicious_coin,
Move prince To castle,
prince trades with king,
Move prince To plains,
prince picks up silver_dagger,
Move prince To garden,
prince solves warlock's riddle,
Move prince To beach,
prince trades with elf,
Move prince To marketplace,
prince picks up magic_sword,
Move prince To beach,
prince kills werewolf,
Move prince To swamp,
prince saved princess,
```

Obr. 5.5: Ukážka viac zaujímavého príbehu.

V tomto príklade môžeme vidieť, že sa príbeh nezačína akciou posúvajúcou príbeh. Toto je známkou dobrého úvodu hráča do deja a zoznámenia sa s príbehom a svetom.

Ďaľej môžme pozorovať, že príbeh nie je zbytočne natiahnutý opakujúcimi sa akciami, čo znamená že hráč sa neunudí v zdĺhavých častiach príbehu. Akcie sa v príbehu striedajú a teda príbeh necháva hráča objavovať svet a hľadať v ňom zaujímavé veci ale zároveň aj ponúka posun príbehu k cieľu. Toto striedanie akcii navyše zaručuje, že žiadna zápletka nie je zbytočná, ako sme mohli vidieť v príbehu z obrázku 5.4, kde jeden z obchodníkov bol nezaujímavý a vlastne zbytočný.

Záver

Cieľom tejto práce bolo navrhnúť a implementovať systém na tvorbu interaktívnych príbehov v dobrodružných textových hrách. Cieľ sme sa rozhodli splniť pomocou systému, ktorý náhodne vygeneruje herné svety, potom pomocou plánovania v nich vytvorí príbeh, následne ohodnotí príbehy podľa kritérií dynamiky príbehu a používateľovi sa vyberie najlepšie ohodnotený. Tento príbeh si potom používateľ môže zahrať v našej dobrodružnej textovej hre, ktorá je súčasťou nášho systému.

Herný svet sme repezentovali pomocou Java objektu HashMap, ktorý obsahuje kľúče a hodnoty. Kľúče simulujú názvy predikátov spolu so vstupnými hodnotami predikátu a hodnoty HashMapy sú kvázi návratovou hodnotou predikátu. Buď generujeme svet náhodne alebo ho vieme aj načítať zo súboru.

Na plánovanie sme použili vyhľadávanie A*. Toto je grafový algoritmus, ktorý pomocou heuristiky hľadá cestu z počiatočného stavu herného sveta do konečného. Cesta sa zkladá z akcií, ktoré ovplyvňujú herný svet a tieto zároveň tvoria náš príbeh.

Ohodnotenie príbehu v našom systéme spočíva v analyzovaní postupnosti akcií vo vygenerovanom príbehu. Máme za to že príbeh by mal byť rovnomerne rozložený a teda, že by sa mali striedať menej zaujímavé akcie s viac zaujímavými. Najlepšie ohodnotený príbeh našim systémom si používateľ môže zahrať v nami implementovanej textovej hre.

Výsledkom našej práce je teda aplikácia, ktorá ponúkne používateľovi zaujímavý príbeh, ktorý si následne môže zahrať, čo poskytuje interaktivitu s príbehom.

Ďalší vývoj

V rámci ďalšieho vývoja aplikácie by sa mohla vylepšiť reprezentácia akcií, ktoré by sa nevytvárali napevno v aplikácii ale dali by sa načítavať zo súboru. Podobne by sme mohli načítavať zo súboru aj typy predikátov, ktoré reprezentujú herný svet. Toto by prispelo ku ľahšej modifikácii príbehov aj osobou, ktorá sa nevyzná v programovaní.

Jedným z vylepšení by mohlo byť aj vytvorenie väčšieho počtu zápletiek. Toto by prispelo ku možnosti lepšie analyzovať vytvorený príbeh a teda príbehy by boli kvalitnejšie.

Posledným aspektom ďalšieho vývoja by bolo vytvoriť lepšiu heuristiku pre plánovanie, čo by viedlo ku optimálnejšiemu plánovaniu a teda by sa dali vytárať aj dlhšie príbehy ako v našej práci.

Bibliografia

[1] ABBOTT, H. Porter - ALBERT, Jan - ALEXANDER, Marc 2009. *The living handbook of narratology* : 2009. .

[2] BRINGSJORD, Selmer - FERRUCCI, David A. 1999. *Artificial Intelligence and Literary Creativity Inside the Mind of BRUTUS, a Storytelling Machine* : 1999. .

[3] DEHN, Natalie aug. 1981. *Story Generation after Tale-Spin*, Drinan, A., ed. University of British Columbia, Vancouver, Canada : Proceedings of the Seventh International Joint Conference on Artificial Intelligence, aug. 1981. Zv. 116, 18.

[4] KLEIN, - SHELDON, et al. 1973. *Automatic novel writing: A status report. Technical Report 186* : Computer Science Department, The University of Wisconsin, Madison, 1973. .

[5] LEBOWITZ, Michael aug. 1983. *Creating a Story-Telling Universe*, Nundy, A., ed. Karlsruhe, Germany : Proceedings of the Eighth International Joint Conference on Artificial Intelligence, aug. 1983. Zv. 1, 63–65.

[6] MEEHAN, James R. Aug. 1977. *Tale-Spin, an interactive program that writes stories*. MIT, Cambridge, MA : Proceedings of the Fifth International Joint Conference on Artificial Intelligence, aug. 1977. 91-98.

[7] ORKIN, Jeff 2003. *"Applying Goal-Oriented Action Planning to Games," AI Game Programming Wisdom 2* : Charles River Media, 2003. .

[8] OWENS, Brent. 2014. *Goal Oriented Action Planning for a Smarter AI*. [online]. Url: `https://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793` (cit. 19. 01. 2017).

[9] PÉREZ, Rafael Pérez, "Mexica: a computer model of creativity in writing", diz. pr., The University of Sussex, 1999.

[10] RIEDL, Mark O. - YOUNG, R. Michael 2010. *Narrative Planning: Balancing Plot and Character* : Journal of Artificial Intelligence Research, 2010. .

[11] STUART, Russell - NORVIG, Peter 2009. *Artificial Intelligence: A Modern Approach* : Prentice Hall, 2009. .

[12] TURNER, Scott R. "Minstrel: a computer model of creativity and storytelling", diz. pr., University of California at Los Angeles, Los Angeles, 1993.