

Univerzita Komenského v Bratislave  
Fakulta Matematiky, Fyziky a  
Informatiky

Tvorba príbehov v počítačových  
hrách

Bakalárska práca

**Univerzita Komenského v Bratislave**  
**Fakulta Matematiky, Fyziky a**  
**Informatiky**

**Tvorba príbehov v počítačových**  
**hrách**

**Bakalárska práca**

Evidenčné číslo: ...

Študijný program: Aplikovaná informatika

Študijný odbor: 9.2.9. aplikovaná informatika

Školiace pracovisko: Katedra Aplikovanej Informatiky

Školiteľ: RNDr. Jozef Šiška, PhD.

**Bratislava 2017**

**Tomáš Bakoš**

# Obsah

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Úvod</b>                             | <b>5</b>  |
| <b>2</b> | <b>Východiská</b>                       | <b>7</b>  |
| 2.1      | Inšpirácia . . . . .                    | 7         |
| 2.2      | Plánovanie . . . . .                    | 7         |
| 2.2.1    | Fluent . . . . .                        | 7         |
| 2.2.2    | Stav . . . . .                          | 7         |
| 2.2.3    | Akcia . . . . .                         | 8         |
| 2.2.4    | Riešenie/Plán . . . . .                 | 8         |
| 2.3      | Výpočtová naratológia . . . . .         | 9         |
| 2.4      | Systémy generujúce príbehy . . . . .    | 10        |
| 2.4.1    | Novel Writer system . . . . .           | 10        |
| 2.4.2    | Talespin . . . . .                      | 11        |
| 2.4.3    | Author . . . . .                        | 11        |
| 2.4.4    | Universe . . . . .                      | 12        |
| 2.4.5    | Minstrel . . . . .                      | 12        |
| 2.4.6    | Mexica . . . . .                        | 13        |
| 2.4.7    | Brutus . . . . .                        | 13        |
| 2.4.8    | Fabulist . . . . .                      | 14        |
| 2.5      | Goal Oriented Action Planning . . . . . | 14        |
| 2.6      | Predošlé bakalárske práce . . . . .     | 16        |
| <b>3</b> | <b>Špecifikácia</b>                     | <b>17</b> |
| 3.1      | Dobrodružná textová hra . . . . .       | 17        |
| 3.2      | Herné objekty . . . . .                 | 17        |

|          |  |           |
|----------|--|-----------|
| 3.3      | Generovanie sveta a príbehu . . . . .  | 18        |
| 3.4      | Generovanie do súboru . . . . .        | 18        |
| 3.5      | Generovanie zo súboru . . . . .        | 18        |
| 3.6      | Nastavenie generátora hodnôt . . . . . | 19        |
| <b>4</b> | <b>Návrh</b>                           | <b>20</b> |
| 4.1      | Jazyk . . . . .                        | 20        |
| 4.2      | Architektúra . . . . .                 | 20        |
| 4.2.1    | Balíček gen . . . . .                  | 21        |
| 4.2.2    | Balíček goap . . . . .                 | 22        |
| 4.2.3    | Balíček game . . . . .                 | 23        |
| 4.3      | Reprezentácia herného sveta . . . . .  | 24        |
| 4.4      | Reprezentácia akcií . . . . .          | 25        |
| 4.5      | Generovanie sveta a akcií . . . . .    | 26        |
| 4.6      | Plánovanie príbehu . . . . .           | 26        |
| 4.7      | Hodnotenie príbehu . . . . .           | 27        |
| 4.8      | Ovládanie aplikácie . . . . .          | 28        |
| 4.9      | Herná časť aplikácie . . . . .         | 28        |
| <b>5</b> | <b>Implementácia</b>                   | <b>29</b> |

# 1. 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 nimi a vyhladáujeme ich. Z vlastnej skúsenosti viem, že človeka dokážu pohltiť, prímeť ho k zamysleniu 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 ľudia 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 disciplína výpočtová naratológia, ktorá študuje tvorbu príbehov z výpočtového hľadiska. Zameriava sa na algoritmické procesy ktoré vytvárajú a interpretujú príbehy a tiež na modelovanie štruktúry príbehu z hľadiska vypočítateľných reprezentácií. Povieme si o nej viac v ďalšej kapitole.

Počítačové hry sú mojou záľubou, ktorá ma sprevádza už od malička a teda mi bolo prirodzené zasadiť problematiku tvorby príbehov do počítačovej hry. Hra príbeh zinterktívni a autori, programátori 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 jednoduchú textovú hru, ktorá bude generovať dobre rozvinutý 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, že príbeh buďe zaujímavý, to znamená, že sa nebudú často opakovať tie isté akcie, akcie budú na seba logicky nadväzovať, budú zmysluplné z hľadiska deju atď. Spôsob akým sa dajú vytvárať takéto príbehy popíšeme v nasledujúcej kapitole.

## 2. Východiská

V tejto kapitole opíšeme všetkú potrebnú teóriu, poznatky, metódy, ktoré sú potrebné alebo nám pomohli pri vypracovávaní témy zadania.

### 2.1 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 sklbením mojich dvoch záujmov.

### 2.2 Plánovanie

Plánovanie je problém alebo oblasť, ktorou sa zaoberá 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 upresním nejaké základné definície z tejto oblasti.

#### 2.2.1 Fluent

Fluent je vo všeobecnosti niečo, čo dokáže plynúť ako tekutina[10]. 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.

#### 2.2.2 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.

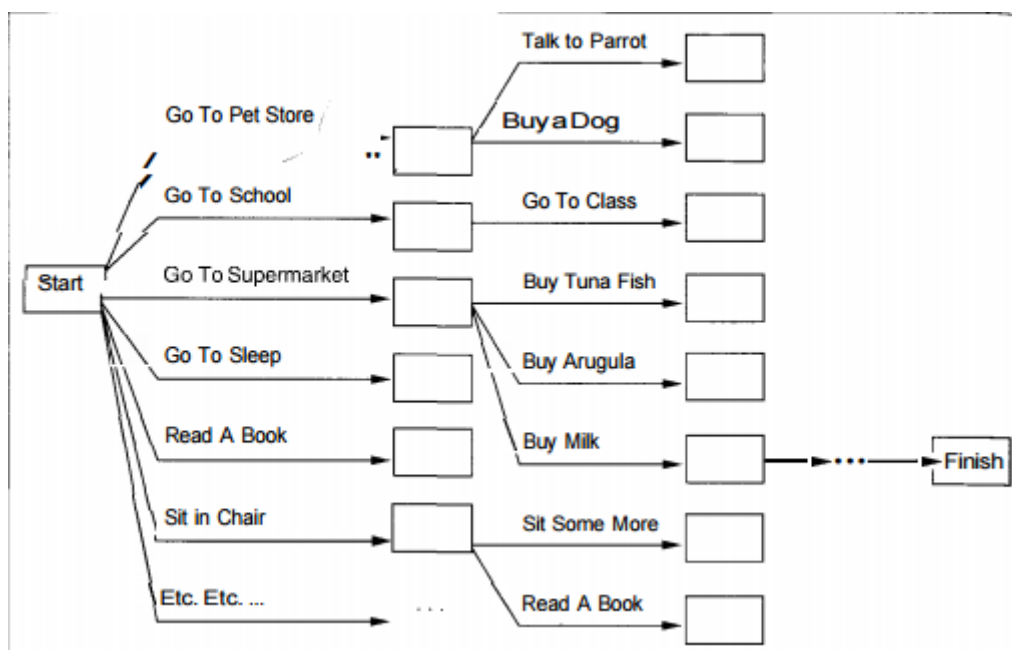
### 2.2.3 Akcia

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 vyvíjať postavy.

### 2.2.4 Riešenie/Plán

Nakoniec k riešeniu (plánu) sa dostávame zadaním 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.

Jedna z reprezentácií ako definovať vzťahy medzi týmito pojmami sú grafové štruktúry. V našom prípade to bude strom. Na obrázku 2.1 môžeme vidieť stavy



Obr. 2.1: Ukážka reprezentácie stavov a akcií [10]

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. Riešenie by mohlo byť aj viac ak by sme mali iný graf alebo doplnili do tohto nové akcie a stavy.

## 2.3 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 štruktúrne prístupy štúdia z ktorých sa vyvinuli rôzne teórie, koncepty a analytické procedúry. Tieto koncepty a modely sú používané ako heuristické nástroje a naratologické teórie hrajú kľúčovú rolu 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 spracovania 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, ďalej 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 princípov 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 ovplyvnená aj lingvistikou napríklad gramatikami príbehov. Je to rýchlo rozvíjajúce sa odvetvie, hlavne vďaka zvýšenému záujmu o interaktívne hry a príbehy, ktoré sa javia ako živé.

## 2.4 Systémy generujúce príbehy

Systémy generujúce príbehy vznikali ako odpoveď na otázky výpočtovej naratológie. Hľadaním všeobecných výpočtových 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 sú 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 ďaleko z troch dôvodov. Po prvé, vyvodenie cieľov zainteresovaných postáv si vyžaduje obrovský priestor na prehľadávanie. Ďalej, ľudia využívajú 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 ľuďom 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 ťažko 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.

### 2.4.1 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 zúvyseli od charakterových črt 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 aktuálneho stavu do nasledujúceho. Druhým bola postupnosť scén ktorá korešpondovala s typom príbehu, ktorý mal byť prerozpráváný. 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.

### **2.4.2 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 ciele 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. súťaživosť, 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, samolúbosti, úprimnosti a inteligencie.

### **2.4.3 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 ciele, ktoré príbehom chce dosiahnuť. A aj ak nemá tak je všeobecne známe, že pri vytváraní príbehu existujú nejaké podvedomé obmedzenia a ciele 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 podciele situácií v príbehu do ktorých chce autor zaviesť niektoré postavy z príbehu alebo do rolí ktoré jednotlivé postavy majú zahrať v príbehu. Môžeme 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 nie sú vidno.

#### **2.4.4 Universe**

O 2 roky neskôr prišiel systém UNIVERSE[5], ktorý generoval scenáre pre radu epizód do telenovely v ktorej hralo rolu veľa postáv a plynulo viaceo súčasne prebiehajúcich a prekrývajúcich sa dejov. UNIVERSE bol prvým storytelingovým systémom, ktorý venoval špeciálnu 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 naplniť. Väčšina 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 transformáciu na samostatný systém generovania príbehu. UNIVERSE priblížil otázku týkajúcu sa vytvárania fiktívneho sveta. Či by sa najprv mal vybudovať svet a potom do sveta vložiť príbeh alebo či by príbeh mal riadiť konštruovanie sveta, kde by sa postavy, lokality a rôzne objekty vytvárali podľa potreby. Tu nastáva rozdiel medzi UNIVERSE-om a AUTHOR-om. Tvorca systému UNIVERSE sa priklonil ku prvej možnosti a implementoval vytváranie postáv nezávisle od deja a na druhej strane bol tvorca AUTHOR-u ktorý sa prikláňal k druhej možnosti.

#### **2.4.5 Minstrel**

Ďalší v chronologickom poradí, ktorý si spomenieme je počítačový program MINSTREL[11]. Tento program vytváral príbehy o Kráľovi Artušovi a jeho rytieroch okrúhleho stola. MINSTREL používal stavebné jednotky pozostávajúce z cieľov a plánov ako dosiahnuť dané ciele. Tieto fungovali na dvoch rôznych úrovniach: autorské ciele a ciele 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 príbeh staval. MINSTREL vytváral príbehy dlhé pól strany až jedna strana.

#### **2.4.6 Mexica**

MEXICA[8] 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 týmito dvoma stavmi tvorí dôležitú časť tvorby príbehu.

Podobne teda bol aj zkonštruovaný program tohto modelu. Pozostáva z dvoch hlavných 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 hypotézy na vytvorenie príbehu. Tento model bol prevratný v tom, že využíval na poháňanie a hodnotenie príbehu aj emocionálne prepojenia a napätia medzi postavami.

#### **2.4.7 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 umožňovali 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 ľuďmi 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.

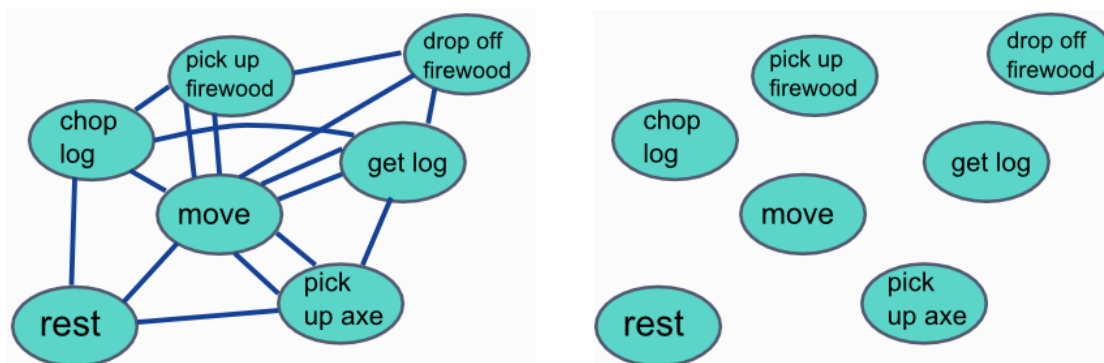
#### **2.4.8 Fabulist**

Najnovším spomedzi spomínaných algoritmov je architektúra FABULIST[9], 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 správanie postavy a následne vytvára plánovacie štruktúry, ktoré vysvetľujú prečo sa tieto postavy zameriavajú na ich cieľ.

### **2.5 Goal Oriented Action Planning**

Goal Oriented Action Planning (GOAP) teda cieľovo orientované plánovanie akcií, 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.

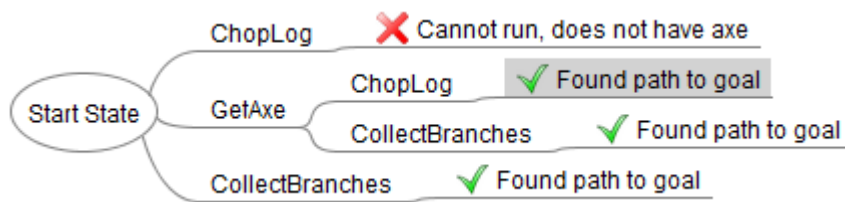
Tento systém vyniká v situáciách, kedy by nám vznikali pri riešení problému veľké a komplikované stavové automaty, kde sa často stáva, že so dôjde do bodu, kedy už nechceme pridávať nové akcie lebo by to vyvolalo priveľa vedľajších efektov a medzier v umelej inteligencii.



Obr. 2.2: Rozdiel medzi reprezentáciou pomocou stavového automatu a GOAP-u[7]

GOAP teda umožňuje z ľavej časti obrázka 2.2 spraviť pravú, kde sú akcie samostatnou jednotkou a nezaoberajú sa vedľajšími efektami svojej činnosti. Tým, že sme jednotlivé akcie odpojili, tak sa môžeme 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 akcií, aby sme nakoniec mohli vybrať postupnosť s najnižšou cenou.

O toto sa stará GOAP plánovač, ktorý sa pozerá na predpoklady a dôsledky akcií a vytvára frontu akcií, ktoré spĺňajú 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 vyzeráť, tak ako popisuje obrázok 2.3.



Obr. 2.3: Plánovací strom GOAP plánovača [7]

## 2.6 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 zápletiiek 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 spúšťaný 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 nehladá na zaujímavosť generovaného príbehu. Touto problematikou sa naša práca zaujímať bude. A nakoniec tvorba príbehov v tejto práci bude zasadená do jednoduchšej 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 rozdielom patrí nepoužitie GOAP plánovanie v práci pána Dobiáša a tiež vygenerovaný príbeh nebol vyjadrený vo forme hry.



## 3. 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.

### 3.1 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 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 použiť.

### 3.2 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.

### 3.3 Generovanie sveta a príbehu

Herný svet bude náhodne generovaný podľa učítých 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.

### 3.4 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.

### 3.5 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.

### 3.6 Nastavenie generátora hodnôt

Náhodné generovanie sveta sa bude generovať podľa pseudo-náhodného generátora hodnôt. Tieto generátory používajú takzvané semiačka na generovanie hodnôt. Používateľ bude mať pri generovaní sveta v našej aplikácii zadať hodnotu semiačka aby sa mu podľa nej 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.

## 4. Návrh

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

### 4.1 Jazyk

Naším 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 rozdielmi 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 objekty sa budú vyskytovať v našej aplikácii.

### 4.2 Architektúra

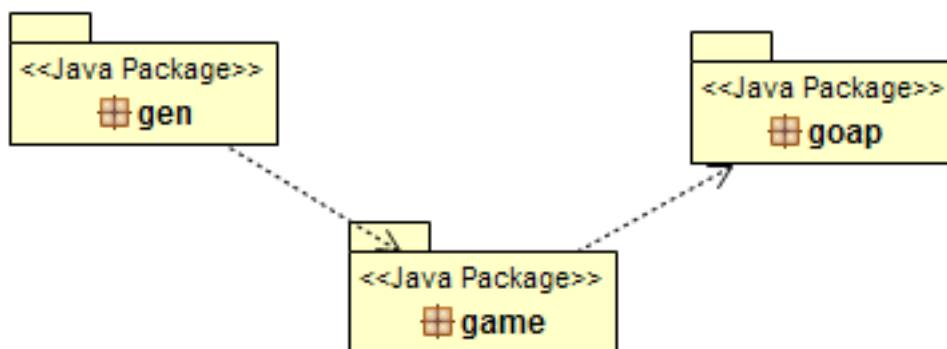
Pred implementáciou aplikácie je vhodné si rozvrhnúť architektúru aplikácie a definovať, čo bude každá z častí obsluhovať. Aplikáciu sme si rozdelili na tri časti a každá z týchto častí bude reprezentovaná balíčkom v Jave. Táto sekcia pojednáva čisto o architektúre aplikácie návrh riešenia jednotlivých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

Prvá časť aplikácie bude obsluhovať načítavanie herných objektov z textových súborov. Potom bude generovať herný svet pomocou načítaných objektov. Ďalej bude sprostredkúvať 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 časťami môžeme vidieť na obrázku 4.1.



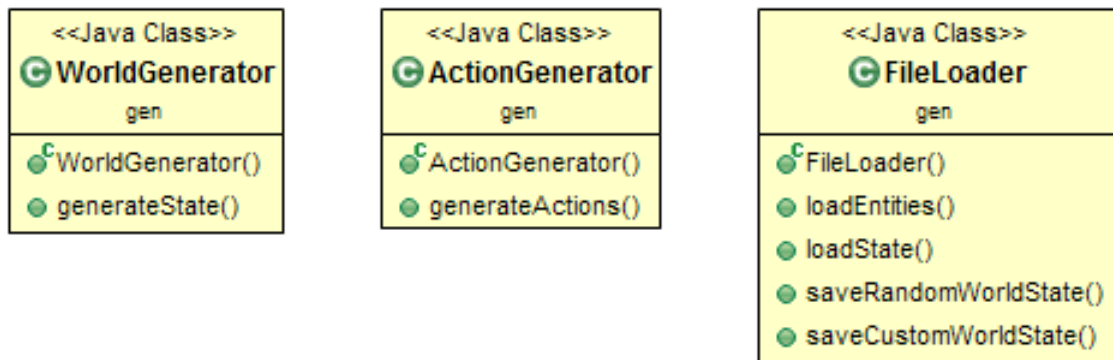
**Obr. 4.1:** Grafické reprezentácia Java balíčkov našej aplikácie.

Na tomto obrázku balíček gen zahrňuje funkcionality generovania spolu s načítavaním a zapisovaním do súboru. Balíček goap v sebe skrýva funkcionality plánovania a reprezentáciu akcií. A posledný balíček game obsahuje samotnú hru a ovládanie aplikácie. Šípky naznačujú vzťahy medzi balíčkami a teda 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.

### 4.2.1 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, ktorý teda bude zabezpečovať prácu so súborom. V tejto triede by sa mali nachádzať štyri funkcie, ktoré obslúžia všetkú interakciu so súborom 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.



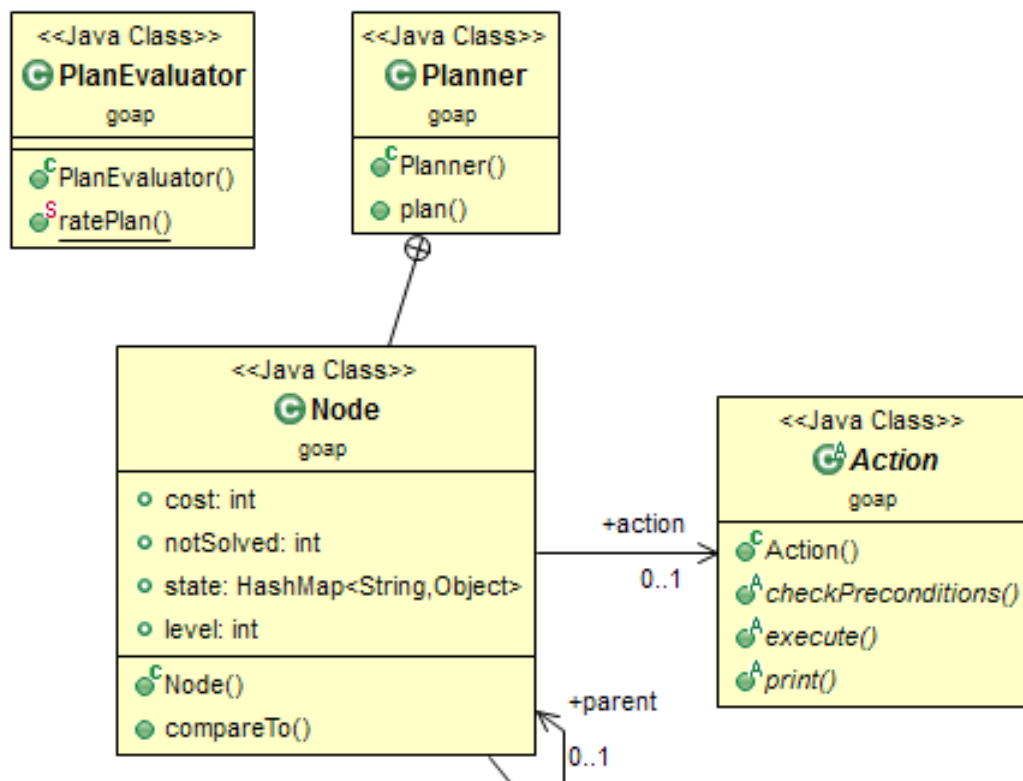
Obr. 4.2: Grafická reprezentácia balíčka gen.

Grafickú reprezentáciu môžeme sledovať na obrázku 4.2. Tento diagram nie je veľmi zložitý, keďže tieto triedy sú od seba nezávislé. Ale majú spoločnú funkcionálnu a preto sú zaradené do tohto balíčka.

#### 4.2.2 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 kritérií. Grafický model môžeme vidieť na obrázku 4.3.

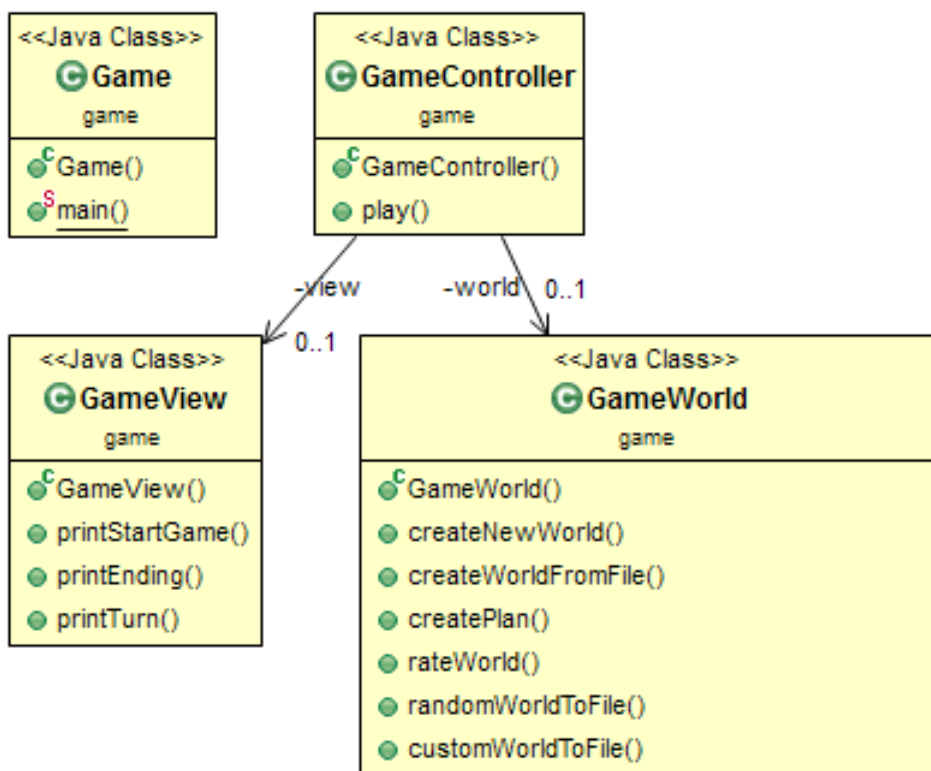


Obr. 4.3: Grafická reprezentácia balíčka goap.

### 4.2.3 Balíček game

V tomto balíčku budeme uchovávať triedy, súvisiace priamo s hracím prostredím aplikácie. Prvú ktorú spomeníme je trieda `GameWorld`. Trieda `GameWorld` bude uchovávať všetky možné informácie o hernom svete. Bude si pamätať všetky načítané objekty 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 využije 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 stvárňovať 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 sprosredkuje koniec príbehu. Ďalšou triedou tohto balíčka by mala byť trieda GameController. Táto sa využije na ovládanie aplikácie. Bude teda poskytovať prepojenie medzi našimi tromi hlavnými funkcionalitami. Umožní používateľovi vybrať si z týchto troch hlavných funkcionalít a spustí vybranú časť aplikácie. A teda bude fungovať aj ako hrací engine. Posledná trieda v tomto balíčku bude trieda Game, ktorá iba spustí GameController a prípadne odchyť výnimky a poskytne spätnú väzbu používateľovi ak sa vyskytne nejaký problém. Na obrázku 4.4 uvádzame grafickú reprezentáciu týchto tried.



Obr. 4.4: Grafická reprezentácia balíčka game.

### 4.3 Reprezentácia herného sveta

Kým sa dostaneme ku generovaniu sveta musíme najprv navrhnuť reprezentáciu herného sveta. Herný svet by mal byť reprezentovaný, niečim ako interpretáciou predikátov. Teda v aplikácii by sme mali definovať 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 vyzeráť podobne ako popisuje obrázok 4.5.

```
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. 4.5:** 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.

## 4.4 Reprezentácia akcií

Akcie budú reprezentované ako jednotlivé triedy v aplikácii. Budú mať za úlohu meniť hodnoty predikátov a tým teda budú môcť ovplyvňovať 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 súčasťou 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.

## 4.5 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 hodnôt 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 inšancií akcie Move(Who, Where) koľko kombinácii dvojíc Who a Where bude načítané zo súboru. Dôvodom je potencionálna možnosť presunúť, každý objekt Who na hociktoré z miest Where.

## 4.6 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 grafový vyhľadávaci algoritmus A\* vyhľadávanie.

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.

Ďalší aspekt, ktorý si A\* vyhľadávanie vyžaduje je akási heuristika alebo niečo čo nám povie ako ďaleko sme od cieľa. V našom prípade si vytvoríme čiastkové ciele príbehu a počet doteraz nevyriešených cieľov v aktuálnom stave použijeme 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. Naša implementácia tejto funkcie, bude spočívať v spočítaní zaujímavostnej hodnoty vrchola a počtu nevyriešených čiastkových cieľov v danom vrchole a podľa tohto súčtu teda bude prioritná fronta usporiadaná.

Po zistení, že sme sa prehľadávaním dostali do cieľového stavu, teda že sme našli konkrétny plán, využijeme referenciu na rodiča v našej triede Node a zrekonštruujeme plán tak ako má byť od začiatku do konca. Použitím tohto algoritmu by nájdená postupnosť akcií mala byť najkratšia možná a naše plánovanie malo byť dostatočné rýchle na to aby sme aj tisícky plánov vedeli naplánovať behom pár sekúnd.

## 4.7 Hodnotenie príbehu

Hodotiť príbeh budeme formou analyzovania štruktúry príbehu. Každý príbeh bude mať svoje hodnotenie v číselnej forme a keď sa vyskytne nejaká vlastnosť v príbehu, ktorú označíme 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, návaznosť akcií teda či sú akcie, ktoré na seba naväzujú, prerušené inou akciou alebo nie, akou akciou začína príbeh. Po tomto ohodnotení sa hodnota priradí k danému svetu, aby sa neskôr mohol vybrať ten najzaujímavejší.

## 4.8 Ovládanie aplikácie

Aplikácia sa bude ovládať pomocou príkazov do konzoli. Používateľ vždy dostane na výber zoznam príkazov, ktoré v danom štádiu aplikácie môže 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.

## 4.9 Herná časť aplikácie

Samotná hra bude taktiež reprezentovaná 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 používateľovi poskytne 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. Ďalším výpisom budú pozície na ktoré hráč môže posunúť svojho hrdinu a 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.

## 5. Implementácia

TODO

# 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] OWENS, Brent. 2014. *Goal Oriented Action Planning for a Smarter AI*. [online]. : 2014. [cit. 8.4.2013]. Url: <https://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793>.
- [8] PÉREZ, Rafael Pérez, “Mexica: a computer model of creativity in writing”, diz. pr., The University of Sussex, 1999.

- [9] RIEDL, Mark O. - YOUNG, R. Michael 2010. *Narrative Planning: Balancing Plot and Character* : Journal of Artificial Intelligence Research, 2010. .
- [10] STUART, Russell - NORVIG, Peter 2009. *Artificial Intelligence: A Modern Approach* : Prentice Hall, 2009. .
- [11] TURNER, Scott R. “Minstrel: a computer model of creativity and storytelling”, diz. pr., University of California at Los Angeles, Los Angeles, 1993.