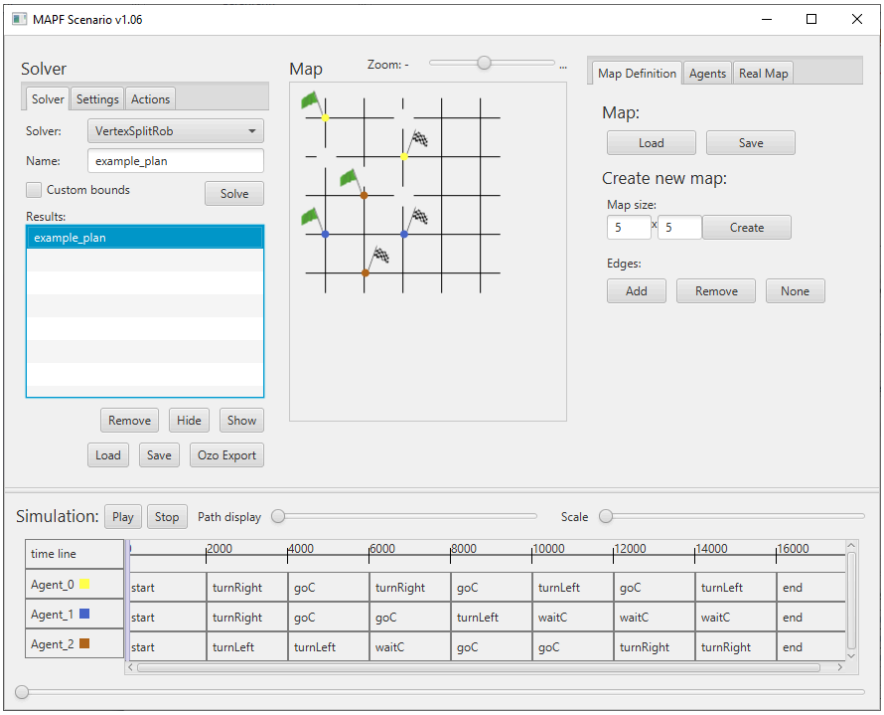
A. Prílohy

A.1 Užívateľská dokumentácia k MAPF Errors and Collision Detection Simulator

V tejto prílohe si ukážeme, čo náš simulátor dokáže a ako sa s ním pracuje. Ilustrujeme si to na klasickom scenári jeho použitia. Pre program najprv vygenerujeme vstup, ten potom do simulátoru načítame, následne agentom do plánov vložíme chyby a nakoniec sa pozrieme na jeho vykonávanie a to ako v prípade, že kolízia nastane, tak aj v prípade, keď kolízia nenastane.

A.1.1 Vygenerovanie vstupu

Ako sme už v kapitole 6.1 povedali, vstupom programu sú textové súbory s príponou .solr, ktoré obsahujú mapu, agentov a ich plány. Tie sú generované programom MAPF Scenario (Obrázok A.1), ktorý vznikol vrámci práce Krasičenko (2018). Na Obrázku A.1 je systém, ktorý budeme po zvyšok dokumentácie používať.

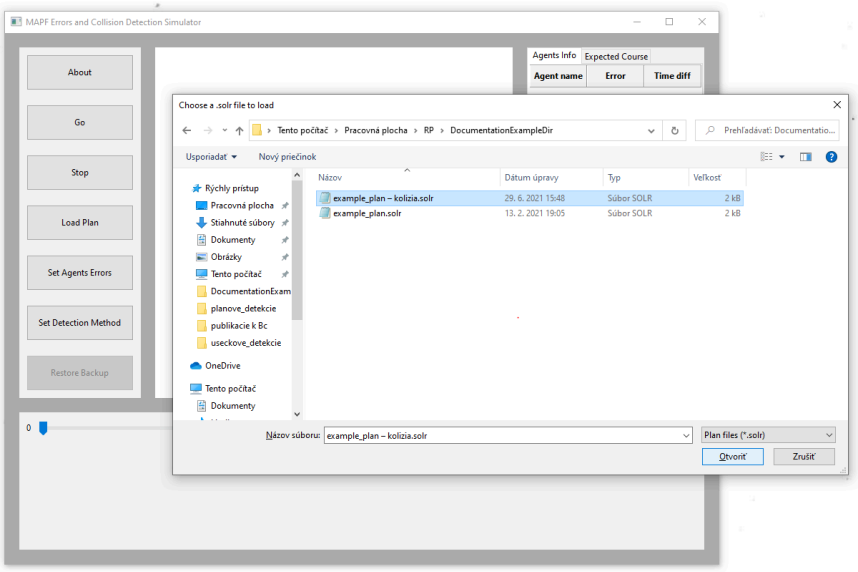


Obrázek A.1: MAPF Scenario so systémom, ktorý budeme po zvyšok dokumentácie používať.

MAPF Scenario má schopnosť generovať plány tvorené rôznymi množinami akcií. Plány vytvorené niektorými množinami akcií ale nemusia byť agentmi priamo vykonateľné. Náš program si preto plány z MAPF Scenario automaticky a transparentne konvertuje na plány, ktoré sú tvorené našimi akciami a sú priamo vykonateľné agentmi.

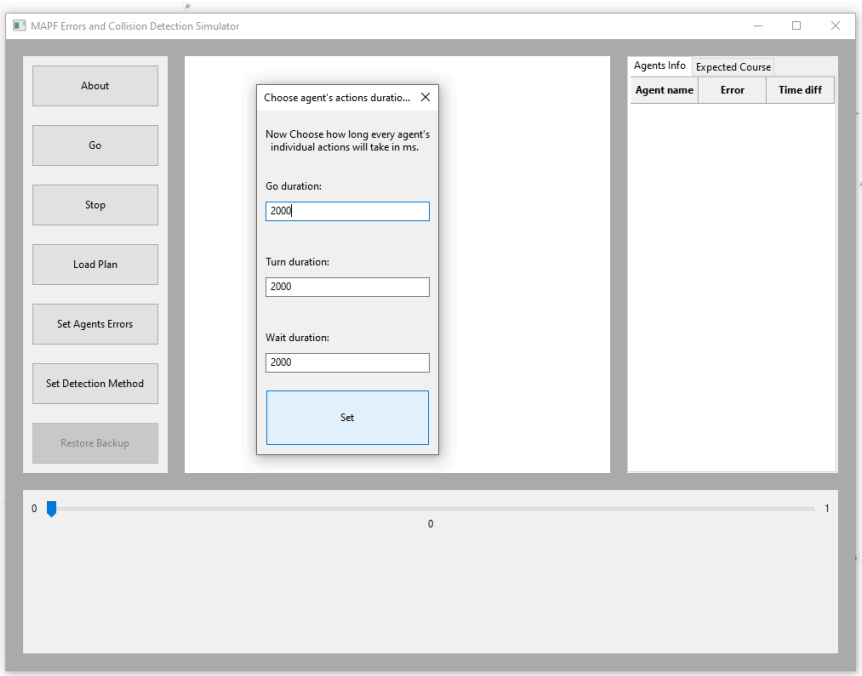
A.1.2 Načítanie vstupu

Keď sme si vytvorili vstup, tak ho do nášho programu načítame. To spravíme krokmi počínajúcimi obrázkom A.2.



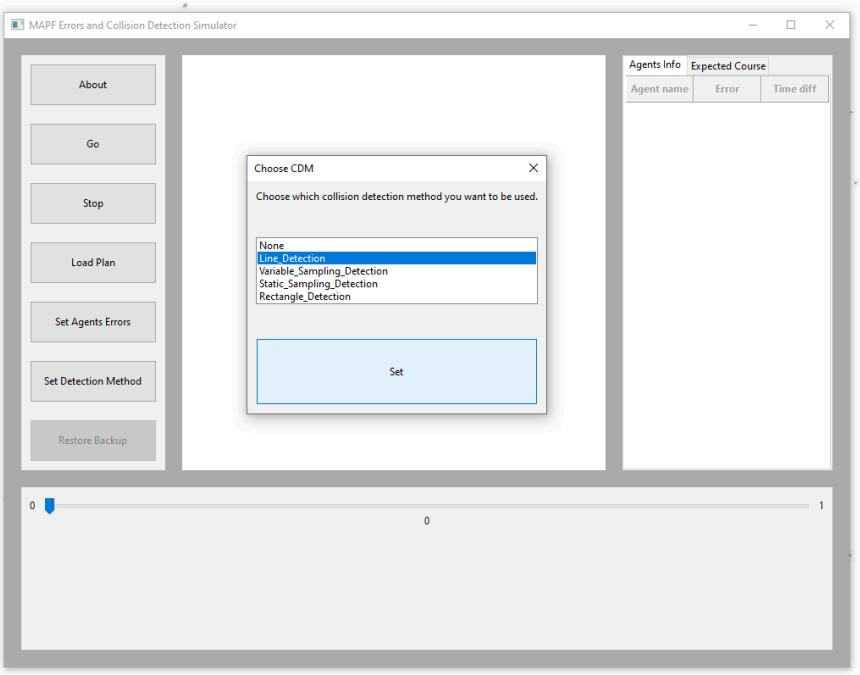
Obrázek A.2: Stlačením tlačítka *Load Plan* sa nám otvorí okno, kde si zvolíme .solr súbor.

Keď si .solr súbor vyberieme, tak sa nám ukáže okno A.3, kde si vyberieme trvanie akcií. Zmena hodnoty *Wait duration* taktiež zmení dobu trvania akcií *Begin* a *End*.



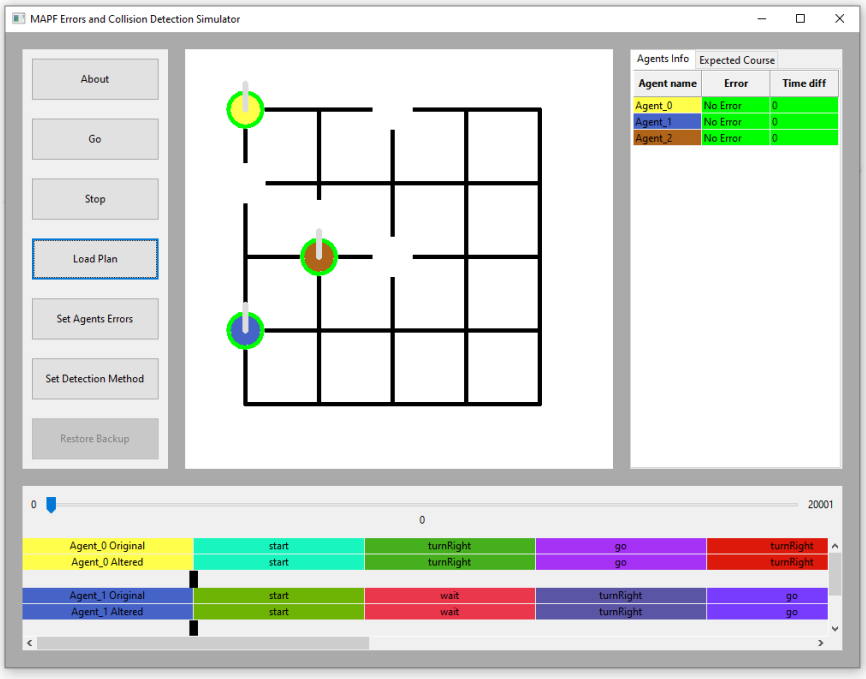
Obrázek A.3: Stlačením tlačítka *Set* nastavíme trvanie akcií krokov plánov.

Po ňom sa dostaneme k oknu výberu metódy detekcie kolízií vyobrazenom na obrázku A.4. Pokiaľ žiadnu nechceme, tak si môžme vybrať možnosť *None*.



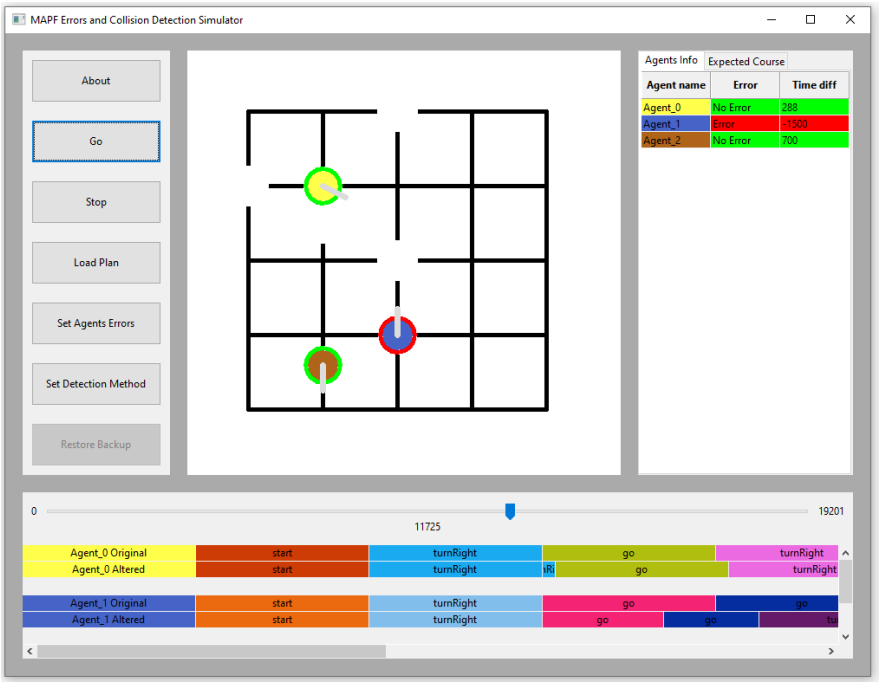
Obrázek A.4: Výber metódy detekcie kolízií, pričom si vybrať môžme iba jednu.

Po vybratí metódy sa proces načítania plánu končí a dostávame sa k načítanému interface-u ako na obrázku A.5.



Obrázek A.5: Program v stave s načítaným vstupom.

Teraz si na obrázku A.6, kde agenti nesprávne vykonávajú svoje plány, vysvetlíme prvky užívateľského prostredia programu.



Obrázek A.6: Agenti chybne vykonávajúci svoj plán

Tabuľka vpravo nám ukazuje, ktorý agent má ako vážne meškanie či náskok, oproti pôvodnému plánu. Tabuľka má tri stĺpce.

Stĺpec *Agent name* obsahuje meno agenta, ktorého sa konkrétny riadok týka. Farba jeho pozadia je farba agenta na mape, ktorá mu bola špecifikovaná v pláne.

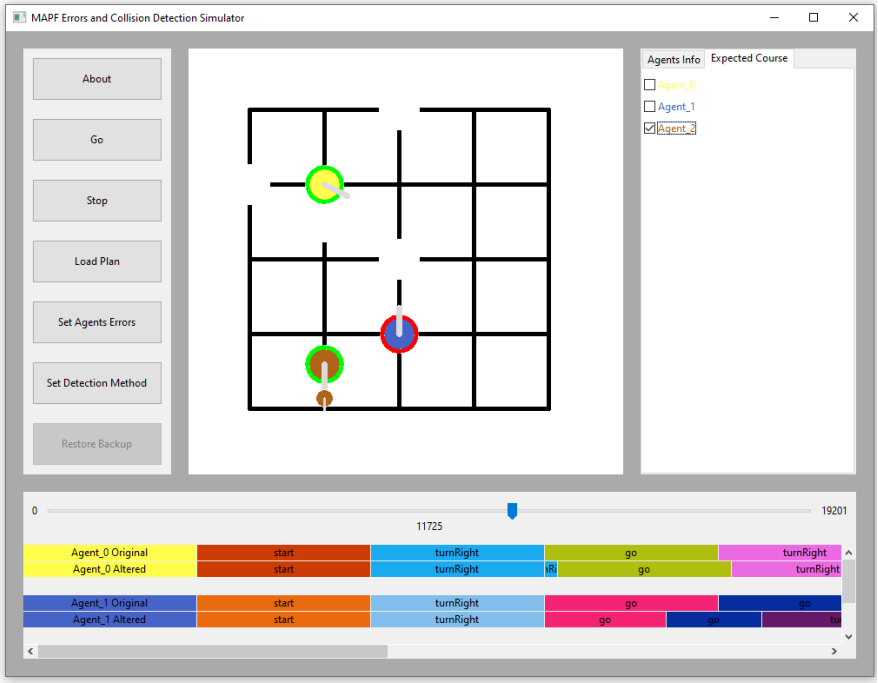
Stĺpec *Error* značí, či aktuálne meškanie či náskok agenta, oproti pôvodnému plánu, je považované za moc veľké, alebo nie. Táto hranica je nastavená na 1000 milisekúnd. Červená farba pozadia znamená moc veľké meškanie či náskok, zelená znamená menšie. Prípadne čierna farba znamená, že sa agent stratil.

Nakoniec stĺpec *Time diff* značí aktuálne meškanie či náskok agenta oproti pôvodnému plánu v milisekundách. Či sa jedná o meškanie alebo náskok spoznáme tak, že náskok je záporné číslo, pričom meškanie je kladné číslo.

Farebné prstence agentov na mape majú rovnakú farbu i sémantiku, ako stĺpec *Error* riadku agenta v tabuľke vpravo. Dovoľujú nám teda lepšie vidieť na mape to, ktorý agent má ako vážne meškanie či náskok oproti pôvodnému plánu.

Pod agentmi môžme vidieť ich plány. Pre každého agenta máme dva riadky. Prvý riadok značí jeho pôvodný plán, pričom druhý riadok značí plán, ktorý reálne vykoná.

Počas behu simuláciu by nás mohlo zaujímať, kde by sa agent, ktorý chybne vykonáva svoj pôvodný plán, mal podľa neho nachádzať. Keď prejdeme vpravo hore z karty *Agent Info* na kartu *Expected Course*, tak sa nám otvorí výber ukázaný na obrázku A.7.

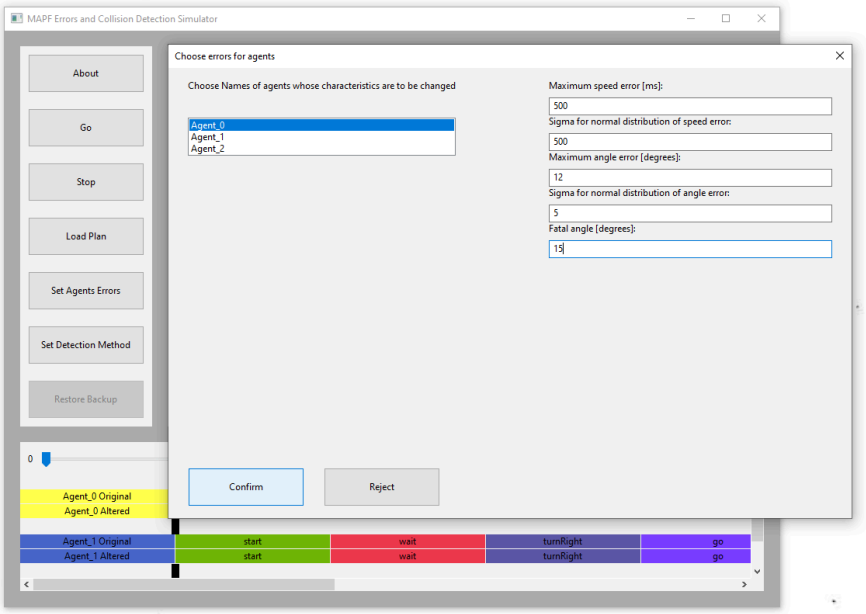


Obrázek A.7: Kde by sa mal agent nachádzať podľa pôvodného plánu.

Tu môžme zaškrtnúť agentov, o ktorých chceme vedieť, kde by sa podľa pôvodného plánu mali nachádzať. Keď nejakého agenta zaklikneme, tak sa nám na mape objaví menší agent rovnakej farby, ktorý túto informáciu reprezentuje.

A.1.3 Vloženie chýb do plánov agentov

Kebyže chceme vložiť agentom chyby do plánov, tak stlačíme tlačítko *Set Agents Errors*. To nám otvorí okno ukázané na obrázku A.8.



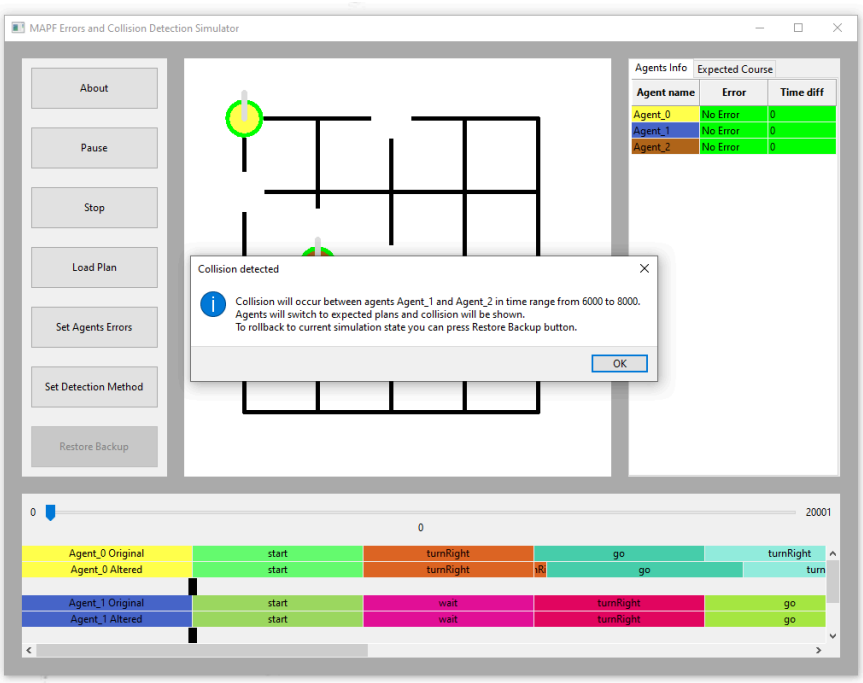
Obrázek A.8: Na ľavo si vyberieme agentov, ktorým chceme chybové charakteristiky sprava priradiť. Následne stlačením tlačítka *Confirm* im tieto charakteristiky nastavíme. Ak máme vybratého len jedného agenta, tak sa nám vpravo zobrazia jeho aktuálne chybové charakteristiky

Ak chceme chyby z plánu agenta odstrániť, tak stačí všetky jeho chybové parametre nastaviť na nula. (Parameter *Fatal angle* je ignorovaný, ak je *Maximum angle error* rovný nule.)

Následne sa znova dostaneme do stavu na obrázku A.5. Na rade je už len pustiť simuláciu pomocou tlačítka *Go* a zistiť, či nami zvolená metóda nájde kolíziu.

A.1.4 Metóda detekovala kolíziu

Ak nami vybratá metóda detekcie kolíziu nájde, tak sa nám zobrazí okno z obrázku A.9.



Obrázek A.9: Metóda našla kolíziu. Simulátor teraz zobrazí systém, v ktorom očakáva kolíziu. Keď sa budeme chcieť vrátiť k pôvodnej simulácií, tak stlačíme tlačítko *Restore Backup*. Pokiaľ nebudeme chcieť dostávať správy o kolízií, jednoducho pomocou tlačítka *Set Detection Method* si zvolíme možnosť *None*.

A.1.5 Kolízia nenastane, alebo sme zvolili metódu None

V prípade že kolízia nenastane, alebo sme si zvolili metódu *None*, budú agenti neprerušene vykonávať svoj plán až do konca. V takom prípade bude simulácia vyzerať ako na obrázku A.6, kde sme si vysvetľovali prvky užívateľského rozhrania.

A.2 Programátorská dokumentácia k MAPF Errors and Collision Detection Simulator

V tejto prílohe sa budeme zaoberať informáciami relevantnými niekomu, kto by mal záujem upravovať zdrojový kód simulátoru, alebo by chcel mať predstavu, ako je simulátor napísaný.

A.2.1 Základné informácie

Simulátor je napísaný v programovacom jazyku C++, pričom pre užívateľské prostredie využíva multiplatformovú open source knižnicu wxWidgets, ktorá je taktiež napísaná v C++.

Program bol otestovaný iba na operačnom systéme Windows 10.

A.2.2 Myšlienkové základy

Program používa takzvaný globálny pohľad na vec. To znamená, že existuje vševediaca entita, ktorou je trieda *Simulation*, ktorá presne vie, kde sa ktorý agent nachádza a aký má plán.

Agenti sú neomylní. To znamená, že svoje plány vždy vykonajú bez chyby. Keďže ale chceme simulovať chyby agentov, tak nám táto skutočnosť implikuje to, že do plánov agentov budeme musieť chyby vnášať vopred a vedome.

Keďže budeme do plánov vnášať chyby vopred a vedome, tak presne vieme, kedy a kde každá jedna chyba nastane. Toto si uložíme a neskôr túto informáciu budeme používať pre ich jednoduché zobrazovanie počas priebehu simulácie.

A.2.3 Hlavné triedy a ich komunikácia

Hlavné triedy programu vieme rozdeliť na dve skupiny.

Prvou budú triedy, ktoré sa zaoberajú samotným simulovaním. Tými sú *Simulation*, *Agent* a triedy implementujúce metódy detekcie kolízií. Tie spĺňajú interface zobrazený na algoritme 1.

Trieda *Simulation* obsahuje pole všetkých agentov, ktorí sú inštancie triedy *Agent*, ktorí v danej simulácií vystupujú. Následne ešte obsahuje mapu, reprezentovanú dvojrozmerným polom, na ktorej sa simulácia odohráva. Zaujímavosťou je, že trieda *Simulation* vôbec nepoužíva žiadnu z tried Zaoberajúcou sa GUI. Z toho plynie, že samotná simulácia nie je viazaná na knižnicu wxWidgets. Dokonca by sa dala upraviť tak, aby fungovala len ako CLI program.

Druhou skupinou budú triedy, ktoré sa zoberajú užívateľským rozhraním.

Hlavnou z nich je *MyFrame*. Tá reprezentuje hlavné okno programu, na ktorom sa všetky ostatné prvky užívateľského prostredia nachádzajú. Táto trieda v sebe taktiež obsahuje samotnú, simuláciu ktorej stav mení.

Ostatné triedy, ktoré si *MyFrame* počas celého behu programu udržuje, si prejdeme len heslovite.

Buttons_Panel reprezentuje panel tlačítok.

Agent_Info_Panel reprezentuje tabuľku agentov a ich meškaní.

Draw_Panel reprezentuje mapu a je zodpovedný za vykresľovanie simulácie.

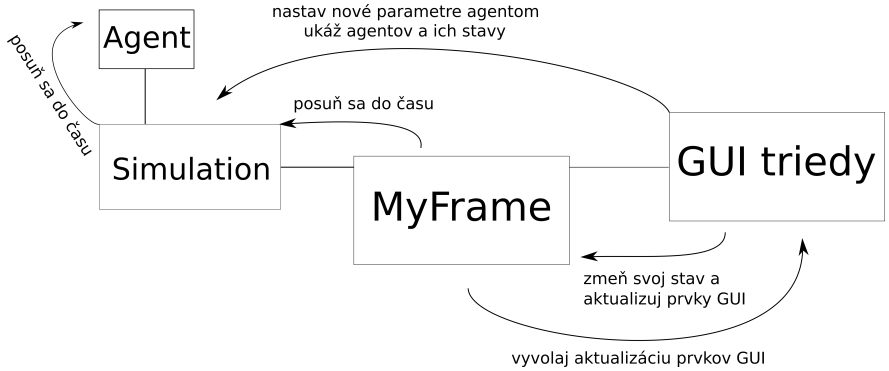
Extended_controls_panel reprezentuje slider času v ktorom sa simulácia nachádza a tabuľky plánov agentov.

Error_Dialog reprezentuje dialóg na nastavovanie chýb agentom.

Na záver *Agents_CheckBoxes_Panel* reprezentuje kartu *Expected Course*.

Skoro Všetky tieto triedy disponujú referenciou na triedu *MyFrame*, cez ktorú si zo simulácie dotazujú nimi potrebné dáta, prípadne menia stav samotnej simulácie ako aj *MyFrame*.

Hlavnú komunikáciu medzi triedami si načrtneme na obrázku A.10.



Obrázek A.10

Kebyže do simulátoru chceme integrovať novú techniku detekcie kolízie, tak musíme pridať presne tri riadky. Prvý riadok sa nachádza v GUI triede *Collision_detection_Dialog*, ktorej musíme v konštruktore povedať, aby užívateľovi v dialógu ponúkala aj našu novú techniku detekcie. Zvyšné dva riadky patria do funkcie *Simulation::move_to_time*, kde sa pomocou if else vetiev inicializuje ukazovateľ na konkrétnu detekciu kolízií, ktorá bude použitá. Tu bude treba pridať kód v štýle ak názov vybratej techniky je názov našej novej techniky, tak ju inicializuj do pred pripraveného ukazovateľa. Po týchto krokoch je už naša nová technika detekcie kolízie plne integrovaná.