ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hofmann** Jméno: **Lukáš** Osobní číslo: **474611**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Použití automatizovaného plánování pro chytré chování hráče v tahové počítačové hře

Název bakalářské práce anglicky:

Using automated planning for smart player behavior in a turn-based computer game

Pokyny pro vypracování:

Úkolem práce je navrhnout a naprogramovat prototyp počítačové hry v prostředí Unity s použitím technik automatizovaného plánování. Pravidla hry budou vycházet z principů tahových logických her jako například Lara Croft GO, nicméně v inverzním smyslu. V navrhované hře bude hrát plánovací algoritmus klasického plánovaní a úkolem hráče bude upravit omezení prostředí tak, aby plánovač nebyl schopen najít řešení.
1) Nastudujte vývojové prostředí Unity a techniky automatizovaného plánování.
2) Navrhněte upravená pravidla hry tak, aby byl schopen daný plánovací systém problémy řešit.
3) Navrhněte a naprogramujte danou hru s integrovaným plánovačem.
4) Validujte a verifikujte (ideálně s lidskými hráči), funkčnost hry.
5) Experimentálně ověřte, že plánovač je schopen dané problémy řešit i pro kombinatoricky těžké konfigurace hry.

Seznam doporučené literatury:

[1] Malik Ghallab, Dana S. Nau, Paolo Traverso: Automated planning - theory and practice. Elsevier 2004, ISBN 978-1-55860-856-6.
[2] Stefan Edelkamp, Stefan Schroedl: Heuristic Search: Theory and Practice. Morgan Kaufmann. 2012, ISBN 978-0-12372-512-7.
[3] Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#, Addison-Wesley Professional; 1 edition (July 21, 2014).

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Antonín Komenda, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.09.2020** Termín odevzdání bakalářské práce: **05.01.2021**

Platnost zadání bakalářské práce: **19.02.2022**

Ing. Antonín Komenda, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Použití automatizovaného plánování pro chytré chování hráče v tahové počítačové hře

Bakalářská práce

Lukáš Hofmann

Studijní program: Softwarové inženýrství a technologie
Vedoucí: Ing. Antonín Komenda, Ph.D.

V Praze, Ledna 2021

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, Ledna 2021

.....
Lukáš Hofmann

Abstrakt

Tato práce popisuje vývoj prototypu tahové počítačové hry, která ke svému fungování využívá automatizované plánování. Před implementací studujeme různé návrhy hry, porovnáváme je s kritérii zadání a na základě této analýzy vybereme nejlepší návrh pro naši hru. Práce charakterizuje herní mechaniky včetně objektů, které se ve hře vyskytují a cíle hry. Po návrhu hry se práce zaměřuje na způsob implementování našeho návrhu i to, jak aplikace generuje PDDL soubory, komunikuje s plánovačem Maplan a jak zobrazuje výsledky hráči. Práce také obsahuje výsledky našeho experimentu, ve kterém jsme měřili rychlost, s jakou plánovač vyřeší problémy se zvyšující se složitostí. Mimo experimentů byla hra testována i lidskými hráči, a podle jejich zpětné vazby jsme vylepšili naši aplikaci a naplánovali další potřebná vylepšení, které dělí prototyp od plnohodnotné hry.
Klíčová slova: tahová počítačová hra, automatizované plánování, PDDL

Abstract

This work describes the development of a prototype of turn-based computer game that uses automated planning. Before implementation, we study various game designs and compare them with a specifications of assignment and based on this analysis, select the best design for our game. The work characterizes game mechanics, including objects that exist in the game and goal of the game. After the design part of the game, the work focuses on how to implement our design. How the application generates PDDL files, communicates with the Maplan scheduler and displays solution to player. The work also contains the results of our experiment, in which we measured the speed with which the scheduler solves problems that increases in complexity. In addition to the experiments, the game was also tested by human players, and according to their feedback, we improved our application and planned other necessary improvements, which separates the prototype from full-fledged game.
Keywords: turn-based computer game, automated planning, PDDL

Poděkování

Děkuji panu Ing. Antonínu Komendovi, Ph.D. za vedení bakalářské práce a paní Ing. Michaele Urbanovské za pomoc při propojování aplikace s Maplanem

Obsah

Kapitola 1

Úvod

Tato práce popisuje vývoj prototypu počítačové hry za využití v herním průmyslu velice nestandartní umělé inteligence, automatizovaného plánování. Tato aplikace bude zahrnovat grafické rozhraní pro hráče, které bude zpracovávat vstupy hráče, skript, který bude přetvářet objekty hry na informace, které dokáže zpracovat náš externí plánovač, a další skript, který dokáže převést výstup z plánovače na výsledek pro grafickou část aplikace, která pak ukáže výsledek hráči.

Tato bakalářská práce je rozdělená na tři části. První část se zabývá přípravou před samotnou implementací. Zde je popsán návrh hry, jednotlivé objekty, se kterými hra pracuje a postup jednotlivých levelů. V druhé části je popsána implementace samotného návrhu, jaké komponenty byli k implementaci použity, jakým způsobem se překládají objekty hry do jazyka PDDL pro plánovač a jednotlivé akce PDDL souborů. Třetí část se zabývá měřením a testováním aplikace. Měří se rychlost výpočtu jednotlivých řešení, ať už kolik akcí je potřeba ke splnění jednotlivých úkolů, tak i reálný čas, který plánovač potřebuje k výpočtu, a to vše ve srovnání s množstvím překážek, které plánovač bude muset překonat. Zároveň se aplikace testuje na lidských hráčích, kteří otestují hratelnost hry i grafické rozhraní.

V závěru si popíšeme zpětnou vazbu z testování, ať už jinými lidskými hráči, nebo při vlastní zkoušce programu a jak nedostatky, které během tohoto testování nastaly, hodláme opravit v budoucích verzích aplikace.

1.1 Motivace

V dnešní době už asi všichni známe videohry. Hry nám nabízejí výzvu, kterou se snažíme překonat. Někdy jsou výzvy tak těžké, že se nám zdá, že si vývojáři, kteří designují levely nebo hráči, kteří samotné levely hrají, libují v bolesti z toho, že ani na stý pokus se nepodaří danou překážku překonat. Jak by to asi vypadalo, kdyby si umělá inteligence, která se stará o obtížnost dané hry ať už ovládáním nepřátelských postav nebo různých pastí, vyměnila role s hráčem, který se (někdy marně) snaží splnit cíle dané hry.

Obsahem této bakalářské práce je vytvořit prototyp počítačové hry, kde umělá inteligence bude v roli postavy, která musí splnit předem určený úkol přes nástrahy dané mapy a kde se hráč stane designérem jedné takové mapy, ve které může ztěžovat postup umělé inteligence k cíli využitím omezených zdrojů.

Kapitola 2

Návrh hry

Tato část popisuje téma hry a základní postup každého levelu. V této kapitole jsou vypsané hry, ze kterých jsme při návrhu čerpali inspiraci a v čem si jsou podobné našemu prototypu. Jsou zde také představené objekty, které se vyskytují v naší hře a jejich role. Navíc je zde popsán hlavní cíl hry a jak ho dosáhnout.

2.1 Téma hry

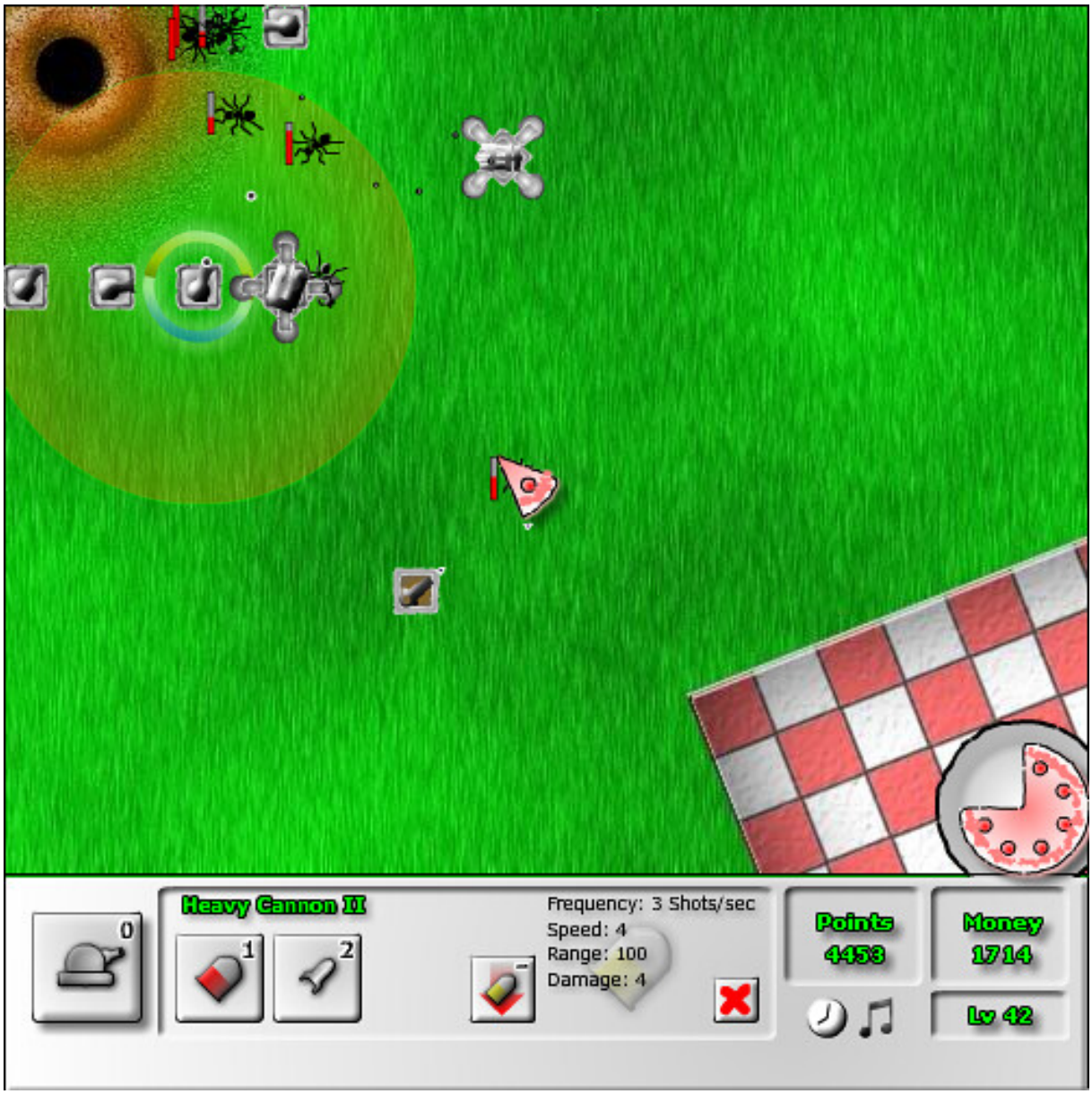
V této části si shrneme jednotlivé návrhy na téma hry. Jejich pozitiva a negativa. Hry, od kterých se můžeme při těchto návrzích inspirovat. V závěru této části si na základě předchozího výzkumu řekneme, na jakém tématu jsme se rozhodli.

2.1.1 Tower Defense

Tower defense hry jsou hry, ve kterých stavíte útočné věže, které střílí po vlnách nepřátel snažící se dostat z bodu A do bodu B. Většina tower defense her se dá rozdělit na dva typy. Na ty, kde nepřátelé chodí po předem vyznačené cestě a hráč staví věže kolem cesty jako například ve hře Kingdom Rush a na ty, kde máte prázdnou plochu a pomocí věží definujete cestu. Jedním takovým příkladem je například hra Tower wars. V prvním typu žádná umělá inteligence většinou není, nepřátele chodí bezmyšlenkovitě po vyznačené trase.

Z druhého typu si můžeme vzít například hru s názvem Antbuster. Jak lze vidět na obrázku 2.1, v této hře hráč staví věže, které střílí po mravencích. Mravenci neustále vylézají z mraveniště a snaží se ukrást dort a přinést ho zpátky do mraveniště. Nejzajímavější na této hře je to, že mravenci si vybírají cestu s nejmenším nebezpečím. Takže pokud umělá inteligence vyslala mravence jednou cestou a zjistila, že mravencům se nedaří projít, jednoduše vyslala mravence cestou jinou.

Ačkoliv by na tomto typu hry bylo vidět, jak automatizované plánování reaguje na vstup hráče v podobě stavění věží, tak tower defense hry nejsou tahové, tudíž by se na tomto typu hry velmi obtížně demonstroval právě tento typ umělé inteligence.



Obrázek 2.1: V této ukázce ze hry Antbuster je vidět základní princip hry. Vlevo nahoře se nachází mraveniště a vpravo dole se nachází dort. Všimněme si, že na tomto obrázku se mravenci snaží jít k dortu přes pravou horní stranu mapy, jelikož směrem dolů od mraveniště mají menší šanci přežít. [1]

2.1.2 Tahová logická hra na styl Lara Croft GO

Lara Croft GO je logická tahová hra, kde hráč hraje za Laru Croft a přes nebezpečné prostředí se snaží dostat na konec mapy. V cestě jí ale stáli nástrahy jako například pilové kotouče nebo hadi, a všechny tyto nástrahy dodržovali jasně daná pravidla, takže hráč věděl, jak se prostředí zachová pokaždé, kdy se pohne. Jeden příklad takového levelu lze vidět na obrázku 2.2.

Pro splnění požadavků bychom museli prohodit role hráče a umělé inteligence, tedy hráč by naplánoval nástrahy mapy a umělá inteligence by se snažila najít cestu do cíle. Na rozdíl od tower defense hry je tato hra tahová, takže by zde bylo lépe vidět, jak se zachová umělá inteligence. Navíc to, že pasti dodržují předem daný patern, zjednodušuje hráči si představit, jaký jím rozmístěné nástrahy budou mít vliv na mapu a samotnou umělou inteligenci.

Tento návrh má ale jeden drobný problém. Ve hře Lara Croft GO jsou levely designovány tak, že se dají projít, bylo by velmi jednoduché vymyslet mapu, kterou by ani umělá inteligence nedokázala porazit. Pokud by hráč měl za úkol zabránit umělé inteligenci k dosažení cíle, jednoduše by využil všechny svoje zdroje a koncentroval by je na co nejmenší plochu. Jelikož v našem případě chceme být schopní vymyslet nemožný scénář pro umělou inteligenci, ale přitom také chceme, aby v tom byla alespoň trochu výzva. Toto by se dalo vyřešit určitým omezením, například že by jedna past musela být vzdálená od druhé o několik polí, ale toto řešení by příliš omezilo volnost hráče při stavbě levelu.

Obrázek 2.2: Zde máme příklad levelu ze hry Lara Croft GO. Zde jsou pole znázorněné pomocí kosočtverců. Na některých polích se nachází překážky. Hadi, kteří hráče uštknou, pokud hráč vstoupí na pole před nimi, a prasklá pole, na které může hráč vstoupit pouze dvakrát, než se pole rozpadne a stane se nepřístupné. [2]

2.1.3 Hitman

Ze hry Lara Croft GO jsme si ujasnili, že kdyby se umělá inteligence snažila projít mapu z jednoho bodu do druhého, hráč by mohl soustředit veškeré zdroje na jednu část v cestě a tím jednoduše zabránit našemu plánovači v průchodu. Na tento problém si můžeme vzít inspiraci například z herní série Hitman. Ukázka ze hry je na obrázku 2.3. V této hře hrajete za vraha a snažíte se zneškodnit předem určené postavy. Zde se postavy, které musíte zabít, pohybují po mapě. Proto jsou i strážci rozmístění po mapě, a nejsou soustředění na jedno místo. Pohyblivé cíle bychom mohli aplikovat na náš prototyp, a tím zabránit soustředění hráčových zdrojů na jedno místo.

Obrázek 2.3: V herní sérii Hitman, ze které pochází tento obrázek, hrajete za profesionálního vraha a vašim cílem je elimininovat všechny cíle na každé mapě. Jak to uděláte, je zcela na vás. [3]

2.1.4 Finální návrh prototypu

Z návrhu Tower defense hry, jsme si ujasnili, že hráč bude potřebovat mít k dispozici překážky, které bude pokládat na mapu a tím ovlivňoval cestu umělé inteligence. Navíc je potřeba, aby hráč viděl jak jeho překážky ovlivňovat chování plánovače, díky čemuž by při následném neúspěchu mohli vidět kde hráče plánovač přelstil a mohl podle toho upravit svou strategii.

Z návrhu tahové logické hry jsme přišli na to, že je potřeba za prvé, nechat umělou inteligenci vybrat si z několika startovních pozic a za druhé, je potřeba umělé inteligenci více možných cílů, nebo udělat cíle pohyblivé. Bez těchto opatření by hra nebyla pro hráče žádnou výzvou. Při návrhu Hitmana jsme si ukázali prostředí hry, ve kterém bychom mohli zasadit naší počítačovou hru. V tomto prostředí je pochopitelné, že umělá inteligence bude moci vstoupit na mapu z vícero možných vstupů a zároveň lze odůvodnit, proč je cíl mise pohyblivý.

Proto se příběh našeho prototypu odehrává v situaci, kde hráč bude plánovat ochranu důležité osoby v našem prototypu je pojmenujeme jako VIP. Umělá inteligence bude hrát za vraha, který se pokusí zabít jednu z těchto VIP postav. Tímto návrhem odstraníme problém, který by se vyskytl při návrhu tahové logické hry na styl Lara Croft GO, jelikož

vrah může přijít jakýmkoliv možným vstupem do oblasti a jelikož vrahovo cílem je osoba, kterou se pokusí zabít, dává smysl, že se vrahův cíl bude hýbat. Hra bude tahová, kde se postavy budou pohybovat po čtvercových polích. Hráč při plánování ochrany bude dávat strážce na tato pole a bude určovat které oblasti má strážce hlídat. Hra bude znázorňovat po celou dobu, které pole strážce vidí, čímž splníme to, co jsme si ujasnili u návrhu tower defense hry. Hráč bude přesně vidět, které pole jsou hlídaná a vrah bude také vědět, kterým polím se má vyhnout. Ukázka hry: Obrázek 2.4.

Obrázek 2.4: Takto vypadá náš prototyp. Stráže, znázorněné modrou elipsou, pokrývají pole zelenou barvou, na které vrah nemůže vstoupit. Vrah, reprezentovaný žlutou elipsou, se snaží těmto polím vyhnout a dostat se k VIP postavám, které jsou reprezentované zelenou elipsou.

2.2 Objekty

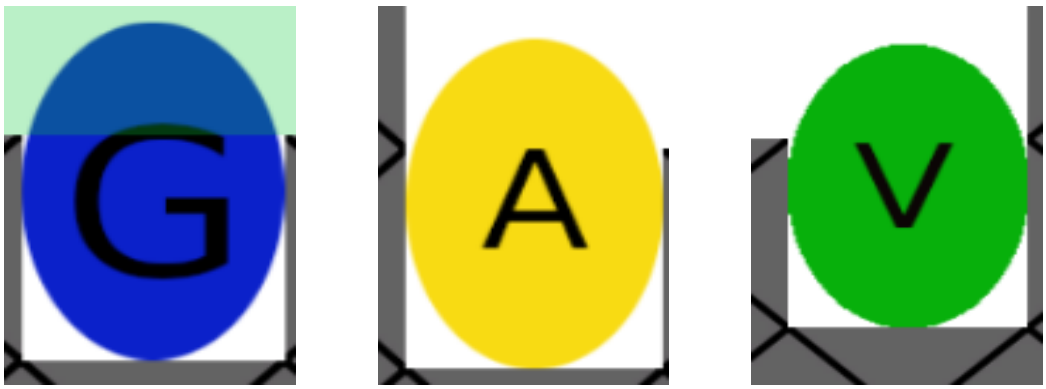
Objekty jsou entity, se kterými hra spolupracuje. Jsou zde zahrnuty postavy, které se pohybují po mapě a pole, ze kterých se mapa skládá. Tato část popisuje důvod, proč se tento typ objektu vyskytuje v naší hře a jak s každým objektem naše hra pracuje.

2.2.1 Postavy

Ve hře jsou 3 typy postav. Jedním z nich jsou strážci (obrázek 2.5a). Strážci jsou jediné postavy, které ovlivňuje hráč, ať už směr, kterým se dívají, cestu, kterou chodí nebo kde vůbec stojí a jsou hráčovo jediným nástrojem proti počítačem ovládaným vrahovi. Stráže vytváří pole vize v závislosti, kterým směrem se dívají a v závislostí na okolních zdech, přes které strážci nevidí.

Druhým typem postavy je vrah (obrázek 2.5b). Vrah je ovládán umělou inteligencí, který se snaží najít mezeru v hlídce stráží. Vrah se na mapu dostane přes tzv. vstupní pole. Vrah se hýbe simultánně se strážemi, ale na rozdíl od stráží, kteří mají pevně daný patern, ve kterém se mají pohybovat. Vrah se může pohnout, nebo vyčkat na lepší příležitost podle potřeby.

Třetím typem postavy je VIP (obrázek 2.5c). Hráč se pomocí stráží snaží zabránit vrahovi, aby se k této postavě dostal. VIP je jediná postava, která je přítomná na mapě již od začátku. VIP se mohou stejně jako ostatní postavy hýbat, čímž mohou hráči značně ztížit úkol. VIP může být na mapě více, při čemž hráč prohrává, pokud se vrah dostane alespoň k jednomu z nich.



(a) Strážce (b) Vrah (c) VIP

Obrázek 2.5: Postavy, které se pohybují po herním plánu.

2.2.2 Pole

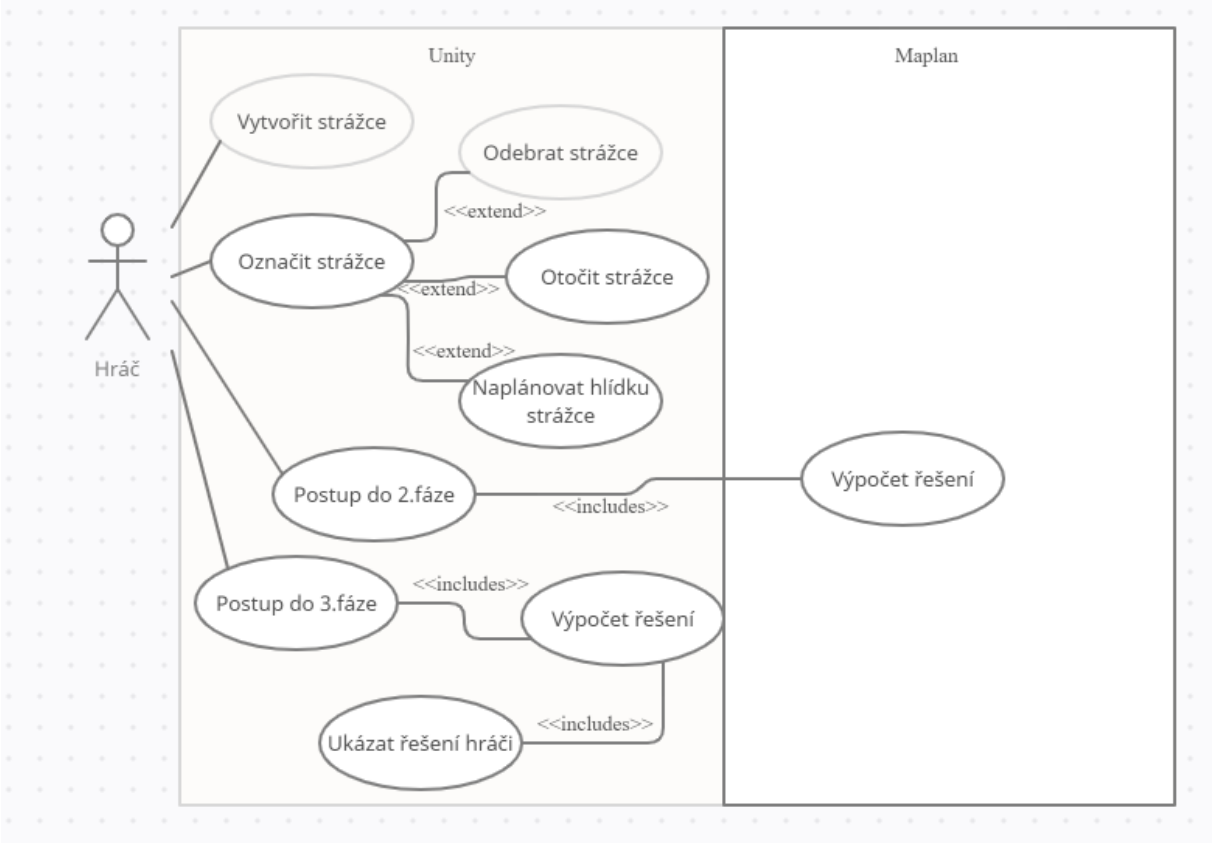
Ve hře jsou 4 typy polí. Bílé pole reprezentují podlahu. Všechny postavy se pohybují po těchto polích, každé takové pole sousedí pouze s políčky, se kterými sdílí hranu. Další je šedé pole s křížkem, které představuje zeď, která blokuje pohyb postav i vizi strážců. Další typ pole má šedou barvu, je pouze na místech, na kterém je i podlaha a slouží jako vstupný bod pro vraha. Na mapě je vždy několik možných vstupů pro vraha, ze kterých si vrah vybere tu nejlepší v závislosti na rozmístění strážců. Posledním typem je pole vize a má průhlednou zelenou barvu. Jsou generované pouze strážci. Pro účel hry působí na vraha stejně jako zdi, zatímco ostatní postavy je ignorují.

2.3 Postup levelu

Hráč uvidí mapu s jednou nebo více VIP postav. U každé VIP postavy se může objevit cesta, která znázorňuje, přes které pole se bude VIP pohybovat. Vpravo nahoře na obrazovce může vidět maximální počet strážců, které může dát na mapu a počet strážců, které hráči ještě zbývá položit. Hráč nemusí položit všechny strážce ale nezíská žádnou výhodu na mapě pokud nepoužije všechny strážce. Hráč bude moci přidávat strážce na mapu dokud nedosáhne maximálního limitu strážců na mapě.

Každého strážce na mapě může hráč označit. Když má hráč označeného strážce, může strážcem otáčet a měnit tím směr, který daný strážce hlídá. Označeného strážce bude moci kdykoliv smazat, což odebere strážce z mapy a přičte volného strážce do jeho rezervy. Hráč může také označenému strážci přidávat pole, po kterých se strážce bude pohybovat. Podle polí, které přidá strážci do hlídky, se bude strážce buď pohybovat ze startovní pozice na poslední pole přidané do hlídky a poté po samé cestě půjde nazpátek, nebo se strážce bude pohybovat po stejné cestě, aniž by se otáčel, pokud mu to rozestavění polí dovolí. Hráč nebude moci přesouvat strážce. Aby strážce přesunul, musí nejdříve daného strážce odstranit a pak postavit na jiné místo.

Až bude hráč s rozestavěním strážců spokojen, bude moci potvrdit svůj plán. Poté hráč bude muset počkat, než plánovač vygeneruje řešení. To by mělo trvat nanejvýš několik sekund. Jakmile plánovač vygeneruje řešení, hra vyčká na potvrzení od hráče. Po potvrzení hráčem hra ukáže řešení hráči. Jakmile se vrah dostane na pole s VIP postavou hra končí. Use case, který znázorňuje jak bude hráč ovládat prototyp, je na obrázku 2.6.



Obrázek 2.6: Use case diagram naší aplikace znázorňující, jak bude moc hráč ovlivňovat náš prototyp.

Kapitola 3

Implementace

V minulé kapitole jsme si představili návrh hry. Tato část popisuje její provedení. Jaké nástroje k implementaci naší aplikace využíváme a jak implementujeme všechny vlastnosti objektů a hry, které jsme si určili, když jsme naší aplikaci navrhovali.

3.1 Komponenty

V této části si představíme veškeré nástroje, které naše aplikace využívá. Je zde zahrnut software, který využíváme k programování prototypu i jednotlivé komponenty, které naše aplikace využívá ke správnému fungováni.

3.1.1 Unity

Unity je multi-platformový herní engine vyvinutý Unity Technologies. Unity se může použít ke tvorbě 2D, 3D i VR her, nebo i ke tvorbě simulací. Unity využívá programovací jazyk C# a bolt. Unity je navíc úplně zdarma pro nekomerční účely, a nebo pro komerční účely kde vývojář nebo firma nevydělala více jak 100,000 dolarů za posledních 12 měsíců.

Unity má velké množství naučných videí i článků, ať už od lidí využívající tento herní engine, nebo od samotných vývojářů. To společně s velkou komunitou lidí, kteří jsou ochotní poradit na internetu, možností bezplatného užití a relativně jednoduché ovládání činí z Unity jeden z nejoblíbenějších herních enginů jak pro začátečníky, tak pokročilé osoby se zájmem o programování video her.[4]

3.1.2 Planning Domains

Planning.Domains je webový pddl editor, který jsem používal pro ověřování správnosti jak mého PDDL domain kódu, tak PDDL problém kódu, který vytvářel mnou napsaný skript v Unity. [5]

3.1.3 Maplan

Maplan [6] je multiagentní plánovač, který implementuje jak více vláknové vyhledávání, tak distribuované prostorové vyhledávání. Plánovač je napsán v jazyce C. Maplan byl vytvořen na základě MAD-A* plánovače [7]. Lze ho použít jako optimální plánovač, nebo jako uspokojivý plánovač.

3.1.4 Docker

."Docker je Open source projekt pro automatizaci nasazení aplikací jako přenosných a vlastních kontejnerů, které mohou běžet v cloudu nebo místně. Docker je také Společnost, která propaguje a vyvíjí tuto technologii a pracuje ve spolupráci s dodavateli cloudů, Linux a Windows, včetně Microsoftu." [8] Maplan se náchází v kontejneru, který nám poskytuje Docker, abychom jsme mohli Maplan použít na operačním systému Windows.

3.2 PDDL

Pro umělou inteligenci využíváme automatizované plánování psané jazykem PDDL [9] za použití plánovače STRIPS [10]. Ve videoherním průmyslu se tato umělá inteligence kvůli její časové náročnosti prakticky nepoužívá. Automatizované plánování je odvětví umělé inteligence, která se snaží vyřešit problém pomocí série akcí. PDDL (Planning Domain Definition Language) je pokus o normalizaci jazyka pro automatizované plánování.

STRIPS (Stanford Research Institute Problem Solver) je automatizovaný plánovač, a je základ pro většinu jazyků vyjadřující automatizované plánování. STRIPS dokument je složený z počátečního stavu, cíle, kterého se plánovač snaží dosáhnout a několika akcí, pomocí kterých se snaží daného cíle dosáhnout. Každá akce se skládá z podmínek, které je nutné splnit, aby akce mohla nastat a důsledků, které se stanou, když akce proběhne.

Při psaní PDDL souborů jsem čerpal inspiraci ze hry Sokoban [11]. Sokoban sice nemá žádnou umělou inteligenci, ale PDDL může být použit pro nacházení řešení jednotlivých levelů dané hry, a jelikož Sokoban v tomto souboru také využívá pohyb postavy

po čtvercové mapě, byla ideální pro první návrhy PDDL souboru pro tuto bakalářskou práci.

3.2.1 Domain PDDL

Doménová část PDDL v tomto projektu definuje pravidla hry. Pro každou mapu stačí jedno společné doménové PDDL.

```
(define (domain game00)
        (:requirements :typing :action-costs)
```

Types sekce

```
  (:types character location timer - object
          assassin vip - character)
```

V části **types** jsou vypsány objekty ve hře. **Character** představuje VIP a vrahy. Stráže nejsou definovány, jelikož pro umělou inteligenci je stráž jen pole, na které vrah nemůže vstoupit. Stejně tak všechna políčka, na která vidí nebo políčka na kterých je zeď. **Location** představují jednotlivá pole a **Timer** definuje kolo, jelikož prototyp je tahová hra.

Predicates sekce

```
(:predicates (clear ?l - location ?k - timer)
             (at ?ch - character ?l - location ?k - timer)
             (entry ?l - location)
             (success ?a - assassin)
             (COMING ?a - assassin ?k - timer)
             (MOVE ?from ?to - location)
             (TIMELINE ?present ?future - timer))
(:functions   (total-cost))
```

V části **predicates** jsou možné stavy jednotlivých objektů a vztahy mezi nimi. Část **(clear ?l - location ?k - timer)** definuje, že vrah může bezpečně vstoupit na pole **l** v čase **k**, tedy že na daném poli nestojí strážce, a ani ho žádný nevidí, takže na něm není pole vize.

Řádek (at ?ch - character ?l - location ?k - timer) znázorňuje na jakém poli **l** se nachází postava **ch**. Zde má kolo k rozdílný důvod. Pro VIP to funguje podobně jako časovač u polí, tedy kde se postava nachází v jaký čas. U vraha čas **k** zaznamenává současný tah.

Řádek (entry ?l - location) definuje všechna pole **l** kde se nachází vstupní bod, kterými vrah může přijít do mapy.

Řádek (success ?a - assassin) značí vítězství vraha a tedy prohru hráče, pokud se vrah dostane do tohoto stavu, zvítězil.

Řádek (COMING ?a - assassin ?k - timer) tento stav znázorňuje, že vrah ještě není v budově a může použít vstupní bod, aby se dostal na nějaké pole.

Řádek (MOVE ?from ?to - location) udává vztah mezi dvěma poli. Tento vztah je mezi poli, které jsou vedle sebe, a proto každá postava může jít z pole **from** na pole **to**. Jelikož toto nefunguje obráceně, pro každá dvě pole která spolu sousedí jsou vypsány dva tyto řádky, každé pole je jednou na pozici **from** a podruhé na pozici **to**.

Řádek (TIMELINE ?present ?future - timer) definuje časovou osu. Znázorňuje, že kolo **future** následuje po kole **present**. Funguje na podobném principu jako řádek **MOVE**.

Actions sekce

```
(:action enter
        :parameters (?a - assassin ?to - location ?present ?future - timer)
   :precondition (and (COMING ?a ?present)
                      (clear ?to ?present)
                      (clear ?to ?future)
                      (entry ?to)
                      (TIMELINE ?present ?future)
                      )
   :effect       (and (not (COMING ?a ?present))
                      (at ?a ?to ?future)
                      (increase (total-cost) 1)
                      )
)

(:action move
        :parameters (?a - assassin ?from ?to - location ?present ?future - timer)
   :precondition (and (at ?a ?from ?present)
                      (clear ?to ?present)
                      (clear ?to ?future)
                      (MOVE ?from ?to)
                      (TIMELINE ?present ?future)
                      )
   :effect       (and (not (at ?a ?from ?present))
                      (at ?a ?to ?future)
                      (increase (total-cost) 1)
                      )
)

(:action kill
        :parameters (?a - assassin ?v - vip ?from ?to - location ?present - timer)
   :precondition (and (at ?a ?from ?present)
                      (at ?v ?to ?present)
                      (MOVE ?from ?to)
                      )
   :effect      (and (not (at ?a ?from ?present))
                      (at ?a ?to ?present)
                      (success ?a)
                      )
)

(:action wait
        :parameters (?a - assassin ?from - location ?present ?future - timer)
   :precondition (and (at ?a ?from ?present)
                (clear ?from ?future)
                (TIMELINE ?present ?future)
   )
   :effect      (and (increase (total-cost) 1)
                (not (at ?a ?from ?present))
                (at ?a ?from ?future)
                )
)
)
```

V části **actions** jsou veškeré možné akce, které vrah může na každé mapě udělat.
Akce **enter** je první akce, kterou musí vrah udělat na začátku každého levelu. Aby vrah mohl použít tuto akci musí být ve stavu **COMING**, pole **to**, na které chce jít, musí být nehlídané tento daný tah i tah následující, a pole **to** musí být označené jako vstupní. Poté co se vrah přesune na toto pole, ztrácí stav **COMING** a získává stav **at**, jelikož se už objeví na mapě a je potřeba to někde zaznamenávat.

Pomocí akce **move** se vrah dostane z jednoho pole **from** na druhé pole **to**. Aby vrah mohl použít tuto akci, musí se nacházet na poli **from**, pole **to** musí být v tomto tahu i tahu následujícím nehlídané, pole **from** a pole **to** musí spolu sousedit a poslední řádek

v podmínkách zajišťuje, že je dodržována časová osa. Pokud se tato akce provede, vrah zmizí z pole **from** a získá stav, že se nachází na poli **to** v čase **future**.

Akce **kill** se od akce **move** příliš neliší. Pořád musí pole **from** a **to** spolu sousedit a vrah se musí nacházet na poli **from**. Navíc na poli **to** musí stát VIP. Tato akce pohne s vrahem na pole s VIP a dá vrahovi stav **(success ?a - assassin)** se kterým hráč prohrává hru. Zde už nezáleží na tom, jestli je pole s VIP hlídané strážcem, neboť hráč prohrává v moment, kdy VIP zemře.

Akce **wait** dává vrahovi možnost vyčkat na místě, tím pádem se pak pohnou pouze strážci. Aby vrah mohl tuto akci použít nesmí se po pohybu strážců ocitnout pole na kterém stojí pod strážcovo dohledem. Vrah poté přijde o stav **at**, ale na místo něj dostane nový s posunutým tahem **future**.

3.3 Integrace do hry

V Unity mají všechny postavy svou vlastní třídu. VIP postavy a stráže mají list polí, na kterých se pohybují a metodu move, která pohne s danou postavou na další pole v listu. Pokud se dostanou na konec listu, tak poté pokračují od znova, pokud první a poslední pole spolu sousedí. Pokud první a poslední pole spolu nesousedí, tak metoda move bere pole z listu postupně od konce.

Pole jsou uložené v takzvaných tile-map, a slouží jako jejich kontejner. Pro tento prototyp máme 4 tile-mapy, jednu pro pole podlahy, jednu pro pole zdí, další pro pole vstupů a poslední pro pole vize. Každé pole má svou x a y souřadnici, na které se nachází. Pole podlahy, pole zdí a pole vstupů jsou součástí mapy a během levelu se nemění. Pole vize jsou generována pomocí skriptu VisionCalculator. VisionCalculator vezme každého strážce a na základě směru jeho pohledu vygeneruje tři řady polí vize. Pole vize se vygeneruje pouze v případě, že na poli není zeď nebo mezi strážcem a polem, na kterém by se mělo vytvořit pole vize, nestojí zeď.

V Unity máme skript PDDLHandler, jehož jediný úkol je vypsat problem.pddl soubor. Při vypisování nejdříve začíná u jednotlivých objektů. Jako první vypíše vraha, poté počet VIP. V problem.pddl se nezmiňujeme o existenci strážců, jelikož pro naše účely je může plánovač brát jako zdi nebo pole vize, tedy pole, na které nemůže vrah vstoupit. Poté PDDLHandler všechna pole, na kterých se nachází pole podlahy nebo pole vstupu. Jelikož plánovač má i omezený čas na splnění svého úkolu, jsou vypsány i jednotlivé tahy.

V našem případě se k tahu nebo času chováme jako k objektu.

Po vypsání objektů začne PDDLHandler vypisovat počáteční stav. Nejdříve PDDLHandler vezme každé pole podlahy a pokusí se najít všechny jeho sousedy. Pro každého souseda co najde, vypíše vztah **MOVE**. Pro zapsání pohybu VIP na herní ploše PDDLHandler nejdříve uloží startovní pozici všech VIP, poté vypíše jejich startovní pozice v čase jedna. Poté pohne s daným VIP a vypíše jeho pozici v čase dva, a takto pokračuje do maximálního počtu tahů. Jelikož problem.pddl nezná postavy strážců, první tah zapíšou všechny pole, kromě pozic strážců a všech polí vize vygenerovaných pomocí skriptu VisionCalculator. Zapsané pole jsou potom volné pro vraha, aby na ně mohl vstoupit, poté PDDLHandler pohne o jedno pole se všemi strážci, VisionCalculator smaže všechny pole vize a vygeneruje nové na základě nového rozestavění strážců. Toto se opakuje do posledního možného tahu. Jako poslední se v počátečním stavu vypíše vztah mezi jednotlivými tahy.

Na závěr problem.pddl se vypíše úkol, kterého se má plánovač snažit dosáhnout a **(:metric minimize (total-cost))**, který zaručí, že řešení, které plánovač najde, je nejkratší možné řešení.

Kapitola 4

Validace a verifikace

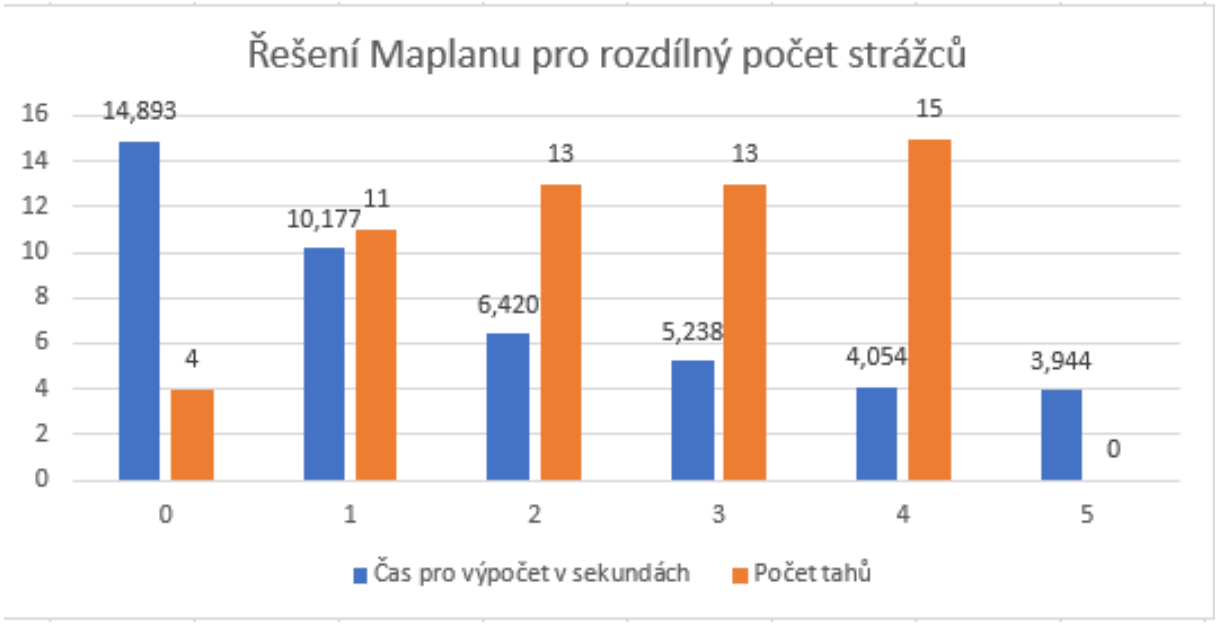
V této části popíšeme průběh a výsledky naších testů a experimentů. V části experimentu měříme jak rychle a efektivně si plánovač dokáže poradit s mapou, kde obtížnost vzrůstá. V testovací části popisujeme výsledky testování na lidských hráčích a jejich zpětnou vazbu, ze které jsme poté vylepšili naši aplikaci z verze 1.0 na verzi 2.0.

4.1 Experimenty

Pro náš experiment využijeme mapu 22x22 polí (obrázek 4.1). Na této mapě se pohybují 3 VIP po vyznačených trasách, jedno VIP vlevo nahoře a dvě VIP, kteří začínají na kraji mapy a dojdou doprostřed mapy, kde se setkají. Měřit budeme rychlost Maplanu při výpočtu nejkratší cesty s postavením různého počtu strážců. Strážci budou přidáváni efektivně. Začneme bez strážců, a pokaždé kdy vrah najde cestu k jednomu z VIP, přidáme strážce, který bude danou cestu hlídat. Takto budeme pokračovat dokud nebudou všechna VIP hlídaná.

V každém měření budeme měřit čas, za který Maplan dokáže najít řešení, a počet tahů, které k tomu potřebuje. Očekáváme, že s každým strážcem se bude zvyšovat nejen počet tahů ale i čas, který Maplan stráví na vypočítávání řešení, jelikož se s každým přidaným strážcem zvyšuje optížnost mapy.

Obrázek 4.1: Mapa 22x22 polí, kterou využíváme k experimentu. Je zde 14 přístupových polí a 3 VIP postavy. VIP postava vlevo nahoře se nachází hned vedle vstupního pole pro vraha. Pokud tedy nedáme na mapu žádného strážce, který by hlídal tento vstupní bod, můžeme si být jistí, že právě toto VIP bude vrahův cíl, jelikož je to možnost s nejmenším počtem akcí ke splnění mise. Další dvě VIP se pohybují z levé a pravé části mapy na její střed, kde se setkají, a poté se otočí na jejich startovní pozici. Tato dlouhá cesta a to, že se nachází uprostřed mapy z nich dělá obtížné postavy na hlídání.



Obrázek 4.2: Graf výsledku našeho experimentu. Dole je počet strážců přítomných na mapě v době měření. Při pěti strážcích je počet akcí nulový, jelikož plánovač nemohl najít řešení, a tedy nemohl splnit daný úkol.

V grafu na obrázku 4.2 vidíme, že ačkoliv plánovač potvrdil naše očekávání ohledně zvyšujícího se počtu akcí, které plánovač musí udělat, aby splnil cíl mise, naše očekávání ohledně zvyšujícího se času, který plánovač potřebuje k výpočtu řešení, nám experiment vyvrátil. Nejen, že se čas potřebný k výpočtu s každým přidaným strážcem nezvyšuje, čas s každým přidaným strážcem klesá. Je to kvůli tomu, že ačkoliv člověk by na řešení s vícero překážkami potřeboval více času na nalezení řešení, pro plánovač znamená více strážců méně polí, na které může s vrahem vstoupit, a tedy méně kombinací, které plánovač musí otestovat.

4.2 Tutoriál

Většina her nemá dostatek místa na obrazovce, aby mohla popsat veškeré vlastnosti jejich GUI, nebo aby mohla mít neustále na obrazovce vypsány veškeré klávesové zkratky. Proto má také většina her tutoriál, který vysvětlí, jaké hra využívá klávesové zkratky, jak pomocí nich ovládat hru, nebo co udělají jednotlivá tlačítka na jejich GUI. Při prvním pohledu na hru, by náš prototyp mohl zmást spousty hráčů, proto jsme vytvořili tutoriál, který má vysvětlit ovládání hry, ybchom mohli hru verifikovat pomocí lidských hráčů.

4.2.1 Obsah tutoriálu

V tomto prototypu se snaží vrah (na mapě se objeví jako žlutá elipsa s písmenem A) ovládaný umělou inteligencí zabít VIP (zelená elipsa s písmenem V). Vašim úkolem je všechny VIP postavy ochránit. To lze učinit pomocí strážců (modrá elipsa s písmenem G). Tento tutoriál vám popíše jak toho dosáhnout.

Všechny postavy se mohou hýbat pouze po bílých nebo šedých polích. Navíc všechny postavy kromě vraha se mohou pohybovat i po zelených polích a na jednom poli, může stát vícero postav. Vrah ovšem nemůže stát na poli se strážcem, pro což existuje jediná výjimka a to, že vrah může vstoupit na pole se strážcem nebo na zelené pole v případě, že při daném tahu zabije VIP a tím pádem vyhraje hru.

Vrah přijde jedním z mnoha průchodů označených světle šedou barvou. Mapa se ovládá pomocí kláves WASD, W pro pohyb nahoru, S pro pohyb dolů, A pro pohyb doleva a D pro pohyb doprava. Pro oddálení mapy lze ještě použít klávesu E a pro přiblížení klávesu Q.

Pro postavení strážce klikněte na tlačítko create guard, to zezelená. Poté, dokud je tlačítko zelené, a máte volné strážce v rezervě, můžete přidávat strážce pomocí klik-

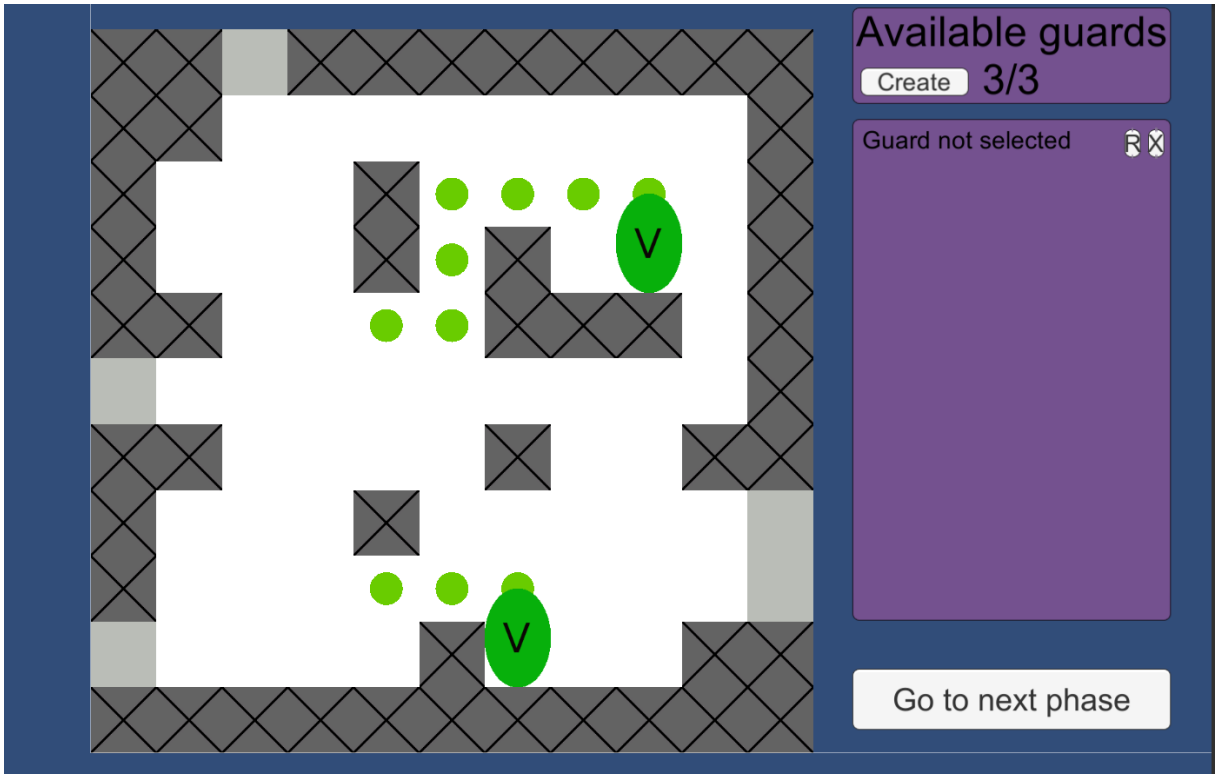
nutí levým tlačítkem na bílé pole. Každý strážce vytváří maximálně 6 zelených polí ve směru, kterým se dívá. Pro změnu směru, kterým se dívá, je nutné strážce označit levým tlačítkem, jakmile strážce zezelená je označen. Poté stačí kliknout na tlačítko R, které otočí strážce o 90 stupňů.Pro vrácení strážce zpátky do rezervy označte strážce stejně jako v minulém bodě a klikněte na tlačítko X pro jeho odstranění.

Pokud pouze postavíte strážce na místo, zůstane tam a bude se neustále koukat stejným směrem. Pokud chcete, aby se váš strážce hýbal, musíte mu přidat pole po kterých se má pohybovat. To učiníte označením strážce. Poté, pokud kliknete na bílé pole vedle strážce, objeví se modrý kruh, který značí, že strážce bude toto pole využívat jako součást své hlídky. Pokud chcete přidat další hlídku musíte kliknout na volné bílé pole vedle posledního pole, které jste přidali. Strážce se poté bude pohybovat po těchto polích, a když se dostane na poslední přidané pole, vrátí se nazpátek. Výjimka je, pokud se poslední přidané pole nachází vedle strážce, pak se strážce na konci otáčet nebude a bude pokračovat dále v cestě.

Při plánování počítejte s tím, že první se hýbe VIP, poté vrah a poté strážci. Všichni se pohybují stejnou rychlostí, jedno pole za tah. Pokud jste s rozestavěním strážců spokojení, klikněte na tlačítko **go to next phase**, které pošle mapu s pozicemi strážců a jejich hlídek plánovači k vyřešení.

4.3 Zpětná vazba

Pro testování s lidskými hráči jsme využili menší mapu 11x11 polí (obrázek 4.3), aby si hráči nejdříve zvykli na mechaniky hry. Hráči dostali psaný tutoriál k přečtení a poté začali plánovat rozmístění strážců bez našeho zásahu. Testování probíhalo v časovém rozmezí mezi 20-60 minutami, pokud se hráči poda5ilo dokončit mapu, mohl se pokusit o zvládnutí větší mapy 22x22 polí.



Obrázek 4.3: Mapa 11x11 polí, kterou používáme k testování na lidských hráčích. Oproti velké mapě je tato menší mapa jednoduší pro první hraní. Je zde 5 vstupních bodů a pouze 2 VIP. Ačkoliv má hráč k dispozici 3 strážce, při testování se jednoduše stávalo, že vrah našel nehlídané místo a úkol splnil.

4.4 Verze 2.0

Při prvním testování lidskými hráči, jsme narazili na několik nedostatků, hlavně v oblasti balancování různých přístupů ke hře a grafického rozhraní hry.

4.4.1 Grafické rozhraní

Ke grafickému rozhraní hry byli připomínky, že tlačítko pro odstranění nebo otáčení strážce by mohlo být přiřazeno ke klávesové zkratce, a že při pokládání strážců není přesně vidět na jaké ukazuje myš. Ačkoliv tento nedostatek nijak neovlivňuje funkčnost našeho prototypu, tato jednoduše proveditelná změna zpříjemní hráčům hraní našeho prototypu. Proto jsme se rozhodli tyto vlastnosti implementovat pro další testování.

4.4.2 Balanční problémy

V balancování hry jsme našli dva problémy. První problém je, že vrah je nucen si vybrat vstupní bod hned v prvním tahu. Takže jeden z hráčů, který hru testoval, zablokoval co nejvíce přístupových bodů a poté se vydal ke všem nehlídaným přístupovým bodům, což mělo za následek velice nepraktický, ačkoliv pro splnění cíle mise efektivní, plán na hlídání VIP. Tento problém se dá vyřešit přidáním nové akce do domain.pddl, která bude podobná jako akce **wait**, ale bude fungovat i když vrah ještě není na mapě. Pokud tedy hráč znova zvolí tuto taktiku, vrah si pouze počká, až přístupový bod nebude hlídán a poté se do mapy dostane nepozorovaně.

Druhý problém byla efektivita strážců pouze s jedním dalším polem hlídky, tedy strážce, který se pohybuje po dvou bodech. Tímto způsobem se strážce otáčel tak rychle, že jeden strážce zvládl hlídat 14 polí, přes které vrah nemohl nikdy projít. Při měření se nám toto chování hodilo, jelikož pro nás bylo jednodušší plánovat naše stráže na efektivní místa. Ale pro účel hry se nám tento problém nelíbí, jelikož takto efektivní metoda hlídky, dělá delší hlídky značně neefektivní. Problém vyřešíme pomalým otáčením strážce, tím že když dojde strážce na poslední pole své hlídky, tak než se začne vracet zpátky na první pole své hlídky, nejdříve se otočí ale zůstane stát na místě, tím dá vrahovi více času na to, aby mohl kolem strážce projít bez povšimnutí.

Kapitola 5

Průběh vývoje aplikace

V této části popisujeme důležité verze aplikace a rozdíl mezi nimi. Jelikož se jedná pouze o prototyp, a ne o celou hru, tato část popisuje také vlastnosti, které dělí prototyp od finální aplikace, a proč jsou tyto vlastnosti důležité před ukončením práce na naší hře.

5.1 Verze hry

Při verzování hry si představíme akorát 4 důležité milníky naší aplikace. Alfa fáze aplikace představuje první spustitelnou verzi hry, kdy hráč mohl potvrdit svůj plán a plánovač vygeneroval řešení na základě poskytnutých problem.pddl a domain.pddl. Verze 1.0 představuje verzi aplikace před implementováním vylepšení zjištěných ze zpětné vazby při testování s lidskými hráči. Ve verzi 2.0 se aplikace nachází po testování. Ačkoliv bakalářská práce popisuje aplikaci ve verzi 1.0, aplikace ve verzi 2.0 je důležitá pro znázornění, jak jsme využili zpětnou vazbu k vylepšení našeho prototypu. Aplikace ve finální verzi představuje proměnu našeho prototypu ve finální hru. Prezentuje, co vše je potřeba udělat, aby se náš prototyp stal video hrou.V tabulce 5.1 můžeme vidět rozdíli mezi jednotlivými verzemi.

	Alfa verze	Verze 1.0	Verze 2.0	Vize do budoucna
domain.pddl	Soubor obsahuje pouze akce pro zabití VIP a pro pohyb	Soubor obsahuje akce pro pohyb, čekání, vstupu do mapy a zabití VIP	Soubor obsahuje akce pro čekání uvnitř i mimo mapu, pohyb a zabití VIP	Soubor obsahuje akce pro čekání uvnitř i mimo mapu, pohyb, zabití VIP a speciální ability, navíc se bude generovat podle vraha
problem.pddl	Soubor se generuje podle objektů a vztahů v domain.pddl	Beze změny	Beze změny	Beze změny
Grafické rozhraní	Bez grafického rozhraní	Grafické rozhraní má tlačítka pro přidání strážce, odebrání strážce a otáčení strážce, navíc lze tlačítkem potvrdit rozestavění strážců	Stejné jako ve verzi 1.0 ale tlačítka mají přiřazené klávesy a při pohybu myší je vidět, na které pole na mapě ukazuje	Grafické rozhraní je upravené podle potřeb a nově implementovaných vlastností
Hratelnost	Jediný kdo se na mapě hýbe je vrah Hráč může přidávat, odebírat a odstraňovat hráče myší a potvrdit musí entrem. Vrah začíná na mapě.	Všechny postavy se mohou hýbat po mapě, vrah začíná mimo mapu, přidáno grafické rozhraní,	Stejné jako ve verzi 1.0 ale s menšími balančními úpravami	Stejné jako verze 2.0 ale hráč si může nakupovat různé druhy strážců a plánuje proti vrahům se speciálními schopnostmi
Grafika	Jednoduchá grafika ze které se dá jednoduše rozeznat typy polí nebo strážců	Beze změny	Beze změny	Grafika, ve které postavy i pole budou reprezentovány jednoduchou 2D texturou, ze které je ale stále možné rozeznat kdo je kdo
Ovládání po mapě	Přesun myši k okraji obrazovky	Pomocí kláves WASD	Pomocí kláves WASD a QE na oddálení a přiblížení mapy	Pomocí myši i klávesnice

Tabulka 5.1: Tato tabulka popisuje různé části naší aplikace v rozdílných verzích.

5.2 Cesta k hotové hře

Tato bakalářská práce popisuje pouze prototyp hry, takže od finálního produktu dělí tuto aplikaci spousta práce. Tato část má za cíl popsat co chybí tomuto prototypu, aby se stal plnohodnotnou hrou.

5.2.1 Nové mapy

V této práci jsme vytvořili pouze dvě mapy. Proto, aby se prototyp stal plnohodnotnou hrou, je potřeba vytvořit více map s rozdílným stupněm obtížnosti. Ačkoliv je aplikace im-

plementovaná tak, aby tvorba nových levelů bylo co nejjednodušší, je to také nejdůležitější část implementace, která dělí tento prototyp od toho, aby se stal video hrou.

5.2.2 Hlavní menu

Ve většině her se nachází hlavní menu, které dá hráči na výběr jak chce pokračovat. Ať už je to pokračováním od místa kde skončil naposledy, vybrání si vlastní mapy, nebo třeba změny nastavení. Jelikož hlavním cílem této bakalářské práce je dokázat, že daný typ hry s využitím automatizovaného plánování je možné vytvořit, hlavní menu nebylo potřeba vytvářet.

5.2.3 Grafika

Je rok 2021, v dnešních hrách se očekává alespoň základní grafika. Animované obrázky postav reprezentující jednotlivý typ osob. Pole, které by svým zevnějškem znázorňovali nejen jaký typ pole reprezentují, ale také budovali prostředí, ve kterém se daný level nachází. Pole zdi by mohlo reprezentovat stromy nebo keře v mapě, která by se odehrávala v přírodě, zatímco v mapě budovy mi reprezentovala klasickou zeď nebo třeba sloup. Ale jelikož jsme pro účely této bakalářské práce potřebovali pouze implementovat funkční prototyp, postačili nám jednobarevné čtverce a elipsy.

5.2.4 Nové schopnosti pro vraha

Po pár hrách začne být vrah předvídatelný, a poté vymyslet způsob, jak zabránit vrahovi ve splnění cíle není žádná výzva. Předvídatelnost byla užitečná při měření a testování základních prvků hry ale pro hraní by se brzo stala nudná a nezáživná. Pokud by vrah mohl mít unikátní schopnosti, například že by se v jeden tah mohl pohnout o dvě pole místo o jedno, nebo měl schopnost podplatit strážce, který by nechal vraha projít, dalo by to hráči nové výzvy, které by musel zahrnou do plánování obrany svých VIP postav.

5.2.5 Více druhů strážců a peníze

Pro testování této aplikace nám jeden strážce stačil, ale pro podporu různých přístupů k problému by bylo zapotřebí přidat více druhů strážců. Strážci by se lišili počtem polí, které by hlídali nebo rychlostí pohybu. Místo toho, aby měl hráč na každou mapu určitý počet strážců, by měl peníze, za které by si mohl kupovat strážce. Každý typ strážce by stál jiný počet peněz, takže by si hráč mohl nakoupit jiný druh strážců v závislosti na rozpoložení mapy, nebo podle toho, jaké strážce preferuje.

5.2.6 Skóre

Implementovat skóre je v této fázi pro náš prototyp naprosto zbytečné, ale v plné hře by představoval možnost soutěžení s ostatními hráči. Hráči by se mohli předhánět kdo dokáže úspěšně zabránit vrahovi ve splnění mise za použití co nejmenšího množství peněz. Toto by přidalo hře menší znovuhratelnost a zároveň donutilo některé hráče přemýšlet nejen na splnění mise, ale i nad tím, jak využít prostředky co nejefektivněji.

Kapitola 6

Závěr

Povedlo se nám vytvořit prototyp počítačové hry využívající automatizované plánování. Hráč může přidávat strážce na mapu a určovat, přes jaké pole budou procházet. Aplikace dokáže převádět objekty ve hře na řádky PDDL kódu a ukládat je do souborů. Takto vygenerované soubory pošle plánovači Maplan, který na základě těchto souborů vypočítá řešení za pomoci automatizovaného plánování, které potom naše aplikace použije pro grafické znázornění řešení hráči. V průběhu implementace jsme prozkoumali různé možnosti návrhů hry a na základě okolností se rozhodly pro tu nejlepší volbu.

Výslednou aplikaci jsme otestovali na lidských hráčích, vzali jsme na vědomí jejich zpětnou vazbu, a podle těchto nových informací jsme vylepšili naší aplikaci. Při měření jsme zjistili, že při větším množství překážek se nezvyšuje čas, který plánovač potřebuje k vyřešení zadaného levelu, ale tento čas se naopak snižuje, jelikož se snižuje počet kombinací, které plánovač musí prozkoumat.

Literatura

[1] MarcelStein, *Antbuster*. WWW: https://armorgames.com/play/522/antbuster.

[2] S. Enix, *Lara Croft GO*. WWW: https://square-enix-games.com/games/lara-croft-go.

[3] I. Interactive, *Hitman II*. WWW: https://www.ioi.dk/hitman2/.

[4] U. Technologies, *Unity*. WWW: https://unity.com/.

[5] C. Muise, "Planning.Domains", in *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*, 2016. WWW: http://www.haz.ca/papers/planning-domains-icaps16.pdf.

[6] D. Fiser a A. Komenda, "Fact-Alternating Mutex Groups for Classical Planning", *J. Artif. Intell. Res. 61*, s. 475–521, 2018.

[7] R. Nissim a R. I. Brafman, "Multi-agent A* for parallel and distributed systems", in *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, W. van der Hoek, L. Padgham, V. Conitzer a M. Winikoff, ed., IFAAMAS, 2012, s. 1265–1266. WWW: http://dl.acm.org/citation.cfm?id=2343955.

[8] N. Anil, *Co je Docker?*, 2020. WWW: https://docs.microsoft.com/cs-cz/dotnet/architecture/containerized-lifecycle/what-is-docker.

[9] A. E. Gerevini, "An Introduction to the Planning Domain Definition Language (PDDL): Book review", *Artif. Intell.*, roč. 280, s. 103 221, 2020. DOI: 10.1016/j.artint.2019.103221. WWW: https://doi.org/10.1016/j.artint.2019.103221.

[10] R. E. Fikes a N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence. 2 (3–4)*, 189–208, 1971.

[11] J. C. Culberson, "Sokoban is PSPACE-Complete", *InFun With Algorithms*, 65–76, 1999.