FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Rogue planet: využití multiagentních systémů ve 2D hře
Student:	Ladislav Strnad
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je vytvořit 2D hru s využitím multiagentních systémů pro kompetitivní hratelnost.

1) Proveďte rešerši problematiky multiagentních systémů a AI ve hrách.
2) Definujte optimální model chování protihráčů (systému agentů) a porovnejte jej se stávajícími podobnými hrami.
3) Navrhněte prototyp aplikace. Zaměřte se především na chování agentů, uživatelské rozhraní a herní mechaniky.
4) Implementujte prototyp hry v Unity Engine.
5) Implementovaný prototyp podrobte vzhledem k účelu aplikace vhodným testům.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 3. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rogue planet: využití multiagentních systémů ve 2D hře

Ladislav Strnad

Katedra softwarového inženýrství
Vedoucí práce: Ing. Radek Richtr, Ph.D.

12. května 2021

Poděkování

Děkuji Ing. Radku Richtrovi, Ph.D. za aktivní vedení této práce. Dále děkuji Kristýně Víškové, Ing. Lence Strnadové a Davidu Primusovi za účast v testování a další pomoc.

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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 12. května 2021 . . . . . .

Odkaz na tuto práci

Strnad, Ladislav. *Rogue planet: využití multiagentních systémů ve 2D hře.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce se zabývá využitím multiagentního systému v počítačové hře. Na základě provedené rešerše multiagentních systémů a chování nehráčských postav v počítačových hrách a následné analýzy současných budovatelských strategií byl navržen prototyp 2D budovatelské strategie. Hlavní součástí návrhu jsou tři typy chování nepřátelských agentů, dále se pak návrh zabývá mechanikami a vzhledem uživatelského rozhraní hry. Dva navržené typy chování jsou inspirovány modely inteligence hejna, zatímco třetí typ byl navržen s důrazem na hratelnost. K implementování navrženého prototypu byl využit herní engine Unity. Implementace obsahuje základní herní mód, jehož součástí jsou všechny navržené mechaniky a ukázkový mód, který demonstruje chování jednotlivých modelů chování agentů a jejich využitelnost v počítačové hře.

Klíčová slova 2D hra, budovatelská hra, multiagentní systém, uživatelské rozhraní, Unity, pixel art

Abstract

This thesis investigates the prospect of using multi-agent systems inside a video game. A 2D base-building strategy game prototype was designed based on the previous research of multi-agent systems and behaviour of non-playable characters in video games and the following analysis of recent building strategy games. The design proposes three types of enemy agent behaviour and it also focuses on game mechanics and graphical user interface of the game. Two of the designed agent behaviour types are inspired by different models of swarm intelligence while the third type was designed primarily with gameplay in mind. The prototype was implemented using Unity game engine. The implementation features a regular game mode which includes all the designed mechanics and a showcase game mode which demonstrates the behaviour of the implemented agent behaviour types and their usability within a video game.

Keywords 2D game, building game, multi-agent system, user interface, Unity, pixel art

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Videoherní průmysl je v současnosti nejvýdělečnějším odvětvím světového zábavního průmyslu [1] a stále roste. Společně s ním se neustále zlepšují nástroje pro herní vývojáře, které dělají tvorbu her snazší a přístupnější. Díky tomu již tvorba videoher není výhradně záležitostí početných vývojářských týmů, ale objevují se i úspěšné indie hry, tedy hry vytvořené jednotlivcem či malým týmem bez závislosti na větším studiu či vydavateli. Tato práce se bude zabývat tvorbou prototypu 2D budovatelské hry obsahující multiagentní systém, k čemuž bude využit herní engine Unity.

Při návrhu a implementaci zmíněného prototypu bude dbáno též na grafickou stránku hry, uživatelské rozhraní a využité herní mechaniky. Při tvorbě her se vývojáři často inspirují v mechanikách již existujících her a snaží se vytvořit nové kombinace, které by hráče oslovily. Součástí práce proto bude analýza podobných her a jejich mechanik. Při tvorbě modelů do vytvářeného prototypu bude využit grafický styl pixel art, který se díky své jednoduchosti stává ve scéně nezávislých her populárním. Do tohoto stylu je pak třeba zasadit i vytvářené uživatelské rozhraní.

Hráč se ve vytvářeném prototypu bude moci setkat s kompeticí ve formě multiagentního systému, jehož agenti budou vůči hráči agresivní. Při návrhu agentů se zaměříme především na jejich chování a vzájemné předávání informací.

Kapitola 1

Cíl práce

Hlavním cílem této práce je navrhnout prototyp 2D budovatelské hry obsahující multiagentní systém, jehož agenti budou soupeřit s hráčem, a tento prototyp implementovat. K implementaci bude použit herní engine Unity. Práce bude dělena do pěti částí.

Rešerše se bude zabývat multiagentními systémy a možnostmi implementace umělé inteligence v počítačových hrách. Dále bude provedeno seznámení s nástroji poskytovanými Unity, které lze k tvorbě prototypu využít.

V části Analýza bude proveden rozbor již existujících her s obdobnou tématikou, přičemž předmětem zájmu budou především užívané mechaniky a využití nepřátelských agentů. Dále pak bude pro vytvářený prototyp provedena analýza požadavků a také bude zvážena využitelnost technik a nástrojů popsaných v předchozí kapitole.

Cílem kapitoly Návrh je vznik kompletního návrhu vytvářeného prototypu, který se bude věnovat plánovanému chování agentů a budov, fungování mechanik hry a také celkovému grafickému vzhledu prototypu včetně uživatelského rozhraní. Implementace vzniklého prototypu pak bude popsána v kapitole Realizace.

Prototyp hry bude po vytvoření otestován, přičemž provedené testy a jejich výsledky budou sepsány v kapitole Testování. Cílem testování je především otestovat funkčnost implementovaných mechanik a uživatelského rozhraní.

Kapitola 2

Rešerše

V této kapitole je popsána problematika multiagentních systémů a modelů inteligence hejna. Dále je pak prozkoumána umělá inteligence v počítačových hrách a také nástroje Unity, které je možné využít v rámci této práce.

2.1 Multiagentní systémy

„*Multiagentní systém je uskupení vícero autonomních agentů interagujících ve sdíleném prostředí za účelem dosažení svých cílů. Agenti v tomto prostředí mají schopnost vzájemné komunikace a případně i koordinace.*“ [2] (přeloženo autorem práce)

Balaji a Srinivasan [3] uvádí, že multiagentní systémy patří společně s paralelní umělou inteligencí (parallel AI) a distribuovaným řešením úloh (distributed problem solving) mezi oblasti, kterými se zabývá výzkum distribuované umělé inteligence. Paralelní umělá inteligence se zabývá především efektivním užíváním klasických technik umělé inteligence v kombinaci s hardwarem umožňujícím paralelizaci výpočtů, jako například více vláknové procesory. Distribuované řešení úloh se zaměřuje na řešení úloh s využitím sdílení informací mezi mnoha výpočetními entitami. Strategie a množství sdílené informace je předem dané návrhem výpočetní entity, což vede k minimální flexibilitě distribuovaného řešení úloh. Dle Vidala [4] se výzkum multiagentních systémů zabývá hledáním způsobů tvorby komplexních systémů obsahujících autonomní agenty, kteří operují pouze na základě jejich omezených schopností a lokálních znalostí, avšak společně dosahují kýžených globálních cílů.

2.1.1 Inteligentní agenti

Russell a Norvig [5] tvrdí, že za agenta můžeme považovat cokoliv, co dokáže užitím senzorů vnímat okolní prostředí a na základě získaných vjemů reagovat působením na dané prostředí skrze efektory. Například lidský agent má oči, uši a další orgány jako senzory, a ruce a nohy efektory. Robotický agent zase jako

senzory může užívat kamery a různé detektory a jako efektory zase různorodé motory. Softwarový agent pak vnímá i působí skrze řetězce bitů.

Abychom agenta mohli považovat podle Woolridge [6] za inteligentního, požadujeme, aby měl následující vlastnosti:

- **Reaktivita.** Inteligentní agent dokáže vnímat okolní prostředí a reagovat na změny v něm za účelem plnění cílů, které mu jsou dány návrhem.
- **Proaktivita.** Inteligentní agent dokáže převzít iniciativu a vykazovat chování za účelem plnění cílů i bez vnějšího podnětu.
- **Sociabilita.** Inteligentní agent je schopný interakce s ostatními agenty za účelem naplnění svých cílů.

Russell a Norvig [5] pak dále považují za podstatnou vlastnost agenta míru jeho autonomie. Pokud jsou agentovy činy založeny pouze na znalostech, které mu byly dány při jeho vzniku, a nikoli na znalostech získaných vnímáním jeho okolí, můžeme tvrdit, že mu chybí autonomie. Agent, který předem zná všechny kroky, které musí učinit k dosažení svého cíle, je sice funkční, avšak jeho schopnost plnění cílů nespočívá v jeho vlastní inteligenci, ale spíše v inteligenci jeho autora. Plně autonomní agent jedná čistě na základě svých vlastních zkušeností, což vede k zcela náhodnému chování v případě, že ještě žádné zkušenosti nezískal. Agent bez autonomie dokáže naopak postupovat pouze dle zkušeností předem vložených při jeho tvorbě. Ve výsledku nemusí být vždy vhodné neusilovat o absolutní autonomii.

Agenty pak můžeme dále rozdělit do čtyř kategorií podle jejich vnitřního fungování [5].

Reflexní agent

Reflexní agent obsahuje soubor pravidel, kdy každé pravidlo přestavuje spojení mezi druhem vjemu a příslušné reakce. Vždy, když agent získá nějaký vjem, nalezne vhodné pravidlo a podle něj se zachová. Ve své podstatě tedy chování reflexního agenta odpovídá nepodmíněným reflexům u lidí, z čehož plyne jeho pojmenování.

Agent, který si zaznamenává informace o okolním prostředí

Někdy není možné se rozhodnout čistě na základě aktuální situace. Například pokud agent potřebuje reagovat na změnu polohy určitého objektu, nestačí znát polohu daného objektu v současnosti, ale je třeba, aby si zmíněný agent ukládal i informace o poloze objektu v předchozích krocích. Tato potřeba plyne z toho, že agent nedokáže skrze své senzory získat kompletní podobu okolního prostředí a potřebuje další informace, aby rozeznal dva odlišné stavy světa poskytující shodné senzorické vjemy. Přirozeným evolučním krokem po reflexním agentovi je tedy agent, který si dokáže kromě aktuálních vjemů

uchovávat i další informace. Agent tohoto typu si uchovává informace dvojího druhu, a to znalosti o změnách prostředí nezávislých na něm, a pak také znalosti o tom, jak jeho vlastní akce působí na prostředí.

Goal-based agent
Ani podrobnější znalosti o okolním prostředí nemusí vždy stačit k výběru vhodné akce. Z toho důvodu je další kategorií takzvaný Goal-based agent, tedy agent založený na cílech. Tento typ agenta asi uchovává stejné informace jako typ předchozí, ale zároveň disponuje i informacemi o cíli, kterého se snaží dosáhnout. Na základě kombinace těchto informací pak rozhoduje, jaká akce ho posune nejblíže ke kýženému cíli a tu pak koná.

Utility-based agent
K cílům se lze často dostat vícero způsoby, přičemž některé jsou výhodnější než jiné. Utility-based agent proto k vlastnostem Goal-based agenta přidává ještě takzvanou utility funkci, která každému možnému stavu přiřazuje reálné číslo, které určuje, jak moc se v něm chce agent nacházet. Toto číslo pak nazýváme utility.

Některé novější publikace pak přidávají ještě pátý typ agenta, který je schopný učení [7].

Learning agent
Učící se agent je rozšířením utility-based agenta. Tento typ agenta vyhodnocuje výsledky svých minulých akcí za účelem zlepšení svých akcí v budoucnu. Hlavními součástmi je výkonný element, který vybírá a vykonává nejvhodnější akce, nebo akce, které ještě nebyly nikdy vyzkoušeny, a učící element, který zodpovídá za postupné zlepšování vlastností agenta vyhodnocováním minulých akcí.

2.1.2 Prostředí

Pro návrh funkčního agenta je třeba znát, v jakém prostředí se bude pohybovat. Russell a Norvig [5] kategorizují prostředí takto:

Dostupné vs. nedostupné
Pokud agent skrze své senzory může vnímat kompletní stav prostředí, říkáme, že je mu dostupné. V takovém prostředí si pak agent nemusí nic uchovávat v podobě vnitřních stavů.

Deterministické vs. nedeterministické
Pokud je možné nadcházející stav prostředí jednoznačně určit podle stavu současného, říkáme o tomto prostředí, že je deterministické. Pokud je prostředí

deterministické, ale není dostupné, může se z pohledu agenta jevit jako nedeterministické, jelikož agent nemá dostatek informací pro přesnou předpověď nadcházejícího stavu.

Epizodické vs. neepizodické
Události v epizodickém prostředí jsou rozděleny do takzvaných epizod. Tyto epizody jsou na sobě vzájemně nezávislé a každá sestává z jednoho vnímání a konání agenta.

Statické vs. dynamické
Pokud se prostředí může měnit v době, kdy se agent rozhoduje, říkáme, že je dynamické pro daného agenta. V opačném případě je pro něj statické.

Diskrétní vs. spojité
Prostředí je diskrétní, pokud v něm může nastat pouze konečný počet různých vjemů a akcí.

2.2 Inteligence hejna

V této podkapitole se seznámíme se třemi běžnými modely inteligence hejna tak, jak je shrnují Lim a Jain [8].

2.2.1 Optimalizace mravenčí kolonií

Optimalizace mravenčí kolonií je model inspirovaný chováním mravenců při hledání potravy. Při nalezení potravy mravenci během cesty zpět do mraveniště vypouští feromony, které slouží jako jednoduchý způsob předávání informací. Když mravenec při hledání potravy narazí na feromonovou cestu vydá se po ní a zároveň ji posílí vlastním feromonem. Čím je feromon dané cesty silnější, tím je vyšší šance, že se po ní mravenci vydají. Feromony se časem vypařují, a díky tomu dlouhé cesty slábnou rychleji než krátké, jelikož na krátkých cestách zvládají mravenci feromon obnovovat lépe. Díky kombinaci vypařování feromonů a preferenci mravenců vůči cestám se silnější feromonovou stopou tak mravenci naleznou nejkratší cestu mezi mraveništěm a zdrojem potravy. Prvním algoritmem založeným na optimalizaci mravenčí kolonií je algoritmus *ant system* [9], prvně popsaný v roce 1991. Dalšími algoritmy založenými na tomto modelu jsou pak například *ant colony system* [10] a *max-min ant system* [11].

2.2.2 Optimalizace hejnem částic

Optimalizace hejnem částic bere inspiraci v chování ptáků a ryb v hejnech při hledání potravy. Model využívá populaci částic pohybujících se prostorem

o n dimenzích určitým směrem a rychlostí. Na počátku mají částice náhodné vlastnosti a v každé další generaci pak upravují své vlastnosti dle vlastní nejlepší pozice a nejlepších pozic okolních částic. Jako první popsali optimalizaci hejnem částic Eberhart a Kennedy [12].

2.2.3 Optimalizace včelím rojem

Optimalizace včelím rojem je založena na chování včel při hledání zdrojů potravy. Včely létají z úlu a hledají potravu, přičemž po nalezení vhodného zdroje se včela vrací zpět do úlu, kde získanou informaci předává ostatním včelám, které se snaží přesvědčit, aby vyrazili k tomuto zdroji. Toto předání probíhá formou včelího tanečku, kterým včela sdělí ostatním včelám vzdálenost, směr a kvalitu nalezeného zdroje. Šance na přesvědčení ostatních včel závisí na vlastnostech nalezeného zdroje, a k lepším zdrojům tedy létá větší množství včel. Příkladem algoritmu založeného na chování včel při hledání potravy je *Artificial bee colony algorithm* [13].

2.3 Umělá inteligence v počítačových hrách

Za umělou inteligenci ve hrách podle Sizera [14] obvykle označujeme způsob, jakým se inteligentní agenti ve hrách dobírají k rozhodnutím, jakou akci konat v reakci na příchozí vjemy. Inteligentní agenti mohou mít ve hrách mnoho podob i úloh. Mohou reprezentovat osoby, zvířata, vozidla, ale i abstraktnější pojmy jako skupinu objektů či stát. Od umělé inteligence ve hrách, na rozdíl od jiných oblastí, nevyžadujeme maximální efektivitu, ale spíše co největší uvěřitelnost a alespoň zdánlivou logiku akcí.

2.3.1 Rozhodování

V této sekci rozebereme několik způsobů, kterými lze zprostředkovat rozhodování umělé inteligence v počítačových hrách dle Mounta [7].

Rozhodovací stromy
Nejjednodušším způsobem zprostředkování rozhodování umělé inteligence jsou rozhodovací stromy. Rozhodovací strom má obvykle podobu binárního stromu, kdy každý uzel daného stromu reprezentuje určitou podmínku a jeho potomci jsou možnými výsledky dané podmínky (obvykle splněna či nesplněna). Potomkem uzlu může být uzel reprezentující další podmínku, nebo list, který reprezentuje zvolenou akci.

Automaty
Automaty přidávají oproti rozhodovacím stromům možnost uložení vnitřního stavu, a tedy umožňují rozhodování založené na kombinaci vnějších vlivů a uloženého stavu. Díky tomu můžeme dosáhnout zajímavějšího chování entit

ve hrách. Automat určující rozhodovací proces lze modelovat jako orientovaný graf, kde každý vrchol reprezentuje možný stav agenta a každá hrana reprezentuje přechod do jiného stavu, který je často doprovázen nějakou akcí.

Hierarchické automaty
Jednou z nevýhod automatů je, že pokud chceme vytvořit komplikovanější chování, kdy může platit více stavů najednou, komplexnost automatu prudce roste, jelikož musí obsahovat vrchol pro každou možnou kombinaci těchto stavů a může dojít až ke kvadratickému růstu počtu přechodů mezi stavy. Jedním z řešení tohoto problému jsou hierarchické automaty. Hierarchické automaty se skládají z obecnějších stavů, které obsahují vícero konkretizujících podstavů, jenž umožňují důmyslnější chování v rámci jednoho obecného stavu.

Behaviorální stromy
Pokud chceme od umělé inteligence chování, které bude v mnoha případech nezávislé na současném stavu, může využití automatů vést k enormnímu množství přechodů mezi stavy. Budoucí úpravy těchto přechodů mohou být pak velmi obtížné. Řešením této situace může být použití behaviorálních stromů. Základním stavebním blokem behaviorálních stromů na rozdíl od automatů není stav, ale úkol, který může mít návratovou hodnotu „running“ (probíhající), „success“ (úspěch) nebo „failure“ (neúspěch). Úkoly tvoří listy behaviorálního stromu, zatímco vnitřní uzly určují logiku vykonávání svých potomků.

2.3.2 Pohyb

Jednou ze základních schopností agentů v počítačových hrách je obvykle pohyb po scéně. Pohyb může mít mnoho podob od přímočarého pohybu mezi dvěma body po pronásledování pohyblivého cíle prostředím plným překážek. V této sekci jsou zmíněny různé druhy pohybu agentů tak, jak je popisuje Millington [15].

Jednoduchý pohyb
Nejjednodušším způsobem pohybu ve hře je rovnoměrný přímočarý pohyb mezi dvěma body. Tento způsob pohybu není náročný na implementaci ani výkon, ale také nepůsobí příliš realisticky, jelikož v reálném světě objekty nedosahují určité rychlosti instantně, ale postupně zrychlují. Z těchto důvodů je používán hlavně ve hrách, kde je na realistický pohyb agentů kladen malý důraz. Ve starých hrách můžeme často vypozorovat, že agenti buď stojí na místě, nebo se pohybují konstantní rychlostí do nového bodu. Jednoduchý pohyb může mít i reprezentativní účel, například v tahových hrách, kde se agenti z pohledu hry při jednotlivých tazích pohybují z místa na místo instantně, může být znázornění přesunu agentů jednoduchým pohybem pro hráče přívětivější.

Steering
Steering je proces, kdy o pohybu agenta rozhodujeme na základě jeho současné pozice, rychlosti a směru pohybu v kombinaci se zvolenou strategií (steering behavior). Většina strategií vyžaduje také informace o agentově současném cíli, či další informace specifické pro danou strategii. Jednoduchými příklady steering strategií mohou být strategie *seek* a *flee*, přičemž v první z nich se agent snaží co nejrychleji dostat na cílovou pozici, zatímco ve druhé se od ní naopak snaží co nejrychleji vzdálit. Příkladem složitější strategie je například *pursue*. Tato strategie je podobná strategii *seek*, avšak agent počítá i s pohybem cíle a svůj pohyb dle odhadu budoucí pozice pronásledovaného cíle vhodně uzpůsobuje. Steering strategie mají vždy jen jeden cíl (např. útěk od zadané pozice, následování zadané cesty, uhýbání překážkám, soudržnost hejna, atd.). Komplexnějšího chování pak můžeme dosáhnout kombinací více strategií.

Pathfinding
Poměrně běžnou úlohou řešenou inteligentními agenty v počítačových hrách je pohyb na nějaké místo co nejkratší cestou. Jedním možným řešením je využití steeringové strategie *seek*. Tato strategie však agenta vede přímo k cíli a nepočítá s možnými překážkami v cestě. Tento problém je někdy možné vyřešit kombinací strategie *seek* se strategií pro uhýbání překážek, avšak větší množství či složitost překážek může vést k situacím, kdy si strategie navzájem odporují. Příkladem je situace, kdy se cíl nachází za překážkou a *seek* agenta vede do překážky, zatímco uhýbací strategie zase od ní, což vede k zasekávání agenta a někdy i k tomu, že agent se vůbec nedostane do zvolené destinace.

Řešením této úlohy užívaným v praxi je proces zvaný *pathfinding* (hledání cesty), někdy také označovaný jako *path planning* (plánování cesty), při kterém agent vypočítá vhodnou cestu a následně se po ní vydá. Algoritmy pro výpočet nejkratší cesty obvykle nedokáží pracovat přímo s videoherní scénou, ale je pro ně třeba scénu reprezentovat vhodnou datovou strukturou. Jednoznačně dominantním algoritmem pro hledání nejkratších cest je algoritmus A*. Pro hledání nejkratší cesty je možné použít i Dijkstrův algoritmus, který však nalezne nejkratší cesty ze zvoleného bodu do všech ostatních bodů. Algoritmus A* je variace Dijkstrova algoritmu zaměřená na efektivní hledání nejkratší cesty mezi dvěma konkrétními body, k čemuž využívá heuristiku.

Jak již bylo zmíněno, pro potřeby algoritmu pro hledání nejkratší cesty je třeba herní scénu reprezentovat vhodnou datovou strukturou. V případě Dijkstrova algoritmu, A* a jejich variant je touto strukturou ohodnocený orientovaný graf bez záporných hran.

2.4 Unity

Unity obsahuje mnoho součástí a nástrojů pro snadnější tvorbu počítačových her. Mimo jiné mezi ně patří podpora zpracování vstupu od uživatele, simulace 2D či 3D fyziky, funkce pro práci s počítačovou grafikou, možnost psaní vlastních skriptů, import modelů a dalších souborů, podpora hraní po síti, nástroje pro práci s rozšířenou realitou a mnoho dalších. Výběr některých nástrojů je zřejmý již z povahy vyvíjené hry (2D hra bude jistě vyžadovat simulaci 2D fyziky, nikoli simulaci 3D fyziky). V této kapitole se zaměříme na nástroje, které je možné využít v praktické části práce.

2.4.1 Render pipeline

V této části práce se seznámíme s Render pipelines (RP), česky vykreslovacími řetězci, tak, jak jsou popsány v oficiální dokumentaci Unity. [16] Render pipeline je posloupnost operací, která je prováděna nad objekty ve scéně, jenž chceme vyobrazit na obrazovce. Různé RP mají odlišné možnosti a výkon, a jsou tak vhodné pro různé druhy her. Unity nabízí hned několik použitelných RP.

Základní, dnes již starší RP užívaný Unity, je *Built-in Render Pipeline*. Mezi jeho výhody patří vysoká univerzálnost a také kompatibilita se staršími projekty a položkami v Unity Asset Store. Jeho vysoká univerzálnost je však také nevýhodou, jelikož kvůli ní není tento RP optimalizován pro žádný konkrétní účel. Vývoj tohoto RP byl upozaděn a v budoucnu by měl být zcela nahrazen URP (viz níže).

Důležitou součástí Unity z hlediska RP je takzvaný *Scriptable Render Pipeline* (SRP). SRP je API umožňující konfiguraci renderování pomocí vlastních C# skriptů. Díky tomuto API můžeme vytvořit vlastní modulární RP pro použití v naší hře, nebo využít jeden ze dvou RP založených na tomto API, jenž Unity nabízí.

První z RP vyvíjených vývojáři Unity, jehož základen je SRP, se nazývá *Universal Render Pipeline* (URP). Silnými stránkami URP je především výkon a nízké nároky na hardware. Je určen pro použití na všech platformách podporovaných Unity včetně mobilních telefonů a přenosných konzolí. Zatímco *Built-in Render Pipeline* šlo měnit výběrem jednoho z několika pevně daných postupů renderování (rendering paths) a dále rozšiřovat vkládáním command bufferů do konkrétních událostí v rámci dané RP, URP tuto rozšiřitelnost dále prohlubuje tím, že umožňuje vývojářům definovat vlastní renderer implementující vlastnosti a strategii osvětlení. Součástí URP je pak předpřipravený forward renderer a 2D renderer. Hotové renderery jde pak dále rozšiřovat užitím specializovaných objektů zvaných renderer features. URP původně nesl název *Lightweight Render Pipeline*, avšak v rámci rebrandingu byl dle článku [17] na oficiálním Unity blogu ve verzi Unity 2019.3 přejmenován právě na *Universal Render Pipeline*. Přejmenování symbolizuje nabrání nového směru vývoje

URP, jehož cílem je vybudovat v budoucnu z URP plnohodnotnou náhradu *Built-in Render Pipeline* s důrazem na lepší výkon.

Druhý z RP vyvíjených vývojáři Unity, jehož základen je SRP, je *High Definition Render Pipeline* (HDRP). HDRP je určena pro renderování fotorealistických vizuálů založené na fyzikálních vlastnostech užitého osvětlení, materiálů a kamery. Je vhodná pro využití v AAA hrách, grafických demonstracích a architektonických renderech. Cílovou platformou pro využití HDRP jsou výkonné počítače a konzole.

2.4.2 Ukládání hry

Unity nabízí mnoho způsobů ukládání stavu hry. V této sekci jsou představeny tak, jak je popisuje článek [18] na oficiálním Unity blogu.

PlayerPrefs
Jednoduše použitelným nástrojem pro perzistenci dat je třída *PlayerPrefs*. Hlavním účelem této třídy je ukládání nastavení hry jako je hlasitost hudby, kvalita grafiky apod. Třída *PlayerPrefs* umožňuje ukládat data ve formě párů sestávajících z klíče a příslušné hodnoty. Exceluje především rychlostí a snadnou použitelností. Jednou z možných nevýhod tohoto způsobu ukládání dat je jejich snadné upravování ze strany uživatele. Dále je také možné používat pouze hodnoty typu int, float a string. Jelikož třída *PlayerPrefs* ukládá data jedné aplikace do jednoho souboru, není ji vhodné používat pro více různých uložených her.

JSON
Člověkem čitelný formát *JSON* je další možnou formou ukládání herních dat. Pro účely testování je možné snadno vytvářet umělá herní data, či kontrolovat správnost dat vytvořených. Snadná čitelnost znamená i snadné úpravy ze strany uživatele, což může být výhodou, pokud chceme podpořit tvorbu modifikací, ale výraznou nevýhodou, pokud se snažíme zabránit podvádění. *JSON* je hojně užívaný formát silně podporovaný většinou platforem, což je výhodou při tvorbě multiplatformních her. Taktéž je tento formát vhodný pro přenos dat po síti.

JsonUtility
Unity obsahuje zabudované API pro serializaci a deserializaci dat do formátu *JSON* zvané *JsonUtility*. Podobně jako *PlayerPrefs* je i *JsonUtility* relativně jednoduché na používání, avšak na rozdíl od *PlayerPrefs* musí ukládání dat do souboru provádět sám vývojář, což znamená vyšší nároky na implementaci, ale zároveň je možné ukládat data jedné aplikace do více různých souborů. Toto API je možné použít pouze pro práci s daty, které dokáže serializovat samo Unity, což je zajisté nevýhodou, avšak zároveň je *JsonUtility* rychlejší než ostatní řešení serializace dat do formátu *JSON*.

2.4.3 Podpora umělé inteligence

V této sekci jsou popsány často užívané či jinak zajímavé nástroje pro snazší implementaci umělé inteligence agentů v Unity, jejich užití v rámci této práce je vhodné zvážit.

Animation State Machines
V závislosti na stavu animované entity je ve hrách často třeba přehrávat různé animace. V Unity animace zprostředkovává animační systém *Mecanim*, který vývojářům umožňuje modelovat vlastní animační automaty (Animation State Machines) [19] sestávající z animačních stavů a přechodů mezi nimi. Z hlediska inteligence agentů jsou animační automaty zajímavé možností přiřazovat jednotlivým stavům skripty určující chování entity v tomto stavu (State Machine Behaviours). V těchto skriptech je možné definovat mimo jiné události při vstupu do stavu, setrvání ve stavu a opuštění stavu.

A* Pathfinding Project
Unity nabízí poměrně pokročilý systém pro navigaci agentů scénou. [20] Tento systém je však vytvořen pro navigaci ve 3D scéně a ve 2D hrách ho nelze využít. Nejrozšířenějším alternativním nástrojem pro navigaci agentů ve scéně v Unity je balíček *A* Pathfinding Project*, [21] který lze použít ve 3D hrách i 2D hrách. Tento balíček umožňuje snadnou reprezentaci scény grafem a následné procházení tohoto grafu. *A* Pathfinding Project* se dělí na placenou a neplacenou variantu, přičemž varianta zdarma je ochuzena o některé pokročilé funkce. Samotné prohledávání grafu scény je nicméně implementováno v obou verzích stejně a to s užitím algoritmu A*.

Unity ML-Agents Toolkit
Unity nabízí sadu nástrojů zvanou *Unity ML-Agents Toolkit*, kterou je možné dodatečně nainstalovat do Unity editoru [22]. Tato sada nástrojů umožňuje využít herní prostředí pro učení agentů využitím metod strojového učení skrze Pythonovské API. Takto vytrénované agenty lze posléze využít k ovládání chování nehráčských postav, či automatizované testování herního obsahu. Implementace této sady nástrojů je založena na open source knihovně strojového učení PyTorch.

Kapitola 3

Analýza

V této kapitole jsou rozebrány hry podobného zaměření, jako je hra, jejíž prototyp je vyvíjen v rámci této práce. Dále je pak provedena analýza funkčních a nefunkčních požadavků a také rozbor využitelnosti konceptů a nástrojů popisovaných v kapitole 2.

3.1 Současná řešení

V této podkapitole jsou hry s budovatelskými prvky rozděleny dle zaměření a užívaných mechanik. Podkapitola taktéž obsahuje příklady zmíněných her, přičemž se jedná zejména o hry vzniklé v posledních deseti letech. Obvykle není možné nahlédnout do vnitřní implementace počítačových her, proto se zaměříme především na jejich obsah a zajímavé mechaniky.

3.1.1 Budovatelské hry bez kompetice

Městské simulátory
Asi nejzákladnějším příkladem budovatelských her jsou různorodé městské simulátory. V současnosti nejpopulárnějším městským simulátorem je dle statistik [23] videoherní platformy Steam hra *Cities: Skylines* [24]. Jako příklad z naší krajiny je pak vhodné zmínit slovenskou hru *Workers & Resources: Soviet Republic* [25]. Základním prvkem tohoto typu her bývá komplexní simulace ekonomiky. Náplní hry je především stavění budov za účelem zisku kapitálu, který následně hráč investuje do dalších staveb. Důležitou mechanikou je možnost odemykání nových staveb, která hráči poskytuje dlouhodobé cíle a motivuje k dalšímu hraní. Díky novým stavbám či dalším odemykatelným prvkům se hra zároveň vyhýbá monotónnosti. Stavěné budovy a jejich obyvatelé dále mívají různorodé potřeby a nároky na infrastrukturu, které hráč musí plnit a které postupem hry nabývají na komplexitě. V principu téměř stejně jako městské simulátory fungují i simulátory různorodých zábavních

parků či zoologických zahrad, jako například *Planet Zoo* [26] nebo *Jurassic World Evolution* [27].

Střídání ročních období
Zajímavou mechanikou pro zpestření hratelnosti, často užívanou v hrách simulujících budování města, je střídání ročních období. Příkladem hry užívající střídání čtyř ročních období odpovídajících mírnému podnebí je simulátor budování středověké vesnice *Banished* [28]. Roční období mají vliv na rychlost růstu rostlin, čímž nepřímo ovlivňují produkci jídla. Dále je pak v zimě potřeba chránit vesničany před chladem, v létě zase před horkem. Díky tomuto střídání hráči nestačí rozhodovat se podle současných potřeb vesnice, ale musí plánovat do budoucna, aby dokázal překonat i nadcházející roční období. Roční období nemusí být samozřejmě jen jaro, léto, podzim a zima. Ve hře *Timberborn* [29] se třeba setkáme s obdobím sucha a obdobím vláhy. Hráč se musí během období vláhy kromě běžného rozvoje města soustředit i na tvorbu zásob vody a potravin, kterých je v období sucha nedostatek. V pozdější fázi hry hráč získává možnost stavět přehrady, které může využít k zadržování vody a tím i zásadně ovlivňovat okolní krajinu. Rostoucí nevlídnost prostředí nemusí být jen řadovou mechanikou, ale může se jednat i o ústřední prvek celé hry. Ve hře *Frostpunk* [30] se setkáme pouze s jedním ročním obdobím, kterým je zima, která však neustále přituhuje. Hráčova rozhodnutí pak nejsou řízena snahou o maximalizaci zisku, ale především úsilím o maximalizaci přežití obyvatel.

Hry s větším důrazem na jednotlivé obyvatele
V dosud zmíněných hrách staví hráči rovnou celé budovy a cesty, je však možné jít i více do detailu. Ve hře *Prison Architect* [31], což je simulátor stavby a správy věznice, hráč nestaví celé budovy, ale v prostředí v podobě čtvercové mřížky staví jednotlivé bloky zdí, podlahy i nábytek. Detailnější pohled se pak týká i postav ve hře. Zatímco v dříve zmíněných hrách můžeme postavy pozorovat pouze při cestě z jedné budovy do jiné (např. z domova do práce), zde můžeme podrobně sledovat jejich celý denní cyklus. Zároveň můžeme postavy snadněji identifikovat a případně si k nim i vytvořit vztah.

Tvary herních plánů
Budovatelské hry nabízejí mnoho variant uspořádání prostředí. Zřejmě nejspořádanější variantou je prostředí rozdělené do čtvercových (*Timberborn*) či šestiúhelníkových (*Before We Leave* [32]) polí, kdy každá budova zabírá jedno či více z nich. Tato varianta poskytuje nejmenší svobodu v umisťování staveb, na druhou stranu ale usnadňuje plánovat využití prostoru do budoucna (víme přesně, kolik máme prostoru a tedy i kolik staveb do něj můžeme postavit) a umožňuje prostor co nejefektivněji využít. Naprosto opačným přístupem je pak možnost stavby umisťovat a rotovat naprosto volně. Příkladem hry uplatňující tento přístup je simulátor budování městečka v osmnáctém století

Ostriv [33]. V této hře hráč ani nestaví cesty mezi budovami. Ty vznikají automaticky putováním vesničanů. Některé budovatelské hry pak volí nějakou formu střední cesty. Například ve hře *Cities: Skylines* je možné cesty umisťovat libovolně, ale budovy lze stavět jen do čtvercové mřížky v okolí postavených cest.

3.1.2 Budovatelské hry s kompeticí

Nepřátelští agenti přicházející z vně mapy
Zásadní mechanikou, která může mít velký vliv na podobu hry, je přítomnost nějaké formy kompetice. Příklad poměrně jednoduché formy kompetice nalezneme ve hře *Rimworld* [34]. *Rimworld* je sci-fi simulátor budování kolonie na cizí planetě. Podobně jako ve dříve zmíněné hře *Prison Architect* je zde stavba vedena velmi dopodrobna. Zároveň je kladen velký důraz na jednotlivé kolonisty, kterých je velmi omezený počet a každý z nich má vlastní schopnosti a povahové rysy. Hráč může své kolonisty i v případě potřeby přímo ovládat, ačkoli jen omezeně. Kompetice má v *Rimworld* formu opozičních agentů, jenž se mohou objevit na okraji mapy v rámci náhodně generovaných událostí a následně se snaží napadnout hráčovu kolonii. Nepřátelští agenti obvykle nejprve nějakou dobu vyčkávají na okraji mapy, čímž dají hráči čas na přípravu obrany, a poté vyrazí nejkratší cestou k nějakému cíli (jeden z kolonistů, zásoba surovin, překážka kterou chtějí zničit, ...). Pokud je nepřátelský agent po cestě k cíli raněn, zvolí si za nový cíl původce utrženého zranění. V případě, že zranění agenta vážně ohrožují na životě, pokusí se o útěk. V případě smrti určitého množství agentů se ostatní vydají na ústup, čímž je simulováno pozbytí vůle bojovat při vysokých ztrátách.

Nepřátelští agenti rozmístění po mapě
Nepřátelští agenti nemusí pouze přicházet z vnějšku mapy, ale mohou pobývat přímo na ní. V české hře *Factorio* [35] se nepřátelští agenti v podobě mimozemské fauny rodí v hnízdech rozmístěných po mapě. *Factorio* je budovatelská hra zaměřená na stavbu komplexních továren a automatizaci. Od dříve zmíněných her se odlišuje také tím, že hráč zde nehraje za nehmotnou entitu udávající rozkazy, ale ovládá přímo postavu v herním světě. Nepřátelé se zde rodí v pravidelných intervalech v hnízdech a po narození se pohybují v jejich okolí. Jednou za několik minut se od nepřátel v okolí hnízda oddělí skupina pěti až dvaceti jedinců a vyrazí založit další hnízdo v blízkém okolí. Důležitou mechanikou je také znečištění ovzduší. Továrny budované hráčem produkují znečištění, které se vzduchem šíří do okolí. Vyšší hladina znečištění v okolí hnízd pak způsobuje častější a masivnější útoky nepřátel. To hráče nutí k výběru mezi využíváním ekologických technologií, které jsou méně efektivní, a výhodnějších neekologických technologií, které však způsobují vyšší intenzitu útoků.

Kombinace přístupů
Přístup zaplnění mapy nepřátelskými agenty předem a užití vln nepřátelských agentů přicházejících zpoza okrajů mapy lze samozřejmě i kombinovat, jako tomu je ve hře *They Are Billions* [36]. *They Are Billions* je budovatelská hra zaměřená na postupné obsazování mapy zaplněné nepřáteli v kombinaci s obranou základny před nárazovými útoky s rostoucí intenzitou. Hráč musí neustále rozšiřovat své pole působnosti, aby získal přístup k novým surovinám a technologiím, ale zároveň nesmí zabírat prostor příliš rychle, aby ho dokázal ubránit před vlnami nepřátel. Další novinkou oproti dříve zmíněným hrám je možnost produkce přátelských agentů v podobě vojáků, které hráč může přímo ovládat a užívat je k průzkumu mapy či boji s nepřítelem.

Zabírání mapy
Ve hře *Rise to Ruins* [37] je mechanika obsazovaní mapy obrácena. Nepřátelští agenti zde postupně obsazují prostor a hráč se jim v tom snaží zabránit. Nepřátelské agenty zde lze rozdělit dle chování na dva druhy a to na „stavitele“ a „válečníky“. Stavitelé se pohybují pouze po území obsazeném nepřítelem a jsou zodpovědní za stavbu budov, které buď dodávají různé bonusy k vlastnostem válečníků (vyšší síla, rychlost, odolnost, ...), nebo chrání nepřátelské území útočením na blízké budovy a agenty hráče. Válečníci jsou zodpovědní za boj s hráčem. Ve dne jsou převážně pasivní a pohybují se převážně po území obsazeném nepřítelem, v noci pak začnou hromadně útočit na území hráče, na což musí být hráč náležitě připraven.

Netradiční formy kompetice
Kompetice v budovatelských hrách nemusí mít nutně podobu nepřátelských agentů. Zajímavou alternativu nabízí např. herní série *Creeper World* [38]. Hráčův úhlavní nepřítel má v této sérii formu žíravé kapaliny zvané „Creeper“, která ničí vše, čeho se dotkne. Creeper je generován několika předem rozmístěnými budovami a postupně zaplňuje celou mapu. Hráč musí především chránit svoji hlavní budovu před zničením. Pro vítězství je pak třeba zničit všechny budovy produkující Creeper, což je možné pouze z krátké vzdálenosti, tudíž hráč musí postupně získat kontrolu nad herním prostorem strategickou výstavbou věží ničících Creeper. Tyto věže je třeba podporovat stavbami produkujícími energii a potřebné suroviny.

3.1.3 Shrnutí

Prostředí v budovatelských hrách může mít tvar pravidelné mřížky, nebo být částečně či zcela volné. Hráčův zážitek je možné ozvláštnit změnami prostředí s dopadem na hratelnost. Tato možnost nebude v rámci prototypu využita, jelikož by měla velký dopad na časovou náročnost implementace, avšak přínos z pohledu cílů této práce by byl minimální. Změny prostředí by zároveň bylo

obtížné skloubit s plánovanou mechanikou postupného osvětlování herní mapy, která bude podrobněji popsána v podkapitole 4.1.

Pohyb agentů je v budovatelských hrách zpravidla ztvárněn jednoduchým hledáním a následováním cesty a steeringová chování se vyskytují jen zřídka. Způsob zpracování rozhodování agentů nelze bohužel bez možnosti nahlédnutí do implementace určit. Obecně lze říci, že chování nepřátelských agentů bývá v budovatelských hrách často velmi přímočaré. V případě komplexních agentů pak bývá kladen důraz spíše na uvěřitelnost chování, než na maximální efektivitu. Strategie rozmístění agentů ve scéně se také mohou lišit, přičemž vhodná strategie záleží na konkrétním stylu hry.

3.2 Využití nástrojů Unity

V této podkapitole je krátce odůvodněno využití či nevyužití různých nástrojů a doplňků Unity popisovaných v podkapitole 2.4.

Z nabízených RP byla pro tuto práci zvolena *Universal Render Pipeline* a to zejména kvůli obsahu 2D rendereru. Ve vyvíjeném prototypu bude osvětlení hojně využíváno a světla určená pro 2D hry, které 2D renderer nabízí, jsou vítanou možností efektivního osvětlení scény.

Pro ukládání hry byla vybrána možnost využití třídy *PlayerPrefs*, která umožňuje snadné ukládání základních údajů o hráčově postupu ve hře. Ostatní varianty ukládání hry byly vyřazeny jako zbytečně implementačně náročné pro použití v prototypu. Je vhodné zmínit, že velmi chudá verze prototypu by si vystačila i zcela bez ukládání stavu hry, čímž by však byla omezena plánovaná hratelnost. Možnost podvádění ze strany hráče nehraje z pohledu naší hry roli, jelikož případné podvádění nebude mít vliv na ostatní hráče. V případě testování prototypu pak může být možnost podvádění i výhodou.

Z nástrojů pro podporu umělé inteligence uvedených v sekci 2.4.3 bude využit balíček *A* Pathfinding Project*, který usnadní navigaci agentů po scéně. Animační stavové automaty využity nebudou, jelikož jsou vhodné spíše pro jednoduché chování v jednotlivých stavech, a je obtížné přenášet informace mezi stavy. *Unity ML-Agents Toolkit* taktéž nebude využit, jelikož je časově náročný na používání a poskytuje malou kontrolu nad výsledným chováním agentů, což je pro tuto práci zásadní nedostatek.

3.3 Analýza požadavků

V této kapitole jsou sepsány požadavky na prototyp vytvářený v rámci této práce. Požadavky jsou rozděleny do dvou kategorií na funkční, které jsou popsány v tabulce 3.1, a nefunkční, které jsou popsány v tabulce 3.2. Každý požadavek obsahuje identifikační kód, popis daného požadavku a prioritu daného požadavku, přičemž priority požadavků mají čtyři možné hodnoty dle prioritizační techniky *MoSCoW* [39].

Tabulka 3.1: Funkční požadavky

Kód požadavku	**FP1**
Požadavek	Hráč může zvolit budovu z nabídky a postavit ji kliknutím na vybraném místě, pokud má dostatek surovin a dané místo to umožňuje.
Priorita	Must have (musí)
Kód požadavku	**FP2**
Požadavek	Hráč může pohybovat kamerou nad scénou do čtyř směrů a měnit její přiblížení.
Priorita	Must have (musí)
Kód požadavku	**FP3**
Požadavek	Prototyp bude hráčovi umožňovat, aby zjistil aktuální počet svých surovin a jejich produkci.
Priorita	Must have (musí)
Poznámka	Počty a produkce jednotlivých surovin v hráčově vlastnictví budou vyobrazeny v uživatelském rozhraní zobrazeném během hry.
Kód požadavku	**FP4**
Požadavek	Hráč se může před nepřátelskými agenty bránit stavbou obranných budov. Aktivní obranné budovy budou automaticky útočit na nepřítele. Obranná budova na nepřítele může reagovat pouze v případě, že je pro ni nepřítel viditelný.
Priorita	Must have (musí)
Kód požadavku	**FP5**
Požadavek	Hráč si může zobrazit podrobné informace o postavené budově kliknutím na danou budovu a se zvolenou budovou provádět další akce. Konkrétní podoba akcí závisí na typu zvolené budovy.
Priorita	Should have (mělo by)
Kód požadavku	**FP6**
Požadavek	Hra bude automaticky ukládat stav některých druhů surovin po dohrání úrovně.
Priorita	Should have (mělo by)
Kód požadavku	**FP7**
Požadavek	Hráč může suroviny získané dohráním úrovně utratit za vylepšení, která jsou perzistentní mezi úrovněmi.
Priorita	Should have (mělo by)
Kód požadavku	**FP8**
Požadavek	Hráč si může prohlédnout druhy chování agentů ve hře v připravených ukázkách.
Priorita	Should have (mělo by)

Tabulka pokračuje na další straně

Pokračování tabulky funkčních požadavků	
Kód požadavku	**FP9**
Požadavek	Hráč si v prototypu může zobrazit stručný popis ovládání a mechanik hry.
Priorita	Could have (mohlo by)
Kód požadavku	**FP10**
Požadavek	Prototyp bude obsahovat zvukové efekty (SFX), které budou vydávány vhodnými objekty.
Priorita	Won't have (nebude součástí)
Poznámka	Zvukové efekty nehrají ve hře významnou roli, proto je jejich priorita v rámci prototypu nízká.
Kód požadavku	**FP11**
Požadavek	Hráč si může uložit stav právě rozehrané úrovně.
Priorita	Won't have (nebude součástí)
Poznámka	Tento požadavek by v plnohodnotné hře měl jistě být naplněn, prototyp však tuto funkcionalitu nevyžaduje.
Kód požadavku	**FP12**
Požadavek	Prototyp bude obsahovat hudbu, která se bude přehrávat na pozadí menu i úrovní.
Priorita	Won't have (nebude součástí)
Poznámka	Hudba je důležitou součástí pro doplnění atmosféry hry, v prototypu však není třeba.

Tabulka 3.2: Nefunkční požadavky

Kód požadavku	**NP1**
Požadavek	Prototyp bude spustitelný v operačním systému Windows 10.
Priorita	Must have (musí)
Poznámka	Prototyp je vyvíjen především pro operačním systém Windows 10, do budoucna je však možné rozšíření i na další OS či platformy.
Kód požadavku	**NP2**
Požadavek	Prototyp bude podporovat ovládání klávesnicí a myší.
Priorita	Must have (musí)
Poznámka	Myš a klávesnice jsou nejběžnějším způsobem ovládání u strategických her. Zároveň tento způsob ovládání odpovídá cílové platformě.
Kód požadavku	**NP3**
Požadavek	Prototyp bude vyvíjen v herním enginu Unity.
Priorita	Must have (musí)
Poznámka	Plyne ze zadání.

Kapitola 4

Návrh

V této kapitole je popsán návrh jednotlivých prvků vytvářeného prototypu hry. Nejprve je věnována pozornost obecnému fungování a mechanikám hry, dále pak konfiguraci herní mapy, chování nepřátelských agentů a na závěr také uživatelskému rozhraní.

4.1 Mechaniky

Ačkoli příběh hry nebude součástí prototypu vytvářeného v rámci praktické části této práce, znalost jeho základních rysů je důležitá pro získání potřebného kontextu hry. Hráč se ujme role kapitána těžařské vesmírné lodi, jejíž úkolem je těžit suroviny na toulavé planetě (anglicky rogue planet), která se přiblížila ke Sluneční soustavě. Při doletu na oběžnou dráhu se ukáže, že planeta, která měla být podle všech předpokladů naprosto pustá, obsahuje mimozemský život, který se jakékoli těžbě aktivně brání.

Základní koloběh
Samotná těžařská loď na planetě nepřistává. Zdržuje se na oběžné dráze a na planetu sesílá pouze základový modul, kolem kterého může hráč stavět další budovy. Pokud hráč natěžil dostatek surovin, nebo je jeho základový modul v ohrožení, může se s ním vrátit zpět na loď a odnesené suroviny využít k zakoupení různých vylepšení či odemknutí nových budov. Získaná vylepšení jsou trvalá a hráč je může využívat při dalších přistáních na povrchu planety. To platí i pro odemčené budovy. Době mezi přistáním a odletem či zničením základového modulu říkáme jeden běh.

Prozkoumávání herní mapy
Jelikož toulavá planeta neobíhá kolem žádné hvězdy, je její povrch zahalen tmou. Hráč musí mapu postupně prozkoumávat s užitím dostupného výběru osvětlovacích budov či dalších nástrojů. Znalost objevených pozic surovin

a dalších prvků mapy pak může hráč využít při následujících přistáních, jelikož základový modul je vždy sesílán na stejné místo na planetě. Taktéž na nepřátelské agenty je možné útočit pouze pokud jsou na světle.

Suroviny a energie
Aby hráč mohl stavět budovy potřebuje suroviny, které lze získat těžbou na povrchu planety. Postavené budovy pak pravidelně spotřebovávají energii a je tedy třeba produkovat dostatek energie pro udržení postavených budov v provozu. Suroviny nevyužité k stavbě může hráč využít k zvýšení produkce energie, nebo je rafinovat. Pouze rafinované suroviny je možné odnést z mapy a využít k vylepšování.

4.2 Mapa

V této podkapitole jsou popsány obecné vlastnosti herní mapy a dále jsou rozebrány jednotlivé prvky, které se na ní vyskytují.

Mapa má tvar čtvercové mřížky, přičemž budovy stavěné hráčem zabírají nejčastěji právě jedno pole, mohou však pokrývat i více polí. Ostatní prvky na mapě se taktéž skládají z jednoho či více polí. Hráč začíná ve středu mapy, kde se nachází jeho základový modul. Nepřátelé se na začátku hry nacházejí na okrajích mapy a postupně zabírají více prostoru. Nejběžnější suroviny jsou rozmístěny po mapě rovnoměrně, zatímco vzácnější se nacházejí pouze dále od středu mapy.

4.2.1 Suroviny

Suroviny na mapě mají podobu těžitelné rudy pokrývající jedno či více polí. Ve hře se bude vyskytovat více druhů rudy s různou vzácností, rozmístěním a využitím.

Bílá ruda
Nejběžnější surovinou na mapě je bílá ruda (white ore). Vyskytuje se po celé mapě včetně středu. K její těžbě není třeba vlastnit žádná vylepšení. Je potřebná k stavbě všech budov.

Červená ruda
Druhou nejběžnější surovinou na mapě je červená ruda (red ore). Vyskytuje se po celé mapě včetně středu. K její těžbě není třeba vlastnit žádná vylepšení. Je možné ji rafinovat na červené krystaly (red crystals), které jsou potřeba k zakoupení základních vylepšení. Je ji možné spalovat a získávat tím malé množství energie.

Zelená ruda
Surovinou vzácnější než červená ruda, je ruda zelená (green ore). Vyskytuje se pouze od určité vzdálenosti od středu mapy a na její získání je třeba zakoupit vylepšení za červené krystaly, takže ji hráč může těžit nejdříve při druhém přistání. Je možné ji rafinovat na zelené krystaly (green crystals), které jsou potřeba k zakoupení pokročilých vylepšení. Je ji možné spalovat a získávat tím střední množství energie.

Modrá ruda
Nejvzácnější surovinou na mapě je modrá ruda (blue ore), která se vyskytuje pouze poblíž okrajů mapy. Na její získání je třeba zakoupit vylepšení za zelené krystaly, takže ji hráč může těžit nejdříve při třetím přistání. Je možné ji rafinovat na modré krystaly (blue crystals), které jsou potřeba k zakoupení pokročilejších vylepšení. Je ji možné spalovat a získávat tím velké množství energie.

4.2.2 Ostatní prvky mapy

Láva
Pole s lávou jsou zdrojem přírodního osvětlení, které hráč může využít k snazší orientaci na mapě, nebo jako levný způsob odhalování nepřátel. Po odemčení příslušné budovy je možné lávu využívat i jako zdroj energie.

Ruiny
Toulavá planeta obsahuje známky pradávných civilizací v podobě ruin. Ruiny jsou budovy předem postavené na mapě, které jsou ignorovány nepřátelskými agenty. Po zakoupení příslušného vylepšení může hráč ruinám dodat energii a tím získat benefity v závislosti na konkrétní aktivované ruině.

Houby
Jedinou formou flóry, která dokázala přežít na planetě bez světla, jsou houby. Houby jsou prvkem mapy, který pokrývá vždy větší počet polí najednou, přičemž v základním stavu nemají žádný efekt. Po zakoupení příslušného vylepšení může hráč houbám dodat živiny, čímž se z nich stane zdroj osvětlení, který nemůže být zničen nepřítelem.

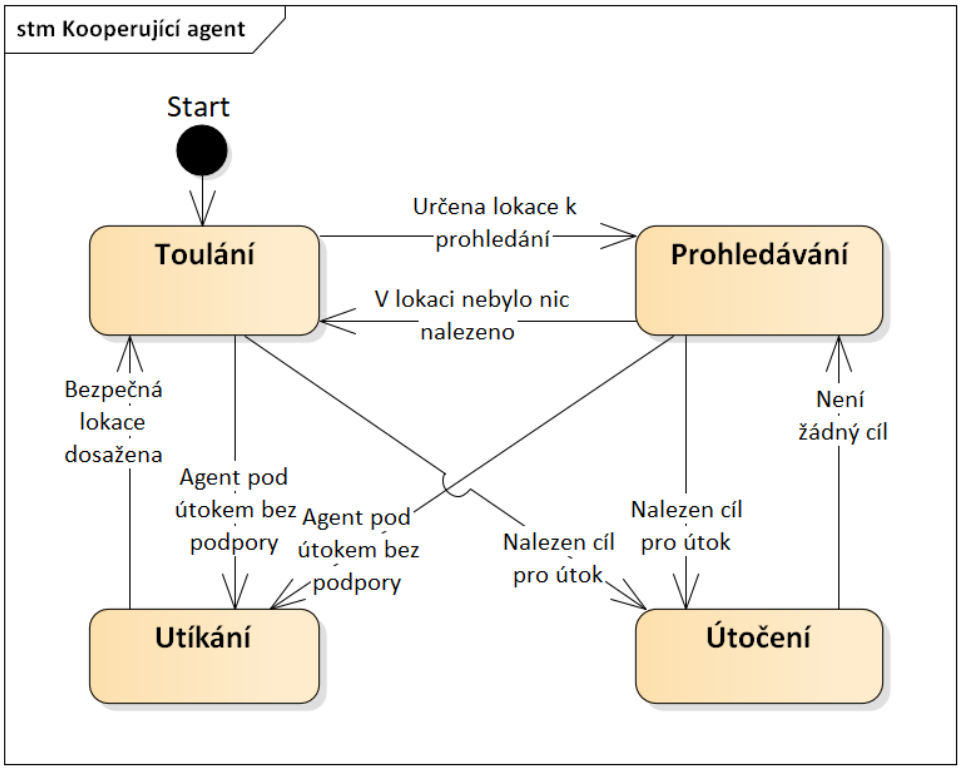
4.3 Nepřátelští agenti

V této podkapitole jsou navrženy tři možné modely chování nepřátelských agentů ve vyvíjeném prototypu hry. První model představuje chování, které je dle autora optimální pro využití v budovatelské hře. Další dva modely jsou pak inspirovány modely inteligence hejna z podkapitoly 2.2. Nejedná se o pokus implementovat konkrétní optimalizační algoritmus, ale spíše o napodobení

přírodních jevů, kterými jsou modely inteligence hejna inspirovány. Ve všech případech se agenti rodí nepřetržitě na určených lokacích u okrajů mapy. Vznik nových agentů může být pozastaven dosažením limitu agentů ve scéně.

4.3.1 Kooperující agenti

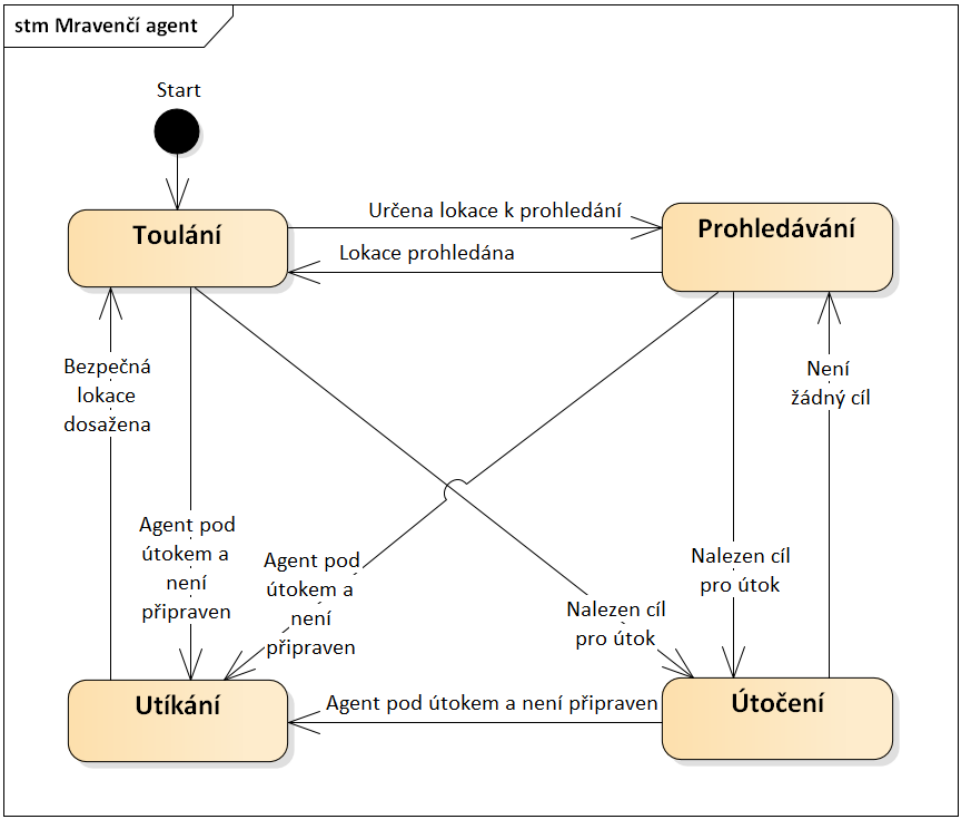
Prvním navrženým typem nepřátelských agentů jsou takzvaní kooperující agenti. Agenti tohoto typu mají jedinou úlohu, kterou je nalezení a zničení hráčových budov. Základem jejich chování je automat na obrázku 4.1. Hlavním účelem tohoto agenta je racionální chování při setkání s hráčem, a výpočetně málo náročné chování ve chvíli, kdy je agent mimo hráčův dohled.



Obrázek 4.1: Stavy kooperujícího agenta

Kooperující agenti začínají ve stavu toulání, při kterém náhodně procházejí mapou, přičemž dokáží detekovat budovy blízko nich. Pokud agent narazí na budovu, zaútočí na ni. Jelikož mají obranné budovy větší dohled než agenti, nastane často situace, kdy se agent ve fázi toulání ocitne pod palbou. V takovém případě si agent spočítá ostatní agenty ve svém okolí a na základě jejich počtu se buď rozhodne pro útěk, nebo se vydá prozkoumat oblast, ze které bylo střídeno, a zároveň vyzve k prohledávání této oblasti okolní agenty.

Pokud při prohledávání oblasti agent narazí na budovy, zaútočí na ně, a po jejich zničení pokračuje dále na prohledávané místo. V případě, že agent dorazí na místo hledání a v oblasti se již nevyskytují žádné budovy, vrací se zpět do fáze toulání. Rozhodl-li se agent pro útěk, vrací se po dosažení bezpečné lokace rovněž zpět do fáze toulání.



Obrázek 4.2: Stavy mravenčího agenta

4.3.2 Mravenčí agenti

Druhým navrženým typem nepřátelských agentů ve hře jsou takzvaní mravenčí agenti. Chování tohoto typu agentů je inspirováno hledáním potravy u mravenců v kombinaci s předáváním informací v hejnech. Jelikož výsledné chování těchto agentů připomíná především chování mravenců, byli zpětně pojmenováni právě podle nich. Agenti tohoto typu se snaží plnit fiktivní úlohu, kdy usilují o posbírání každého ze tří druhů barevné rudy vzestupně dle vzácnosti a poté se snaží nalézt budovy hráče a zaútočit na ně. Základem jejich chování je automat na obrázku 4.2. Automat, který je základem chování mravenčích agentů, je velmi podobný automatu předchozího typu agentů.

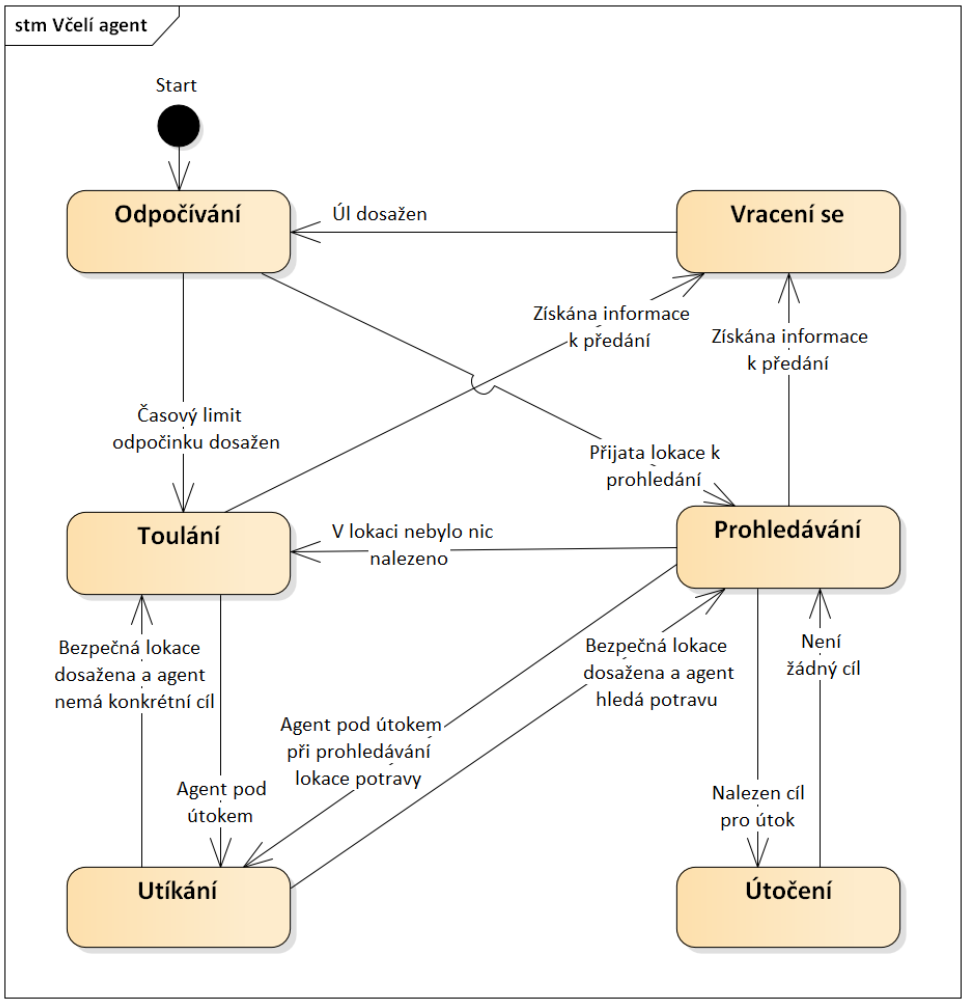
Výrazným rozdílem oproti kooperujícím agentům je však kromě pravidelné komunikace a zmíněné fiktivní úlohy především to, že mravenčí agenti si pamatují nejlepší lokace cílů zjištěné buď předáním informace od svých druhů, nebo samostatně nalezené. Mravenčí agenti také vědí, jaký druh cíle právě chtějí hledat. Rozhodování agentů není založeno pouze na základním vnitřním stavu a vnějších podnětech, ale také na dalších uložených informacích a cíli daného agenta.

Podobně jako kooperující agenti začínají mravenčí agenti ve stavu toulání, při kterém náhodně procházejí mapou. Během pohybu mapou se agent setkává s různými prvky mapy. Pokud agent narazí na barevnou rudu, uloží si její pozici. V případě, že se jedná o barevnou rudu, kterou agent právě hledal, sebere ji a vydá se hledat rudu další barvy. Informace o nalezených rudách si agenti pravidelně předávají s druhy ve svém okolí. Když agent obdrží lepší informaci o rudě, než dosud měl, novou informaci si uloží. V případě, že se jedná o rudu, kterou agent právě hledá, vydá se prohledat obdrženou lokaci. Totéž platí i pro nalezené budovy. Když se agent ocitne pod palbou hráčových obranných budov, rozhoduje se pro boj či útěk v závislosti na tom, zda již získal všechny druhy barevné rudy.

4.3.3 Včelí agenti

Třetím navrženým typem nepřátelských agentů jsou takzvaní včelí agenti. Základem jejich chování je automat na obrázku 4.3. Chování těchto agentů je inspirováno chováním včel při hledání potravy. Agenti tohoto typu plní fiktivní úlohu nošení barevných rud do úlu, přičemž hodnota rud se liší v závislosti na jejich vzácnosti.

Včelí agenti začínají ve stavu odpočinku, kdy vyčkávají v úlu. Odpočinek může být přerušen buď přijmutím zprávy od jiného agenta, nebo dosažením časového limitu odpočinku. Pokud agent odpočíval příliš dlouho bez přijetí jakékoli zprávy, začne náhodně procházet mapou a hledat zdroje barevných rud. Po nalezení vhodného zdroje nebo budovy hráče se agent vrací zpět do úlu, kde získanou informaci předává odpočívajícím agentům. Každý z právě odpočívajících agentů se může rozhodnout, zda informaci přijme a vydá se prozkoumat danou lokalitu, či nepřijme, a bude pokračovat v odpočívání. Pokud agent informaci přijal, prohledá určené místo, a v závislosti na úspěchu se vrací zpět do úlu nebo přechází do stavu toulání. V případě, že se agent ocitne pod palbou, závisí jeho chování na důvodu, proč se v dané lokalitě vyskytl. Pokud agent na místo dorazil na základě předané zprávy o hráčových budovách, rozhodne se zaútočit. V opačném případě se vrací do úlu, aby předal informaci ostatním agentům.



Obrázek 4.3: Stavy včelího agenta

4.4 Budovy

V této podkapitole jsou popsány budovy, které hráč může stavět, a jejich možná vylepšení. Množství surovin spotřebovaných k výstavbě je pouze orientační. Konkrétní hodnoty se mohou měnit v rámci balancování. To platí i pro průběžnou spotřebu energie a surovin. Hráč může postavit budovu pouze pokud vlastní množství surovin požadované na její výstavbu. Budovy lze stavět jen na místech, která mají přístup k energii. Velikost všech budov je právě jedno pole, není-li uvedeno jinak. Každá budova je popsána v samostatné tabulce:

- Reflektor v tabulce 4.1
- Základový modul v tabulce 4.2
- Rozvodna energie v tabulce 4.3
- Kulomet v tabulce 4.4
- Těžební vrták v tabulce 4.5
- Bodové světlo v tabulce 4.6
- Generátor v tabulce 4.7
- Rafinérie v tabulce 4.8

Tabulka 4.1: Reflektor

Budova	
Název	**Reflektor** (Spotlight)
Popis	Rotující světlo, které může v jeden moment svítit jen jedním směrem, avšak má větší dosah. Pokud osvítí nepřítele, začne ho sledovat.
Cena	Malé množství bílé rudy
Spotřeba	Střední množství energie
Produkce	Žádná
Možná vylepšení	
Vylepšení	Zvýšení dosahu světla
Popis	Zvětší vzdálenost, na jakou reflektor dosvítí. Toto vylepšení lze koupit třikrát.
Cena	Červené krystaly při prvním nákupu, zelené při druhém, modré při třetím.

Tabulka 4.2: Základový modul

Budova	
Název	**Základový modul** (Base)
Popis	Základový modul je hráčovou hlavní budovou. Je postaven automaticky na začátku běhu a hráč nemůže jeho lokaci změnit, ani postavit více instancí této budovy. Hráč může kdykoli odstartovat vzlet základového modulu zpět na oběžnou dráhu, čímž ukončí současný běh a získá všechny rafinované suroviny. Běh může být ukončen také zničením základového modulu a v takovém případě hráč obdrží pouze malou část rafinovaných surovin. Základový modul zajišťuje přístup k energii okolním polím. Součástí modulu je zdroj světla osvětlující okolí. Tato budova také poskytuje základní produkci energie a bílé rudy, aby hráč měl přístup k surovinám na začátku běhu.
Cena	Žádná
Spotřeba	Žádná
Produkce	Malé množství energie a bílé rudy
Možná vylepšení	
Vylepšení	Zvýšení dosahu energie
Popis	Větší oblast okolo základového modulu, které je dodávána energie. Toto vylepšení lze koupit mnohokrát.
Cena	Červené krystaly, množství roste s každým dalším nákupem.
Vylepšení	Zvýšení dosahu světla
Popis	Zvýší dosah světla vyzařovaného základovým modulem. Toto vylepšení lze koupit mnohokrát.
Cena	Červené krystaly, množství roste s každým dalším nákupem.
Vylepšení	Zvýšení základní produkce energie
Popis	Zvýší množství energie produkované základovým modulem. Toto vylepšení lze koupit vícekrát.
Cena	Kombinace různých barev krystalů v závislosti na úrovni vylepšení od nejméně vzácných po nejvzácnější.
Vylepšení	Zvýšení základní produkce bílé rudy
Popis	Zvýší množství bílé rudy produkované základovým modulem. Toto vylepšení lze koupit vícekrát.
Cena	Kombinace různých barev krystalů v závislosti na úrovni vylepšení od nejméně vzácných po nejvzácnější.

Tabulka 4.3: Rozvodna energie

Budova	
Název	**Rozvodna energie** (Power station)
Popis	Rozvodna energie dodává energii okolním polím a umožňuje stavbu budov na těchto polích.
Cena	Střední množství bílé rudy
Spotřeba	Střední množství energie
Produkce	Žádná
Možná vylepšení	
Vylepšení	Zvýšení dosahu energie
Popis	Zvětší oblast, které je dodávána energie okolo Rozvodny energie. Toto vylepšení lze koupit třikrát.
Cena	Červené krystaly při prvním nákupu, zelené při druhém, modré při třetím.
Vylepšení	Symbióza
Popis	Pokud je rozvodna postavena na pole s houbami, dodá jim energii. Toto vylepšení lze koupit pouze jednou.
Cena	Mnoho červených krystalů.
Vylepšení	Aktivace ruin
Popis	Rozvodna energie aktivuje všechny ruiny v dosahu. Toto vylepšení lze koupit pouze jednou.
Cena	Mnoho zelených krystalů.

Tabulka 4.4: Kulomet

Budova	
Název	**Kulomet** (Machinegun)
Popis	Obranná věž střílející automaticky po osvětlených nepřátelích v dosahu.
Cena	Velké množství bílé rudy
Spotřeba	Malé množství energie
Produkce	Žádná
Možná vylepšení	
Vylepšení	Zvýšení dostřelu
Popis	Zvětší vzdálenost, na jakou kulomet dostřelí. Toto vylepšení lze koupit třikrát.
Cena	Červené krystaly při prvním nákupu, zelené při druhém, modré při třetím.

Tabulka 4.5: Těžební vrták

Budova	
Název	**Těžební vrták** (Mining drill)
Popis	Budova pro těžbu rudy. Pokud je umístěna na políčko s rudou libovolné barvy, začne rudu této barvy produkovat. V základu může těžit pouze červenou a bílou rudu.
Cena	Střední množství bílé rudy
Spotřeba	Střední množství energie
Produkce	Ruda v závislosti na umístění
Možná vylepšení	
Vylepšení	Zvýšení efektivity
Popis	Mírně zvýší množství produkované rudy za vteřinu.
Cena	Kombinace různých barev krystalů v závislosti na úrovni vylepšení od nejméně vzácných po nejvzácnější. Toto vylepšení lze koupit vícekrát.
Vylepšení	Těžba zelené rudy
Popis	Umožňuje těžit zelenou rudu. Toto vylepšení lze koupit pouze jednou.
Cena	Malé množství červených krystalů
Vylepšení	Těžba modré rudy
Popis	Umožňuje těžit modrou rudu. Toto vylepšení lze koupit pouze jednou.
Cena	Malé množství zelených krystalů
Vylepšení	Těžba lávy
Popis	Po umístění na políčko s lávou začne vrták produkovat energii. Toto vylepšení lze koupit pouze jednou.
Cena	Mnoho zelených krystalů

Tabulka 4.6: Bodové světlo

Budova	
Název	**Bodové světlo** (Point light)
Popis	Bodové světlo má krátký dosah, ale osvětluje své okolí konzistentně ve všech směrech
Cena	Malé množství bílé rudy
Spotřeba	Střední množství energie
Produkce	Žádná
Možná vylepšení	
Vylepšení	Zvýšení dosahu světla
Popis	Zvětší osvětlenou oblast okolo bodového světla. Toto vylepšení lze koupit třikrát.
Cena	Červené krystaly při prvním nákupu, zelené při druhém, modré při třetím.

Tabulka 4.7: Generátor

Budova	
Název	**Generátor** (Generator)
Popis	Generátor spaluje rudu vybrané barvy (libovolná mimo bílé) a přetváří ji na energii. Barvu spalované rudy lze měnit i po postavení generátoru.
Cena	Střední množství bílé rudy
Spotřeba	Ruda zvolené barvy
Produkce	Energie v závislosti na vzácnosti spalované rudy
Možná vylepšení	
Vylepšení	Zvýšení efektivity
Popis	Zvýší produkci energie beze změny spotřeby rudy. Toto vylepšení lze koupit vícekrát.
Cena	Kombinace různých barev krystalů v závislosti na úrovni vylepšení od nejméně vzácných po nejvzácnější. Toto vylepšení lze koupit vícekrát.

Tabulka 4.8: Rafinérie

Budova	
Název	**Rafinérie** (Rafinery)
Popis	Rafinerie slouží k produkci barevných krystalů z rud příslušné barvy.
Cena	Střední množství bílé rudy
Spotřeba	Ruda zvolené barvy, malé množství energie
Produkce	Krystaly v závislosti na rafinované rudě
Možná vylepšení	
Vylepšení	Zvýšení efektivity
Popis	Zvýší produkci krystalů bez změny spotřeby rudy a energie. Toto vylepšení lze koupit vícekrát.
Cena	Kombinace různých barev krystalů v závislosti na úrovni vylepšení od nejméně vzácných po nejvzácnější. Toto vylepšení lze koupit vícekrát.

4.5 Uživatelské rozhraní

V této podkapitole je popsán obsah a plánovaný vzhled uživatelského rozhraní vyvíjeného prototypu hry. Prvky navrženého uživatelského rozhraní lze logicky rozdělit do tří kategorií, kterými jsou: hlavní menu, uživatelské rozhraní v průběhu hry a obrazovka s vylepšeními. Obrázky wireframů navržených obrazovek naleznete na konci podkapitoly.

4.5.1 Hlavní menu

Hlavní menu (obrázek 4.4) je první obrazovka, kterou hráč po spuštění hry uvidí. Z této obrazovky je možné navigovat užitím tlačítek přímo do hry (viz sekce 4.5.2), nebo některé z podobrazovek hlavního menu. Dále má hráč v hlavním menu samozřejmě i možnost ukončit celou hru.

První podobrazovkou hlavního menu je obrazovka s krátkým manuálem obsahující informace o mechanikách a prvcích hry. (obrázek 4.5). Tato obrazovka se skládá z několika úseků textu, mezi nimiž může hráč přepínat tlačítky ve spodní části obrazovky. Jelikož vyvíjený prototyp hry nebude obsahovat žádný tutoriál ani wiki, je vhodné hráči předat informace o fungování hry alespoň jednoduchou textovou formou. Implementována nebude ani možnost změnit nastavení ovládání. O použitelných klávesách bude hráč informován v rámci této obrazovky.

Druhou podobrazovkou hlavního menu je obrazovka s výběrem ukázek chování agentů (obrázek 4.6). Obrazovka sestává ze tří stránek, mezi kterými lze přepínat tlačítky na okraji obrazovky. Každá stránka se vztahuje k jednomu modelu chování nepřátelských agentů a obsahuje několik ukázek chování daného modelu agentů s různými nastaveními.

Třetí podobrazovkou hlavního menu je obrazovka s obecnými informacemi o hře (obrázek 4.7), jako je jméno autora, či informace, že byla vyvíjena v rámci bakalářské práce. Budou zde také zmíněni autoři volně použitelných materiálů využitých ve hře, pokud hra bude nějaké obsahovat.

Čtvrtou podobrazovkou hlavního menu je obrazovka pro potvrzení obnovy dosaženého postupu (obrázek 4.8). Pokud se hráč rozhodne, že by rád vymazal veškerý dosažený postup, aby mohl hrát znovu od začátku, poslouží mu právě tato obrazovka. Vymazání postupu by bylo možné zprostředkovat i tlačítkem přímo v hlavním menu bez potvrzovací obrazovky, jelikož se však jedná o nevratnou operaci, která může potenciálně smazat mnoho hodin postupu, počítá návrh s požadavkem potvrzení tohoto mazání.

4.5.2 Uživatelské rozhraní v průběhu hry

Uživatelské rozhraní zobrazené během hry (obrázek 4.9) ukazuje informace o příjmu a současném počtu všech dostupných surovin, energie a rafinovaných

krystalů v tabulce v levém horním rohu. V levém dolním rohu je pak osm tlačítek reprezentujících budovy, které může hráč stavět.

Při zvolení jednoho z tlačítek s dostupnými budovami se v pravém dolním rohu zobrazí panel s informacemi o zvolené budově (obrázek 4.10). Tento panel obsahuje počet surovin potřebný k výstavbě, spotřebu a produkci vybrané budovy a krátký popis toho, jak funguje.

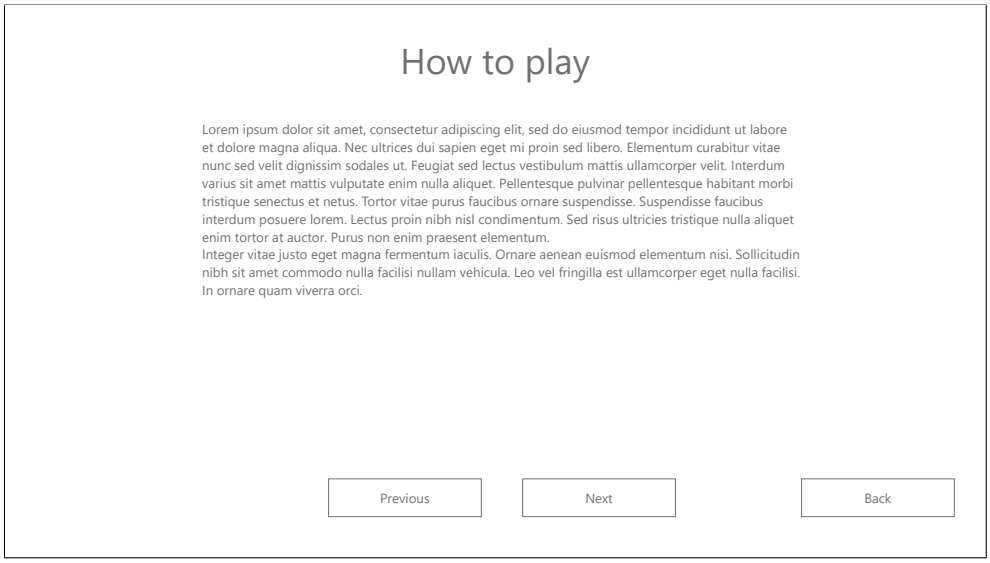
Když hráč klikne na již postavenou budovu, zobrazí se ve spodní části obrazovky panel s informacemi o této budově (obrázek 4.11). Panel vyobrazuje produkci, spotřebu, životy a popis dané budovy. Dále jsou pak součástí panelu tlačítka pro zdemolování a aktivaci či deaktivaci dané budovy. Panel může obsahovat i další tlačítka v závislosti na typu dané budovy.

Hru je možné kdykoli pozastavit. Když je hra pozastavena, objeví se menu s nabídkou několika akcí (obrázek 4.12). Toto menu nabízí hráči vrátit se zpět do hry, zobrazit obrazovku s informacemi o hře, odejít do hlavního menu, nebo kompletně hru ukončit.

4.5.3 Obrazovka s vylepšeními

Poté, co se hráč rozhodne se základovým modulem odletět zpět na loď na orbitě a odnese si získané krystaly, zobrazí se obrazovka s vylepšeními (obrázek 4.13). Ve vrchní části této obrazovky se nachází tlačítka reprezentující jednotlivé typy budov, přičemž po vybrání budovy se zobrazí možná vylepšení, která může hráč pro danou budovu zakoupit. Dále pak obrazovka s vylepšeními obsahuje ukazatele s množstvími získaných krystalů. Z této obrazovky se hráč může vrátit zpět do hlavního menu, nebo začít další běh.

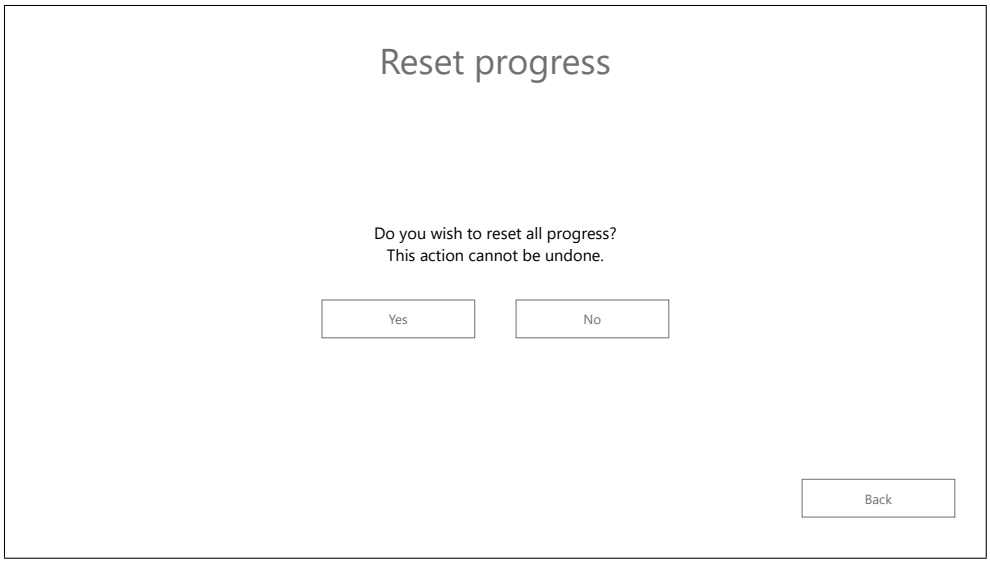
Obrázek 4.4: Hlavní menu



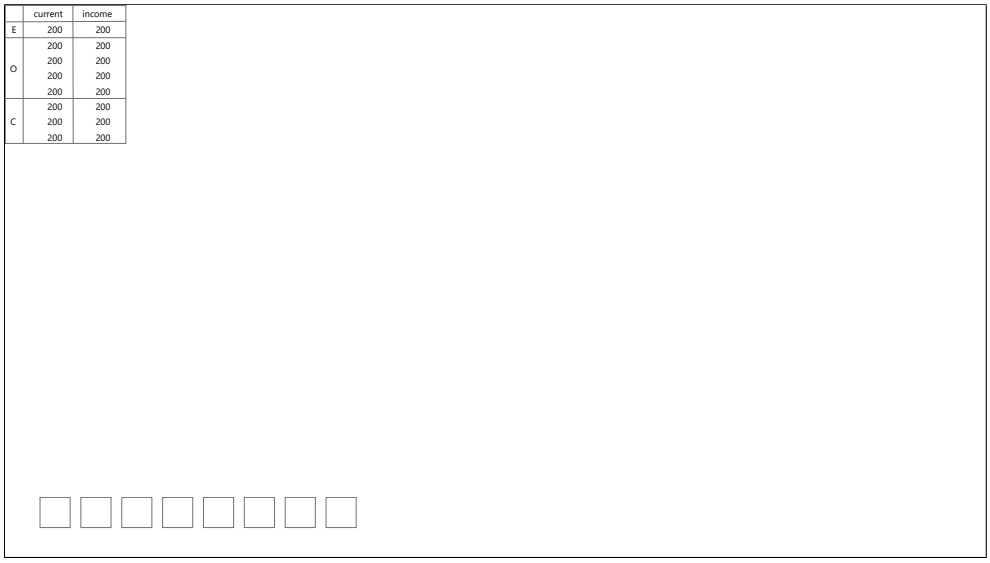
Obrázek 4.5: Obrazovka s krátkým herním manuálem

Obrázek 4.6: Výběr ukázek chování agentů

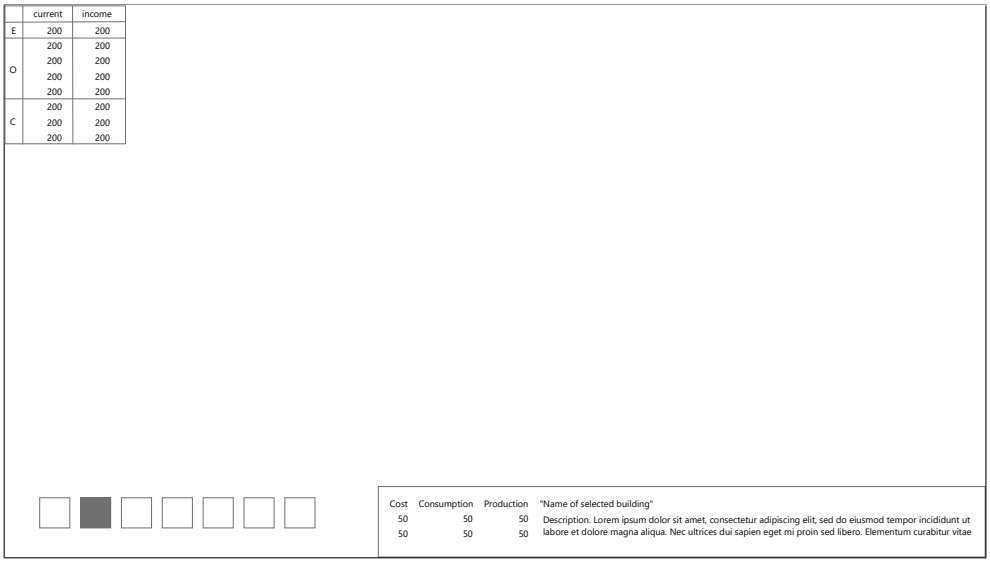
Obrázek 4.7: Titulky



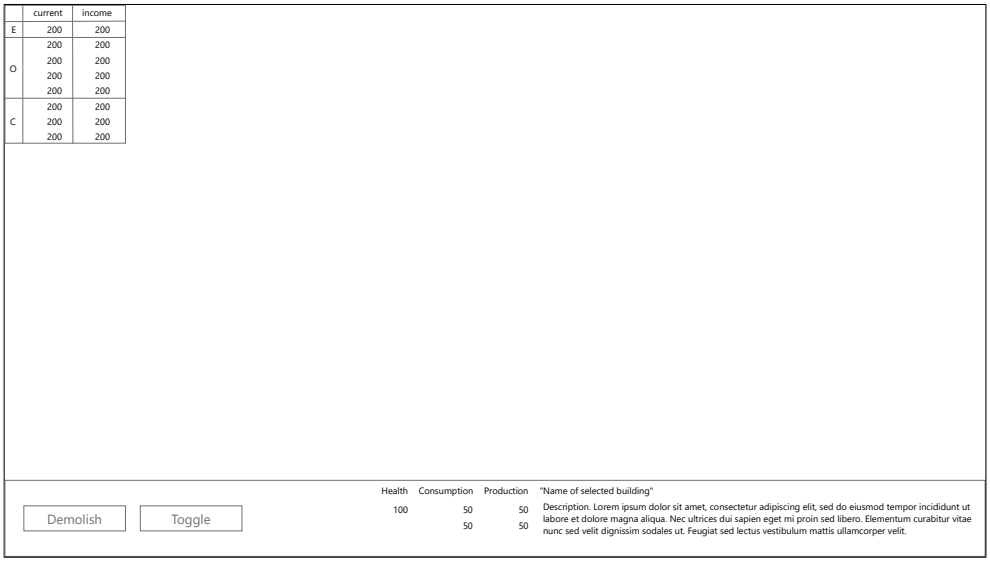
Obrázek 4.8: Smazání postupu



Obrázek 4.9: uživatelské rozhraní ve hře

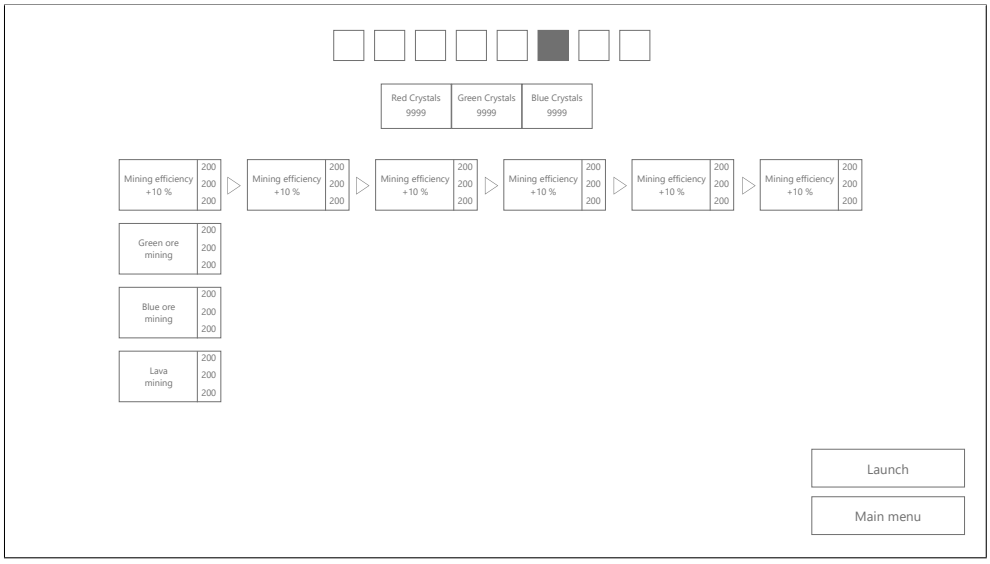


Obrázek 4.10: Informace o vybrané budově



Obrázek 4.11: Informace o postavené budově

Obrázek 4.12: Pauza



Obrázek 4.13: Obrazovka s vylepšeními

Kapitola 5

Realizace

V této kapitole jsou popsány všechny prvky, jenž byly implementovány v praktické části práce. Nejprve jsou zmíněny součásti vzniklého uživatelského rozhraní, dále pak ostatní prvky hry a nakonec samotní agenti. Veškeré grafické prvky ve hře byly vytvořeny autorem práce. Výjimkou je pouze využitý font *Thaleah Fat* [40]. Celý prototyp byl vytvořen v anglickém jazyce. Byly splněny všechny požadavky popsané v kapitole 3.3 s prioritou vyšší než *Won't have*. Spustitelný prototyp byl pro snazší přístup umístěn taktéž do veřejného GitHub repozitáře.

5.1 Uživatelské rozhraní

Tato podkapitola popisuje obrazovky a další prvky uživatelského rozhraní implementované ve vytvořeném prototypu hry. Popis uživatelského rozhraní je rozdělen do dvou sekcí. První sekce se věnuje vzhledu hlavního menu a dalších obrazovek, kdy je uživatelské rozhraní středem zájmu. Druhá sekce se pak věnuje prvkům uživatelského rozhraní v průběhu samotného hraní, kdy je středem zájmu herní mapa a prvky uživatelského rozhraní po okrajích obrazovky tvoří především doplňující informační roli.

5.1.1 Hlavní menu a další obrazovky

Všechny obrazovky využívají kombinace tmavého pozadí a světlého textu. Díky tomu působí uživatelské rozhraní velmi temně, což koreluje se zcela temnou herní mapu, na které se hra odehrává. Tím je zároveň udržován tón celé hry v jednotných barvách. Na obrázku 5.1 můžete vidět ukázku vzhledu hlavního menu hry. Jedinou obrazovkou s výrazným pozadím je obrazovka s vylepšeními. Pozadí této obrazovky má evokovat interiér vesmírné lodi, do které se hráč přesunul po odletu základovým modulem z povrchu bludné planety.

Obrázek 5.1: Ukázka vzhledu hlavního menu

Součástí implementovaného prototypu jsou následující obrazovky:

- Nabídka hlavního menu
- Krátký herní manuál
- Výběr z ukázek chování agentů
- Titulky
- Smazání postupu ve hře
- Nabídka vylepšení k zakoupení

Byly implementovány všechny obrazovky, které byly navrženy v podkapitole 4.5. Vzhled obrazovek převážně odpovídá korespondujícím wireframům. Drobné změny se dočkala obrazovka s krátkým herním manuálem, jejíž spodní tlačítka pro posun mezi stránkami textu kupředu a zpět byla nahrazena řadou tlačítek na pravé straně obrazovky, přičemž každé tlačítko se vztahuje k právě jedné stránce s textem. Textové stránky na této obrazovce byly taktéž doplněny obrázky pro vyšší srozumitelnost.

5.1.2 Součásti uživatelského rozhraní v průběhu hry

V rámci vytvořeného prototypu hry byly implementovány následující prvky uživatelského rozhraní aktivní během samotného hraní:

- Tabulka surovin

Obrázek 5.2: Ukázka vzhledu uživatelského rozhraní ve hře

- Tlačítka s budovami
- Panel s informacemi o budově k postavení
- Panel s informacemi o již postavené budově
- Textové pole pro upozornění hry
- Panel s nastavením rychlosti hry
- Panel pro aktivaci globálního světla a vizualizace komunikace agentů

Byly implementovány všechny prvky uživatelského rozhraní, jenž byly navrženy v sekci 4.5.2. Dále pak byl oproti původnímu návrhu přidán panel pro aktivaci globálního světla a vizualizace komunikace agentů, který je obsažen pouze v rozhraní ukázek chování agentů, a také panel s nastavením rychlosti hry. Kromě prvního ze zmíněných panelů a větší nabídky rychlostí času je rozhraní ukázek shodné s rozhraním v běžné hře. Pro zlepšení zpětné vazby na hráčovy akce a zvýraznění událostí ve hře byl přidán systém na zobrazování varovných zpráv v pravém horním rohu obrazovky. Varovné zprávy mohou mít různé barvy v závislosti na kontextu samotné zprávy.

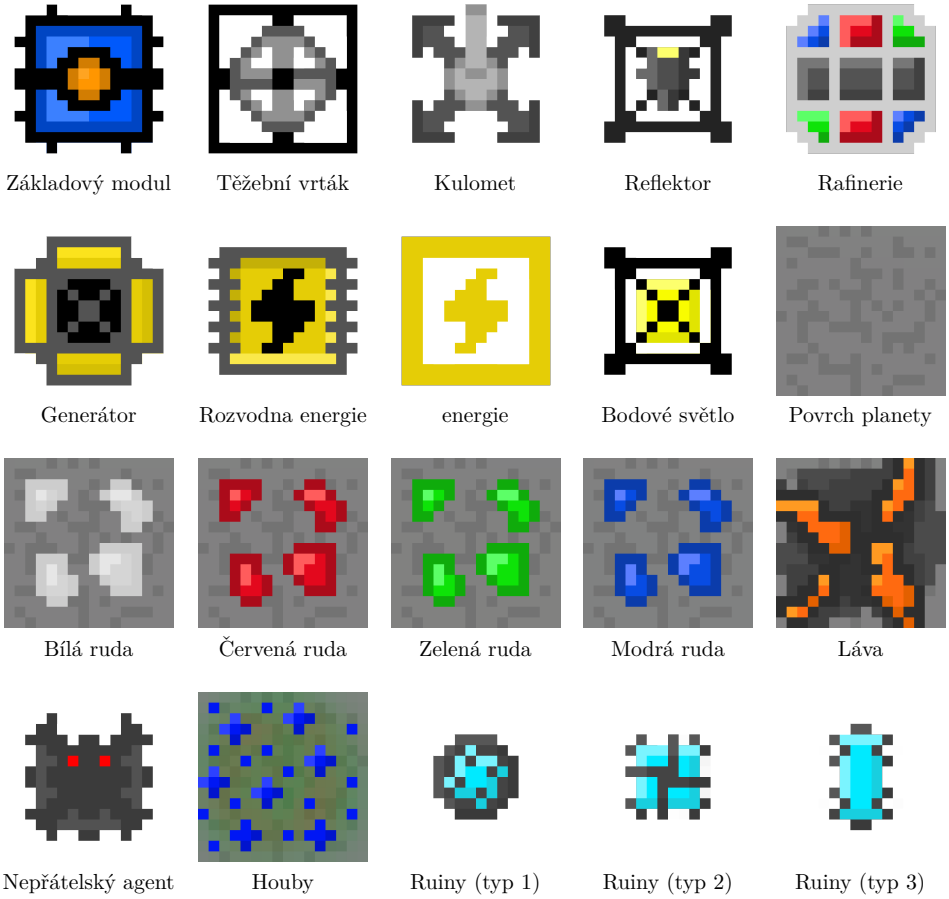
Podobně jako hlavní menu a jeho obrazovky bylo i uživatelské rozhraní v průběhu hry laděno do tmavých barev a grafického stylu pixel art. Informace o produkci a spotřebě surovin v informačních panelech se aktualizují v závislosti na získaných vylepšeních a v případě panelu s informacemi o již postavené budově i na současném stavu a nastavení dané budovy. Texty popisující množství určité suroviny mají vždy barvu dané suroviny, přičemž

všechny barevné texty si svojí barvu nastavují při načtení scény dle společné třídy s nastavením. Díky tomu lze snadno upravovat barvu všech textů popisujících konkrétní surovinu. Ukázku vzhledu uživatelského rozhraní naleznete na obrázku 5.2.

5.2 Prvky a mechaniky hry

Vytvořený prototyp obsahuje všechny podstatné mechaniky popisované v podkapitole 4.1. Za zmínku stojí především implementace mechaniky odhalování nepřátel použitím světel, kdy hráčovy kulomety pálí pouze po osvětlených nepřátelských agentech. Další významnou implementovanou mechanikou je možnost nakupování trvalých vylepšení, která mění vlastnosti příslušných budov.

Tabulka 5.1: Vzhled prvků hry



Vzhled všech podstatných prvků hry je vyobrazen v tabulce 5.1. Implementovány byly následující prvky:

- 4 druhy surovin navržené v sekci 4.2.1
- 3 druhy dalších prvků mapy navržené v sekci 4.2.2
- 8 druhů budov navržených v podkapitole 4.4

Prvky na mapě
Fungování surovin a lávy je v souladu s návrhem. V případě hub a ruin byla implementována pouze jednodušší verze, kdy tyto prvky fungují pouze jako zdroje světla rozličných tvarů bez spotřeby energie. Rozložení prvků mapy v implementovaném prototypu odpovídá rozložení popsanému v podkapitole 4.2. Suroviny se na mapě vyskytují v oblasti tvaru čtverce okolo základového modulu o velikosti přibližně 200 na 200 jednotek. Samotná herní mapa nemá žádné pevné hranice, a hráč je tedy omezen především množstvím surovin. V případě ukázek chování agentů je využito jednodušší rozložení surovin oproti základnímu hernímu módu.

Budovy
Při stavbě libovolné budovy je kontrolováno, zda hráč vlastní potřebné suroviny a budova je stavěna na vhodném místě. Pokud není splněna některá z podmínek pro stavbu budovy, k postavení vybrané budovy nedojde a vypíše se příslušná varovná zpráva. Každá aktivní budova pravidelně spotřebovává určené množství surovin či energie, a v případě, že hráč nevlastní dostatek surovin, budova se deaktivuje a vypíše se varovná hláška. Hráč může budovy deaktivovat také manuálně za účelem snížení spotřeby a deaktivované budovy opět manuálně aktivovat. Deaktivované budovy nevykazují žádnou produkci, spotřebu, ani jinou činnost. V případě potřeby může hráč postavené budovy bourat. Každá budova má určitý počet životů, přičemž hráč si aktuální stav životů budovy může zobrazit prostřednictvím informačního panelu. Když je libovolná hráčova budova zničena nepřítelem, je hráč na tuto skutečnost upozorněn textovým varováním.

5.3 Nepřátelští agenti

Ve vytvořeném prototypu budovatelské hry byly implementovány tři modely chování nepřátelských agentů dle návrhu popsaného v podkapitole 4.3. V této podkapitole jsou popsány nejprve společné vlastnosti všech agentů a v jednotlivých sekcích pak fungování každého implementovaného modelu.

Pohyb všech nepřátelských agentů je realizován skrze pathfinding, což je zprostředkováno balíčkem *A* Pathfinding Project* [21]. Samotné rozhodování o cíli cesty, předávání informací mezi agenty a veškeré další chování agentů je

již zprostředkováno kódem napsaným autorem této bakalářské práce v rámci praktické části práce. V rámci prototypu je pohyb agentů limitován na omezenou vzdálenost od středu mapy, aby bylo zamezeno situacím, kdy agenti při náhodném prohledávání opustí dění hry, nebo se ocitnou na okraji grafu pro pathfinding. Maximální vzdálenost agenta od středu mapy určuje konstanta `maxDistance`. Aktivnímu zapojení agentů do hry je taktéž napomoženo konstantou `centering`, která určuje posun náhodně generované destinace směrem ke středu mapy. Vždy, když agent dorazí do cíle během náhodného procházení a rozhodne se vygenerovat nový cíl, kterým se stává náhodně vybraný bod v okolí agenta, tento bod je před prohlášením za cíl posunut o vektor délky určené zmíněnou konstantou směrem ke středu mapy. Obě zmíněné konstanty jsou serializovány a je tyto konstanty tedy možné nastavit přímo v editoru bez zásahu do kódu.

Na srážku s jiným agentem agent reaguje tak, že si zvolí náhodný bod v blízkém okolí za dočasný cíl cesty, čímž udělá úkrok stranou, a poté pokračuje v původním úkolu.

Prototyp obsahuje několik ukázek každého druhu agentů s různými hodnotami konstant ovlivňujících jejich chování. Hodnoty zmíněných konstant jsou společně s krátkým popisem výsledného chování uvedeny na stránce uživatelského rozhraní pro výběr ukázek chování. Jednotlivým ukázkám byla pro přehlednost přiřazena krátká jména vystihující jejich chování.

5.3.1 Kooperující agenti

Chování kooperujících agentů bylo implementováno dle návrhu v sekci 4.3.1. Tato sekce se zaměřuje na hodnoty, které byly v rámci ukázek chování tohoto druhu agentů modifikovány, a dále pak na jednotlivé ukázky chování.

Konstanta `communicationDistance` určuje vzdálenost, na kterou dokáží agenti vzájemně komunikovat.

Konstanta `wanderingFriendRequirement` určuje minimální počet agentů (včetně agenta samotného), které vyžaduje agent ve svém komunikačním dosahu, aby se po zásahu od obranné budovy ve stavu toulání rozhodl prohledat lokaci, ze které byl střelen, a zároveň k tomuto kroku vyzvat i okolní agenty.

Konstanta `searchingFriendRequirement` určuje minimální počet agentů (včetně agenta samotného), které vyžaduje agent ve svém komunikačním dosahu, aby po zásahu od obranné budovy ve stavu prohledávání pokračoval v prohledávání a nevydal se na útěk.

Zjištění nedostatku agentů v okolí pro útok je v ukázkách reprezentováno bílým kruhem o velikosti prohledávané oblasti. Pokud je agentů pro útok naopak dostatek, je tato skutečnost zvýrazněna červeným kruhem o velikosti prohledávané oblasti. Přijetí zprávy k útoku je reprezentováno malým červeným kruhem.

Loose packs
Toto nastavení představuje základní chování nepřátelských agentů využité v základním herním módu. Agenti mají běžnou vzdálenost pro komunikaci i požadavky na počty okolních agentů. Na počátku ukázky je hustota agentů ve scéně velmi malá, a tedy nejprve převážně ustupují. Postupem času hustota agentů v okolí hráčovy základny roste, čímž roste i šance, že se agenti v reakci na zásah vydají do útoku.

Tight packs
V tomto nastavení mají agenti nižší komunikační vzdálenost a vyšší nároky na počty okolních agentů něž v chování předchozím. Proto déle trvá, než hustota agentů dosáhne požadované výše, ale poté agenti útočí v početnějších a kompaktnějších smečkách než v předchozí ukázce.

Highly aggressive
V této ukázce mají agenti vysoký dosah komunikace a nulové nároky na počty okolních agentů. Výsledkem je chování, kdy agenti vždy útočí a zároveň přilákají i agenty v širším okolí.

Lone wolves
V této ukázce je simulována situace, kdy agenti prakticky nejsou schopni vzájemné komunikace. Toho je docíleno nastavením minimální komunikační vzdálenosti v kombinaci s nulovými nároky na počty okolních agentů.

5.3.2 Mravenčí agenti

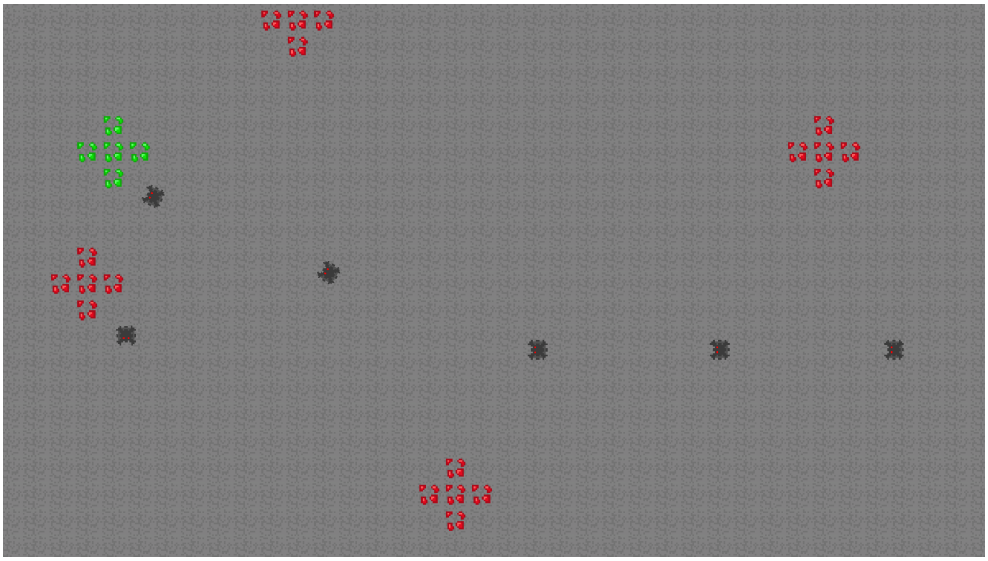
Chování mravenčích agentů bylo implementováno dle návrhu v sekci 4.3.2. Tato sekce se zaměřuje na hodnoty, které byly v rámci ukázek chování tohoto druhu agentů modifikovány, a dále pak na jednotlivé ukázky chování.

Konstanta `communicationDistance` určuje vzdálenost, na kterou dokáží agenti vzájemně komunikovat, přičemž mravenčí agenti komunikují pravidelně.

Konstanta `communicationDelay` určuje dobu, po které proběhne další komunikace agenta. Agent tedy předává informace okolním agentům pravidelně každých `communicationDelay` sekund.

Konstanta `randomWanderChance` určuje pravděpodobnost, že se mravenčí agent místo přechodu do stavu prohledávání rozhodne provést jeden krok toulání.

Objevení suroviny, o které agent do té doby neměl žádné informace, je v ukázkách reprezentováno velkým kruhem s barvou dané suroviny. Obdržení informace o surovině, o které agent do té doby neměl žádné informace, je reprezentováno malým kruhem příslušné barvy. Totéž platí i pro informace o nalezených budovách, které jsou reprezentovány bílou barvou. V případě, že agent dorazí na lokaci budovy a zjistí, že na daném místě se již žádná budova



Obrázek 5.3: Ukázka chování mravenčích agentů

nenachází, pošle tuto informaci okolním agentům, kteří si ji dále předávají a tato komunikace je reprezentována malými kruhy s černou barvou.

Orderly
V této ukázce mají agenti vysokou komunikační vzdálenost a nulovou šanci na náhodné prohledávání. Výsledkem je, že poté, co časem agenti naleznou nějakou cestu z hnízda skrz suroviny do báze hráče, je tato informace díky vysoké komunikační vzdálenosti předávána novým agentům hned po jejich narození. Nově narození agenti tedy znají všechny čtyři body jejich budoucí cesty, a těchto bodů se drží, což způsobuje na pohled spořádané chování, které si můžete prohlédnout na obrázku 5.3.

Explorers
V této ukázce mají agenti stejnou komunikační vzdálenost, jako v ukázce předchozí, avšak mají vysokou šanci na náhodné procházení před prohledáním známé lokace cíle. Díky tomu agenti hojně prozkoumávají mapu, a na základnu hráče útočí nárazověji a z vícero směrů, než v předchozí ukázce.

Forgetful
V této ukázce mají agenti stejnou šanci na náhodné prohledávání, jako v první ukázce, avšak jejich komunikační dosah je výrazně nižší. To způsobuje, že agenti často bloudí, dokud jejich hustota na mapě není dostatečně vysoká pro efektivní přenos informací. Při vysoké hustotě agentů se informace rychle šíří, agenti naleznou své cíle a vydají se posbírat suroviny a zaútočit na základnu

hráče. Útok na základnu hráče obvykle končí smrtí všech zúčastněných agentů. Vzhledem k nízké komunikační vzdálenosti tak může dojít k tomu, že agenti s informacemi vymřou, a nově zrození agenti musí lokace surovin opět hledat.

Inverted
Agenti v této ukázce mají stejnou komunikační vzdálenost, jako v prvních dvou ukázkách. Šance agentů na náhodné toulání je nízká. Hlavním specifikem této ukázky je posunutí mravenišť blíže k hráčově základně.

5.3.3 Včelí agenti

Chování včelích agentů bylo implementováno dle návrhu v sekci 4.3.3. Tato sekce se zaměřuje na hodnoty, které byly v rámci ukázek chování tohoto druhu agentů modifikovány, a dále pak na jednotlivé ukázky chování.

Konstanta `randomWanderChance` udává, s jakou pravděpodobností se včelí agent místo přechodu do stavu vracení se rozhodne provést jeden krok toulání.

Konstanta `receiveOrderModifier` upravuje šanci, že se agent rozhodne přijmout informaci o lokaci k prohledání od jiného agenta a vydá se na danou lokaci prohledat. V případě, že agent obdrží informaci, jejíž hodnota je vyšší než maximum hodnot všech informací, se kterými se agent za svůj život setkal, automaticky zprávu přijme. V opačném případě se agent rozhoduje dle následující podmínky:

```
(UnityEngine.Random.Range(0f,1f) < (infoValue/best) * receiveOrderModifier)
```

Výpočet hodnoty zprávy byl proveden již při nalezení cíle agentem, který zprávu odnesl do hnízda, aby ji předal ostatním. Výpočet provádí následující funkce:

```
private float calculateValue(TargetType targetType, Vector3 targetPosition)
{
   return
   (
      (1f - (Vector3.Distance(hivePosition, targetPosition)/orderMaxDistance))
      * targetValues[(int)targetType]
   );
}
```

Konstanta `maxrestingTime` určuje v sekundách maximální dobu, jakou může agent setrvat ve stavu odpočívání. Při překročení maximální doby se agent vydává toulat po mapě.

Konstanta `minrestingTime` určuje v sekundách minimální dobu, jakou může agent setrvat ve stavu odpočívání. Před uplynutím této doby agent reaguje pouze na zprávy o nalezených budovách hráče a všechny ostatní zprávy zcela ignoruje.

Předání informací o nalezené surovině je v ukázkách zobrazeno kruhem příslušné barvy o velikosti odpovídající komunikačnímu dosahu agenta. Rozhodnutí o přijmutí zprávy je zvýrazněno malým kruhem příslušné barvy. Aby hromadné útočení včelích agentů na základnu hráče působilo více jako roj včel, kontrolují včelí agenti během útoku prostor před sebou a za sebou a mírně upravují svoji rychlost pro lepší soudržnost s okolními agenty. Na začátku ukázek chování včelích agentů se obvykle nějakou dobu nic neděje, což je způsobeno tím, že agenti se rodí ve stavu odpočívání.

Balanced
Tato ukázka reprezentuje vyvážené chovaní nejvíce odpovídající návrhu. Maximální doba odpočinku je půl minuty a minimální doba odpočinku je deset vteřin. Hodnoty konstant upravujících náhodné toulání a šanci na přijetí informace jsou střední.

Hunters
V této ukázce mají agenti minimální šanci na přijímání zpráv či náhodné toulání. Zároveň mají dlouhou minimální dobu odpočinku. Kombinace těchto faktorů způsobuje, že zpráva o nalezení hráčových budov má velkou šanci nalákat větší skupinu agentů.

Gatherers
V této ukázce mají agenti poměrně vysoké šance na přijímání zpráv a náhodné toulání. Dále pak nemají žádnou minimální dobu odpočinku. Výsledkem je chování agentů, kdy agenti setrvávají v hnízdech pouze krátkou dobu a většinu času tráví mimo něj a to buď nošením surovin, nebo prohledáváním. Agenti nejprve hojně prohledávají prostor, následně začnou nosit suroviny z několika zdrojů zároveň a časem konvergují k nošení z pouze jednoho až dvou nejlepších zdrojů.

Kapitola 6

Testování

V této kapitole jsou popsány provedené testy vyvíjeného prototypu. Za zmínku stojí zejména provedené testování použitelnosti.

6.1 Playtesting

Všechny vyvinuté části vytvořeného prototypu hry byly průběžně testovány vývojářem, přičemž předmětem testování byla především správná funkčnost. V pokročilejší fázi vývoje pak byl prototyp několikrát podroben i krátkému uživatelskému testování, při kterém byly odhaleny některé obtížně nalezitelné chyby. Toto průběžné testování se zároveň stalo užitečným zdrojem zpětné vazby.

Jediným známým nedostatkem, který byl v rámci playtestingu odhalen a nebyl v prototypu řešen, je schopnost dvou a více rozvoden předávat si energii navzájem i po odpojení od základny hráče. V rámci prototypu je význam tohoto nedostatku marginální, avšak v případné plné verzi hry by ho bylo vhodné buď opravit, nebo jeho existenci prohlásit za úmyslně ponechanou mechaniku.

6.2 Testování použitelnosti

Po dokončení implementační části byl vzniklý prototyp budovatelské strategie podroben kvalitativnímu testování použitelnosti. Vzhledem k epidemiologické situaci nebyla využita fakultní laboratoř pro testování uživatelského rozhraní. Testování použitelnosti se zúčastnili tři uživatelé s různorodým vztahem k počítačovým hrám. Testování s jedním uživatelem proběhlo vzdáleně, s dalšími dvěma uživateli proběhlo testování v domácím prostředí. Během testování uživatelé spouštěli hru vždy na vlastních počítačích, čímž bylo docíleno vyššího komfortu uživatelů a zároveň byla ozkoušena bezproblémová funkčnost prototypu na vícero zařízeních.

Testování proběhlo dle testovacího scénáře v příloze B. Poznatky získané během testování jsou v této podkapitole rozděleny dle části hry, ke které se vztahují. Každý poznatek je doplněn návrhem na možné zlepšení, které z něj vyplývá. Jelikož byl testován prototyp počítačové hry, jsou zúčastnění uživatelé v rámci vyhodnocení výsledků testování označováni za hráče. Ve vyvíjeném prototypu byla implementována pouze část návrhů na zlepšení, přičemž ostatní návrhy by měly být zohledněny v případě budoucího vývoje plnohodnotné hry.

6.2.1 Hlavní menu

V úvodní obrazovce a podobrazovkách pro smazání postupu a zobrazení titulků se uživatelé orientovali bez potíží a bez problémů chápali významy a fungování všech tlačítek. Při testování podobrazovek s informacemi o hře a výběrem ukázek agentů se projevily drobné nedostatky obrazovek spojené především s vysokým obsahem textu. Všechny podstatné nalezené nedostatky hlavního menu a jeho součástí jsou popsány v tabulce 6.1.

6.2.2 Uživatelské rozhraní ve hře

Význam jednotlivých prvků uživatelského rozhraní ve hře nedělal hráčům větší potíže. Problémy hráčům spíše než neporozumění uživatelskému rozhraní způsobovalo především nedostatečné chápání funkce jednotlivých budov a surovin. Toto počáteční zmatení lze vzhledem k absenci tutoriálu považovat za očekávatelné a všem hráčům se podařilo prvkům hry a jejich fungování časem porozumět. Problémy především méně zkušeným hráčům způsobovala také obtížnost hry, která je však vyšší záměrně, aby byl hráč více motivován k využívání mechaniky nákupu trvalých vylepšení.

Jednotlivé nedostatky nalezené při testování hlavního herního módu a jeho uživatelského rozhraní jsou popsány v tabulce 6.2. Součástí této fáze testování bylo též otestování obrazovky pozastavené hry a obrazovky pro nákup vylepšení.

6.2.3 Uživatelské rozhraní v ukázkách

Jelikož je uživatelské rozhraní v ukázkách chování agentů téměř shodné s rozhraním ve hře, nedělalo hráčům žádné problémy se v něm po předešlém testování orientovat. Jediným objeveným nedostatkem bylo, že ukázky chování agentů začínají s kamerou zaměřenou na základnu hráče jako v běžné hře, avšak chování agentů se zpočátku odehrává především na okrajích mapy. Tento nedostatek byl vyřešen úpravou počátečního nastavení hlavní kamery v ukázkových scénách.

Tabulka 6.1: Nedostatky nalezené pří testování hlavního menu

Poznatek č. 1
Hráčům se často nechce číst obsah obrazovky s manuálem a vynechané informace později postrádají během hry.
Řešení
Tento problém byl již částečně adresován umožněním přístupu k manuálu hry během pozastavení hry bez návratu do hlavního menu. Optimálním řešením by pak byla implementace tutoriálu, který hráčům potřebné informace předá více interaktivním způsobem.
Poznatek č. 2
Většině hráčů se nechtějí číst popisy chování ukázek agentů. Ukázku chování pak vybírají víceméně náhodně, přičemž preferují především chování na prvním místě v seznamu.
Řešení
Ukázkám chování agentů byla přidána krátká jména stručně vystihující dané chování. Zajímavější ukázky chování byly ve výběru přesunuty nahoru.
Poznatek č. 3
Hráč s větší zkušeností s hrami se snažil využít pro návrat z podobrazovky do hlavní nabídky klávesu Escape.
Řešení
Problém lze vyřešit implementací příslušné funkce dané klávesy.
Poznatek č. 4
Méně zkušeným hráčům nebyl na první pohled jasný význam zkratek LMB a RMB.
Řešení
Problém byl vyřešen nahrazením zmíněných zkratek za jejich plný název.

Tabulka 6.2: Nedostatky nalezené pří testování hlavního herního módu

Poznatek č. 1
Všem hráčům dělalo do určité míry problémy rozeznat jednoznačně každou budovu pouze na základě obrázku dané budovy na tlačítku pro výběr budovy.
Řešení
Tento problém byl v prototypu hry adresován přidáním názvů budov k příslušným tlačítkům. Dále by pak mohlo pomoci seznámení hráče s jednotlivými budovami v rámci případného tutoriálu.
Poznatek č. 2
Ukázalo se, že je pro hráče v některých případech obtížné rozpoznat, zda je budova aktivovaná.
Řešení
Tento problém byl vyřešen přidáním ikony, která upozorňuje na neaktivní stav budovy.
Poznatek č. 3
Hráči často ignorují varovná oznámení.
Řešení
Důležitá varovná oznámení mohou být doplněna zvukovým efektem, čímž by hráč informaci obdržel i bez čtení oznámení.
Poznatek č. 4
Hráči si někdy nevšimnou, že je některá z jejich budov pod útokem.
Řešení
Hráč může být na útok upozorněn textovým či zvukovým varováním. Budova pod útokem může být též graficky zvýrazněna pro snazší orientaci. V současné době je hráč upozorněn pouze na zničení budovy.
Poznatek č. 5
Hráči si někdy nevšimnou, že jejich produkce některé suroviny či energie klesla do záporných hodnot.
Řešení
Problém lze vyřešit přidáním zvukového upozornění, nebo intenzivnějším zvýrazněním záporné produkce oproti stávajícímu.
Poznatek č. 6
Jeden z hráčů si prvně neuvědomil, že zakoupené vylepšení se vztahuje pouze k jedné budově.
Řešení
Problém lze řešit zvýrazněním jména vybrané budovy, nebo úpravou jmen vylepšení.

Závěr

Hlavním cílem této práce bylo navrhnout prototyp 2D budovatelské hry obsahující multiagentní systém, jehož agenti budou soupeřit s hráčem, a tento prototyp implementovat s využitím herního enginu Unity.

Byla provedena rešerše problematiky multiagentních systémů a umělé inteligence v počítačových hrách. Součástí rešeršní části byl taktéž přehled nástrojů a prvků Unity použitelných pro účely této práce. Následovala analytická část, která se věnovala žánru budovatelských her včetně příkladů her tohoto žánru a také funkčním a nefunkčním požadavkům na vyvíjený prototyp. Dále pak byla v této části zvážena použitelnost konceptů a nástrojů z rešeršní části.

V návrhové části byla popsána základní myšlenka hry a důležité mechaniky. Byla navržena podoba herní mapy včetně prvků, které se na ní mohou vyskytovat, a jejich role z pohledu hratelnosti. Také vznikl kompletní návrh uživatelského rozhraní včetně wireframů všech obrazovek prototypu. Rovněž byly popsány všechny budovy, které může hráč stavět, a jejich role ve hře. Důležitou součástí návrhu je pak popis možného chování nepřátelských agentů. Realizační část sestává z popisu všech prvků hry implementovaných v Unity. Vytvořený prototyp byl podroben testování, jehož výsledky jsou popsány v poslední části práce.

Výstupem praktické části práce je samotná implementace prototypu 2D budovatelské hry. Implementovaný prototyp hry obsahuje všechny obrazovky uživatelského rozhraní navržené v návrhové části, stejně tak i všechny navržené budovy a většinu navržených prvků mapy. Grafický vzhled uživatelského rozhraní, podoba jednotlivých prvků mapy, budov stavěných hráčem i samotných nepřátelských agentů byl ztvárněn tvůrcem této práce a to v grafickém stylu pixel art.

Důležitým výstupem práce jsou pak tři různé návrhy modelů chování agentů ve hře, které byly v praktické části implementovány. První model je zaměřen na použitelnost v budovatelské hře a odpovídá autorově představě o nejvhodnějším multiagentním systému z hlediska hratelnosti. Agenti v tomto

modelu procházejí mapou a pokud se agent setká s budovami hráče, dá se v závislosti na počtu agentů v oblasti buď na útěk, nebo se pustí do společného útoku. Druhý model je inspirován chováním mravenců. Každý agent v něm usiluje o sebrání každého druhu barevné rudy a následný útok na nejbližší budovy nepřítele. Sbírání rudy se agenti snaží optimalizovat vzájemnou komunikací, při které si předávají informace o blízkých lokalitách rudy. Třetí model se zakládá na chování včel při hledání potravy a agenti se v něm snaží optimalizovat rychlost nošení surovin do jejich hnízda. Hledající agenti se po nalezení surovin vrací do hnízda, kde se snaží přesvědčit čekající agenty k cestě pro nalezenou surovinu, přičemž jejich šance na úspěch je závislá na kvalitě nalezené suroviny.

Na tuto práci je možné navázat prací na tvorbě plnohodnotné hry založené na vzniklém prototypu. Taktéž v případě vytvořených modelů multiagentních systémů je zajisté prostor pro další rozšiřování, tvorbu přídavných modelů, nebo jejich kombinaci.

Bibliografie

1. RICHTER, Felix. Gaming: The Most Lucrative Entertainment Industry By Far. *Statista* [online]. 2020 [cit. 2021-04-22]. Dostupné z: `https://www.statista.com/chart/22392/global-revenue-of-selected-entertainment-industry-sectors/`.
2. JAKOB, Michal; ROLLO, Milan. *Introduction to Multi-Agent Systems* [online]. [N.d.] [cit. 2021-04-22]. Dostupné z: `https://cw.fel.cvut.cz/old/_media/courses/a3b33kui/prednasky/prednaska_11.pdf?cache=nocache`.
3. BALAJI, P. G.; SRINIVASAN, D. An Introduction to Multi-Agent Systems. In: *Innovations in Multi-Agent Systems and Applications - 1*. Ed. SRINIVASAN, Dipti; JAIN, Lakhmi C. Berlin, Heidelberg: Springer, 2010, s. 1–27. ISBN 978-3-642-14435-6. Dostupné z DOI: `10.1007/978-3-642-14435-6_1`.
4. VIDAL, José. *Fundamentals of Multiagent Systems: with NetLogo Examples* [online]. 2006 [cit. 2021-04-22]. Dostupné z: `https://github.com/josemvidal/FMAS/blob/master/mas2.pdf`.
5. RUSSELL, Stuart; NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. USA: Prentice Hall, 1995. ISBN 0-13-103805-2.
6. WOOLRIDGE, Michael. *Introduction to Multiagent Systems*. USA: John Wiley & Sons, Inc., 2002. ISBN 978-0-471-49691-5.
7. MOUNT, David M. *Game Programming* [online]. 2016 [cit. 2021-04-22]. Dostupné z: `https://www.cs.umd.edu/class/spring2016/cmsc425/Lects/cmsc425-spring16-lects.pdf`.
8. LIM, Chee Peng; JAIN, Lakhmi C. Advances in Swarm Intelligence. In: *Innovations in Swarm Intelligence*. Ed. LIM, Chee Peng; JAIN, Lakhmi C.; DEHURI, Satchidananda. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 1–7. ISBN 978-3-642-04225-6. Dostupné z DOI: `10.1007/978-3-642-04225-6_1`.

9. DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. 1996, roč. 26, č. 1, s. 29–41. Dostupné z DOI: 10.1109/3477.484436.

10. GAMBARDELLA, L.M.; DORIGO, M. Solving symmetric and asymmetric TSPs by ant colonies. In: *Proceedings of IEEE International Conference on Evolutionary Computation*. 1996, s. 622–627. Dostupné z DOI: 10.1109/ICEC.1996.542672.

11. STÜTZLE, Thomas; HOOS, Holger H. MAX–MIN Ant System. *Future Generation Computer Systems*. 2000, roč. 16, č. 8, s. 889–914. ISSN 0167-739X. Dostupné z DOI: https://doi.org/10.1016/S0167-739X(00)00043-1.

12. KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995, sv. 4, 1942–1948 vol.4. Dostupné z DOI: 10.1109/ICNN.1995.488968.

13. KARABOGA, Dervis. *An Idea Based on Honey Bee Swarm For Numerical Optimization*. 2005. Tech. zpr.

14. SIZER, Ben. *The Total Beginner's Guide to Game AI* [online]. 2018 [cit. 2021-04-22]. Dostupné z: https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/.

15. MILLINGTON, Ian. *Artificial Intelligence for Games*. USA: Morgan Kaufmann Publishers Inc., 2006. ISBN 978-0-12-497782-2.

16. UNITY TECHNOLOGIES. *Render pipelines* [online]. 2021 [cit. 2021-04-22]. Dostupné z: https://docs.unity3d.com/Manual/render-pipelines.html.

17. LIRA, Felipe. How the Lightweight Render Pipeline is evolving. *Unity Blog* [online]. 2019 [cit. 2021-04-22]. Dostupné z: https://blogs.unity3d.com/2019/09/20/how-the-lightweight-render-pipeline-is-evolving/.

18. ZGEB, Bronson. Persistent data: How to save your game states and settings. *Unity Blog* [online]. 2021 [cit. 2021-04-22]. Dostupné z: https://blogs.unity3d.com/2021/02/23/persistent-data-how-to-save-your-game-states-and-settings/.

19. UNITY TECHNOLOGIES. *Animation State Machines* [online]. 2021 [cit. 2021-04-22]. Dostupné z: https://docs.unity3d.com/Manual/AnimationStateMachines.html.

20. UNITY TECHNOLOGIES. *Navigation and Pathfinding* [online]. 2021 [cit. 2021-04-22]. Dostupné z: https://docs.unity3d.com/Manual/Navigation.html.

21. GRANBERG, Aron. *A* Pathfinding Project* [online]. 2020. Ver. 4.2.15 [cit. 2021-04-22]. Dostupné z: `https://arongranberg.com/astar/features#`.

22. UNITY TECHNOLOGIES. *The Unity Machine Learning Agents Toolkit* [online]. 2021. Release 16 [cit. 2021-04-22]. Dostupné z: `https://github.com/Unity-Technologies/ml-agents/blob/release_10_docs/docs/ML-Agents-Overview.md#summary-and-next-steps`.

23. VALVE CORPORATION. *Statistiky služby Steam a her* [online]. 2021 [cit. 2021-04-22]. Dostupné z: `https://store.steampowered.com/stats/`.

24. COLOSSAL ORDER LTD. *Cities: Skylines* [soft.]. Paradox Interactive, 2015 [cit. 2021-04-22]. Dostupné z: `https://www.paradoxplaza.com/cities-skylines/CSCS00GSK-MASTER.html`.

25. 3DIVISION. *Workers & Resources: Soviet Republic* [soft.]. 2019 [cit. 2021-04-22]. Dostupné z: `https://www.sovietrepublic.net/`.

26. FRONTIER DEVELOPMENTS. *Planet Zoo* [soft.]. 2019 [cit. 2021-04-22]. Dostupné z: `https://www.planetzoogame.com/`.

27. FRONTIER DEVELOPMENTS. *Jurassic World Evolution* [soft.]. 2018 [cit. 2021-04-22]. Dostupné z: `https://www.jurassicworldevolution.com/en-GB`.

28. SHINING ROCK SOFTWARE LLC. *Banished* [soft.]. 2014 [cit. 2021-04-22]. Dostupné z: `http://www.shiningrocksoftware.com/`.

29. MECHANISTRY. *Timberborn* [soft.]. 2021 [cit. 2021-04-22]. Dostupné z: `https://timberborn.mechanistry.com/`.

30. 11 BIT STUDIOS. *Frostpunk* [soft.]. 2018 [cit. 2021-04-22]. Dostupné z: `https://www.frostpunkgame.com/`.

31. DOUBLE ELEVEN; INTROVERSION SOFTWARE. *Prison Architect* [soft.]. Paradox Interactive, 2015 [cit. 2021-04-22]. Dostupné z: `https://www.prisonarchitect.com/`.

32. BALANCING MONKEY GAMES. *Before We Leave* [soft.]. 2021 [cit. 2021-04-22]. Dostupné z: `https://www.balancingmonkeygames.com/`.

33. YEVHENIY. *Ostriv* [soft.]. 2020 [cit. 2021-04-22]. Dostupné z: `https://ostrivgame.com/home/`.

34. LUDEON STUDIOS. *Rimworld* [soft.]. 2018 [cit. 2021-04-22]. Dostupné z: `https://rimworldgame.com/`.

35. WUBE SOFTWARE LTD. *Factorio* [online]. 2020 [cit. 2021-04-22]. Dostupné z: `https://www.factorio.com/`.

36. NUMANTIAN GAMES. *They Are Billions* [soft.]. 2019 [cit. 2021-04-22]. Dostupné z: `http://www.numantiangames.com/theyarebillions/`.

37. DOERR, Raymond. *Rise to Ruins* [soft.]. SixtyGig Games, 2019 [cit. 2021-04-22]. Dostupné z: `https://risetoruins.com/`.

38. KNUCKLE CRACKER. *Creeper World* [soft.]. 2016 [cit. 2021-04-22]. Dostupné z: `https://knucklecracker.com/creeperworld/playcwts.php`.

39. CLEGG, Dai; BARKER, Richard. *Case Method Fast-Track: A Rad Approach*. USA: Addison-Wesley Longman Publishing Co., Inc., 1994. ISBN 020162432X.

40. HOPPMANN, Rick. *Free Pixel Font - Thaleah Fat* [online]. 2019 [cit. 2021-04-22]. Dostupné z: `https://assetstore.unity.com/packages/2d/fonts/free-pixel-font-thaleah-140059`.

Příloha **A**

Seznam použitých zkratek

RP Render Pipeline

URP Universal Render Pipeline

SRP Scriptable Render Pipeline

HDRP High Definition Render Pipeline

API Application Programming Interface

LMB Left Mouse Button

RMB Right Mouse Button

Příloha B

Scénář testování použitelnosti

1 Před testem

1.1 Seznámení s testováním použitelnosti

- Obeznámení uživatele s problematikou uživatelského testování
  - K čemu testování složí
  - Jak obvykle testování probíhá
- Poučení o právech
  - Možnost kdykoli testování přerušit
  - Využití získaných dat v rámci BP

1.2 Úvodní dotazy

- Zkušenost uživatele s počítačovými hrami obecně
- Zkušenost uživatele s budovatelskými hrami

1.3 Krátký úvod do problematiky

- Obecné seznámení s budovatelskými hrami
- Stručné seznámení s BP

Připomenutí, že předmětem testování je aplikace, nikoli uživatel

2 Během testu

1) Pročtení informací o hře
   - Upgrady
   - Ovládání
2) Spuštění hry
   - Zpět z How to play do Menu
   - Tlačítko hry
3) Zakoupení vylepšení
   - Použití světel
   - Těžba červené rudy
   - Získání červených krystalů
   - Launch
   - Výběr upgradu
   - Zakoupení
   - Návrat do hry
4) Spuštění ukázky chování agentů
   - Výběr
   - Zrychlení
5) Smazání postupu

3 Po testu

3.1 Závěrečné dotazy

- Část, která dělala největší problémy
- Co by se dalo zlepšit
- Prostor pro případné další postřehy

Poděkování za účast

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
  - `BPstrnalad.exe` ..... soubor pro spuštění implementace
- `src`
  - `RoguePlanetProject` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... adresář s textem práce
  - `thesis.pdf` ..... text práce ve formátu PDF