FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Do roboty! Samoorganizující se model starověké vesnice.
Student:	Petr Bureš
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je vytvořit vizualizaci samoorganizujícího se prostředí zvoleného typu obydleného samočinně jednajícími agenty. Jeho vývoj může být usměrňován, přičemž model reaguje na akce uživatele. Model se organizuje vzhledem k potřebám svých obyvatel - agentů (např. hlad) a omezení svého okolí (např. úrodnost polí) a to formou produkce vhodných zdrojů a úpravou své struktury (např. stavba mlýnu, pekárny, sbírání borůvek).
1) Proveďte rešerši problematiky modelů pro vytvoření a vývoj samoorganizující se (urbanistické) scény.
2) Definujte na základě požadavků zadavatele soubor pravidel (model) prostředí a chování agentů, podle kterých se bude město rozvíjet.
3) Navrhněte prototyp aplikace. Zaměřte se především na chování (AI) agentů a jeho přizpůsobivost vygenerovanému prostředí.
4) Implementujte prototyp vizualizace modelu města pomocí Godot Engine.
5) Implementovaný prototyp podrobte vzhledem k účelu aplikace vhodným testům

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Do roboty! Samoorganizující se model starověké vesnice

Petr Bureš

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

13. května 2019

Poděkování

Děkuji svému vedoucímu Ing. Radku Richtrovi, Ph.D. za ochotu, pozitivní přístup a mnoho užitečných rad. Dále bych chtěl poděkovat všem, kteří se jakýmkoli způsobem podíleli na této práci a hlavně své rodině za neustávající podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. § 46 odst. 6 tohoto zákona 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 13. května 2019 .....

Odkaz na tuto práci

Bureš, Petr. *Do roboty! Samoorganizující se model starověké vesnice*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá simulací samoorganizujícího modelu starověké vesnice a její vizualizací. Vesnice je zasazena do náhodně generovaného dynamického prostředí. Vesničané se rozhodují pro dostupné akce na základě znalosti jejich důsledků pro jednotlivce i celou vesnici. Prototyp hry umožňuje hráči usměrňování akcí vesničanů, které není vyžadováno pro průběh simulace.

Klíčová slova počítačová hra, umělá inteligence, izometrické 2D, urbanistická simulace, samoorganizace, generování mapy, rozmístění objektů, Godot Engine

Abstract

This thesis simulates a self organizing model of an ancient village and its visualization. The village is placed into a randomly generated dynamic environment. Villagers make decisions based on their knowledge of the consequences for both the individuals and the whole village. The prototype lets the player direct the actions of villagers, but it's not required for the simulation process.

Keywords computer game, artificial intelligence, isometric 2D, urbanistic simulation, self organization, map generation, object placement, Godot Engine

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Úvod

Samoorganizující se model starověké vesnice je velmi složité a různorodé téma. Existuje mnoho herních i jiných studií, které vyvíjí, nebo se jinak zajímají o hry a simulace složitějších prostředí, jako je například starověká vesnice. Příklady her, kterými je tato práce inspirována jsou Knights and Merchants, The Settlers, Banished a Rimworld.

Tato práce je zaměřena na simulaci dynamického prostředí. Model je graficky vizualizován ve stylu izometrické 2D hry a je možné v reálném čase sledovat jeho stav. Vesničané se rozhodují v závislosti na jejich potřebách, ale i stavu celého prostředí. Povolání, ze kterých si vesničané vybírají, jsou vzájemně závislé a je důležité, aby měli dostupné suroviny potřebné k jejich práci. Pro vykonávání těchto povolání vesničané staví příslušné budovy na vhodná místa.

Dále práce obsahuje náhodné generování volné přírody. Generovaná mapa volné přírody musí obsahovat dostatečné množství dosažitelných surovin pro úspěšný rozvoj vesnice a prvky volné přírody jako vodu, stromy a kameny. Přes vodu a budovy vesničané projít nemohou, a proto je potřeba vytvořit mapu těchto oblastí, kterou lze dynamicky upravovat se změnami prostředí a použít pro hledání cesty do určitého bodu.

Postupy použité při implementaci vlastností modelu popsaných výše jsou přizpůsobené dostupným funkcím a možnostem Godot Engine. Použité grafické podklady jsou s výjimkou obrázků zvířat, fontu a animací vesničanů vytvořeny autorem práce.

Přehled kapitol:

Cíl práce V této kapitole jsou zmíněna relevantní témata a krátký popis zvoleného postupu k dosažení cíle práce.

Rešerše Tato kapitola obsahuje přehled relevantních témat a přístupů použitých v této práci. To zahrnuje informace a odkazy na hry, které sloužily jako inspirace pro tuto práci. Dále jsou obsaženy informace z odborných článků pro reprezentaci a generování prostředí a metody pro rozhodování vesničanů.

Analýza Témata a přístupy zmíněné v rešerši jsou zhodnoceny v této kapitole. Zhodnocení zahrnují jak pozitivní, tak negativní vlastnosti relevantních her a přístupů.

Návrh Kapitola popisuje strukturu navržené vesnice a pro práci vybrané přístupy. Sepsány jsou základní prvky hry a metody použité pro jejich umístění do scény. Zahrnuty jsou také návrhy grafického uživatelského rozhraní.

Realizace V této kapitole jsou doplňující informace o částech práce, které nejsou dostatečně popsány v návrhu, nebo se jejich implementace od návrhu liší. Dále kapitola obsahuje informace o dostupných způsobech usměrňování simulace hráčem.

Testování Tato kapitola popisuje provedené testy a jejich výsledky. To zahrnuje jak uživatelské, tak výkonnostní testy a také chyby, které byly odhaleny při delší simulaci a způsob jejich opravení.

Kapitola 1

Cíl práce

Cílem této práce je získání přehledu o současných i starších podobných projektech a přístupech použitelných v této práci. Práce se zabývá tématy generování mapy volné přírody, grafickému zpracování této mapy a vesnice a metodami rozhodování vesničanů.

Náhodné generování volné přírody je důležité pro simulaci různorodého chování vesničanů v odlišných prostředích. Rozhodování vesničanů je založeno na jejich dostupných akcích a znalosti jejich důsledků. Tyto důsledky zahrnují ty, co ovlivňují stav daného vesničana, ale i celé vesnice. Z těchto přístupů je také potřeba při analýze vybrat ty nejvhodnější.

Poté je nutno provést popsání modelu prostředí, implementaci vybraných metod pro vizualizaci, způsobu rozhodování vesničanů a generování mapy volné přírody. Dále provedení testů vývoje vesnice, interakce vesničanů a uživatelských testů s následným vyhodnocením jejich výsledků.

Model tohoto prostředí je velmi rozsáhlý a je potřeba popsat všechny budovy, resp. povolání, suroviny, výrobky a potřeby vesničanů a vytvořit pro ně obrázky, použitelné pro 2D vizualizaci modelu.

Kapitola 2

Rešerše

Tato kapitola se zabývá inspirací a relevantními pojmy vzhledem k tématu práce. Nejdříve seznámení se samoorganizujícími modely a budovatelskými hrami, kterými je práce inspirována, v podkapitolách 2.1 a 2.2. Dále probírá témata generování terénu a metody pro rozhodování agentů s pojmy z teorie her v podkapitolách 2.3 a 2.4.

2.1 Samoorganizující modely

Modelů pro simulaci různých prostředí existuje mnoho, pro simulaci starověké vesnice v této práci je třeba, aby se na výsledném stavu simulace podíleli hlavně vesničané, dále také nazýváni agenty, budující tuto vesnici. Níže tedy o relevantních agentních modelech, které již dříve úspěšně využil ve své práci například Daniel Laube [1].

2.1.1 Agentní modely

Agentní modely podle Craiga Reynoldse [2] simulují globální dopad lokálních interakcí členů populace. Tyto interakce probíhají v daném prostředí podle chování členů a jejich charakteristických parametrů.

Eric Bonabeau [3] agentní modely popisuje spíše jako přístup než technologii. Tento přístup popisuje systém z perspektivy jeho základních částí. Jako výhody tohoto přístupu pak uvádí:

- umožňuje vznik jedinečných jevů
- přirozeným způsobem popisuje systém
- je flexibilní

A jako vhodné oblasti použití agentních modelů mimo jiné jmenuje:

Trhy – burza, nákupní boti a strategické simulace.

Organizace – operační risk a organizační design.

Pohyb – evakuace, doprava a chování zákazníků.

2.2 Budovatelské hry

Tato práce je inspirována hrami žánru budovatelských strategických her. Budovatelské hry se rozdělují na hry zaměřené na budování města a na hry zaměřené na boj s nepřátelskými městy nebo postavami. Dále je také lze rozlišovat podle implementace času v těchto hrách na:

Real-time hry aktualizují svůj stav pravidelně a nezávisle na akcích hráčů.

Tahové hry aktualizují svůj stav po ukončení tahu hráče nebo hráčů.

Následuje malý výběr úspěšných budovatelských her, kterými je práce inspirována, od starších titulů po ty současné.

2.2.1 Inspirace starších budovatelských her

Paul Bellow [4] tento žánr označil jako různorodý v preferencích hráčů ke složitosti a estetice těchto her. Jako první hru uvedl SimCity [5] z roku 1989, tato hra je široce uznávaná jako zakladatel žánru. V době kdy hra vyšla bylo velmi neobvyklé, aby hráči neměli určený žádný přesně určený cíl nebo podmínky výhry a prohry, tento styl hry se nazývá sandbox.

Další z důležitých her tohoto žánru je Tropico [6] z roku 2001. Tento příspěvek změnil prostý simulátor města v simulátor diktátorského národu na ostrově. Hráč se tedy dostává do kůže diktátora a snaží se udržet svou moc a zastavit revoluce nespokojené populace.

Velmi inovativní hrou se stalo i Anno 1701 [7] vydané v roce 2006, které přidalo stupně úrovně populace. Zaměřuje se tedy více přímo na obyvatele města. S každou úrovní obyvatele se zvýší nejen daně, které platí, ale i jejich potřeby jako například tabák a parfémy.

2.2.2 Inspirace současných budovatelských her

Největší inspirací pro tuto práci je hra Banished [8]. Tato hra nemá žádný daný cíl a hráč začíná pouze se skladem a omezeným množstvím surovin, které jsou potřeba k chodu vesnice. Rychle se musí postarat, aby měli vesničané domov a aby nedošlo jídlo. Později lze také obchodovat s okolními vesnicemi a hráč se setká i s přírodními katastrofami a epidemiemi.

Jedna z komplexnějších budovatelských her je Cities: Skylines [9] z roku 2015, velmi úspěšná hra konkurující titulům SimCity. Hlavní inovací této hry je přidání realistického dopravního ruchu a spravování sousedství. Pozemní

komunikace a rozložení města se stává mnohem větším problémem než u dříve zmíněných titulů.

Hrou vybočující z řady zmíněných her je Rimworld [10]. Počet obyvatel osady je velmi omezený, ale každý z nich je unikátní. Úkolem hráče je hlavně micro management, tedy kontrola detailních herních mechanik. Určení priorit akcí podle schopností jednotlivých obyvatel a rozdělení vybavení jako například pušek těm, co umí střílet a nemají problémy s psychikou.

Tento žánr je velice rozsáhlý a jistě si zmínku zaslouží i jiné hry jako Knights and Merchants [11] nebo The Settlers [12], ale tyto tituly, které jsou použité jako inspirace, postačí pro seznámení s žánrem budovatelských her.

2.3 Mapy

Ve své práci o generování terénu Jean-David Génevaux [13] napsal, že virtuální terény hrají důležitou roli v počítačové grafice. Oblasti využití jsou například letecké simulátory, počítačové hry, design krajin a filmové prostředí. Terén je v těchto oblastech dominantním vizuálním elementem většiny scén. V posledních letech se podařilo pokročit ve vývoji efektivních metod generování syntetických terénů. Existující techniky se dají zhruba rozdělit na:

- procedurální
- fyzikálně založené
- založené na příkladech terénu
- založené na nákresech terénu

Procedurální a fyzikálně založené metody často postrádají v šíři možností nastavení generování. Metody založené na nákresech vyžadují kvalitní manuálně vytvořené předlohy, což je značně zdlouhavé. Metody založené na příkladech terénu jsou omezeny rozsahem jejich vstupu. Z pohledu geologie je korektní pouze fyzikálně založená metoda.

Při pohledu na reálné terény je zřejmé, že jejich strukturu určují z velké části sítě řek. Je důležité, aby řeky vždy tekly z vyššího terénu směrem dolů a aby neměnily svůj směr příliš náhle.

2.3.1 Generování řek

Většina způsobů jak generovat řeky je pochopitelně spjata s informací o výšce terénu, někdy dále s informacemi o objemu srážek, větru, vlhkosti atd. Často tedy začínají vygenerováním výškové mapy pomocí například Perlinova šumu (metoda pro generování grafického šumu) nebo případně manuální tvorbou této výškové mapy.

Amit Patel [14] použil ve své práci systém s trojúhelníkovou reprezentací, kde každá část mapy má pouze tři hrany. Tato reprezentace velmi omezí

složitost generování. Zůstávají tedy 3 možnosti, jak může každá část řeky vypadat. Tyto možnosti jsou:

Pramen nemá žádný vstup a jeden výstup.

Ohyb má jeden vstup a jeden výstup.

Soutok má na rozdíl od reality pouze dva vstupy a jeden výstup.

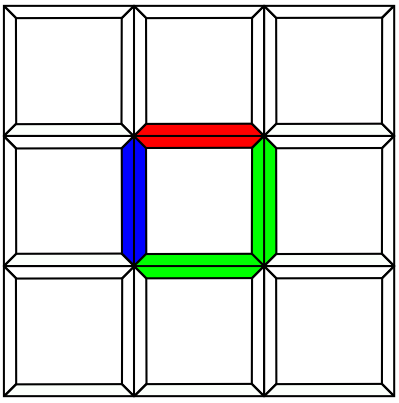
Tento způsob se dá představit jako binární strom. Pramen je list bez potomků, ohyb má pouze jednoho potomka a soutok má potomky dva. Nejlehčím způsobem implementace je potom náhodné rozdělení těchto možností na všechny části mapy. Potom už lze pouze určit, kde řeky jsou a jejich tvar a směr se určí podle toho, jaký druh části mapy na daném místě je.

2.3.2 Dlaždicová reprezentace prostředí

Ares Lagae [15] popisuje dlaždicové mapy ve 2D jako takové uspořádání obrázků, že vyplní celou plochu prostředí bez mezer nebo překrývání. Každý z těchto obrázků se nazývá dlaždice. Množina obrázků použitých v dlaždicové mapě je dlaždicová množina.

Důležitou metodou je Wangovo dláždění [16], které používá konečnou množinu čtvercových dlaždic. Všechny tyto dlaždice mají stejnou velikost a každá hrana dlaždice má přidělenou barvu. Barvy jsou kombinovány několika určenými způsoby. Celá plocha je potom dlážděna libovolně velkým množstvím kopií dlaždic z množiny tak, aby sousední hrany měli stejnou barvu.

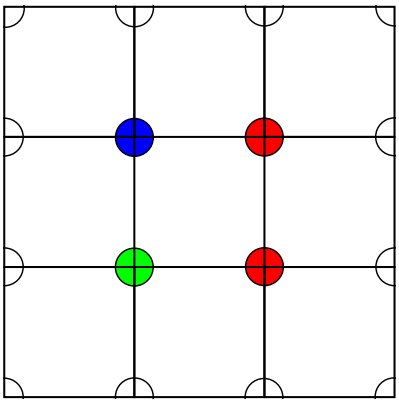
Tento přístup, ale bohužel nezaručuje spojitost signálu v rozích dlaždic, protože nijak neomezuje sousední dlaždice na diagonálách. V počítačové grafice toto způsobuje nežádoucí chyby. Příklad Wangova dláždění je znázorněn na obrázku 2.1.



Obrázek 2.1: Wangovo dláždění

Problém spojitosti v rozích dlaždic řeší rohové dlaždice. Tyto dlaždice jsou velmi podobné Wangovým, ale místo barevných hran mají barevně označené

rohy. Tato metoda zajišťuje spojitost signálu na hranách i v rozích dlaždic. Rohové dlaždice jsou znázorněny na obrázku 2.2.



Obrázek 2.2: Rohové dlaždice

2.3.3 Hledání cesty v generovaném prostředí

Nejzákladnějším způsobem hledání cesty je podle Marcelo Kallmanna [17] geometrické plánování cest. Tento způsob využívá různých variant široce známých algoritmů jako například A* a je důležité se s ním seznámit pro pochopení omezení a vlastností moderních přístupů. Základem hledání cest jsou Eukleidovské nejkratší cesty.

Tyto cesty vždy splňují dvě podmínky, nesmí obsahovat žádné kolize a musí mít minimální délku. Nechť n je počet uzlů v množině S s mnohoúhelníkovými překážkami v $\mathbb{R}^2$. Jsou dány body p a q v $\mathbb{R}^2$, řekneme, že cesta $\pi(p,\ q)$ existuje, pokud π spojuje tyto dva body bez křížení jakékoli hrany překážek v S. Pokud neexistuje jiná kratší cesta spojující tyto dva body, tato cesta je *globálně optimální* a nazývá se Eukleidovská nejkratší cesta.

Pro počítačové hry existuje datová struktura navržená pro podporu hledání cest a navigačních výpočtů. Tato struktura se nazývá navigation mesh, ale neexistuje k ní žádná formální definice. V posledních letech se navigation mesh stala jedním z nejpoužívanějším způsobů implementace navigačních systémů v počítačových hrách.

Hlavní funkcí této struktury je reprezentovat volné prostředí efektivně pro velké množství dotazů vyhledání cesty. Také by měla podporovat ostatní prostorové dotazy, jako například viditelnost. Její základní vlastnosti jsou:

- Lineární počet buněk, aby vyhledávací algoritmy mohly se strukturou pracovat v přijatelném čase.
- Kvalitní cesty, protože Eukleidovskou nejkratší cestu nelze vždy efektivně nalézt. Vždy je ale potřeba efektivně nalézt alespoň lokálně nejkratší cestu.

- Vyhodnocování průchodu, struktura by neměla potřebovat dopředu znát hodnoty pro vyhodnocení průchodu mezi překážkami.
- Robustnost reprezentace, z důvodu častých přesahů a průniků definovaných překážek. Tyto konflikty by měla struktura dokázat vyřešit.
- Dynamická aktualizace, protože prostředí se ve hrách velmi často mění a obzvlášť pokud je uživateli povoleno prostředí měnit, musí být aktualizace časově nenáročná.

2.4 Metody pro rozhodování agentů

Pro tuto část práce je vhodné seznámit se s teorií her. Teorie her je matematická teorie rozhodování v konfliktních situacích [18]. Rozhodnutí hráčů závisí na rozhodnutích ostatních hráčů a předpokládá se, že hráči jsou inteligentní racionální agenti.

2.4.1 Inteligentní racionální agent

Jeden z důležitých pojmů je právě inteligentní racionální agent. Takovýto agent nebo hráč má v teorii her tyto vlastnosti:

- Zná množinu možných akcí.
- Ví jaké důsledky má každá z daných akcí.
- Dokáže seřadit důsledky na základě váhy hodnot.
- Umí zvolit akci garantující maximální přínosnost.

2.4.2 Vězňovo dilema

Další z pojmů teorie her je vězňovo dilema [19]. Dva muži jsou společně obviněni porušením zákona a jsou drženi odděleně policií. Oběma bylo řečeno nebo mají důvod věřit, že:

- Pokud se jeden přizná a druhý ne, první dostane odměnu s hodnotou 1 a druhý bude pokutován hodnotou -2.
- Pokud se oba přiznají, budou pokutováni hodnotou -1.
- Pokud se ani jeden nepřizná, oba budou propuštěni.

Ani jeden z mužů si nemůže být jistý, jakou akci zvolí ten druhý a musí se rozhodnout, co udělat. První muž ví, že pokud se oba přiznají, dostane pokutu hodnoty -1, pokud se ale nepřizná a druhý muž ano, dostane pokutu hodnoty -2. Je tedy lepší, pokud se přizná. Když se první muž přizná a druhý

se nepřizná, bude volný a navíc dostane odměnu s hodnotou 1, a proto je lepší se přiznat. To si uvědomuje i druhý hráč, a proto se oba přiznají, přestože optimálním rozhodnutím je, že se ani jeden nepřizná.

Následky výběru strategie v závislosti na výběru druhého muže jsou znázorněny v tabulce 2.1.

Tabulka 2.1: Vězňovo dilema

	Druhý muž se přizná	Druhý muž se nepřizná
První muž se přizná	$-1, -1$	$1, -2$
První muž se nepřizná	$-2, 1$	$0, 0$

2.4.3 Nashova rovnováha a Paretovo optimum

Nashova rovnováha [20] je taková kombinace herních strategií, že pro hráče není žádný důvod změnit volbu. Tedy pokud ostatní hráči nezmění svoji strategii, hráč nemůže změnit svoji strategii tak, aby si zlepšil svůj výsledek [21].

Pro případ vězňova dilematu v tabulce 2.1 je tedy Nashovou rovnováhou možnost, kdy se oba přiznají, protože pokud jen jeden z nich změní strategii a nepřizná se, dostane pokutu hodnoty -1 navíc.

Paretovo optimum [22] je taková kombinace herních strategií, že neexistuje jiná kombinace taková, aby si alespoň jeden z hráčů polepšil a nikdo z ostatních hráčů nepohoršil [23].

Pokud se oba rozhodnou, že se nepřiznají, nastávají dvě různé možnosti. Když se pouze jeden rozhodne přiznat, polepší tím sobě, ale druhému muži pohorší. Druhou možností je, že se oba přiznají, ale to uškodí oběma mužům. Tedy pro případ vězňova dilematu v tabulce 2.1 je zřejmé, že Paretovo optimum připadá na kombinaci, kdy se ani jeden nepřizná.

2.4.4 Evoluční algoritmus

Evoluční algoritmus dříve pro stejný účel využil Daniel Laube [1]. Tento algoritmus využívá šlechtění většího množství často náhodně inicializovaných jedinců, zde například vesničanů. Na tuto množinu jedinců (potomků) zvanou populace se používají operátory. Mezi důležité operátory pro vznik další populace patří:

Mutace pouze náhodně modifikuje potomky.

Křížení vybere dva jedince a vytvoří z nich nové potomky.

Selekce zvolí vhodné kandidáty pro příští populaci.

Tato metoda využívá k hodnocení jedinců takzvanou fitness funkci, která dokáže určit, jak blízko je jedinec k optimálnímu řešení problému [24].

Pro vyšlechtění uspokojivého jedince je ale často potřeba mnoho iterací. Při spuštění nové simulace je tedy potřeba čekat dlouhou dobu a výsledek může stále skončit jako lokální maximum.

2.4.5 Konečné stavové automaty

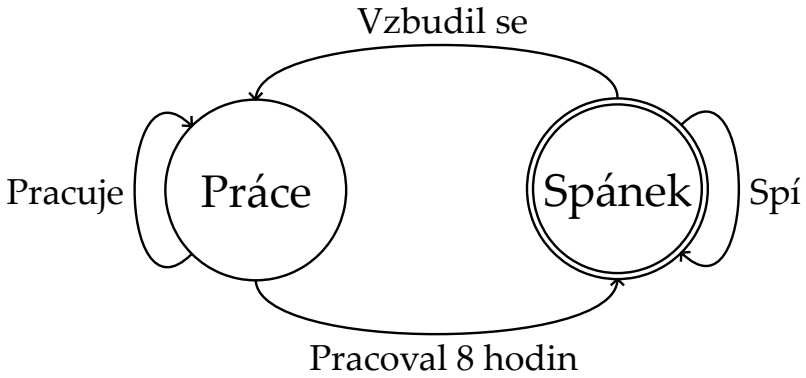
Konečný stavový automat [25] je teoretický výpočetní model definovaný touto pěticí:

- konečná neprázdná množina stavů
- konečná neprázdná množina vstupních symbolů
- přechodová funkce nebo tabulka
- počáteční stav
- množina koncových stavů

Dle Allena Kenta [26] tento model popisuje daný systém jako černou krabici s určeným počtem vstupů a výstupů. Předpokládá se, že akce systému se provádí v diskrétním časovém okamžiku, aby byly spojité vstupy vzorkovány za signálu ze synchronizující složky, jako jsou například hodiny. Dále se předpokládá, že je systém konečný v počtu možných vnitřních stavů a také sekvenční, tedy závisí nejen na současném vstupním signálu, ale i posloupnosti těch předchozích.

Konečné stavové automaty označuje Fernando Bevilacqua [27] jako perfektní pro implementaci umělé inteligence ve hrách s výbornými výsledky bez složitého kódu. Často jsou používány pro organizaci a reprezentování průběhu rozhodování.

Na obrázku 2.3 je zobrazen konečný stavový automat reprezentující velmi zjednodušenou umělou inteligenci člověka ve hře. *Uzly* na tomto obrázku reprezentují stavy automatu a *hrany* jsou přechody mezi těmito stavy. *Dvojitý uzel* označuje počáteční stav.



Obrázek 2.3: Stavový automat

Kapitola 3

Analýza

Tato kapitola je zaměřena na shrnutí informací z kapitoly 2 a vyhodnocení zmíněných přístupů. První podkapitolou 3.1 je struktura budovatelských her, kterými je práce inspirována. Zmíněny jsou jejich dobré i špatné vlastnosti. Druhá podkapitola 3.2 analyzuje dostupné platformy. Další podkapitola 3.3 je o přístupech ke generování map, jejich reprezentaci a manipulaci s nimi. Nakonec shrnutí a vyhodnocení metod používaných pro rozhodování agentů v podkapitole 3.4.

3.1 Struktura budovatelských her

Tato podkapitola shrnuje a zhodnocuje hlavní vlastnosti her, které byly zmíněny v podkapitole 2.2. Zaměřuje se na to, co tyto hry mají společné a čím uspěly nebo naopak selhaly.

3.1.1 Simcity a Cities: Skylines

Hry SimCity [5] a Cities: Skylines [9] jsou si velmi podobné v důležitosti rozložení města a návrhu dopravní sítě. Město které vzniká je tedy velmi rozrostlé a je potřeba propojit různé části města tak, aby měli obyvatelé přístup k práci a co nejvíce službám.

Různorodost stavby budov je v těchto hrách velmi omezená, protože většina budov vzniká sama v průběhu hry. Tyto budovy jsou postaveny v zónách (rezidenční, industriální a komerční) označených hráčem. Hlavní náplní her je tedy návrh robustní sítě města, který vyžaduje pokročilé plánování a strategii. Tento typ her plní požadavky většiny zkušených hráčů, ale pro začátečníky může být příliš složitý a nezáživný, protože většinu hry pouze čekají a sledují budovy, upozornění a dopravní prostředky.

3.1.2 Anno 1701 a Banished

Anno 1701 [7] a Banished [8] zde zastupují budovatelské hry, které používají například pro stavbu nejen peníze, ale i suroviny, jako je dřevo nebo kámen. Při porovnání s hrami simulujícími města jdou tyto hry více do detailu a vznikající vesnice nebývají tak rozrostlé.

Hráči mají větší kreativní svobodu nad rozložením města a více se přizpůsobují okolnímu prostředí, jako jsou například hory, řeky nebo zdroje surovin. Díky menšímu počtu obyvatel také hráč sleduje jejich akce a stav jako je spokojenost nebo zdraví. Stavbou budov poté určuje, na co by se měli obyvatelé soustředit a určuje jejich povolání. Tato povolání na sobě často závisí, protože k vykonávání práce potřebují nějakou surovinu, kterou vyrábí někdo jiný.

Dále může hráč obchodovat, aby získal suroviny, kterých nemá dostatek, nebo je nemůže produkovat. Tyto hry jsou tedy relativně komplexní, ale lépe hráči znázorňují, co vesnice v danou chvíli potřebuje.

3.1.3 Rimworld

Rimworld [10] je hrou, která se zaměřuje spíše na obyvatele, než samotnou osadu. Cílem této hry je uniknout z planety, na které ztroskotala jejich vesmírná loď. Na této planetě ale nejsou sami.

Zatímco se hráč snaží vybudovat osadu a prostředek pro únik z planety, hra vytváří události. Tyto události jsou například útok ostatních obyvatel planety, volání o pomoc nebo vlna extrémní zimy. Hráč musí na tyto události rychle reagovat, nebo může například přijít o obyvatele, kvůli jeho zraněním.

Ve hře je dostupných mnoho různých surovin a způsobů, jak je získat (produkce, obchodování, nalezení nebo od poražených nepřátel). Obyvatelé mají vždy několik charakteristických vlastností, které je ovlivňují jak pozitivně, tak negativně. Hráč určuje priority obyvatel a snaží se efektivně rozdělit potřebné úkoly mezi obyvatele.

Tato hra se tedy zaměřuje na obyvatele a jejich únik z planety, zatímco osada je jen prostředkem, jak přežít. Hráč staví osadu na základě potřeb jednotlivých obyvatel a současných událostí.

3.2 Platformy

Výběr platformy z velké části ovlivňuje, na co je vývoj zaměřen, jeho omezení a výhody. Vybraná platforma by měla vývoj značně zjednodušit s co nejméně omezeními. Zvažované platformy pro tuto práci jsou:

- Unity Engine [28]
- Godot Engine [29]
- Qt Framework [30]

Unity je současně jedna z nejpoužívanějších platforem zejména pro vývoj 3D her. Proto je k dispozici mnoho článků o dostupných funkcích. Qt je velmi robustní platforma, použitelná pro vývoj velkého množství různých aplikací a uživatelského rozhraní. Godot Engine, platforma vyvíjena její komunitou, je z tohoto výběru nejmladším projektem, zaměřujícím se na umožnění vývoje her bez potřeby použití jiných nástrojů.

Pro tuto práci je použit Godot Engine zejména pro jeho jednoduchost a otevřenost díky zveřejňování pod licencí MIT. Pro tuto platformu je navržen a optimalizován vlastní dynamický programovací jazyk GDScript, se syntaxí velmi podobnou známému jazyku Python.

3.3 Mapy

V této podkapitole jsou vyhodnoceny možné postupy pro reprezentaci, vytvoření a manipulaci s prostředím. Nejdříve se v části generování mapy řeší vytvoření tohoto prostředí a následně jeho reprezentace v další části. V poslední části je zhodnocen přístup k navigaci vesničanů.

3.3.1 Generování mapy

Jak bylo zmíněno v podkapitole mapy 2.3, terén tvoří velikou část scény a má tedy velký dopad na celkový dojem hry a její funkcionalitu. Byly zmíněny techniky používané pro generování prostředí. Následují zhodnocení těchto technik.

Procedurální generování funguje na základě určení základních pravidel generovaného světa. Tato technika je velmi užitečná pro rychlé vytvoření velkého množství různých prostředí za použití omezených prostředků. Pokud je potřeba vytvořit komplexnější nebo reálnější prostředí, je vhodné použít jednu z následujících technik.

Pro realističtější prostředí se využívá fyzikálně založené generování, které se zaměřuje na následky působení vody, změn teploty nebo lidské interakce s prostředím. Pro tuto metodu je potřeba mít dostatečné množství informací o těchto fyzikálních dopadech a implementovat je do procesu generování.

Pokud jsou k dispozici příklady nákresů nebo jiná forma informací o prostředí, jako je například výška terénu, je možné jejich kombinací generovat nové, podobné prostředí. Tato metoda dokáže vytvořit velmi kvalitní terén, ale nikdy nemůže vzniknout takové prostředí, které není zastoupeno v použitých příkladech.

3.3.2 Reprezentace prostředí

Pro generování prostředí je vhodné mít určené části světa, ze kterých se generované prostředí skládá a pravidla umístění těchto částí. Čím jednodušší a vzájemně podobnější jsou tyto části, tím lépe se budou skládat do výsledného prostředí. Je tedy důležité, aby na sebe tyto části správně navazovaly. Z tohoto

důvodu a pro dynamickou interakci s prostředím je potřeba vhodná reprezentace prostředí.

Jedna z možných reprezentací, zmíněna v sekci 2.3.2, je pomocí čtvercových dlaždicových map. Pro správné použití Wangova dláždění je potřeba relativně vysoký počet obrázků v závislosti na počtu použitých barev neboli různých typů dlaždic.

Ideálním způsobem reprezentace jsou rohové dlaždice, které se správně vytvořenou množinou dlaždic perfektně navazují na jakékoliv části hran.

3.3.3 Navigace prostředím

Navigace prostředím je důležitá pro samotnou funkci hry, aby mohli vesničané provádět různé akce, musí být schopni se přemístit tam, kde je lze provést. Také je důležité, aby navigace fungovala s aktuálním stavem prostředí dynamického světa. Pro větší počet aktivních vesničanů je potřeba, aby hledání cesty nebylo příliš časově náročné.

V podkapitole 2.3 byla pro tento účel zmíněna struktura navigation mesh. Rychlé vytváření a používání této struktury je podporováno platformou Godot Engine. To znamená, že je jako většina výpočetně náročných základních funkcí platformy implementována v jazyce C++. Pro použití této struktury je nutné vytvořit hlavní mnohoúhelník, který reprezentuje plochu, po které je možné se pohybovat. Dále je potřeba seznam mnohoúhelníků, které reprezentují plochu, po které se pohybovat nelze.

Potřebnou časovou náročnost struktura splňuje díky lineárnímu počtu buněk, které mohou být například mnohoúhelníky, trojúhelníky nebo čtverce. Reprezentace buněk závisí na implementaci navigation meshe.

Online aktualizace této struktury, tedy aktualizace za běhu hry jsou možné přidáním nových mnohoúhelníků a aktualizováním buněk, které byly novou překážkou ovlivněny. Tato operace je náročnější než hledání cesty a neměla by se používat příliš často.

3.4 Rozhodování agentů

Rozhodování agentů je důležitou částí práce a je potřeba zhodnotit možné přístupy z podkapitoly 2.4. Jako první se tato podkapitola zabývá definováním vesničana jako části simulace. Dále zhodnotí reprezentaci pomocí konečných stavových automatů a šlechtění populace použitím evolučního algoritmu.

3.4.1 Inteligentní racionální agent

Definice reprezentace schopností vesničanů a informací jim dostupných, viz sekce 2.4.1, je pro samotné rozhodování velmi důležité. Rozhodování je založeno na zhodnocení všech akcí a výběru té nejlépe ohodnocené. Nejdůležitější částí agentů je tedy dokázat co nejlépe zhodnotit akce na základě jejich cíle.

Rozhodnutí se mohou zakládat na cílech samotného agenta, ale i skupiny agentů nebo celé simulace. V tomto případě je také potřeba, aby agenti byli schopni určit důležitost svých cílů vůči těm ostatním. Například pokud by agent nadhodnotil cíl celé simulace, místo svého cíle jít se najíst a umře hladem, simulace přijde o mnoho práce, kterou by agent dále provedl, pokud by se šel najíst včas a přežil.

Dostupné akce mohou být hodnoceny staticky nebo měnit hodnocení dynamicky na základě úspěchu předchozích akcí nebo stavu simulace.

3.4.2 Konečné stavové automaty

Konečné stavové automaty zmíněné v sekci 2.4.5 jsou velmi přímočarým způsobem, jak popsat rozhodování agentů v různých situacích. Rozhodování agentů závisí na aktuálním stavu agenta, ale i prvků simulace. Akce agent provádí při přechodu mezi stavy a je možné přejít znovu do stejného stavu.

Tato reprezentace je ideální pro definování základního cyklu akcí agentů. Je možno ji upravovat manuálně i automatizovaně za běhu simulace.

3.4.3 Evoluční algoritmus

Evoluční algoritmus dokáže dosáhnout velmi uspokojivých výsledků. Jak bylo zmíněno v sekci 2.4.4, pro správnou funkci této metody je ale potřeba navrhnout kvalitní fitness funkci, která dokáže jedince ohodnotit a není časově náročná, protože bude použita mnohokrát.

Jedince je také potřeba reprezentovat takovým způsobem, aby na ně bylo možno fitness funkci a operátory použít. Obvyklé typy reprezentací jsou:

Genetický algoritmus je reprezentace jedinců pomocí binárních řetězců, tedy vektorů z $\{0, 1\}^n$.

Genetické programování reprezentuje jedince jako stromy, kde listy jsou vstupem a vnitřní uzly jsou funkce používané na tento vstup.

Evoluční programování používá stavové automaty jako jedince, kde optimalizuje přechody mezi stavy.

Genetický algoritmus se pro reprezentaci rozhodování agentů nehodí, protože je příliš omezená binárními prvky vektoru. Genetické programování je možno použít pro reprezentaci reakcí agenta na různé události, ale evoluční programování má výhodu jednoduchosti a intuitivní podoby chování a aktuálního stavu agenta.

Nevýhodou evolučních algoritmů je potřeba před každým spuštěním hry šlechtit populaci a možnost uváznutí v lokálním maximu, kdy je možné, že se agenti budou chovat velmi neobvykle a bude potřeba spustit šlechtění znovu.

Kapitola 4

Návrh

Tato kapitola se zabývá mnoha různými algoritmy a přístupy vybranými pro vytvoření dynamické scény. To zahrnuje vizualizace a reprezentace prostředí, vesničanů a budov v podkapitole 4.4. V podkapitolách 4.1 a 4.2 jsou popsány požadavky práce a návrh uživatelského rozhraní. Návrh generování prostředí je v podkapitole 4.3 a dále struktura vesnice v podkapitole 4.5.

4.1 Funkční a nefunkční požadavky

V této podkapitole jsou sepsány požadavky pro tuto práci. Nejdříve se zabývá požadavky funkčními, tedy požadavky na možnosti a funkce hry. Poté jsou zmíněny požadavky nefunkční, tedy použitelnost, spolehlivost, výkon a podpora údržby.

4.1.1 Funkční požadavky

- Simulace bude zasazena do 2D prostředí a její stav je dynamicky vizualizován.
- Simulaci bude možno usměrňovat, ale dokáže se vyvíjet plnohodnotně bez jakékoli vnější interakce.
- Náhodně generované prostředí a schopnost vesničanů se tomuto prostředí přizpůsobit.
- Suroviny potřebné pro vývoj vesnice a uživatelské rozhraní zobrazující aktuální počet surovin.
- Stavba budov a specializace vesničanů pro budovou dané povolání a zpracování surovin.
- Interakce s herními objekty, jako například stromy, pro přístup k surovinám podle možností generované mapy.

- Uživatelské rozhraní pro stavbu budov určenou hráčem.
- Vesničané budou schopni rozhodnout, která budova je v danou chvíli nejpotřebnější a vybrat místo, kam ji postavit.

4.1.2 Nefunkční požadavky

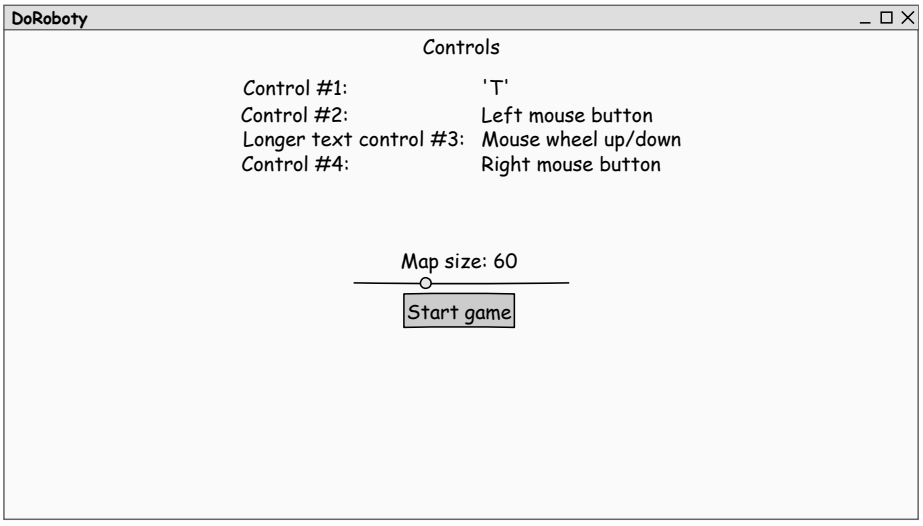
- Aplikace bude plně ovladatelná pomocí klávesnice a myši.
- Pro implementaci bude použita platforma Godot Engine.
- Lze rozlišit plochu, po které se můžou vesničané pohybovat nebo stavit budovy a plochu nepřístupnou.
- Pohyb vesničanů by měl být omezen pouze na přístupné plochy.
- Spustitelnost aplikace alespoň na systémech Windows a Linux.

4.2 Uživatelské rozhraní

Tato kapitola obsahuje návrh uživatelského rozhraní použitého ve hře. První je návrh hlavního menu, které se zobrazí po spuštění hry. Dále je znázorněno grafické rozhraní dostupné po spuštění hry.

4.2.1 Hlavní menu

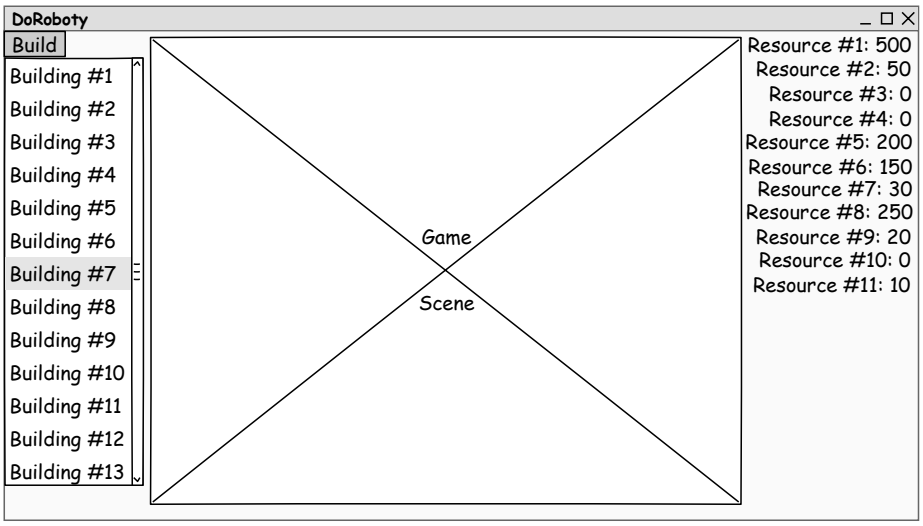
Hlavní menu hry zobrazuje potřebné instrukce pro ovládání hry a nastavení parametru velikosti mapy. Návrh hlavního menu je znázorněn na obrázku 4.1.



Obrázek 4.1: Návrh uživatelského rozhraní hlavního menu

4.2.2 Grafické uživatelské rozhraní hry

Po spuštění hry z hlavního menu se zobrazí generovaná scéna a uživatelské rozhraní pro stavbu budov a zobrazení dostupných surovin. Návrh uživatelského rozhraní je znázorněn na obrázku 4.2.



Obrázek 4.2: Návrh uživatelského rozhraní ve hře

4.3 Generování mapy

Pro reprezentaci mapy je použito dláždění zmíněné v podkapitole 2.3. Tato podkapitola se zabývá generováním dlaždic této mapy a umístěním objektů do scény. Pro generování jsou použity základní principy procedurálního generování map.

4.3.1 Generování dlaždic

Z důvodu omezeného počtu obrázků různých typů dlaždic není možno využít výhod Wangova nebo rohového dláždění, které byly zmíněny v sekci 2.3.2. Na obrázku 4.3 je zobrazena použitá dlaždicová množina, která byla vytvořena autorem práce.

Obrázek 4.3: Dlaždicová množina

Při generování dlaždic na začátku hry jsou použity pouze typy voda a tráva, ostatní typy jsou tvořeny v průběhu hry. Hra používá 4 typy dlaždic zobrazené na obrázku 4.3:

Hlína označuje zastavenou plochu.

Voda označuje nepřístupnou plochu.

Tráva označuje nevyužitou přístupnou plochu.

Pole označuje plochu využitou farmou.

Mapa je vždy čtvercová a velikost je určena parametrem. Na začátku generování dlaždic se náhodně vybere bod na mapě, který určuje střed jezera a další bod, který označuje začátek řeky směřující do jezera.

Poté se rozhodne typ každé dlaždice s 99% pravděpodobností pro typ tráva a 1% pravděpodobností pro typ voda, aby nevznikla pouze velká travnatá plocha. Dále se dlaždice typu tráva změní na typ dlaždice voda pokud platí nerovnice 4.1.

$$g \cdot l \cdot m > d \tag{4.1}$$

V rovnici 4.1 je g pseudonáhodné číslo generované algoritmem napodobujícím vlastnosti náhodných čísel v intervalu $(1, 2)$. Dále l označuje velikost jezera, m označuje velikost mapy a d označuje vzdálenost mezi pozicí dlaždice a středem jezera.

Nakonec se pomocí lineární interpolace, tedy vybrání bodu v intervalu mezi body jezera a řeky, změní typ dlaždic na vybraných bodech a okolí na vodu. Tento proces je popsán pseudokódem 1. Vstupní parametry algoritmu jsou l označující bod jezera a r označující bod řeky.

Algoritmus 1: Generování řeky

```
   Vstup: Dva 2D vektory l a r
 1 t ← 0,0
 2 while t < 1,0 do
 3 |   p ← lineární_interpolace(l, r, t)
 4 |   for x ← −1 to 2 do
 5 |   |   for y ← −1 to 2 do
 6 |   |   |   Nastav dlaždici v bodu p + (x, y) na typ voda
 7 |   |   end
 8 |   end
 9 |   t ← t + 0,01
10 end
```

Výpis kódu 1: Algoritmus pro generování řeky

4.3.2 Generování objektů

Objekty jsou generovány velmi podobným způsobem jako dlaždice. Všechny objekty jsou umístěny vždy jen na dlaždice typu tráva. Při generování každé dlaždice typu tráva je $0,5\%$ pravděpodobnost, že na ni bude přidán strom a stejná šance, že bude přidána hromada kamení. Nízká hodnota $0,5\%$ byla zvolena pouze pro zvýšení různorodosti scény a vytvoření překážek pro stavbu budov.

Hlavní způsob přidání těchto objektů je podobný generování vody na základě blízkosti k jezeru. Nejdříve se určí dva náhodné body na mapě, jeden určuje střed naleziště kamení a druhý střed lesa.

Pro každou dlaždici se zjistí, ke kterému z těchto dvou bodů je blíže. Poté se určí typ objektu a pravděpodobnost jeho umístění. Pravděpodobnost je menší, čím dále se nachází dlaždice od bodu. Typ objektu (strom nebo hromada kamení) se určí podle bodu, ke kterému je blíže. Nakonec se algoritmem napodobujícím vlastnosti náhodných čísel vygeneruje pseudonáhodné číslo v intervalu $(0, 1)$ a podle určené pravděpodobnosti se objekt buď umístí na dlaždici, nebo zůstane dlaždice prázdná.

4.4 Základní prvky hry

Tato podkapitola popisuje návrh základních prvků hry. Důležitou částí jsou suroviny, které budou ve hře dostupné a interaktivní objekty, které jsou potřebné pro některé suroviny. Nakonec jsou sepsány všechny budovy, které mohou vesničané postavit.

4.4.1 Suroviny

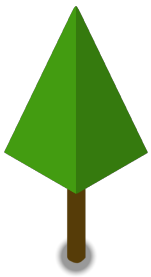
Pro stavbu budov a uspokojování potřeb vesničanů jsou potřeba suroviny. Seznam surovin dostupných ve hře podle kategorií (stavební materiály, částečné produkty, jídlo a luxusní výrobky) je:

- Stavební materiály:
  - Dřevo
  - Kámen
- Částečné produkty:
  - Obilí
  - Mouka
  - Chmel
  - Bavlna
  - Kůže

  - Mléko
  - Kuře
  - Kráva
  - Divoké zvíře
- Jídlo:
  - Chleba
  - Maso
  - Vajíčko
  - Sýr
- Luxusní výrobky:
  - Sud piva
  - Oblečení

4.4.2 Interaktivní objekty

Pro získání stavebních materiálů je potřeba, aby vesničané interagovali s objekty, které jsou částí generovaného prostředí. Obrázek 4.4a je použit ve hře pro zobrazení stromů v prostředí a slouží vesničanům jako přístup ke dřevu. Obrázek 4.4b je použit pro zobrazení hromady kamení a zpřístupňuje vesničanům kámen. Pro zajištění stálého přístupu k těmto surovinám se při odstranění objektu ze scény přidá nový objekt stejného typu na náhodné volné pozici na mapě.



(a) Strom

(b) Kameny

Obrázek 4.4: Interaktivní objekty

4.4.3 Budovy

Pro zpracování a produkování surovin vesničané potřebují postavit budovy. Pro postavení různých budov jsou potřeba různé počty stavebních materiálů.

Zároveň má každá budova svoji velikost, která určuje, kolik dlaždic v prostředí zabírá. Všechny budovy ve hře používají obrázky vytvořené autorem práce. Informace o jednotlivých budovách jsou v části (a) následujících tabulek a podoba budov v části (b).

- Trh 4.1
- Dřevorubecký srub 4.2
- Kamenický srub 4.3
- Farma 4.4
- Zvířecí farma 4.5
- Lovecký srub 4.6
- Mlýn 4.7
- Pekárna 4.8
- Řeznictví 4.9
- Pivovar 4.10
- Hospoda 4.11
- Mlékárna 4.12
- Krejčovství 4.13

Trh	
Potřebné dřevo	50
Potřebný kámen	50
Výška	6
Šířka	4
Produkuje	–
Zpracovává	–

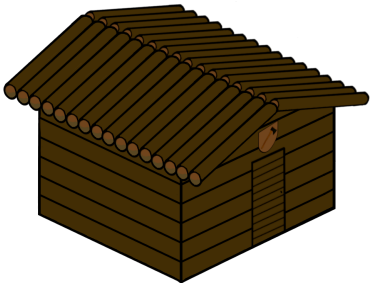
(a) Vlastnosti trhu

(b) Podoba trhu ve hře

Tabulka 4.1: Trh

Dřevorubecký srub	
Potřebné dřevo	50
Potřebný kámen	20
Výška	4
Šířka	3
Produkuje	Dřevo
Zpracovává	–

(a) Vlastnosti dřevorubeckého srubu

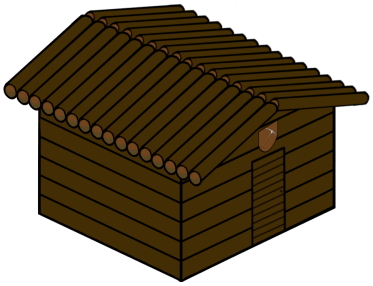


(b) Podoba dřevorubeckého srubu ve hře

Tabulka 4.2: Dřevorubecký srub

Kamenický srub	
Potřebné dřevo	70
Potřebný kámen	20
Výška	4
Šířka	3
Produkuje	Kámen
Zpracovává	–

(a) Vlastnosti kamenického srubu



(b) Podoba kamenického srubu ve hře

Tabulka 4.3: Kamenický srub

Farma	
Potřebné dřevo	200
Potřebný kámen	120
Výška	6
Šířka	4
Produkuje	Obilí nebo Bavlna nebo Chmel
Zpracovává	–

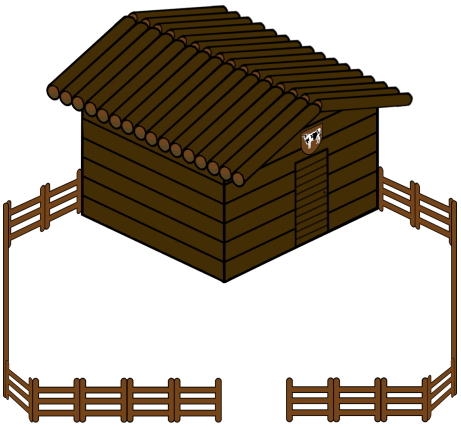
(a) Vlastnosti farmy

(b) Podoba farmy ve hře

Tabulka 4.4: Farma

Zvířecí farma	
Potřebné dřevo	200
Potřebný kámen	100
Výška	6
Šířka	4
Produkuje	Kráva a Mléko nebo Slepice a Vejce
Zpracovává	–

(a) Vlastnosti zvířecí farmy

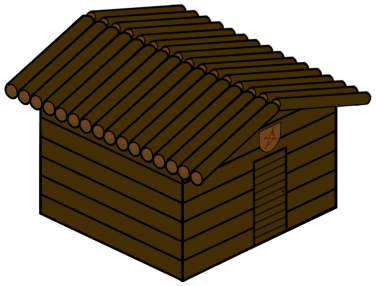


(b) Podoba zvířecí farmy ve hře

Tabulka 4.5: Zvířecí farma

Lovecký srub	
Potřebné dřevo	100
Potřebný kámen	60
Výška	4
Šířka	3
Produkuje	Divoké zvíře
Zpracovává	–

(a) Vlastnosti loveckého srubu

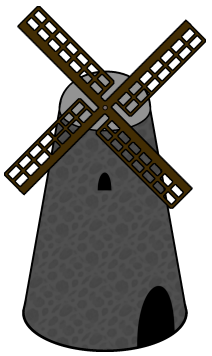


(b) Podoba loveckého srubu ve hře

Tabulka 4.6: Lovecký srub

Mlýn	
Potřebné dřevo	50
Potřebný kámen	150
Výška	5
Šířka	4
Produkuje	Mouka
Zpracovává	Obilí

(a) Vlastnosti mlýnu

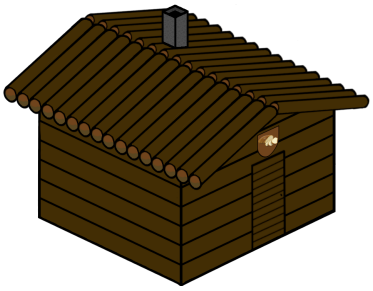


(b) Podoba mlýnu ve hře

Tabulka 4.7: Mlýn

Pekárna	
Potřebné dřevo	150
Potřebný kámen	120
Výška	4
Šířka	3
Produkuje	Chleba
Zpracovává	Mouka

(a) Vlastnosti pekárny

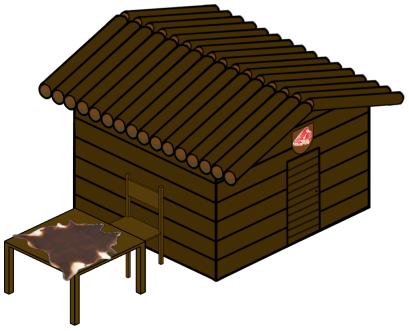


(b) Podoba pekárny ve hře

Tabulka 4.8: Pekárna

Řeznictví	
Potřebné dřevo	150
Potřebný kámen	80
Výška	5
Šířka	4
Produkuje	Maso a Kůže
Zpracovává	Divoké zvíře nebo Kráva nebo Kuře

(a) Vlastnosti řeznictví

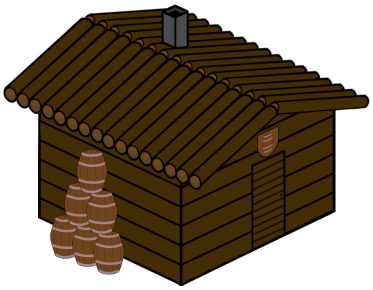


(b) Podoba řeznictví ve hře

Tabulka 4.9: Řeznictví

Pivovar	
Potřebné dřevo	150
Potřebný kámen	150
Výška	4
Šířka	3
Produkuje	Sud piva
Zpracovává	Chmel

(a) Vlastnosti pivovaru

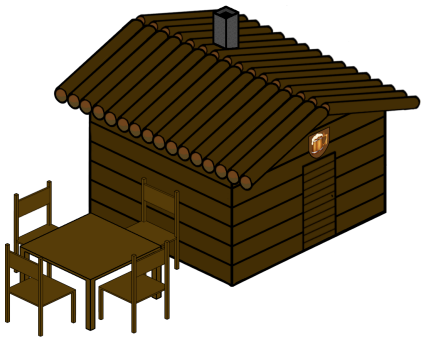


(b) Podoba pivovaru ve hře

Tabulka 4.10: Pivovar

Hospoda	
Potřebné dřevo	200
Potřebný kámen	150
Výška	5
Šířka	4
Produkuje	–
Zpracovává	Sud piva

(a) Vlastnosti hospody

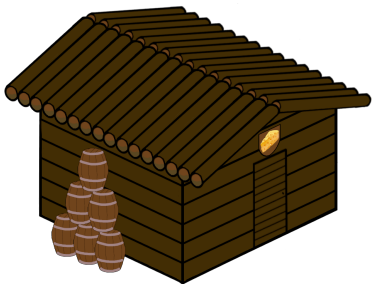


(b) Podoba hospody ve hře

Tabulka 4.11: Hospoda

Mlékárna	
Potřebné dřevo	150
Potřebný kámen	90
Výška	4
Šířka	3
Produkuje	Sýr
Zpracovává	Mléko

(a) Vlastnosti mlékárny



(b) Podoba mlékárny ve hře

Tabulka 4.12: Mlékárna

Krejčovství	
Potřebné dřevo	180
Potřebný kámen	100
Výška	5
Šířka	4
Produkuje	Oblečení
Zpracovává	Bavlna nebo Kůže

(a) Vlastnosti krejčovství

(b) Podoba krejčovství ve hře

Tabulka 4.13: Krejčovství

4.5 Struktura vesnice

Tato podkapitola popisuje prvky a procesy nutné pro chod vesnice. Hlavní a první částí jsou pochopitelně vesničané. Dále je popsán proces stavby budov, který úzce souvisí s výběrem povolání vesničanů a výrobou surovin.

4.5.1 Vesničané

Potřeby vesnice se odvíjí od potřeb vesničanů. Tyto potřeby jsou jídlo a spokojenost, kterou ovlivňuje počet druhů dostupného jídla, čas strávený v hospodě a přístup k oblečení.

Každému vesničanovi roste hodnota hladu a klesá hodnota spokojenosti s uplynulým časem. Pokud se dostane hodnota hladu nad určenou hranici, vesničan přestane pracovat a jde na trh. Na trhu si vybere tolik jídla, aby dostal svůj hlad alespoň na hodnotu 0 (pokud je dostupný dostatek jídla) a aby snědl co nejvíce druhů jídla. Za každý rozdílný druh jídla, které sní, se zvýší jeho spokojenost.

Dalším způsobem, jak zvyšuje vesničan svoji spokojenost je pitím piva. Pokud se dostane jeho spokojenost pod hranici 200, vesničan vyhledá nejbližší hospodu a jde se napít. Vypije tolik piv, aby dostal svoji spokojenost alespoň na hodnotu 1000 (pokud je dostupný dostatek piva).

Posledním způsobem pro udržení vysoké spokojenosti vesničana je oblečení. Pokud není oblečen a na trhu je dostupné oblečení, dojde se obléct. Oblečení vydrží 5 minut, a potom si vesničan musí pro oblečení dojít znovu. Pokud je vesničan oblečen, jeho spokojenost se snižuje poloviční rychlostí.

Obrázky použité pro reprezentaci vesničanů a jejich animace jsou volně dostupné a použitelné pro jakýkoli projekt [31]. Podoba vesničana je znázorněna na obrázku 4.5.

Obrázek 4.5: Podoba vesničana

4.5.2 Chování vesničanů

V předchozí části této podkapitoly bylo popsáno chování vesničanů v závislosti na jejich potřebách. Ve chvíli, kdy je vesničan přidán do hry, se začne rozhodovat, jaké povolání je potřeba.

Výběr povolání závisí na spotřebě a výrobě surovin. Každých 10 sekund hra spočítá přibližnou hodnotu, o kterou se mění dostupné množství každé

suroviny na trhu. Tato hodnota začíná pro každou surovinu s hodnotou nula a dále se počítá podle rovnice 4.2.

$$z_{n+1} = a_{n+1} - a_n + \frac{z_n}{2} \tag{4.2}$$

V rovnici 4.2 je n počet provedených výpočtů, z_{n+1} označuje současnou hodnotu změny dostupného množství dané suroviny a z_n označuje předchozí hodnotu. Obdobně a_{n+1} označuje současné množství dané suroviny na trhu a a_n označuje množství při posledním výpočtu.

Při výběru povolání vesničan zjistí, která surovina nejrychleji ubývá. Pokud se tato surovina vyrábí z jiných surovin, nejdříve zjistí, jestli pro výrobu potřebná surovina přibývá (její hodnota změny množství je kladná). Pokud je záporná, vesničan si vybere povolání, které vyrábí surovinu potřebnou pro výrobu. Pokud je kladná, vybere si povolání vyrábějící surovinu, která ubývá nejrychleji.

Výjimkou je stav, kdy ještě není postavený trh, dřevorubecký srub a kamenický srub. Tyto budovy jsou nezbytně potřebné pro chod vesnice. Pořadí stavby těchto budov pokud nejsou alespoň jednou postaveny je trh, dřevorubecký srub a kamenický srub.

Výběr povolání určuje i kterou budovu vesničan postaví, protože k vykonávání každého povolání je potřeba právě jedna budova. Pokud vesničan postaví trh, zůstává bez povolání, dokud nepostaví jinou budovu.

4.5.3 Stavba budov

Pro stavbu budov je potřeba najít volné místo podle počtu dlaždic, které zabere. Každá budova nějakým způsobem potřebuje interagovat s trhem, a proto se volné místo vždy hledá od nejbližšího trhu. Výjimkou jsou budovy dřevorubecký srub a kamenický srub, pro které se hledá místo nejblíže k lesu nebo nalezišti kamení.

Vesničan hledá místo v kruhu kolem trhu a postupně zvyšuje vzdálenost, dokud nenajde místo odpovídající velikosti budovy. Pokud chybí dřevo nebo kámen, vesničan čeká, dokud není dostupný dostatek těchto surovin.

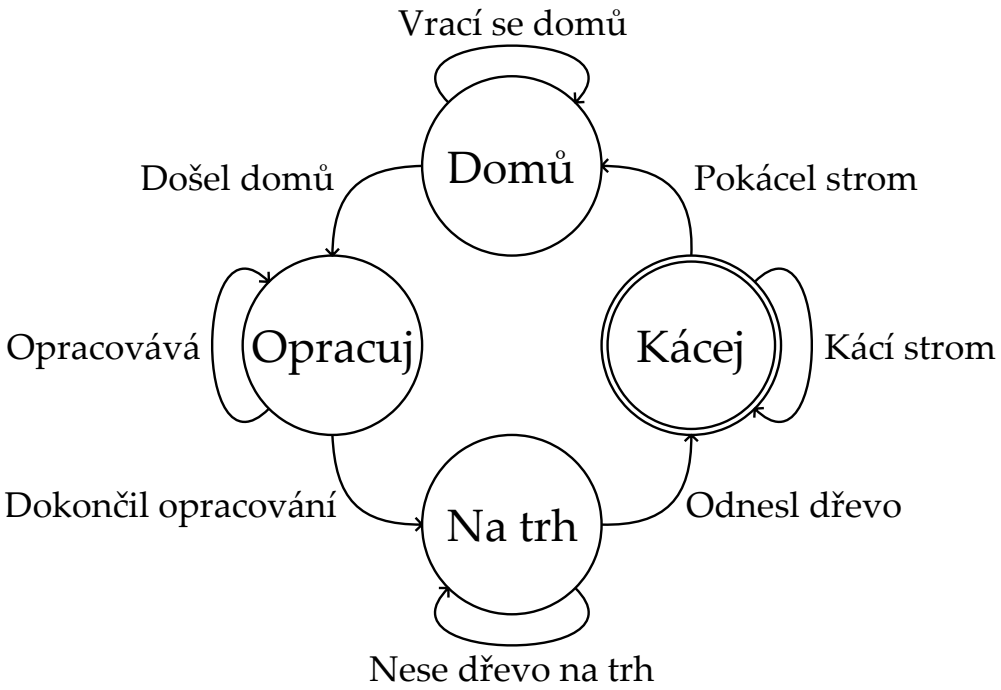
4.5.4 Povolání

Chování všech povolání je navrženo pomocí konečných stavových automatů zmíněných v sekci 2.4.5. Každé povolání má několik stavů, které určují vesničanovi akce. Dostupná povolání jsou:

- Dřevorubec
- Kameník
- Lovec

- Farmář
- Chovatel zvířat
- Mlynář
- Pekař
- Pivovarník
- Hospodský
- Řezník
- Krejčí
- Mlékař

Například dřevorubec má 4 stavy. Tyto stavy jsou kácení, opracování, vracení se domů a odnášení dřeva na trh. Navržený stavový automat dřevorubce je znázorněn na obrázku 4.6. *Uzly* na tomto obrázku reprezentují stavy automatu a *hrany* jsou přechody mezi těmito stavy. *Dvojitý uzel* označuje počáteční stav.

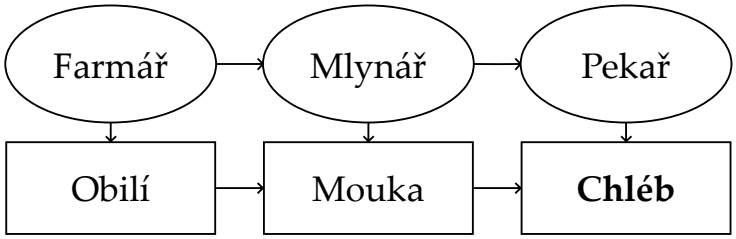


Obrázek 4.6: Stavový automat dřevorubce

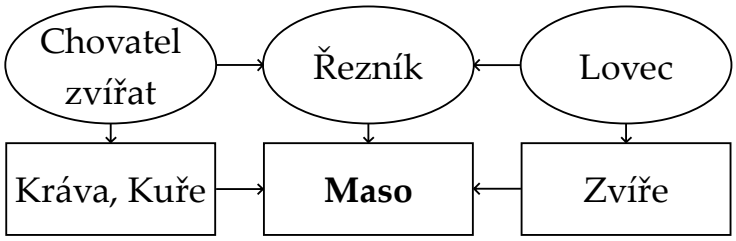
4.5.5 Výroba surovin

Výroba surovin je pro simulaci velmi důležitý proces. Tento proces je pro většinu surovin unikátní a závislý na surovinách potřebných k výrobě. Grafy na obrázcích 4.7, 4.8 a 4.9 znázorňují proces výroby některých jídel.

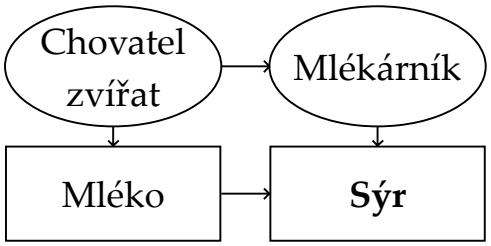
Kulaté uzly grafu jsou povolání potřebná k výrobě daného jídla a *obdélníkové uzly* jsou produkty, které vyrábí. *Hrany* mezi povoláními znázorňují, kdo zásobuje které povolání potřebnými surovinami, *hrany* mezi povoláními a surovinami znázorňují, co dané povolání vyrábí a *hrany* mezi surovinami znázorňují suroviny, ze kterých jsou vyráběny. *Tučný text* označuje finální produkt.



Obrázek 4.7: Výroba chleba

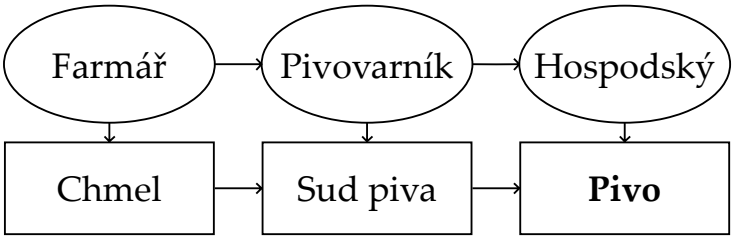


Obrázek 4.8: Výroba masa

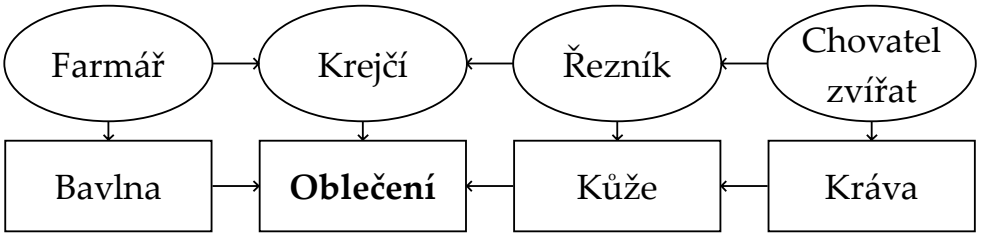


Obrázek 4.9: Výroba sýru

Z grafů je patrné, že některé povolání mohou produkovat více než jeden druh produktů. Dále obrázky 4.10 a 4.11 znázorňují výrobu dalších potřebných produktů, piva a oblečení:

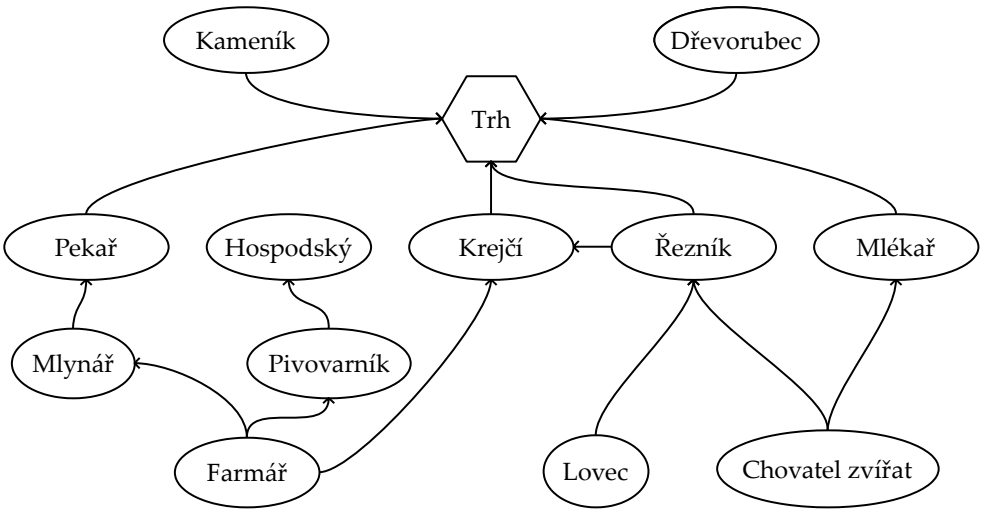


Obrázek 4.10: Výroba piva



Obrázek 4.11: Výroba oblečení

Všechny tyto produkty putují na trh, kde jsou dostupné ostatním vesničanům. Pivo z trhu putuje do hospody, kam chodí vesničané pít, pokud mají nízkou spokojenost. Vztahy všech povolání jsou znázorněny na obrázku 4.12. Finální výrobky jsou po dokončení výroby vesničanům dostupné na trhu, který je označený *šestiúhelníkem*.



Obrázek 4.12: Vztahy povolání a trhu

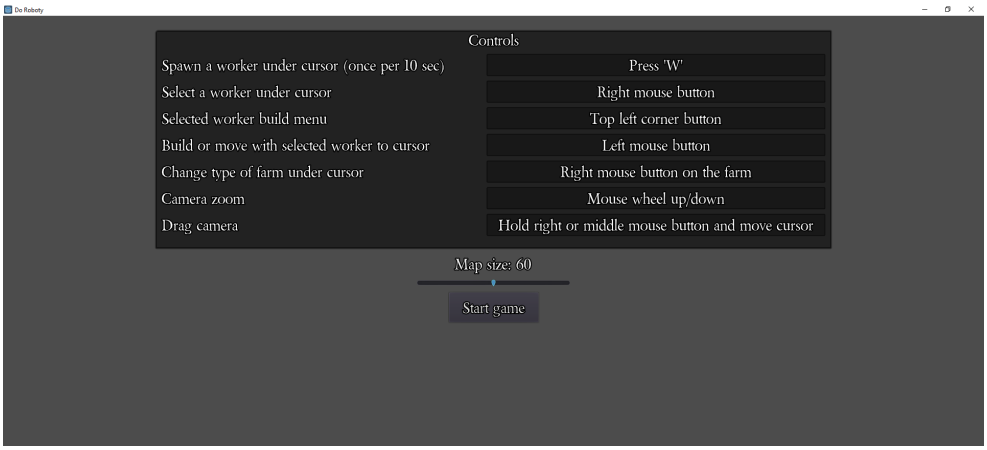
Kapitola 5

Realizace

Tato kapitola popisuje části práce, které nebyly dostatečně popsány v návrhu nebo se nějakým způsobem liší. V podkapitole 5.1 je zmíněn způsob předání potřebných informací hráči a v podkapitole 5.2 je popsáno definování nepřístupných ploch vody. Dále jsou v podkapitole 5.3 uvedeny doplňující informace o chování vesničanů a v podkapitole 5.4 jsou shrnuty způsoby, které jsou hráči k dispozici pro usměrňování simulace.

5.1 Spuštění hry

Po spuštění hry je potřeba předat uživateli informace o ovládání. Proto se ihned zobrazí tři hlavní prvky. První z nich je panel s informacemi o použití kláves, myši a uživatelského rozhraní v průběhu hry. Dále je použit posuvný prvek pro umožnění nastavení velikosti mapy s minimální hodnotou 20 a maximální hodnotou 100. Poslední částí úvodní obrazovky je tlačítko pro spuštění simulace. Úvodní obrazovka hry je znázorněna na obrázku 5.1.



Obrázek 5.1: Úvodní obrazovka

5.2 Prostředí

Po spuštění simulace se vygeneruje prostředí se zadanou velikostí a je potřeba vytvořit strukturu reprezentující přístupné a nepřístupné plochy, jak bylo zmíněno v sekci 3.3.3. Tato struktura se vytváří po dokončení generování celého prostředí. Použitý algoritmus využívá rekurze, tedy vnořené volání stejné funkce, a postupně prochází všechny krajní dlaždice typu voda. Nejdůležitější částí tohoto algoritmu je funkce, která rozhoduje, jestli navazuje na dlaždici další dlaždice vody a jaké vrcholy přidat do struktury. Tato funkce je zobrazena ve výpisu kódu 2.

```
func next_tile(x, y, direction):
    # all connected tiles checked
    if(check_end(x, y) == false):
        var newDirection = next_direction(x, y, direction)
        # at the first checked tile again
        if(tiles[y][x] != 3):
            tiles[y][x] = 2
        # add corners to the polygon
        check_corners(x, y, direction, newDirection)
        match newDirection:
            'N':
                next_tile(x, y-1, newDirection)
            'E':
                next_tile(x+1, y, newDirection)
            'S':
                next_tile(x, y+1, newDirection)
            'W':
                next_tile(x-1, y, newDirection)
    else:
        # mark all connected tiles as checked
        clear_poly(x, y)
```

Výpis kódu 2: Funkce next_tile

5.3 Chování vesničanů

Jakmile je vesničan přidán do scény, jeho spokojenost se začne snižovat a hlad zvyšovat. Každý vesničan sdílí stejné chování při naplňování těchto potřeb. Povolání určují akce, které vesničan provádí, ale při naplňování potřeb se úplně přeruší práce, kterou vykonává. Pokud se vesničanovi nepovede naplnit

svoji potřebu, kvůli například nedostatku jídla, vrací se do práce a zkouší naplnit svoji potřebu později.

Vesničané postupně tvoří systém, který dokáže naplnit potřeby všech vesničanů ve scéně co nejefektivněji. Jejich prospěšnost pro vesnici je maximalizována pomocí výběru jejich povolání na základě aktuálních potřeb vesnice. Samotný vesničan není schopen naplnit svoje potřeby.

5.4 Usměrňování simulace hráčem

Jak je znázorněno na obrázku 5.1, hráči je umožněno usměrňování simulace. Tyto akce jsou přidání a vybrání vesničana, pohyb nebo stavba vybraným vesničanem a změna typu farmy. Po dokončení akce vesničana, kterou určil hráč, se vesničan ihned vrací zpět do vykonávání akcí, které byly přerušeny.

Pro změnu typu farmy se zobrazí menu, které obsahuje dostupné typy farmy. Toto menu zmizí pokud hráč přesune kurzor mimo jeho plochu. Menu je znázorněno na obrázku 5.2.

Obrázek 5.2: Menu pro změnu typu farmy

Kapitola 6

Testování

V této kapitole je popsán průběh a vyhodnocení všech provedených testů. Podkapitola 6.1 popisuje způsob průběžného testování a v podkapitole 6.2 jsou informace o výsledcích uživatelského testování. Dále podkapitola 6.3 se zabývá testováním delšího průběhu simulace a podkapitola 6.4 se zabývá provedeným výkonnostním testem.

6.1 Průběžné testování

Všechny funkce, které byly do hry přidány, prošly ihned po implementaci manuálním testováním. Pokud se při testování vyskytla chyba, ať už v samotné implementaci nebo v začlenění do projektu s ostatními funkcemi, vždy došlo k analýze kódu a následnému opravení.

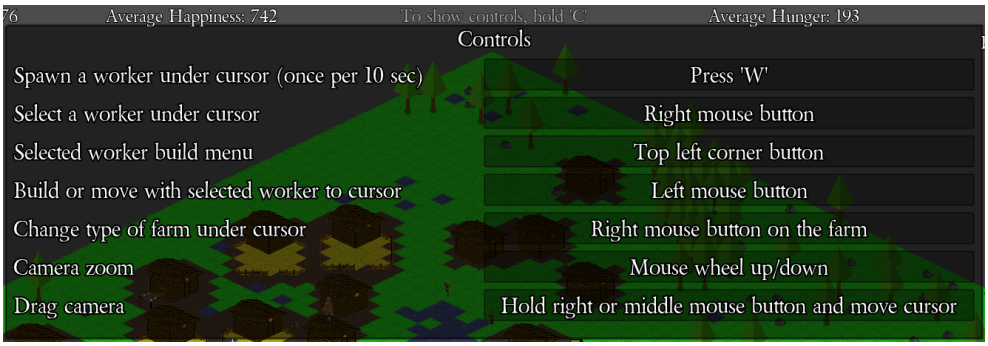
Pro toto testování bylo využito ladícího prostředí platformy Godot Engine. Také bylo odhaleno a opraveno špatné uvolňování prvků odstraněných ze scény, které se projevilo při ukončení hry.

6.2 Uživatelské testování

Hra byla podrobena uživatelskému testování. Toto testování je zaměřeno zejména na předání informace o ovládání hry a jednoduchost použití. Předpokladem pro uživatele testující tuto hru byla znalost anglického jazyka, protože veškeré uživatelské rozhraní je implementováno pouze anglicky.

Pro testování bylo vybráno více uživatelů s různými zaměřeními. Zastoupeni byli jak zkušení uživatelé, jako je například analytik informačních technologií, tak méně zkušení uživatelé. Cíl každého uživatele při testování byl úspěšně použít všechny dostupné možnosti ovládání a porozumět uživatelskému rozhraní. Uživatelé většinou používali pro testování své počítače a nenastal žádný problém se spuštěním nebo výkonností.

Při testech byly dostupné informace o ovládání pouze na úvodní obrazovce, která je na obrázku 5.1. Toto se projevilo jako jeden z nedostatků, protože naprostá většina uživatelů zapomněla nebo si nepřečetla ovládání na úvodní obrazovce. Dále uživatelům také chyběla informace o úspěšnosti vesnice nebo uspokojování potřeb vesničanů. Nedostatky byly vyřešeny přidáním možnosti zobrazení panelu s informacemi o ovládání přímo ve hře a přidáním informace o průměrné spokojenosti a hladu vesničanů. Tyto prvky jsou znázorněny na obrázku 6.1.



Obrázek 6.1: Přidané prvky uživatelského rozhraní

Dalším výstupem testování byla odhalená chyba menu pro změnu typu farmy. Menu zmizelo pouze ve chvíli, když uživatel opustil kurzorem jeho plochu. Při zobrazení menu bylo možné mít kurzor mimo menu, a proto vznikla situace, kdy menu zůstalo zobrazené, i když uživatel přestal menu používat. Tato chyba byla vyřešena zvětšením plochy, kterou musí kurzor opustit, aby menu zmizelo.

6.3 Testování delší doby simulace

Stavba vesnice a zpracování surovin dostupných v prostředí má vliv na strukturu a vzhled prostředí. Při tomto testování lze pozorovat změny rozmístění objektů a způsob, kterým se vesnice postupně rozrůstá a přizpůsobuje prostředí.

Při testech byly zjištěny některé neefektivně implementované části práce. Například vybrání vhodného místa pro stavbu trhu, které bylo vyhledáváno po celé mapě, a proto byl rozsah tohoto hledání omezen na dvacet dlaždic. Dalším problémem byla příliš častá aktualizace stavu vesničanů, která byla z tohoto důvodu také omezena.

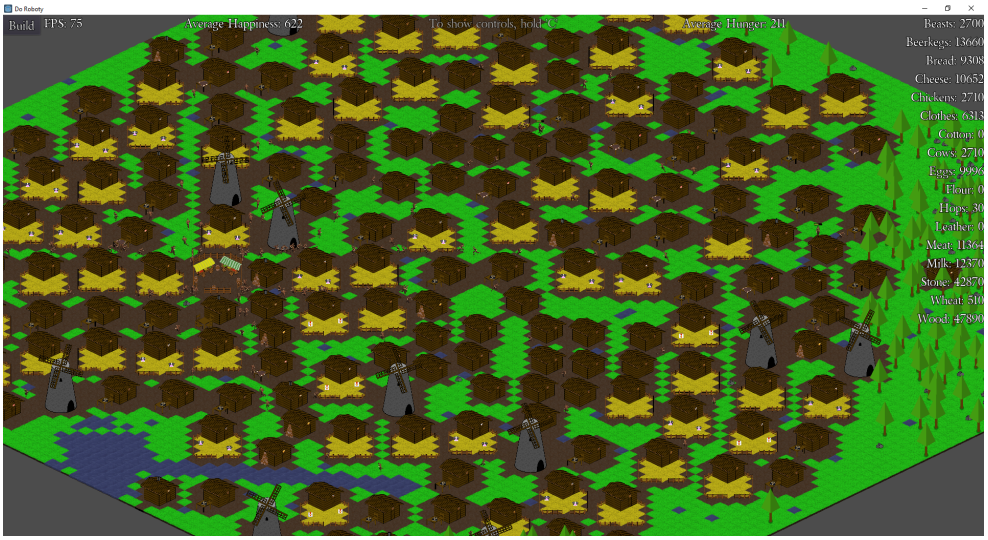
6.4 Výkonnost

Výkonnostní testy byly provedeny na počítači s následující konfigurací:

- Intel Core i7-8700K
- 16 GB DDR4 2666 MHz
- NVIDIA GeForce GTX 1060 3GB

Při tomto testování bylo ve scéně 200 pracujících vesničanů a nedošlo k větším výkonnostním problémům, pouze při stavbě budovy mírně klesá počet snímků za sekundu. Snímek hry pořízený při tomto testu je na obrázku 6.2. Průměrné vytížení počítače při tomto testu:

- Procesor byl vytížen na 10 %.
- Grafická jednotka byla vytížena na 16 %.
- Hra využívala 120 MB paměti.



Obrázek 6.2: Snímek hry při výkonnostním testu

Závěr

Cílem této práce bylo získání přehledu o podobných projektech a metodách, jejich analýza a návrh použití těchto metod pro následnou implementaci samoorganizujícího modelu vesnice a její vizualizace.

Byla provedena rešerše o existujících hrách, které sloužily jako inspirace pro tuto práci. Dále rešerše obsahuje získané informace o často používaných přístupech generování volného prostředí a rozhodování vesničanů. Pro reprezentaci tohoto prostředí byl vybrán přístup dlaždicových map. Pomocí těchto map lze velmi efektivně vytvořit rozsáhlé prostředí s velmi nízkými požadavky. Pro generování dlaždicových map a objektů bylo využito metod procedurálního generování.

Rozhodování vesničanů je založeno na jejich vlastních potřebách a povolání, ale výběr tohoto povolání závisí na stavu celé vesnice a probíhá ihned po přidání vesničana do scény. Tímto způsobem postupně vzniká složitý systém pro uspokojování potřeb všech vesničanů.

Pro práci se podařilo navrhnout jednoduchý, ale zajímavý model prostředí s nemalým počtem vzájemných závislostí herních prvků. Byla použita izometrická reprezentace herních objektů, které ve 2D hrách působí mnohem lépe než ty ploché při pohledu ze shora. Pro práci bylo navrženo grafické uživatelské rozhraní, a díky uživatelským testům se podařilo eliminovat větší nedostatky. Všechny grafické podklady použité pro reprezentaci herních prvků, s výjimkou použitého fontu, obrázků zvířat a vesničanů, byly vytvořeny autorem práce.

Úspěšně bylo dále implementováno usměrňování hráčem. To zahrnuje ovládání vesničanů (pohyb a stavba budov), ale i změnu typu suroviny, které se vyrábí ve vybrané budově. Toto usměrňování ve většině případů úspěšnosti simulace spíše škodí, ale po dokončení akce určené hráčem se vesničané vrací k práci, kterou přerušili.

Možná vylepšení Hlavní částí pro vylepšení je generování prostředí. Současné omezení velikosti prostředí zdaleka neodpovídá maximální možné velikosti. V aktuální verzi implementace je pro generování použitý rekurzivní algoritmus pro vytvoření mnohoúhelníků reprezentujících přístupné i nepřístupné plochy. Tento algoritmus není optimální a jeho přepsání by umožnilo vytváření mnohem větších map. Z tohoto důvodu je v aktuální verzi omezena i velikost a složitost generovaného jezera a řeky.

Další možné vylepšení je umožnění vytvoření nových oddělených vesnic. Tyto nové vesnice by upravily své parametry podle úspěšnosti předchozích vesnic. Tímto způsobem by vznikaly nové vesnice a bylo by možné sledovat změny jejich struktury.

Bibliografie

1. DANIEL, Laube. *Babylon: Samoorganizující se parametrický model starověkého města*. Praha, 2015. Dostupné také z: https://dspace.cvut.cz/handle/10467/65996. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra teoretické informatiky. Vedoucí práce Radek Richtr.
2. REYNOLDS, Craig. *Individual-Based Models* [online]. 1999 [cit. 2019-04-05]. Dostupné z: https://www.red3d.com/cwr/ibm.html.
3. BONABEAU, Eric. Agent-based modeling: methods and techniques for simulating human systems. *PubMed* [online]. 2002, roč. 99 Suppl 3 [cit. 2019-04-05]. ISSN 0027-8424. Dostupné z DOI: 10.1073/pnas.082080899.
4. BELLOW, Paul. *History of City Building Games* [online]. 2017 [cit. 2019-04-06]. Dostupné z: https://litrpgreads.com/blog/history-of-city-building-games.
5. WRIGHT, Will. *SimCity* [software]. 1989 [cit. 2019-04-28]. Dostupné z: https://www.ea.com/games/simcity.
6. POPTOP SOFTWARE. *Tropico* [software]. 2001 [cit. 2019-04-28]. Dostupné z: https://www.worldoftropico.com/us/.
7. RELATED DESIGNS. *Anno 1701* [software]. 2006 [cit. 2019-04-28]. Dostupné z: https://store.ubi.com/eu/anno-1701/58ab30190c8ee463688b4567.html.
8. HODOROWICZ, Luke. *Banished* [software]. 2014 [cit. 2019-04-28]. Dostupné z: http://www.shiningrocksoftware.com.
9. COLOSSAL ORDER. *Cities: Skylines* [software]. 2015 [cit. 2019-04-28]. Dostupné z: https://www.paradoxplaza.com/cities-skylines/.
10. LUDEON STUDIOS. *Rimworld* [software]. 2018 [cit. 2019-04-28]. Dostupné z: https://rimworldgame.com/.

11. JOYMANIA ENTERTAINMENT. *Knights and Merchants* [software]. 1998 [cit. 2019-04-28]. Dostupné z: http://www.knightsandmerchants.net/.

12. BLUE BYTE. *The Settlers* [software]. 1993 [cit. 2019-04-28]. Dostupné z: https://www.thesettlers-alliance.com/en/.

13. GÉNEVAUX, Jean-David; GALIN, Éric; GUÉRIN, Eric; PEYTAVIE, Adrien; BENES, Bedrich. Terrain Generation Using Procedural Models Based on Hydrology. *ACM Trans. Graph.* [online]. 2013, roč. 32, č. 4, s. 143:1–143:13 [cit. 2019-04-07]. ISSN 0730-0301. Dostupné z DOI: 10.1145/2461912.2461996.

14. PATEL, Amit. *Procedural river drainage basins* [online]. 2017 [cit. 2019-04-07]. Dostupné z: http://www.redblobgames.com/x/1723-procedural-river-growing/.

15. LAGAE, Ares; KAPLAN, Craig S.; FU, Chi-Wing; OSTROMOUKHOV, Victor; DEUSSEN, Oliver. Tile-based Methods for Interactive Applications. In: *ACM SIGGRAPH 2008 Classes*. Los Angeles, California: ACM, 2008, 93:1–93:267. SIGGRAPH '08. Dostupné z DOI: 10.1145/1401132.1401254.

16. WANG, Hao. *Games, Logic and Computers* [online]. 1965 [cit. 2019-05-02]. Dostupné z DOI: https://doi.org/10.1007/978-94-009-2356-0_10.

17. KALLMANN, Marcelo; KAPADIA, Mubbasir. Navigation Meshes and Real-time Dynamic Planning for Virtual Worlds. In: *ACM SIGGRAPH 2014 Courses* [online]. Vancouver, Canada: ACM, 2014, 3:1–3:81 [cit. 2019-04-08]. SIGGRAPH '14. ISBN 978-1-4503-2962-0. Dostupné z DOI: 10.1145/2614028.2615399.

18. TOMASSINI, Marco. Introduction to Evolutionary Game Theory. In: *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation* [online]. Amsterdam, The Netherlands: ACM, 2013, s. 765–778 [cit. 2019-04-08]. GECCO '13 Companion. ISBN 978-1-4503-1964-5. Dostupné z DOI: 10.1145/2464576.2480808.

19. TUCKER, Albert W. *A Two-Person Dilemma* [online]. 1950 [cit. 2019-04-24]. Dostupné z: http://www.rasmusen.org/x/images/pd.jpg.

20. NASH, John F. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*. 1950, roč. 36, č. 1, s. 48–49. ISSN 0027-8424. Dostupné z DOI: 10.1073/pnas.36.1.48.

21. POLICONOMICS. *Game theory II: Nash equilibria* [online]. 2017 [cit. 2019-04-08]. Dostupné z: https://policonomics.com/lp-game-theory2-nash-equilibrium/.

22. MORNATI, Fiorenzo. *Pareto Optimality in the work of Pareto* [online]. 2013 [cit. 2019-04-24]. Dostupné z DOI: 10.4000/ress.2517.

23. POLICONOMICS. *Pareto optimality* [online]. 2017 [cit. 2019-04-08]. Dostupné z: https://policonomics.com/pareto-optimal/.

24. MALLAWAARACHCHI, Vijini. *How to define a Fitness Function in a Genetic Algorithm?* [online]. 2017 [cit. 2019-04-08]. Dostupné z: https://towardsdatascience.com/how-to-define-a-fitness-function-in-a-genetic-algorithm-be572b9ea3b4.

25. WRIGHT, David R. *Finite State Machines* [online]. 2005 [cit. 2019-05-03]. Dostupné z: https://web.archive.org/web/20140327131120/http://www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf.

26. KENT, A.; WILLIAMS, J.G. *Encyclopedia of Computer Science and Technology: Volume 25 - Supplement 10: Applications of Artificial Intelligence to Agriculture and Natural Resource Management to Transaction Machine Architectures* [online]. Taylor & Francis, 1991 [cit. 2019-05-03]. Encyclopedia of Computer Science Series. ISBN 9780824722753. Dostupné z: https://books.google.cz/books?id=W2YLBIdeLIEC.

27. BEVILACQUA, Fernando. *Finite-State Machines: Theory and Implementation* [online]. 2013 [cit. 2019-05-03]. Dostupné z: https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867.

28. UNITY TECHNOLOGIES. *Unity* [software]. 2019 [cit. 2019-05-02]. Dostupné z: https://unity.com/.

29. LINIETSKY, Juan. *Godot Engine* [software]. 2019 [cit. 2019-05-02]. Dostupné z: https://godotengine.org/.

30. THE QT COMPANY. *Qt* [software]. 2018 [cit. 2019-05-02]. Dostupné z: https://www.qt.io/.

31. KENNEY. *Isometric Dungeon Tiles* [online]. 2019 [cit. 2019-04-14]. Dostupné z: https://www.kenney.nl/assets/isometric-dungeon-tiles.

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
  - `DoRobotyWin` ..... adresář se spustitelným souborem pro Windows
  - `DoRobotyLinux` ..... adresář se spustitelným souborem pro Linux
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF