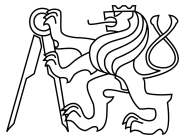
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Babylon: Samoorganizující se parametrický model starověkého města
Student:	Daniel Laube
Vedoucí:	Ing. Radek Richtr
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Babylon je vizualizovaný samoorganizující se modulární model města (urbanistické scény) včetně obyvatel a okolí. Jeho vývoj může být usměrňován, přičemž model reaguje na akce uživatele. Model se organizuje vzhledem k potřebám svých obyvatel (např. hlad) a omezení svého okolí (např. úrodnost polí) a to formou produkce vhodných zdrojů a úpravou své struktury (např. stavba mlýnu, pekárny).
1) Analyzujte problematiky modelů pro samoorganizující se rozvoj urbanistické scény.
2) Definujte na základě požadavků zadavatele soubor pravidel (model) prostředí, podle kterých se bude město rozvíjet.
3) Pomocí metod softwarového inženýrství navrhněte prototyp aplikace simulující samoorg. rozvoj urbanistické scény s ohledem na jeho využití pro strategickou hru a to včetně vhodného rozhraní.
4) Implementujte prototyp vizualizace modelu města pomocí engine Unity.
5) Implementovaný prototyp podrobte vzhledem k účelu aplikace vhodným testům.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 16. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Babylon: Samoorganizující se parametrický model starověkého města

Daniel Laube

Vedoucí práce: Ing. Radek Richtr

16. května 2016

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Radku Richtrovi za čas, trpělivost a mnoho užitečných rad mně věnovaných při tvorbě této práce. Dále bych chtěl poděkovat své rodině, přítelkyni a přátelům za podporu v průběhu celého bakalářského studia.

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, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2016 .....

Odkaz na tuto práci

Laube, Daniel. *Babylon: Samoorganizující se parametrický model starověkého města*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Práce se zabývá vytvořením a vizualizací samoorganizujícího se modelu města (Babylonu), s možností využití ve strategické hře, tematicky zasazené do období stavby Babylonské věže. Vytvořený model je optimalizován pomocí evolučního algoritmu. Na základě uživatelem definovaných, nebo náhodných vstupních podmínek (terén města, ekonomické podmínky, počasí), probíhá proces vývoje města, jež je možné ovlivňovat. Vývoj města je řízen samoučícím se modelem reagujícím na zásahy uživatele. Výsledkem je vizualizace celého učícího procesu a zásahů uživatele vhodnou formou, vzhledem k záměru vytvoření strategické hry. Model je vytvořen v prostředí herního enginu Unity 5.

Klíčová slova Unity, umělá inteligence, virtuální model města, vizualizace, genetický algoritmus, evoluční výpočetní techniky

Abstract

This paper is about creation and visualization of self-organizing model of city (Babylon) with possibility of use in strategic game set into era of Babylon tower construction. Created model is optimized using evolution algorithms.

Based on user defined or random input conditions (terrain, economical conditions, weather) runs process of city development which can be affected by user. Development of city is managed by self learning model which reacts to users actions. Result is suitable visualization of learning process and users actions considering its purpose to be part of strategic game. Model is created using environment of game engine Unity 5.

Keywords Unity, artificial intelligence, virtual model of city, visualization, genetic algorithm, evolution computation techniques

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tato práce se zabývá vytvořením samoorganizujícího se modelu města. Tento model by měl být poté využitelný ve strategické hře, jako dynamické prostředí v kterém bude hra zasazena. Vzhledem k záměru využití modelu, jako dynamického prostředí je vhodné, aby model netvořil pouze změť budov blízko sebe, ale aby měly rozumné rozložení. Strukturou měst se zabýval například Ernest Burgess, jehož model města [26] je rozdělen do zón, každé specifické jiným typem budov. Centrální část tvoří obchodní čtvrť a zbytek města lze rozdělit na zóny soustředných mezikruží. V těchto mezikružích se směrem od centra nachází zóny: průmyslová, obytná dělnické třídy, obytná zóna vyšší třídy a zóna dojíždějících.

Problematika samoorganizce, je téma velice obsáhlé a zasahuje do oblasti umělé inteligence, fyziky, chemie, ale i běžného života a mnoha dalších. Samoorganizace je proces, při kterém celkový řád vyvstane z interakce jeho nezávislých menších částí, které původně nebyly uspořádané [19]. Pro ilustraci je uveden obrázek 0.1 mravenců, kteří postupně naleznou nejkratší cestu. Práce se ovšem zabývá především zjednodušeným modelem používaným například v počítačových hrách a využitím některých dostupných přístupů řešení z oblasti umělé inteligence. Součástí je tedy návrh i implementace modelu. V této práci je k optimalizaci parametrů modelu testováno několik variant evolučního algoritmu. Jednotlivé varianty se liší především ve zpracování fenotypů, jejich selekce a mutace. Součástí práce je i grafická vizualizace celého procesu. Proces je vizualizován ve 3D a stylizován do low-poly (3D modely tvořené relativně malým množstvím polygonů). Vizualizace je pro tuto práci druhotná a v první řadě je věnován prostor algoritmické části zajišťující chod města a prostředí v němž se vyvíjí.

V ideálním případě, by měl výsledný model být co nejblíže realitě, co do počtu surovin, různých budov, profesí, potřeb dělníků a jejich chování. Výsledkem by bylo vlastní život žijící starověké město se vším, co k němu patří. Obsahoval by řadu profesí nutných pro fungovaní města přes rolníky, kováře, pekaře, alchymisty, lékaře, obchodníky, vojáky a mnohé další. Každá

Obrázek 0.1: Mravenci hledající nejkratší cestu

z profesí by se aktivně podílela na podobě modelu.

Profese by zastávali dělníci chovající se co nejpodobněji opravdovému člověku. Každý dělník/agent by tak měl různé potřeby, které by musel naplňovat a různé preference v naplňování těchto potřeb. Například jeden raději večeří chleba s klobásou, druhý polévku a třetí pečené kuře. Každý by byl na něco trochu šikovnější a to by ovlivňovalo jakou profesi bude zastávat a jak se mu v ní bude dařit.

Každá věc by měla stanovený co nejreálnější způsob získání. Ke stavbě budovy by tak nestačilo jen kácet stromy, ale dále je i zpracovávat do použitelného materiálu. Případně získávání jídla by nebylo omezeno na získávání generického jídla, jako tomu je v některých hrách, kde ze srnky, keře s bobulemi i pole dostáváme tutéž surovinu. Ale zvlášť by stála každá zahrnutá potravina a způsob jejího získání. Například pokud by chtěl dělník uspokojit hlad, bude si moci koupit chleba. Chleba samotný je ovšem až výsledným produktem řady procesů. Tato řada (zde oproti realitě stále zjednodušená) by se mohla sestávat z vypěstování obilí, jeho zpracování na mouku a teprve poté upečení chleba.

Takovýto ideální případ by byl nejen pro potřeby hry zbytečně obsáhlý a detailní, ale i velice náročný na implementaci. Požadavky na model budou tedy muset být zredukovány natolik, aby implementace v daném čase, s danými zdroji, byla reálná, ale zároveň postačovala pro požadovaný účel.

Rozbor zadání

1. Analyzujte problematiky modelů pro samoorganizující se rozvoj urbanistické scény.

    - Součástí tohoto bodu je seznámení se strategickými počítačovými hrami, výčet použitelných metod z oblasti evolučních algoritmů a modelů používaných v počítačových hrách a dostupnou literaturou, věnující se těmto oblastem. Více informací se nachází v kapitole Rešerše 1.

2. Definujte na základě požadavků zadavatele soubor pravidel (model) prostředí, podle kterých se bude město rozvíjet.

    - Požadavky na model se budou týkat jak dostupných surovin, budov a jednotek, tak způsobu hledání nejvhodnějších parametrů ovlivňujících chování modelu. Podstatnou částí modelu je řetězec parametrů ovlivňujících chování modelu. Hodnoty částí tohoto řetězce jsou modifikovány při hledání nejvhodnějšího nastavení parametrů modelu, jiné jsou záměrně konstantní. Metody a modely z kapitoly Rešerše 1 jsou zhodnoceny v kapitole Analýza 2.

3. Pomocí metod softwarového inženýrství navrhněte prototyp aplikace simulující samoorg. rozvoj urbanistické scény s ohledem na jeho využití pro strategickou hru a to včetně vhodného rozhraní.

    - Návrh prototypu je přizpůsoben možnostem Unity 5 s využitím jejich nástrojů. V kapitole Návrh 3 je popsána podoba a požadavky na výsledný model s využitím vybraných metod z kapitoly Analýza 2.

4. Implementujte prototyp vizualizace modelu města pomocí engine Unity.

    - Vybrané části výsledné implementace jsou popsány v kapitole Implementace 4. Mezi tyto části patří diagramy popisující chování dělníků, algoritmy generování terénu a surovin, konkrétní podoby testovaných evolučních alogoritmů a jejich výsledky a grafy vývoje populace z vybraných běhů.

5. Implementovaný prototyp podrobte vzhledem k účelu aplikace vhodným testům.

    - Testy jsou směřovány nejen na bezchybný běh algoritmu, ale také na výslednou podobu a chod města a také evoluční algoritmy. Výsledky testů jsou součástí kapitoly Implementace 4.

Kapitola 1

Rešerše

Kapitola Rešerše je rozdělena na čtyři podkapitoly.

- První podkapitola 1.1 se věnuje kromě samoorganizujícím modelům v počítačových hrách, také historii počítačových her a přístupům určování vývoje, které modely používají.
- Druhá podkapitola 1.2 se zabývá Použitelnými metodami z oblasti umělé inteligence, evolučními výpočetními technikami, typy prostředí a typy agentů a jejich rozhodování.
- Ve třetí podkapitole 1.3 jsou popsány používané modely prostředí, včetně používaných surovin, budov a jednotek.
- Čtvrtá podkapitola 1.4 je zaměřena na použitelné platformy.

1.1 Samoorganizující modely v počítačových hrách

Dle Helbinga [5] se samoorganizující agentní modely v počítačových hrách od modelů s vědeckým zaměřením liší v několika bodech. Zatímco vědecky zaměřené modely si kladou za cíl, aby jejich vnitřní mechanismy byly co nejpodobnější realitě a byly tak schopné co nejpřesněji odhadnout výsledek experimentu nebo dané situace v reálném světě. Tak modely využívané v počítačových hrách si zakládají především na realistickém dojmu z výsledku, byť některé vnitřní mechanismy a rozhodnutí nejsou dostatečně opodstatněny. Takovéto modely vytržené z kontextu dané hry, proto nebývají použitelné. Dalším, co do funkčnosti, méně podstatným rozdílem je vizuální zpracování těchto modelů. Hry snažící se o realistický dojem mají propracovanější vizuální stránku. Agentní modely tvořené pro vědecké účely tuto část často záměrně zjednodušují a zaměřují se pouze na důležité vlastnosti vizualizovaných objektů. Naopak Mandelbrot [27] tvrdí, že modely generující prostorové nebo fyzikální předpovědi musí také “vypadat dobře”.

1.1.1 Historie strategických her

Tato kapitola čerpá z [1]. Potřeba nějaké umělé inteligence, schopné ovládat a rozvíjet město v prostředí počítačových her, vznikla s příchodem prvních strategických her, tedy v průběhu 70. let minulého století. Hry byly inspirovány často hlavně již existujícími hrami, jako například šachy, nebo Go. Mezi první z nich patří strategická tahová[1] hra Invasion z roku 1972. Tato hra je velmi podobná deskové hře Risk!. Nejedná se ovšem o čistě počítačovou hru a k jejímu hraní byla potřeba hrací deska a další herní předměty. Počítač byl využíván jen k soubojům při dobývání pevností. Tato hra však stále nenabízela virtuálního protihráče, pouze virtulání souboje o dobytí pevností [2].

Mezi první hry, které nabízely výzvu v podobě virtuálního protihráče patří počiny firmy Strategic Simulation, Inc. a to Computer Ambush, Computer Bismarck nebo Computer Conflilict určené pro platformy Apple II, Commodore 64 či TRS-80. Hry byly stále tahové, nicméně již zcela virtuální, bez potřeby hrací desky, nebo jiné hrací pomůcky, kromě počítače. Nabízely možnost vžít se do různých historických období, ve kterých se odehrávaly velké válečné konflikty.

Velkým milníkem strategických her je příchod titulu Herzog Zwei (koncem roku 1989). Herzog Zwei je totiž považován za první realtime strategii[2]. Nutno však podotknout, že prvky realtime strategie se objevovaly již v dřívějších titulech například Cytron Masters (1982), Utopia (1982) nebo Herzog (1989). Realtime strategie přinesly další výzvu pro umělou inteligenci, protože bylo nutné, aby byla schopna reagovat okamžitě a nemohla si brát čas navíc, jako doposud. Mezi největší průkopníky nového žánru realtime strategie patří titul z roku 1992 s názvem Dune II. Postupem času tento žánr převládl nad tahovými strategiemi, které si však stále najdou své příznivce i v dnešní době a nic nenasvědčuje tomu, že by tento žánr zcela zanikl.

Co do počtu prodaných kopií je za jednu z nejúspěšnějších považovaná série Total War se 7,5 miliony prodaných kopií k roku 2012 [6]. První hrou série vydanou v roce 2000 je Shogun: Total War. Hry série Total War kombinují oba zmíněné přístupy. V jednotlivých tazích hráč plánuje získávání nových zdrojů a území a realtime se odehrávají souboje armád a dobývání pevností.

Mezi další velmi úspěšné série patří například série Age of Empires, Civilization, Command and Conquer nebo Warcraft, s mnoha různými díly, datadisky, rozšířeními a v případě Warcraftu i MMORPG (massively multiplayer online role-playing game, hra více hráčů o velkém množství lidí s RPG prvky) adaptace World of Warcraft.

[1] Tahová strategie – Hráči se střídají na tahu podobně jako v šachách. Hráč může zasahovat do hry jedině pokud je zrovna na tahu.

[2] Realtimová strategie – Všichni hráči hrají zároveň.

1.1.2 Přístupy určování vývoje

V herních titulech můžeme pozorovat různé přístupy k určování směru vývoje hráčova města (základny/osady/...), pro účely této práce jsou rozděleny na tyto dva hlavní přístupy.

- První z nich by se dal shrnout jako přístup založený na úplné kontrole, kdy hráč řeší chování každé jednotky, každé budovy. Jednotky, mají jen velmi základní omezené chování. Mezi hry s tímto přístupem patří například výše zmiňovaný titul Age of Empires, Warcraft, nebo Empire Earth.

- Druhý přístup dává hráči menší kontrolu nad přesným vývojem dění a jednotliví agenti/dělníci, mají více či méně složité samostatné chování. Hráč tedy neovládá každou jednotku zvlášť, pouze některé (nejčastěji vojenské) a rozhoduje spíše o tom, kde bude stát jaká budova, nebo kam potáhne s vojskem. Mezi strategie využívající tento typ ovládání patří například Knights and Merchants nebo Settlers.

S těmito dvěma přístupy a jejich různými modifikacemi a kombinacemi se setkáváme v naprosté většině strategických her. Každý má své výhody a nevýhody, které jsou v následující kapitole Analýza rozebrány.

V obou případech je hráč vázaný pravidly daného modelu, kterými se musí řídit. Mezi tyto pravidla patří jaké suroviny může získávat, jak je může využít, jaké jednotky a budovy používat a mnohé další. Soubor surovin je více či méně obsáhlý, od naprosto jednoduchých, například Command and Conquer, kde jedinou surovinou jsou peníze. Po složité systémy v hrách jako je Knights and Merchants, Stronghold nebo hry série Heroes of Might and Magic, ve kterých hráč musí kontrolovat získávání mnoha různých typů surovin (až desítky různých). Některé suroviny se mohou do jisté míry zastupovat. Příkladem může být ve zmiňovaném Knights and Merchants možnost nasytit jednotky chlebem, vínem, klobásami, nebo rybami. Každá z těchto surovin má svůj specifický způsob opatření a dává tak v každé dané situaci možnost se rozhodnout, kterou bude hráč preferovat.

Podle modelu, nebo daného nastavení hry, musí hráč prozkoumávat své okolí, aby zjistil konkrétní podobu terénu a rozmístění zdrojů. Tento mechanizmus bývá realizován pomocí válečné mlhy. *„Termín válečná mlha se používá pro situace, kdy jedna, nebo obě z válčících stran mají špatný, nebo žádný přehled o situaci protivníka a svá rozhodnutí konají na základě odhadů“* [3]. V případě strategických her pak označuje tu část mapy, kterou neprozkoumal, nebo nemá aktuální informace o dění v dané oblasti. Mechanizmus je často implementován pomocí černých míst na mapě, která se odkryjí v případě přítomnosti jednotek. Jakmile však jednotky dané místo opustí, bývá oblast opět zahalena do mlhy a místo černého místa na mapě hráč vidí poslední známý obraz krajiny.

1.2 Použitelné metody z oblasti umělé inteligence

V této podkapitole jsou popsány základní myšlenky evolučních algoritmů. Dále jsou vysvětleny vybrané základní pojmy z této oblasti. Věnován je prostor především agentům a prostředím v kterých se pohybují.

1.2.1 Evoluční výpočetní techniky a genetické algoritmy

Dle Řehořka [10] jsou evoluční výpočetní techniky rodina metod stochastické populační optimalizace. Inspiraci čerpají z Evoluční biologie, tedy z práce Charlse Darwina a Gregora Johanna Mendela. Fungují na principu "šlechtění" populace kandidujících řešení. Vhodná řešení "rodiče" jsou pak vybrána k reprodukci, během níž může dojít ke křížení a mutaci.

Evoluční algoritmy používají několik pojmů, z nichž několik základních je v této práci vysvětleno.

- Genotyp – reprezentace řešení používána evolučním algoritmem, například vektor bitů
- Fenotyp – konkrétní podoba řešení
- Fitness – označení pro kriteriální funkci optimalizačního problému, která vyjadřuje míru adapatace kandidujícího řešení
- Jedinec – označení pro kandidující řešení
- Populace – množina šlechtěných jedinců
- Generace – čítač hlavních cyklů algoritmu
  - 0. generace populace byla vygenerována operátorem inicializace
  - n. generace populace prošla n cykly selekce, reprodukce a náhrady
- Inicializace – vytvoření počáteční populace
- Selekce – výběr jedinců na základě fitness, výslednou množinu jedinců nazýváme rodiče
- Křížení – výběr částí z vybraných kandidujících řešení a jejich použití k vytvoření nového kandidujícího řešení "potomka".
- Mutace – náhodná, nebo řízená modifikace fenotypu "potomka"
- Reprodukce – proces tvorby potomků z rodičů, typicky křížením a mutací, výsledná množina se nazývá potomci

Genetické algoritmy spadají do rodiny evolučních výpočetních technik. Jsou hojně využívány v mnoha problémech k nalézání vhodných řešení. Mezi než patří strojové učení, vědecké modelování, optimalizace a inženýrství [8]. Genetické algoritmy jsou algoritmy založené na heuristickém adaptivním hledání. Jsou inspirovány myšlenkami evoluce o přirozeném výběru a genetikou. Jako takové představují inteligentní využití náhodného vyhledávání k řešení optimalizačních problémů. Ačkoliv využívají náhodnost, nejsou náhodné, místo toho využívají historickou informaci k nasměřování hledání řešení lepším směrem. Základní techniky genetických algoritmů jsou navrženy k simulování procesů, vyskytujících se v přírodě, nutných pro evoluci, především pak těch následující princip Charlse Darwina "přežití nejzdatnějších" [9].

Genetické algoritmy implementují používané operátory mnoha způsoby z nichž si některé uvedeme.

- Inicializační operátor
  - Neinformovaná selekce – náhodně vygeneruje počáteční populaci jedinců. Tento způsob je nejpoužívanější.
  - Informovaná selekce – pomocí jednoduché heuristiky vychýlý počáteční populaci slibným směrem. Přináší ovšem nebezpečí uváznutí v lokálním minimu.
- Operátor selekce
  - Ruletová selekce – pravděpodobnost výběru jedince je přímo úměrná jeho fitness.
  - Turnajová selekce – náhodně se vylosuje k jedinců a vybereme toho s největší fitness.
- Operátor křížení
  - Křížení probíhá v n bodech fenotypu a získáváme tak dva nové jedince.
  - Tento operátor může být vzhledem k povaze dat vynechán, případně nahrazen specializovaným pro daný problém.

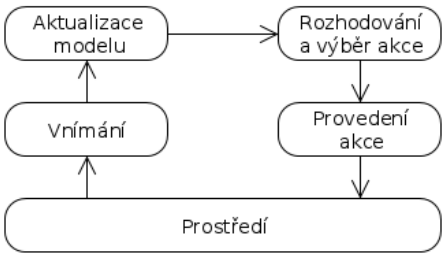
1.2.2 Typy agentů

Různé typy agentů jsou velmi často využívány ve hrách. Mohou představovat hlídajícího vojáka, dělníka nebo i budovu. Pro každý účel se hodí trochu jiné vlastnosti agenta.

Dle Bittnera [7] je pro všechny agenty společný cyklus Prostředí->Vnímání->Aktualizace modelu->Rozhodování a výběr akce->Provedení akce->Prostředí viz diagram 1.1.

Agenty můžeme rozdělit na několik typů:

- Reflexivní [11]
  - Tento agent volí akce pouze na základě aktuálních vjemů, bez využití předchozích zkušeností.
- Model-based [12]
  - Narozdíl od reflexních agentů udržují navíc vnitřní stav, který spolu s aktuálními vjemy používá k volbě akce.
- Goal-based [13]
  - Má seznam cílů a akcí. U každé akce má přiřazený předpokládaný výsledek dané akce. Agent pak volí takové akce, kterými naplní zadaný cíl. To vyžaduje plánování, které předchozí dva typy agentů postrádaly.
- Utility-based [14]
  - Navíc oproti goal-based agentovi přiřazuje jednotlivým řešením efektivitu a vybírá nejefektivnější z dostupných řešení.
- Učící se [15]
  - Oproti utility-based agentovi potřebuje navíc externího hodnotitele dopadu jeho akcí na prostředí, ve kterém se pohybuje. Na základě hodnocení se pak agent může modifikovat, aby dosáhl lepších výsledků.



Obrázek 1.1: Cyklus agenta

1.2.3 Typy prostředí

Agenti i model jsou umístěni v prostředí, ve kterém vykonávají svou činnost. Prostředí má vždy své specifické vlastnosti, kterým by měl být agent přizpůsoben. Prostředí, ve kterých se agenti pohybují, lze rozdělit podle různých kritérií. Patří mezi ně například tyto čtyři[16]:

- Deterministické/Nedeterministické
  - Deterministické prostředí je takové prostředí, jehož stav v následujícím kroku závisí pouze na činnosti agentů.
- Epizodické/Neepizodické
  - V epizodickém prostředí jsou agentovy zkušenosti a akce rozdělovány do nezávislých epizod. Tím, že jsou epizody nezávislé, nemusí agent znát následky svých akcí.
- Statické/Dynamické
  - Dynamické prostředí je takové, které se mění i v průběhu rozhodování agenta.
- Diskrétní/spojité
  - Pokud je počet vjemů a akcí v daném prostředí konečný, je dané prostředí diskrétní.

1.2.4 Rozhodování agentů

Dle Bittnera [7] je rozhodování agentů několik typů. Pro představu jsou uvedeny tyto tři:

- Rozhodovací strom
  - Jsou rychlé, snadno implementovatelné a jednoduché na pochopení. V základní podobě jsou jedním z nejjednodušších řešení, ovšem různými rozšířeními se mohou stát poměrně sofistikovanými. Rozhodovací strom může být použit pro učícího agenta. V současné době to je však ve hrách poměrně vzácné a to i přes dobrou rychlost učení [21].
- Konečný stavový automat
  - Konečné stavové automaty berou v úvahu jak vjemy prostředí, tak svůj vnitřní stav. Jsou tak vhodné například pro NPC reprezentující hlídkujícího vojáka[22].

- Fuzzy logic
  - Oba předchozí přístupy měly přesně nastavené podmínky, podle kterých se chovaly. Přístup fuzzy logic je soubor matematických technik navržených k vypořádáním se s šedými zónami. Tento přístup dává možnost vytvořit plynulejší přechod mezi jednotlivými stavy agenta [23].

1.3 Používané modely prostředí v počítačových hrách

Na základě pozorování lze říci, že naprostá většina herních titulů se snaží vytvořit svůj vlastní model prostředí, který se více či méně liší od ostatních. Tyto modely, ve většině případů, zahrnují správu zdrojů, jak surovin, tak jednotek, různé mechanismy a logiky dané hry a jiná pravidla určující vlastnosti tohoto modelu. Relativnost času je v těchto modelech velmi často až absurdní. Dělníci staví hrady během několika dní, nebo žijí i více jak sto let, bez újmy na pracovním nasazení. Modely se velmi prolínají a čerpají ze sebe navzájem inspiraci. A tak, i kdybychom chtěli, tyto modely roztřídit do jednotlivých kategorií, jen těžko bychom hledali zástupce spadajícího jen do jedné kategorie. I přesto v této práci jsou zmíněny některé vlastnosti typické pro větší skupinu modelů.

1.3.1 Suroviny

Jednou z vlastností modelů je spektrum dostupných surovin. To se mezi modely často liší, byť můžeme pozorovat, že některé suroviny jsou častěji použité než jiné. Mezi často používané suroviny patří například peníze[3], jídlo[4] nebo zlato[5]. Mezi málo používané pak třeba síra[6], znalosti[7] nebo úsně[8].

Kromě typů surovin je často rozdíl v jejich počtu. Některé modely spoléhají pouze na jedinou surovinu například Command and Conquer: Generals, jiné naopak na desítky různých surovin například Knights and Merchants.

Zdroje surovin pak mohou být vyčerpatelné, případně jejich alternativní způsob získávání nevýhodný.

1.3.2 Budovy

Budovy lze rozdělit podle různých kritérií. Jednu z možností předkládá Age of Empires II, kde jsou budovy rozděleny na ekonomické a vojenské.

[3] Peníze – Command and Conquer: Generals
[4] Jídlo – Rise of Nations, série Age of Empires
[5] Zlato – Age of Mythology, Empire Earth
[6] Síra – Heroes of Might and Magic
[7] Znalosti – Rise of Nations
[8] Usně – Knights and Merchants

Spektrum budov na rozdíl od surovin bývá v naprosté většině titulů poměrně pestré. Hlavní rozdíl mezi jednotlivými tituly je tedy spíše v zastoupení v jednotlivých kategoriích budov.

1.3.3 Jednotky

Spektrum jednotek je podobně jako budov také poměrně pestré. Modely se mezi sebou liší například ve specializaci jednotek. Zatímco některé modely mají větší množství specializovaných jednotek, schopných vykonávat pouze omezené množství úkonů, jiné naopak využívají univerzálních jednotek schopných zastat více úkonů.

Některé modely vyžadují údržbu jednotek například v podobě stravy, v jiných modelech stačí naopak zaplatit za jednotku jednou a dál již hráče nic nestojí.

1.4 Použitelné platformy

Použitelných platforem je v dnešní době nepřeberné množství. V této práci je zmíněno pouze několik z nich.

Jednou z možností je vytvoření vlastní sofistikovanější platformy vybudované nad některou již dostupnou OpenGL, GLUT nebo DirectX.

Další možností je využití sofistikovanějšího nástroje například herního enginu. V této práci jich je uvedeno pouze několik.

- Unreal Engine 4
  - Zdarma dostupný pro nekomerční účely. Má poměrně silnou základnu vývojářů sestávající z indie developerů, studentů, ale i profesionálních herních vývojářů.
- CryENGINE 3
  - Po zakoupení licence má uživatel přístup ke zdrojovému kódu. Dle Hapaly [17] dokumentace v některých částech není dostatečná a vývojářská fóra nejsou příliš obsáhlá.
- Unity 5
  - Jsou dostupné verze Personal a Professional. Personal verze je zdarma dostupná. Professional verze však nabízí prioritní řešení chyb a některé další služby navíc. Podoně jako Unreal Engine 4 má silnou základnu vývojářů čítající 4,5 milionu uživatelů [18].

Kapitola 2

Analýza

Kapitola Analýza se vrací k tématům kapitoly Rešerše 1, tentokrát se však zaměřuje na zhodnocení zmiňovaných řešení.

- První podkapitola 2.1 je věnována samoorganizujícím modelům v počítačových hrách a přístupům k určování vývoje.
- Ve druhé podkapitole 2.2 jsou zhodnoceny použitelné metody z oblasti umělé inteligence. Patří mezi ně evoluční výpočetní techniky, typy agentů a jejich rozhodování a typy prostředí.
- Třetí podkapitola 2.3 je věnována používaným modelům prostředí počítačových her, surovinám, budovám a jednotkám.
- Poslední podkapitola 2.4 hodnotí použitelné platformy.

2.1 Přístupy určování vývoje v samoorganizujících modelech v počítačových hrách

V této podkapitole jsou zhodnoceny oba přístupy určování vývoje zmíněné v Rešerši 1. Je zde rozebrán přístup založený na absolutní kontrole a druhý založený jen na částečné.

Nejprve zhodnocení přístupu založeného na absolutní kontrole nad jednotkami a budovami. Na první pohled je zřejmé, že první přístup má velkou výhodu v krátkém reakčním čase a lepší organizaci jednotek, dle hráčovi vůle. Hráč může snáze usměrňovat úsilí svých jednotek směrem, který uzná za vhodné. To je ovšem za cenu toho, že žádná jednotka sama od sebe většinou neprojeví žádné úsilí směrem k vítězství, pokud ji to hráč nepřikáže. Tím pádem se může stát, že jednotky, které by se mohly aktivně podílet na vývoji, jsou pasivní. Zadávané úkoly hráčem jednotce, mohou být různého rozsahu, od poměrně krátkých, například přesun z bodu A do bodu B, po někdy i celoživotní například práce na poli. Tento přístup je typický pro méně rozsáhle

modely, jak do počtu surovin a budov, tak i jednotek. Do této kategorie spadají hry jako například Warcraft III, Rise of Nations nebo Empire Earth II.

Výhodu druhého přístupu je, že každá jednotka má svůj cíl, který se snaží naplnit a jedná více, či méně samostatně. Hráč ji tak musí hlavně vytvořit prostředí v kterém může fungovat a ona se postará o zbytek. Nehrozí tak, že pokud bude moci vykonávat nějakou aktivitu, tak ji vykonávat nebude. Největší úskalí je, jak již bylo zmíněno, pomalejší reakční čas. Toto úskalí je o to větší pokud nejsou jednotky šikovně naprogramovány. Může tak docházet ke zbytečným ztrátám při obraně nebo jiným potížím. Tento přístup je typický pro obsáhlejší modely se složitější strukturou. Zástupci této kategorie jsou například Settlers, Stronghold nebo Knights and Merchants.

2.2 Použitelné metody z oblasti umělé inteligence

V této podkapitole jsou zhodnoceny metody z oblasti umělé inteligence, především pak evoluční výpočetní techniky. Dalším tématem této kapitoly jsou agenti. Jsou zde zhodnoceny jednotlivé typy agentů a jejich rozhodování.

2.2.1 Evoluční výpočetní techniky a genetické algoritmy

Evoluční výpočetní techniky se prokázaly jako mocný nástroj hledání co nejoptimálnějších řešení nejrůznějších problémů. Dobrým příkladem je práce Thomase Geijtenbeeka [20], který využívá evoluční programování k animování pohybu virtuálních postav.

2.2.2 Typy agentů

Reflexivní agenti jsou pro svou jednoduchou rozhodovací strukturu složenou z konkrétních podmínek a jim přiřazených reakcí vhodní pouze pro méně náročné úkoly. Tedy pro tuto práci nevhodní [11].

Model-based agent si je již schopen uložit stav do kterého se dostal po poslední akci. Díky uloženému stavu jsou do jisté míry schopni odhadnout následek svých akcí [12]. Takovýto agent je již o něco blíže potřebám této práce, nicméně stále nevhodný.

Goal-based agent také není vhodným řešením. Důvodem je absence vyhodnocování efektivity nalezených řešní vedoucích k jeho cíli [13]. Dělník, který dobrovolně dělá věci zbytečně složitě by neobstál.

Zbývají tedy poslední dvě možnosti utility-based agent a učící se agent.

2.2.3 Rozhodování agentů

- Rozhodovací strom
  - Rozhodovací strom nevyžaduje žádnou pamět navíc a jeho časová náročnost je lineární vůči počtu navštívených uzlů. Pokud budeme předpokládat, že každé rozhdnutí zabere konstantní množství času a rozhodovací strom je vyvážený, pak je jeho časová náročnost $O(\log_2 n)$. [21]
- Konečný stavový automat
  - Konečný automat vyžaduje pouze paměť k udržení spuštěného přechodu a současného stavu. Jeho paměťová náročnost je $O(1)$ a časová $O(m)$, kde m je počet přechodů na stav. Algoritmus volá ostatní funkce v stavových a přechodových třídách. Ve většině případů je nejvíce času stráveno právě prováděním těchto funkcí.
  - I přestože jsou snadno implementovatelné, jsou známé složitou správou a konečné automaty ve hrách mohou být velmi obsáhlé [22].
- Fuzzy logic
  - Algoritmus má $O(m+n)$ paměťovou složitost, kde n je počet stavů na vstupu a m je počet výstupních stavů. Mimo algoritmus je nutné uložit pravidla. To vyžaduje $O(\prod_{k=0}^{i} n_k)$, kde n_i je počet stavů na vstupu na proměnou a i je počet vstupních proměnných.
  - Velkou slabinou je absence škálovatelnosti. Funguje dobře pro malý počet vstupních proměnných a malým počtem stavů na proměnou. Pro zvládnutí 10 vstupních proměnných, každou s 5 stavy by vyžadovalo téměř 10 milionů pravidel, což dalece přesahuje naše možnosti [23].

2.3 Používané modely prostředí

Vzhledem k využití principů samoorganizace, je vhodnější pojmout model způsobem založeném na menší kontrole nad každou jednotkou. Každý dělník je v takovémto pojetí nezávislým malým činitelem přispívajícím k celkovému řádu systému. Roli hráče pak převezme centrální regulující algoritmus, který stejně jako hráč, bude mít jen omezenou kontrolu nad dělníky.

Nabízí se například možnost, že by vliv tohoto algoritmu na dělníka se vzdáleností od jeho centra klesala. Jeho centrum by bylo reprezentováno nějakou budovou, třeba radnicí. Pro účely této práce pojem radnice označuje nejen budovu, ale i centrální regulující algoritmus.

Další možností, jak může algoritmus nepřímo ovládat dělníky, je vytvoření potřeby peněz, které dělník získá za provedenou práci pro radnici. Radnice vypisuje pracovní nabídky pro dělníky. Dělník, který chce získat něco za peníze, tak musí vykonat nějakou práci, třeba jít kácet dříví a až poté si může koupit víno/maso/... Radnice tak opět má možnost nepřímo dělníka kontrolovat a usměrňovat jeho chování potřebným směrem.

2.4 Použitelné platformy

Použitá platforma byla specifikována již v zadání. Volba byla učiněna na základě autorových znalostí této platformy, dobré dokumentace a početné komunity uživatelů.

Kapitola 3

Návrh

V kapitole Návrh jsou popsané části navržené pro výslednou implementaci.

- První podkapitola 3.1 se věnuje funkčním a nefunkčním požadavkům na model. Dále jsou zde rozepsány případy užití modelu.
- Druhá podkapitola 3.2 je věnována mapě, především pak její struktuře a terénu.
- Třetí podkapitola 3.3 se zabývá přístupem k ovládání jednotek.
- Čtvrtá podkapitola 3.4 popisuje pojetí času v tomto modelu.
- Pátá až sedmá podkapitola jsou zaměřeny na suroviny 3.5 a jejich získávání, budovy 3.6, dělníky 3.7 a jejich potřeby a jejich uspokojování.
- Poslední podkapitola 3.8 obsahuje možné podoby evolučního algoritmu.

3.1 Požadavky

3.1.1 Funkční požadavky

- Výsledný model je použitelný jako dynamické prostředí pro strategickou hru.
- Uživatel má možnost měnit nastavení modelu a tím i jeho chování. Toto nastavení je možné provést, jak před spuštěním simulace, tak v jejím průběhu.
- Model umožňuje spuštění evolučního algoritmu s cílem optimalizovat nastavení parametrů modelu.
- Parametry evolučního algoritmu bude možné nastavit, před spuštěním simulace v hlavním menu.

- Model bude obsahovat vhodné grafické rozhraní, které bude poskytovat informace o aktuálním stavu modelu, budov a oblastí na mapě. Dále bude poskytovat přístup k nastavením modelu.
- Proces vývoje města bude vhodně vizualizován.
- Součástí vizualizace bude terén, zdroje surovin, budovy a dělníci.
- Terén bude možné obarvit dle zadaného parametru, například vzdálenosti od konkrétního typu suroviny, budovy, nebo úrodnosti půdy.
- Bude možné se pohybovat po mapě a pozorovat vybranou oblast mapy.
- Model bude ukládat vybraná data do databáze. Data jako stav populace v daném čase, výslednou výšku věže, nastavení parametrů a jejich výslednou fitness.

3.1.2 Nefunkční požadavky

- Model bude implementován pomocí enginu Unity 5.
- Zpracování vizualizace bude v 3D.
- K ukládání parametrů modelu bude využito databáze SQLite.
- Systém bude schopný vizualizovat alespoň 100 agentů při průměrných 30 fps (frames per second, počet snímků za sekundu) při konfiguraci hardwaru (nebo podobně výkonné):
  - Intel Core i5-3210M 2,5GHz
  - NVIDIA GeForce GT640M, 2048MB
  - 6 GB DDR3
- Výsledný model bude spustitelný na platformě Windows.

3.1.3 Případy užití

V této podkapitole jsou rozebrány případy užití modelu. Nejprve je uveden diagram případů užití 3.1 a poté je každému případu věnována tabulka 3.1 3.2 3.3, v které je detailněji popsán.

Tabulka 3.1: Případ užití – Spustit simulaci

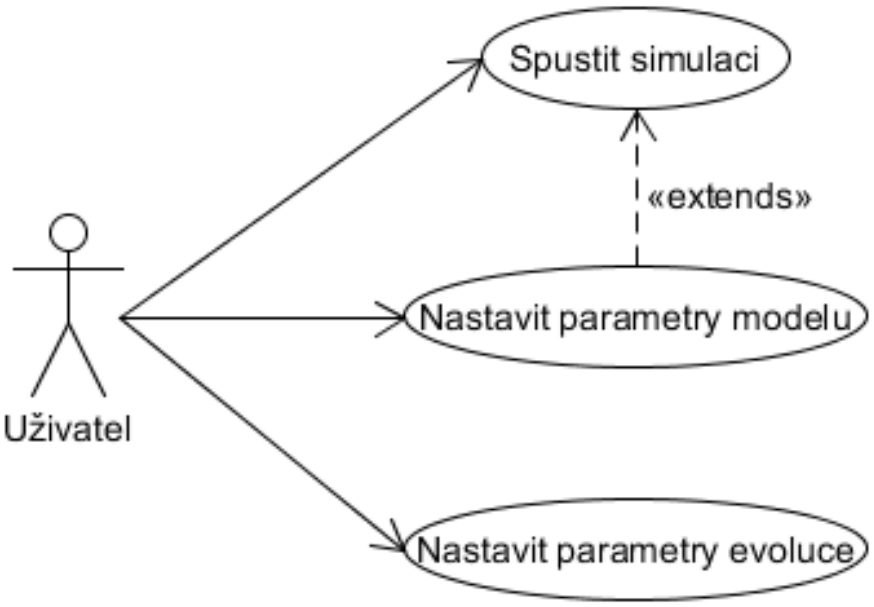
Název případu užití	Spustit simulaci
Cíl případu užití	Spustit simulaci běhu modelu
Primární aktér	Libovolný uživatel
Vstupní podmínky	Uživatel se nachází v hlavním menu
Výstupní podmínky	Simulace je spuštěna, dle zadaných podmínek
Základní scénář	Uživatel spustí program. Nastaví parametry simulace a spustí simulaci.

Tabulka 3.2: Případ užití – Nastavit parametry modelu

Název případu užití	Nastavit parametry modelu
Cíl případu užití	Nastavit parametry ovlivňující chování budov, dělníků a světa
Primární aktér	Libovolný uživatel
Vstupní podmínky	Uživatel se nachází v hlavním menu, nebo ve spuštěné simulaci
Výstupní podmínky	Parametry modelu jsou nastaveny dle uživatelových představ
Základní scénář	Uživatel spustí program, nebo simulaci a v nabídce si otevře menu, v kterém nastaví požadované parametry. Změny pak může uložit, nebo stornovat.

Tabulka 3.3: Případ užití – Nastavit parametry evoluce

Název případu užití	Nastavit parametry evoluce
Cíl případu užití	Nastavení parametrů evolučního algoritmu
Primární aktér	Libovolný uživatel
Vstupní podmínky	Uživatel se nachází v hlavním menu
Výstupní podmínky	Parametry evoluce jsou nastaveny dle představ uživatele
Základní scénář	Uživatel v hlavním menu vybere počet iterací, maximální čas každé z nich, pravděpodobnost mutace a její míru. Dále pak nastaví jaké vstupní fenotypy se mají použít pro šlechtění. Má tři možnosti, první možností je parametry nastavit uživatelem, druhou je použití defaultních, třetí pak použití vybraných pomocí operátoru selekce.



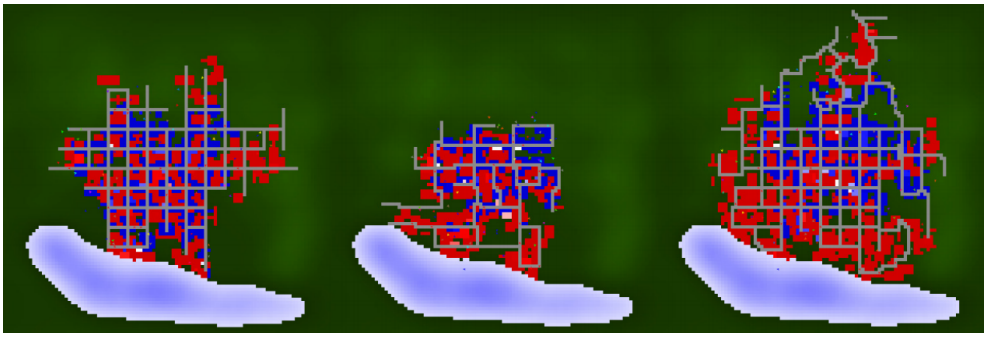
Obrázek 3.1: Usecase diagram

3.2 Mapa

3.2.1 Struktura mapy

Mapa je rozdělena na čtverce. Každý čtverec je stejně velký a velikost strany čtverce slouží v této práci jako základní jednotka délky. Vzdálenost dvou bodů je vypočítána Eukleidovskou metrikou. Eukleidovská metrika je definována takto: vektory $\vec{a}$ a $\vec{b}$ obsahují souřadnice bodů jejichž vzdálenost $m_e(\vec{a}, \vec{b})$ lze výpočítat vzorcem $m_e(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n}(a_i^2 - b_i^2)}$. Budovy mohou být umisťovány jedině v souladu s čtvercovou mřížkou. Čtvercová mřížka omezuje možné podoby města. Není například možné budovy rotovat a umisťovat jinak než je předem definováno. Pro lepší představu je uveden ilustrační obrázek 3.2. To neplatí pro dělníky, kteří nejsou mřížkou vázáni a mohou se volně pohybovat.

Mapa je dále rozdělena na větší čtverce sloužící k určení znalostí terénu každého dělníka, pro účely práce nazvané znalostní čtverce. Role znalostních čtverců v životě dělníka je podrobněji rozepsána v kapitole věnované dělníkům 3.7. Každý znalostní čtverec má stranu velikosti 16 jednotek. V každém znalostním čtverci se nachází oblast, jejímž navštívením dělník získá znalost tohoto čtverce. Oblast pro přidání znalostního čtverce, má podobu čtverce o velikosti strany 8 jednotek. Tato oblast je umístěna uprostřed znalostního čtverce.



Obrázek 3.2: Výslednou strukturu popisují nejlépe levé a prostřední město[28]

3.2.2 Terén

Podoba terénu je v Unity 5 určována výškovou mapou. Výšková mapa je jednokanálová textura ve formátu raw, ideálně velikosti $1 + 2^n$ pixelů na výšku i na šířku. Tato textura se přiřadí komponentu Terrain, který pak na jejím základě vypočítá výšky bodů terénu. Výsledná výška terénu v daném bodě je určena tímto vzorcem:

- $F(f) = (f/255)h_m$
  - f – číslo z intervalu $< 0, 255 >$ určené barvou pixelu ve výškové mapě odpovídající danému bodu v terénu
  - h_m – maximální výška terénu, která je nastavená v komponentu Terrain

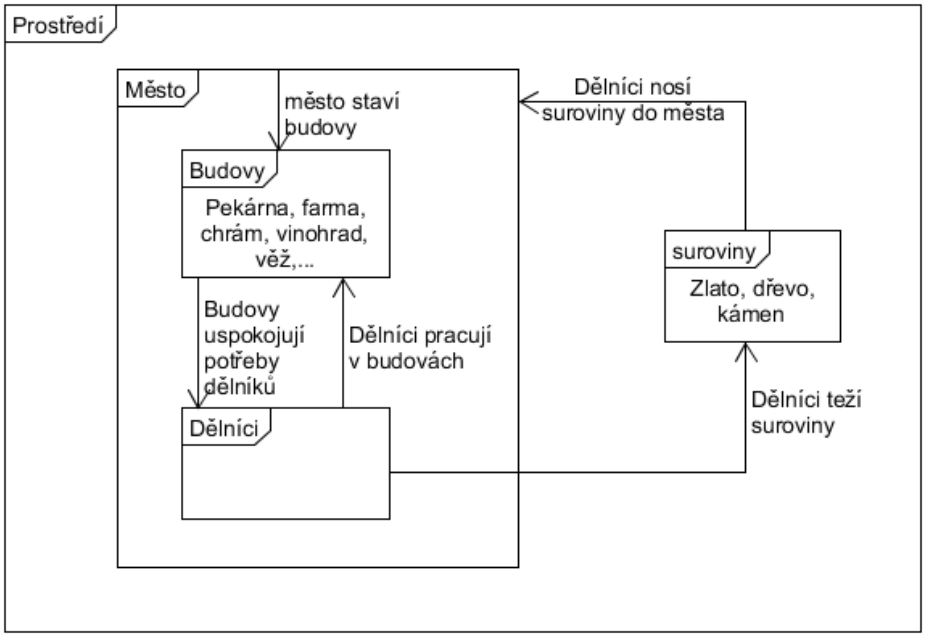
Editovat tuto texturu je možné před spuštěním simulace i v jeím průběhu. K editaci před spuštěním může posloužit vnitřní nástoroj Unity Terrain Editor nebo grafický editor schopný exportu do požadovaného formátu. Za běhu lze terén urpavovat pouze pomocí skriptu.

V této práci je implementován, autorem práce navržený, jednoduchý generátor náhodného terénu. Generátor je schopný vygenerovat řeku podle zadaných parametrů, jako jsou body kterými má protékat, její hloubka, šířka a klikatost. Generátor dále generuje zdroje surovin jako jsou lesy, zlatou rudu a kamenolomy. Více o fungování generátoru, je v kapitole Implementace4.

3.3 Přístup ovládání jednotek

V této práci se kombinují oba zmíněné přístupy (absolutní a částečná kontrola). Agent reprezentující dělníka má svůj vlastní algoritmus, kterým se řídí. Tím ovlivňuje podobu světa, v kterém se pohybuje (více v kapitole Dělníci 3.7). Součástí modelu je algoritmus, vizuálně reprezentovaný jako radnice, který usměrňuje dělníky a výstavbu města.

Provázanost jednotlivých částí modelu popsaných v následujících kapitolách ilustruje tento zjednodušený diagram 3.3.



Obrázek 3.3: Provázanost částí modelu

3.4 Čas

Čas je dle teorie relativity relativní a jinak tomu není ani v tomto modelu. V modelu je zaveden cyklus střídaní dne a noci. Proto jsou k popisu časových úseků zvoleny jednotky dní. Jejich délkou se však nesnaží přiblížit realitě, pouze nastavit pravidla fungování modelu.

3.5 Dostupné suroviny

Pro potřeby práce byly zvoleny následující suroviny:

- Chleba 3.4
- Mouka 3.5
- Obilí 3.6
- Maso 3.7

- Voda 3.8
- Víno 3.9
- Zlato 3.10
- Kámen 3.11
- Dřevo 3.12
- Populace 3.13

V případné navazující práci by seznam surovin mohl být rozšířen blíže k ideální podobě modelu. Nejspíše o další zdroje jídla a víry.

3.5.1 Způsoby získávaní surovin

Každá surovina má definovaný jeden nebo více způsobů získávání. Ty budou v této kapitole probrány.

Budovy, které neslouží k těžení surovin, ale samy nějaké produkují, tak činí periodicky, vždy po uplynutí $\frac{1}{6}$ dne. Produkované množství se řídí vzorcem:

- $a = P_r min(W, R)$
  - P_r – Konstanta určující množství produkované jedním dělníkem z jedné jednotky zpracovávané suroviny
  - W – počet pracujících dělníků v budově
  - R – dostupné množství suroviny potřebné k produkci
  - $min(a, b)$ – vrací menší z čísel a a b

Množství produkované suroviny získávané těžbou (zlato/kámen/dřevo) se odvíjí od množství, doneseného dělníky do zlatého dolu/kamenolomu/pily. Jeden dělník unese jednu jednotku dané suroviny.

Tabulka 3.4: Chleba

Surovina	Chleba
Popis	Chleba je produkován pekárnou. $P_r = 3$

Tabulka 3.5: Mouka

Surovina	Mouka
Popis	Mouka je produkována větrným mlýnem z obilí. $P_r = 3$

Tabulka 3.6: Obilí

Surovina	Obilí
Popis	Obilí produkuje obilná farma. Produkce obilí je závislá na dvou faktorech. Jedním z nich je počet dělníků pracujících na farmě a druhým je úrodnost půdy, na které stojí farma. Produkované množství se řídí tímto vzorcem: • $a = P_r W F_a$ – P_r – Konstanta určující množství produkované jedním dělníkem – W – počet dělníků pracujících na farmě – F_a – průměrná úrodnost půdy na které farma stojí Úrodnost se postupem času snižuje v závislosti na množství vyprodukovaného obilí. Nová úrodnost daného čtverce se vypočte takto: • $U_1 = U_0(1 - (a/(cP_r))/10)$ – U_1 – Nová úrodnost – U_0 – Stávající úrodnost – a – Naposledy vyprodukované množství – c – Celkový počet pracovních míst na farmě $P_r = 3$

Tabulka 3.7: Maso

Surovina	Maso
Popis	Maso je produkováno v řeznictví, které vyžaduje k jeho produkci dodávky obilí. $P_r = 2$

Tabulka 3.8: Voda

Surovina	Voda
Popis	Voda je dostupná ze dvou zdrojů. Prvním z nich je řeka a druhým je studna.

Tabulka 3.9: Víno

Surovina	Víno
Popis	Víno je produkováno ve vinohradu a pro jeho produkci platí stejná pravidla jako pro obilí. $P_r = 2$

Tabulka 3.10: Zlato

Surovina	Zlato
Popis	Zlato je těženo z hor se zlatou rudou. K jeho těžbě je třeba vybrat vhodné místo a postavit zlatý důl, do kterého budou dělníci schromažďovat natěžené zlato. Zásoby zlata jsou pro každou mapu omezené. Dělník za jednu směnu získá jednu jednotku zlata.

Tabulka 3.11: Kámen

Surovina	Kámen
Popis	Kámen je získáván obdobně jako zlato. Těží se z hor a je opět třeba postavit kamenolom sloužící jako schromažďovací bod. Zásoby kamene jsou omezené. Dělník za jednu směnu získá jednu jednotku kamene.

Tabulka 3.12: Dřevo

Surovina	Dřevo
Popis	Dřevo se těží v lese a je schromažďováno na pile. Dřevo je v této verzi práce také omezeným zdrojem. Do navazující práce se počítá s postupnou obnovou lesa a tím i zdrojů dřeva. Dělník za jednu směnu získá jednu jednotku dřeva.

Tabulka 3.13: Populace

Surovina	Populace
Popis	Populace, přesněji počet míst pro bydlení dělníků, je zajišťována výstavbou domů. Jeden dům poskytuje obydlí čtyřem dělníkům.

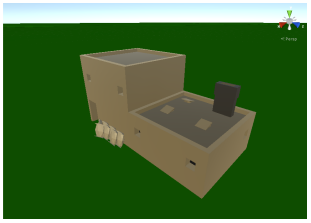
3.6 Dostupé budovy

Dostupných budov je celkem třináct. Každá má svůj specifický účel, jak již bylo předestřeno v předchozí kapitole. Z kapitoly věnované mapě také víme, že budova musí být umístěna v souladu s čtvercovou mřížkou. Každá budova může zabrat jeden, nebo více čtverců.

3.6.1 Přehled budov

Tabulka 3.14: Pekárna

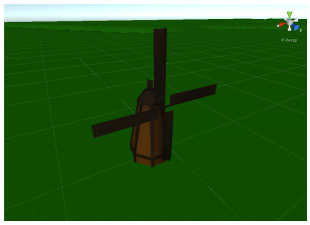
Budova	Pekárna
Popis	Slouží k pečení chleba. Ke svému chodu potřebuje mouku a dělníky. Jedna pekárna poskytuje práci najednou pěti dělníkům.
Velikost	1 × 1 čtverec
Cena	zlato – 5 jednotek kámen – 10 jednotek dřevo – 5 jednotek



Obrázek 3.4: Pekárna

Tabulka 3.15: Větrný mlýn

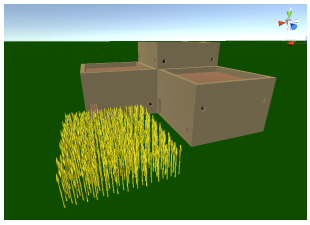
Budova	Větrný mlýn
Popis	Slouží k produkci mouky z obilí. Ke svému chodu potřebuje obilí a dělníky. Jeden větrný mlýn poskytuje práci třem dělníkům.
Velikost	2 × 2 čtverce
Cena	zlato – 5 jednotek kámen – 5 jednotek dřevo – 10 jednotek



Obrázek 3.5: Mlýn

Tabulka 3.16: Obilná farma

Budova	Obilná farma
Popis	Produkuje obilí. Produkce je závislá nejen na počtu pracujících dělníků, jichž je schopna pojmout až deset, ale také na úrodnosti půdy 3.6.
Velikost	2×2 čtverce
Cena	zlato – 5 jednotek kámen – 5 jednotek dřevo – 15 jednotek



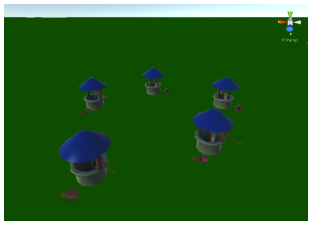
Obrázek 3.6: Obilná farma

Tabulka 3.17: Řeznictví

Budova	Řeznictví
Popis	Produkuje maso. Produkce je závislá na dodávkách obilí a počtu pracujících dělníků.
Velikost	2×2 čtverce
Cena	zlato – 5 jednotek kámen – 10 jednotek dřevo – 10 jednotek

Tabulka 3.18: Studna

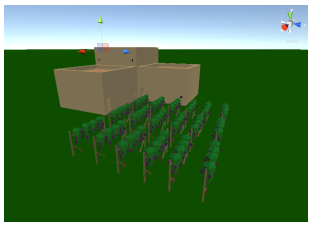
Budova	Studna
Popis	Zdroj vody pro dělníky.
Velikost	1×1 čtverec
Cena	zlato – 0 jednotek
	kámen – 5 jednotek
	dřevo – 5 jednotek



Obrázek 3.7: Studna

Tabulka 3.19: Vinohrad

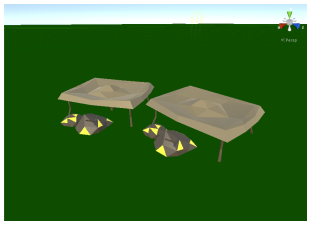
Budova	Vinohrad
Popis	Produkuje víno. Produkce je stejně jako u obilné farmy závislá na úrodnosti půdy 3.6 a počtu pracujících dělníků, kterých zde může najednou pracovat až deset.
Velikost	2×2 čtverce
Cena	zlato – 15 jednotek
	kámen – 5 jednotek
	dřevo – 20 jednotek



Obrázek 3.8: Vinohrad

Tabulka 3.20: Zlatý důl

Budova	Zlatý důl
Popis	Slouží jako schromažďovací bod pro vytěžené zlato.
Velikost	1×1 čtverec
Cena	zlato – 0 jednotek
	kámen – 10 jednotek
	dřevo – 5 jednotek



Obrázek 3.9: Zlatý důl

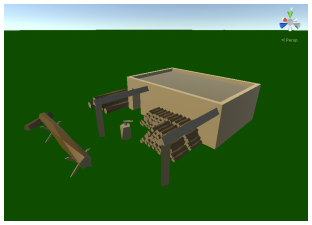
Tabulka 3.21: Kamenolom

Budova	Kamenolom
Popis	Schromažďovací bod pro vytěžený kámen.
Velikost	1×1 čtverec
Cena	zlato – 0 jednotek
	kámen – 0 jednotek
	dřevo – 5 jednotek

Obrázek 3.10: Kamenolom

Tabulka 3.22: Pila

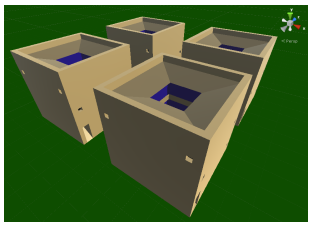
Budova	Pila
Popis	Schromažďovací bod pro vytěžené dřevo.
Velikost	1×1 čtverec
Cena	zlato – 0 jednotek kámen – 0 jednotek dřevo – 5 jednotek



Obrázek 3.11: Pila

Tabulka 3.23: Dům

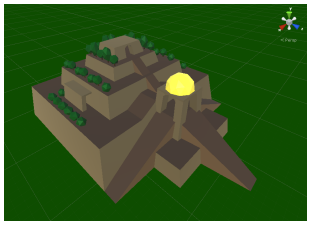
Budova	Dům
Popis	Poskytuje ubytování pro dělníky. Jeden dům poskytuje přístřeší čtyřem dělníkům.
Velikost	2×2 čtverce čtverec
Cena	zlato – 0 jednotek kámen – 0 jednotek dřevo – 10 jednotek



Obrázek 3.12: Dům

Tabulka 3.24: Chrám

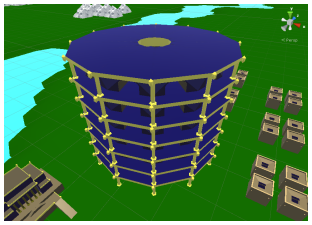
Budova	Chrám
Popis	Umožňuje naplnění potřeby duchovna dělníkům.
Velikost	4×3 čtverce čtverec
Cena	zlato – 30 jednotek kámen – 25 jednotek dřevo – 10 jednotek



Obrázek 3.13: Chrám

Tabulka 3.25: Babylonská věž

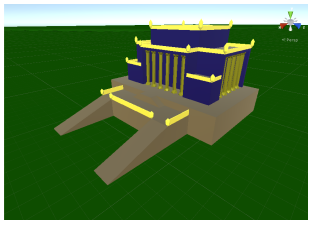
Budova	Babylonská věž
Popis	Stejně jako chrám slouží k naplnění potřeby duchovna dělníků. Navíc je její výstavba jedním z cílů modelu. Věž je jednou ze dvou budov, kterými město disponuje již od začátku běhu. Věž se může vyskytovat v celé mapě pouze jedna. Cena je v tomto případě za patro a postupem času roste. Cena každého dalšího patra je jeden a půl násobek ceny předchozího patra. Uvedená cena je za první patro.
Velikost	4×4 čtverce čtverec
Cena	zlato – 1 jednotek kámen – 2 jednotek dřevo – 1 jednotek



Obrázek 3.14: Babylonská věž

Tabulka 3.26: Radnice

Budova	Radnice
Popis	Radnice je druhou budovou, kterou město disponuje již od začátku běhu. Stejně jako věž, nemůže být postavena žádná další. Slouží především k reprezentaci algortimu regulujícího vývoj města. Toho dociluje jednak rozhodováním o výstavbách budov. Radnice také vytváří úkoly pro dělníky, které pak dělníci plní a přispívají tím k chodu města. Slouží také jako místo zrodu nových dělníků. Přestože v této verzi práce se nepočítá s možností výstavby více radnic, v navazující práci by tomu mělo být právě naopak. I proto je zde stanovena cena této budovy.
Velikost	4×2 čtverce čtverec
Cena	zlato – 25 jednotek kámen – 25 jednotek dřevo – 10 jednotek



Obrázek 3.15: Radnice

3.6.2 Stavba budov

Stavbu budov má na starosti centrální algoritmus regulující chování modelu. Centrální algoritmus na základě několika parametrů rozhoduje která budova bude postavena. Tyto parametry jsou v této verzi tři.

- Suroviny
  - Nejdůležitějším parametrem jsou dostupné suroviny, bez kterých není možné budovu postavit.
- Potřebnost
  - Potřebnost budovy je určená množstvím požadavků dělníků, kteří o budovu mohou požádat, nebo požadavky na získávání surovin.
- Efektivita
  - Efektivita se pro různé typy budov počítá trochu jinak. Budovy jsou pro účely výpočtu efektivity rozděleny na tři typy. První skupinou jsou budovy určené k těžbě surovin a druhou skupinou jsou ostatní budovy produkující suroviny. Třetí skupinou jsou budovy u nichž se efektivita nepočítá, patří sem radnice, věž a domy. Na jejím základě se rozhoduje o nutnosti vybudovat další budovu stejného typu, která již někde stojí. Nutná efektivita pro danou budovu, je předmětem šlechtění.

O umístění budovy se rozhoduje na základě dvanácti parametrů. Ty jsou:

- Místo kde byla budova vyžádána dělníky, nebo poloha radnice
- Úrodnost půdy
- Nebezpečí požáru
- Nebezpečí povodně
- Vzdálenosti od:
  - zdroje kamene
  - zdroje zlata
  - zdroje dřeva
  - vody
  - obilné farmy
  - větrného mlýnu
  - studny
  - budovy stejného typu

Váha těchto parametrů je pro každou budovu individuální a stejně jako nutná efektivita, je předmětem šlechtění. Proces výběru umístění budovy probíha tak, že se zjistí možná umístění budovy a ty jsou následně ohodnocena na základě těchto parametrů. Výsledné hodnocení budovy vznikne takto:

- Parametry které nejsou vztaženy ke vzdálenosti, se vynásobí příslušnou vahou parametru a přičtou k výslednému hodnocení
- Parametry vztažené k vzdálenosti, jsou pak zpracovány podle toho, zda je jejich váha kladná, nebo záporná
- Hodnota přičtená k výslednému hodnocení kladných parametrů se získá tímto vzorcem:
  - $v = (w - d)p/w$
    - w – velikost úhlopříčky mapy
    - d – vzdálenost místa od dané budovy/zdroje suroviny
    - p – váha parametru
- pro zápornou hodnotu váhy parametru je vzorec upraven takto
  - $v = 1 - (w - d) * (-p)/w$
    - w – velikost úhlopříčky mapy
    - d – vzdálenost místa od dané budovy/zdroje suroviny
    - p – váha parametru
- Zpracováním všech parametrů vznikne výsledné ohodnocení, čím vyšší je jeho hodnota, tím je místo vhodnější pro stavbu.

3.7 Dělníci

Dělník je agent, který má své potřeby, které se snaží naplňovat. Pokud se mu stanovenou dobu nebude dařit naplnit některou z vitálních potřeb, tak zemře. Zemřít však nemusí jen z nenaplnění vitální potřeby, ale i stářím. Smrt stářím je náhodně určena v rozmezí mezi 14 až 56 dnů. Slouží k získávání surovin a umožňuje tak rozšiřovat město a budovat věž.

3.7.1 Pohyb dělníků po mapě

Pohyb dělníků po mapě by neměl být přímo vázaný na čtvercovou síť, ale měl by být spojitý.

Znalost prostředí je u každého dělníka individuální a závisí na pohybu dělníka. Dělník tak pro uspokojení potřeb může využívat pouze budovy nacházející se v oblasti, kterou prozkoumal. K rozšiřování známé oblasti dochází při průzkumu terénu, nebo pokud je dělník radnicí vyslán pracovat na jemu dosud neznámé místo. Přidání známé oblasti je vázáno na navštívení znalostního čtverce, čímž dělník získá znalost nové části mapy.

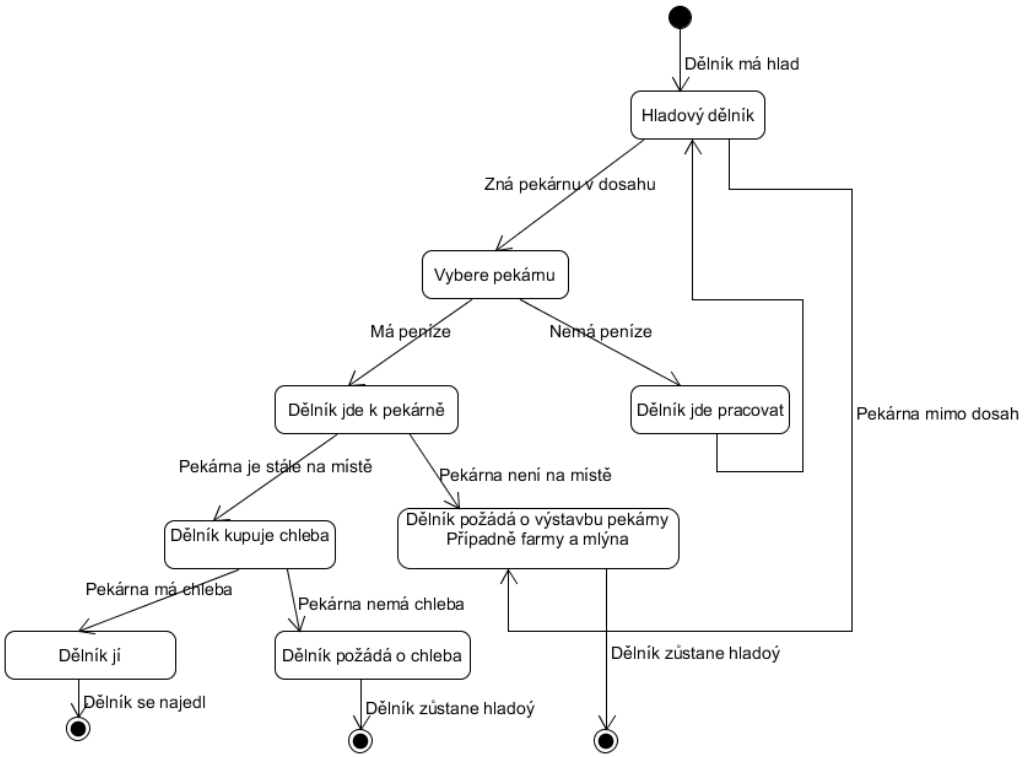
3.7.2 Potřeby dělníků a jejich uspokojení

Pro účely této práce mají dělníci několik potřeb. Potřeby jsou rozděleny na dvě skupiny vitální a sekundární.

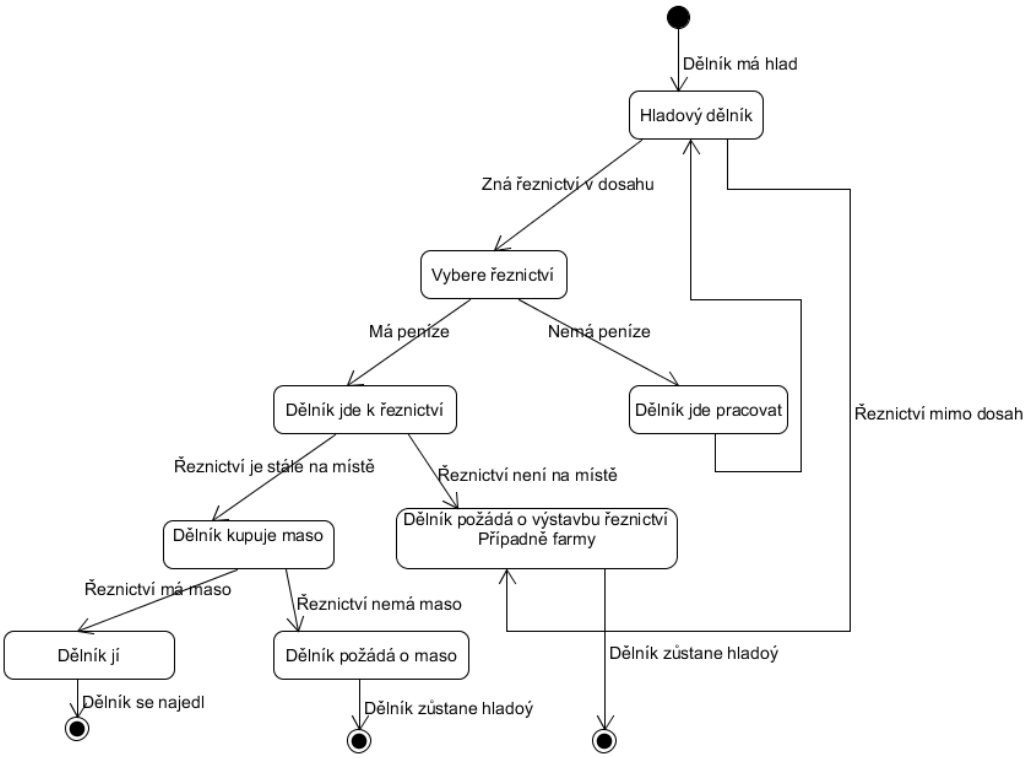
Mezi vitální patří hlad 3.27 a žízeň 3.28. Mezi sekundární pak duchovno 3.29, odpočinek 3.30, průzkum 3.30, práce 3.32 a rozmnožování se 3.33. Každou z těchto potřeb se snaží dělník v určitých časových intervalech uspokojovat. Tento interval je předmětem šlechtění a v případě hladu a žízně má pevně nastavený horní limit, po jehož uplynutí dělník zemře. Pokud dělník potřebuje pro uspokojení potřeby nějakou budovu hrozí, že má již neaktuální informace a budova byla již zničena. Pokud není některá z budov potřebná k uspokojení dělníkovy potřeby v jeho dosahu, požádá o její výstavbu radnici. V následujícím seznamu jsou jednotlivé potřeby detailněji popsány a opatřeny diagramem popisujícím rozhodování dělníka.

Tabulka 3.27: Hlad

Potřeba	Hlad
Popis	Hlad je jednou ze dvou vitálních potřeb. Dělník ho tak musí uspokojovat nejméně jednou za 7 dní. Jak již bylo zmíněno časová realističnost není cílem této práce, nicméně pro přiblížení realitě, lze chápat tento čas jako maximální čas na který si může dělník nakoupit jídlo dopředu. K jeho uspokojení má v této práci možnost koupit si chleba 3.16, nebo maso 3.17.



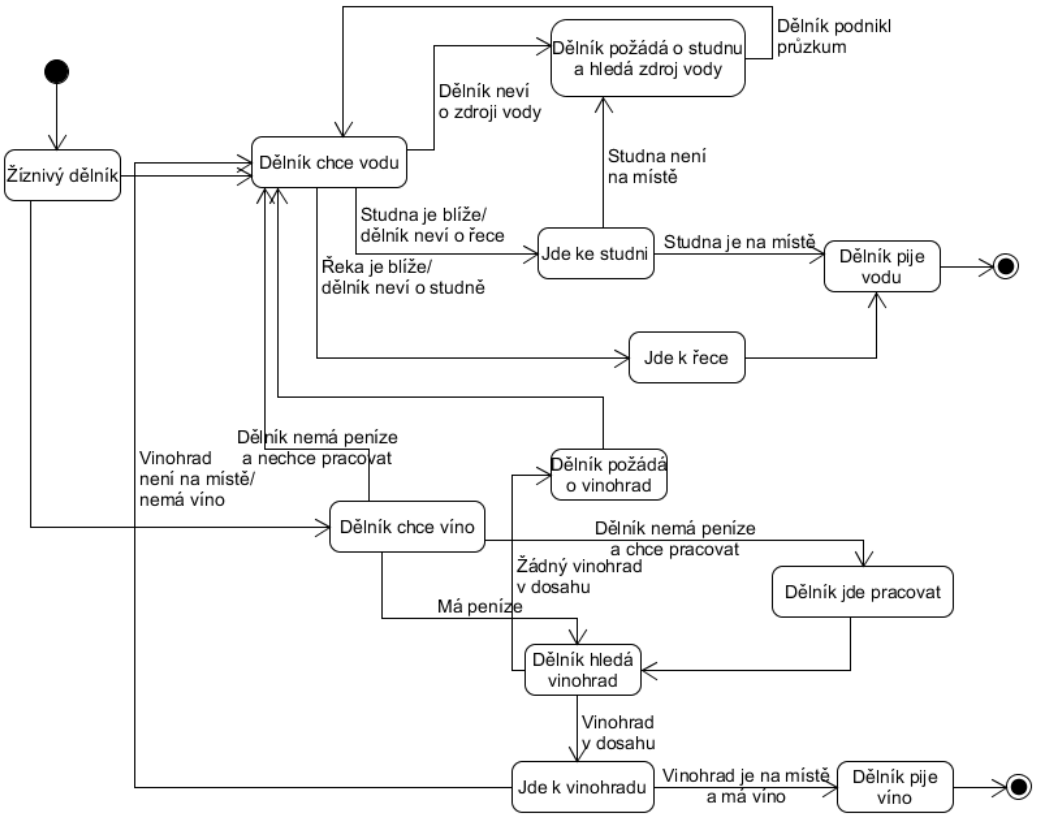
Obrázek 3.16: Diagram uspokojení hladu chlebem



Obrázek 3.17: Diagram uspokojení hladu masem

Tabulka 3.28: Žízeň

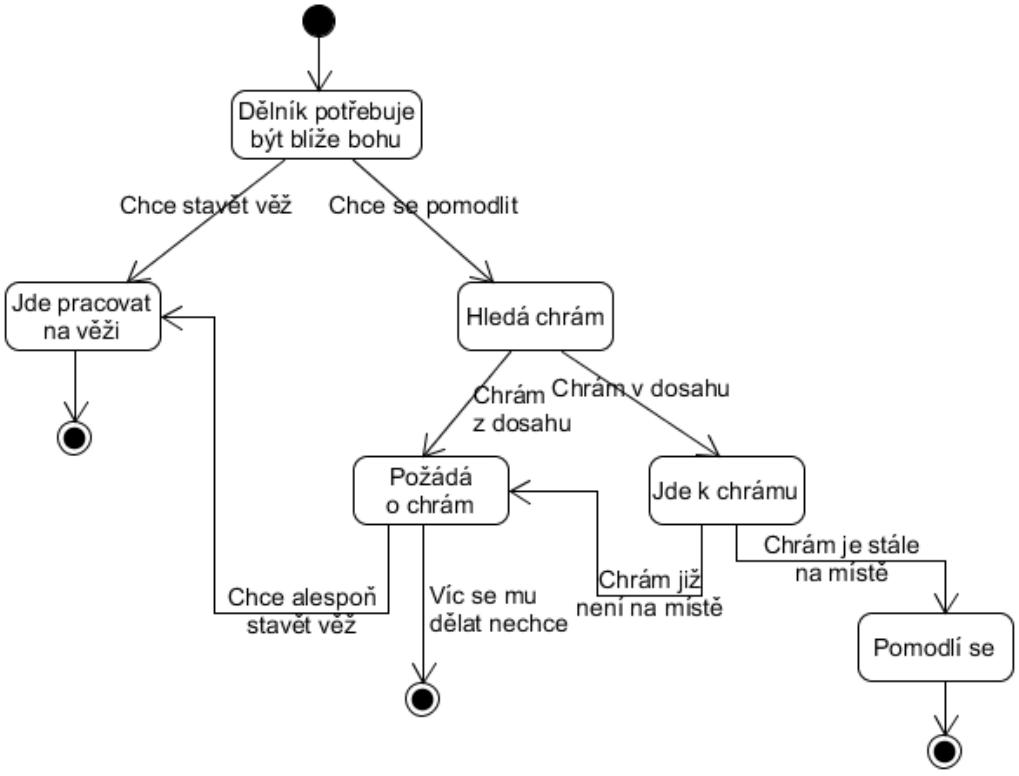
Potřeba	Žízeň
Popis	Žízeň je druhou vitální potřebou. Na její uspokojení má oproti hladu, pouze dva dny. K uspokojení žízně má dělník na výběr ze tří možností. První z možností je napít se vody z řeky. Druhou je napít se ze studny. Obě tyto možnosti dělníka nestojí žádné peníze. Třetí možností je napít se vína. Tato možnost je pro dělníka zpoplatněna. V diagramu 3.18 je znázorněno chování dělníka, v případě, že dostane žízeň.



Obrázek 3.18: Diagram uspokojení žízně

Tabulka 3.29: Duchovno

Potřeba	Duchovno
Popis	Dělník může potřebu duchovna uspokojit dvěma způsoby. Prvním je modlitba v chrámu. Pokud chrám nebyl postaven, nebo ho nezná má dělník druhou možnost uspokojit potřebu duchovna prací na věži, za kterou nedostane zaplaceno. Preference mezi modlitbou a prací na věži je předmětem šlechtění. Konkrétní postup při uspokojování hladu je v diagramu 3.19



Obrázek 3.19: Diagram uspokojení duchovna

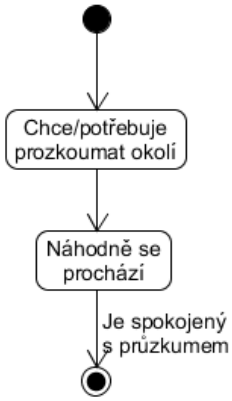
Tabulka 3.30: Odpočinek

Potřeba	Odpočinek
Popis	V současné verzi práce k odpočinku stačí dělníkovi zůstat stát na místě a nic nedělat. Do další verze se však počítá s tím, že by každý dělník měl svůj dům do kterého by si šel odpočinout. Případně se více přizpůsobilo realitě plynutí času a dělník by chodil v noci spát. Detailněji viz diagram 3.20 uspokojení hladu.

Obrázek 3.20: Diagram uspokojení odpočinku

Tabulka 3.31: Průzkum

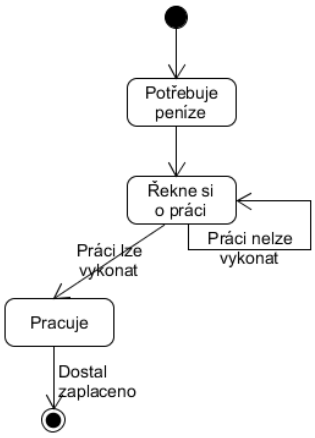
Potřeba	Průzkum
Popis	Dělník se náhodně pohybuje a tím dochází k průzkumu nových oblastí. Postup uspokojení průzkumu viz 3.21



Obrázek 3.21: Diagram uspokojení průzkumu

Tabulka 3.32: Práce

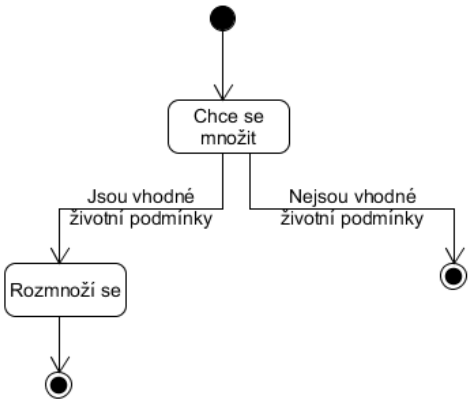
Potřeba	Práce
Popis	Potřeba práce může být upřednostněna některou z vitálních potřeb v případě, že dělník nemá dostatek peněz na její uspokojení. Práce může být uspokojena vykonáním práce v některé z dostupných budov. O práci si dělník říká radnici a ta rozdělováním práce reguluje vývoj města. Za vykonání jednoho úkolu dostane dělník peníze na jeden chleba, maso nebo jedno víno. Tímto množstvím uspokojí vitální potřebu. Postup při uspokojování práce je znázorněn v diagramu 3.22



Obrázek 3.22: Diagram uspokojení práce

Tabulka 3.33: Rozmnožovat se

Potřeba	Rozmnožovat se
Popis	K tomu, aby se dělník mohl rozmnožit musí být splněno několik podmínek. První z nich je, že daný dělník musí dosáhnout určitého věku, v tomto případě pěti dnů. Dalším požadavkem je dostatek surovin. Na nového dělníka je potřeba jeden chleba a jedna jednotka populace. Tyto požadavky jsou pevně dány. Tímto se model přibližuje Verhulstovu modelu, pro který je typické zavedení nosné kapacity prostředí vyjadřující množství jedinců, kterří jsou schopni se na daném území uživit [25]. Předmětem šlechtění je nutná životní úroveň, která se zohledňuje v následujícím vzorci, který rozhoduje o tom, zda se dělník chce rozmnožit: • $(T_{va} < T_{lv} * L) \wedge (Tja < Tlj * L)$ – T_{va} – je průměrný čas bez vody – T_{ja} – je průměrný čas bez jídla – T_{lv} – je časový limit bez vody – T_{lj} – je časový limit bez jídla – $L \in \langle 0, 1 \rangle$ – číslo vyjadřující nutnou životní úroveň S postupem času se také mění frekvence s jakou se chce dělník množit. Vzhledem k tomu, že je zde potřeba rozmnožit vztažena vždy vůči jednotlivci a nikoliv dvojci, lze populaci označit za průkopnickou [25]. I zde je uveden diagram 3.23 postupu uspokojení potřeby.



Obrázek 3.23: Diagram uspokojení rozmnožování

3.8 Šlechtění

Vzhledem k množství nastavitelných parametrů ovlivňujících chování modelu se přímo nabízí využití evolučních algoritmů k optimalizaci chování modelu. Množství parametrů (168) však přináší poměrně složitý optimalizační problém proto nelze dopředu přesně odhadnout vhodnou podobu evolučního algoritmu, ani úspěšnost ve výpočetně omezených podmínkách. V této podkapitole jsou vybrány konkrétní možnosti řešení, které jsou při implementaci uplatněny a v závěru zhodnoceny.

3.8.1 Genotyp

Součástí genotypu jsou priority parametrů pro výstavbu budov, efektivity potřebné pro výstavbu budov a parametry dělníka – rychlost růstu potřeb a preference uspokojování potřeb. Celkem genotyp obsahuje 168 parametrů.

Pro zpřehlednění výpočtu fitness a lepšího ohodnocení fenotypu je genotyp rozdělen na více částí. Lze tak vyhodnocovat fitness pro každou část zvlášť a následně z těchto fitness vypočítat celkovou fitness fenotypu. Genotyp je rozdělen na tyto části. První část, která popisuje nastavení dělníka a 13 dalších částí pro každou budovu zvlášť.

3.8.2 Fitness a její výpočet

Podoba fitness funkce je závislá na požadovaném chování modelu. V prním případě se předpokládá, že cílem je pouze co nejvyšší věž, bez ohledu na další požadavky. Fitness funkce se v takovémto případě nabízí takováto:

- $F(f) = h_v$
  - f – fenotyp
  - h_v – výška věže

Tato funkce by pak platila pro všechny části genotypu.

V druhém případě se předpokládá, že kromě co nejvyšší věže jsou cílem i co nejlepší životní podmínky pro dělníky.

K zhodnocení výšky věže, není třeba žádných složitých vzorců a použije se samotná výška věže vyjádřená počtem pater, včetně rozestavěné části nedokončeného patra.

Určení kvality podmínek pro dělníky bude již o něco složitější. K jejímu výpočtu je využito několik parametrů. Těmi jsou průměrný čas mezi uspokojením hladu, průměrný čas mezi uspokojením žízně, limit pro uspokojení hladu, limit pro uspokojení žízně, úmrtnost hlady, úmrtnost žízní a celková populace. Podoba fitness funkce pro nastavení dělníka by pak mohla vypadat třeba takto:

- $F_d(f) = h_v(T_{lv}/T_{va} + T_{lj}/T_{ja})/2$
  - f – fenotyp
  - h_v – výška věže
  - T_{va} – průměrný čas bez vody
  - T_{ja} – průměrný čas bez jídla
  - T_{lv} – časový limit bez vody
  - T_{lj} – časový limit bez jídla

Tato fitness funkce by tak měla upřednostňovat dělníky, kteří často jedí a pijí, ale zároveň se podílejí na stavbě věže. Fitness funkce pro budovy, jsou pro tyto účely rozdělené do třech kategorií:

- Budovy podílející se na uspokojení hladu
  - Obilná farma, větrný mlýn a pekárna
- Budovy podílející se na uspokojení žízně
  - Studna, vinohrad
- Ostatní

  - Kamenolom, zlatý důl, domy, pila,...

Fitness funkce pro budovy umožňující uspokojení hladu:

- $F_h(f) = h_v(U_h/C_p)$
  - f – fenotyp
  - h_v – výška věže
  - U_h – úmrtnost hladem
  - C_p – celková populace

Fitness funkce pro budovy uspokojující žízeň:

- $F_z(f) = h_v(U_z/C_p)$
  - f – fenotyp
  - h_v – výška věže
  - U_z – úmrtnost hladem
  - C_p – celková populace

Fitness ostatních budov:

- $F_o(f) = h_v$
  - h_v – výška věže

Celková fitness řešení se určí vzrorcem

- $F(f) = ((F_o(f)O + F_h(f)H + F_z(f))/(O + H + Z) + F_d(f))$
  - f – fenotyp
  - h_v – výška věže
  - $F_o(f)$ – fitness ostatní budov
  - $F_h(f)$ – fitness budov uspokojujících hlad
  - $F_z(f)$ – fitnes budov uspokojujících žízeň
  - $F_d(f)$ – fitnes dělníků
  - O – počet ostatní budov
  - H – počet typů budov uspokojujících hlad
  - Z – počet typů budov uspokojujících žízeň

3.8.3 Inicializace

K inicializaci populace je testována náhodná inicializace. A v druhém případě ručně nastavená inicializace, která by měla poskytnout dobrý odrazový bod pro šlechtění.

3.8.4 Selekce

I v případě selekce je vyzkoušen více než jeden přístup.

Prvním přístupem, je výběr několika nejlepších jedinců na základě jejich částečných fitness. Hodnoty jejich parametrů jsou zprůměrovány do výsledného fenotypu. Tento přístup zvýhodňuje nejsilnější známé řešení a podobá se tak metodě hillclimbing. Nevýhodou tohoto přístupu je riziko uváznutí v lokálním minimu.

Druhým přístupem, je využití selekce pomocí turnaje. Vybere se nějaké určité, nebo náhodné množství kandidujících řešení a z nich se vybere nejsilnější. Tento přístup dává šanci i méně dobrým řešením.

3.8.5 Mutace

Mutace je provedena přičtením náhodného čísla k náhodně vybraným parametrům fenotypu, pro každý parametr se generuje nové číslo. Poté se znovu náhodně vyberou parametry fenotypu a vynásobí se náhodným číslem, opět pro každý parametr zvlášť generovaným. Čísla jsou v obou případech generovány v daném rozsahu, tak že všechna čísla mají stejnou pravděpodobnost, aby byly vygenerovány. Čím vyšší je fitness, tím menší je pravděpodobnost na zmutování některého z parametrů fenotypu.

I zde není pouze jedniná možnost. V druhém případě je vynechána násobící část. Zaměřeno je na přičítací část, kde je změněněný způsob výběru přičítaného čísla. Ke generování náhodných čísel, které je následně přičíteno k vybranému parametru, je využito polární formy Box-Mullerovy transformace [24].

```
float x1, x2, w, y1, y2;

do {
    x1 = 2.0 * rand() - 1.0;
    x2 = 2.0 * rand() - 1.0;
    w = x1 * x1 + x2 * x2;
} while ( w >= 1.0 );

w = sqrt( (-2.0 * log( w ) ) / w );
y1 = x1 * w;
y2 = x2 * w;
```

Kde `rand()` generuje čísla z intervalu $\langle 0, 1 \rangle$. Box-Mullerova transformace umožňuje transformovat rovnoměrně rozloženou náhodně generovanou množinu čísel do nové množiny náhodných čísel s gaussovým/normálním rozložením. Tímto přístupem bychom měli dosáhnout pozvolnějších změn a tím stabilnějšího vývoje. Pokud by vývoj byl příliš nestálý, můžeme vygenerované číslo dělit zvolenou konstantou.

Kapitola 4

Implementace

V této kapitole jsou detailněji rozebrány vybrané zajímavé části implementace.

- První podkapitola 4.1 obsahuje výčet implementovaných částí.
- Druhé podkapitola 4.2 popisuje implementaci dělníka, dělníkových potřeb a jeho pohyb po mapě.
- Třetí podkapitola 4.3 je věnována implementaci mapy a generátoru náhodného terénu.
- Čtvrtá podkapitola 4.4 je zaměřena na výkonnost výsledné implementace.
- Předposlední podkapitla 4.5 shrnuje výsledky aplikace evolučních algoritmů.
- Poslední kapitola 4.6 obsahuje výsledky testů.

4.1 Výčet implementovaných částí

Výsledná implementace zahrnuje:

- Suroviny
  - chleba, mouka, obilí, voda, víno, zlato, kámen, dřevo, populace
- Budovy
  - pekárna, mlýn, farma, studna, vinohrad, zlatý důl, kamenolom, pila, dům, radnice, Babylonská věž, chrám
- Všechny zmiňované parametry na základě, kterých se rozhoduje o výstavbě budovy.

- Potřeby dělníků
  - hlad, žízeň, duchovno, odpočinek, pracovat, prozkoumávat, množit se
- Uspokojení potřeb
  - hlad – chlebem
  - žízeň – vodou z řeky/study a vínem z vinohradu
  - duchovno – modlitbou nebo prací na věží
  - odpočinek – odpočíváním
  - práci – pracovním v některé z budov, nebo těžbou surovin
  - průzkum – náhodným procházením
  - množit se – plozením dalších dělníků
- Generátor náhodného terénu a zdrojů surovin
  - generuje řeku ze zadaných parametrů
  - rozmístí suroviny podle zadaných parametrů
- Grafické uživatelské rozhraní
- Vizualizaci procesu vývoje města
  - modely budov, zdrojů surovin, dělníka
  - potřebné textury a materiály
- 5 verzí evolučních algoritmů, testovaných k optimaizaci parametrů
  - různé kombinace operátoru selekce, mutace
  - různé způsoby výpočtu fitness a rozdělení genotypu
- Napojení modelu na SQLite databázi, sloužící k ukládání parametrů a výsledků šlechtění.

I přes neúplnou implementaci všech částí z návrhu, je implementace v tomto rozsahu více než dostačující pro fungování modelu.

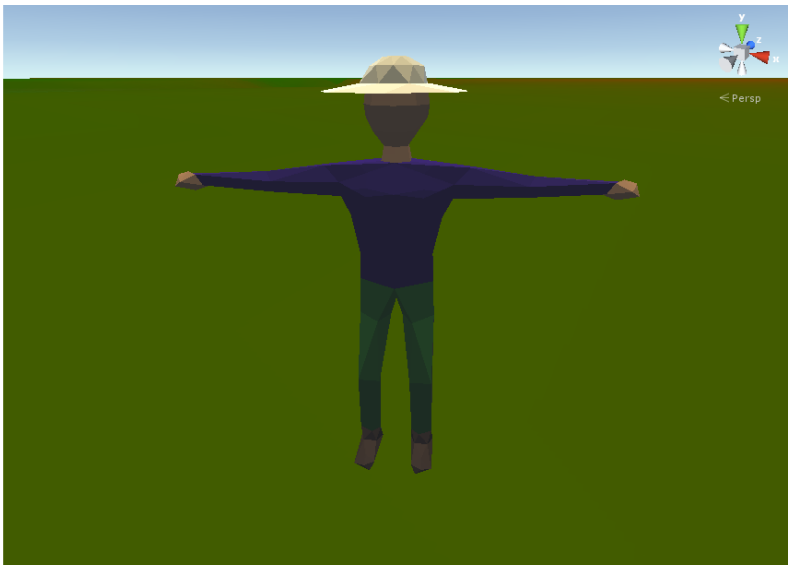
4.2 Implementace dělníka

4.2.1 Struktura dělníka a jeho potřeb

Třída dělníka se stará o výběr aktuálně nejnutnější potřeby a jejího uspokojení. Třída dělníka `Worker` je poděděná z třídy `Unit`, která je předpřipravena pro rozšíření této práce o další typy jednotek. Na začátku běhu si tak načte všechny

komponenty (třídy) potřeb připojené k objektu dělníka a uloží do prioritní fronty, kde jsou řazeny podle aktuální priority daných potřeb. Všechny potřeby jsou potomky třídy `Need`.

Poté v cyklu vybírá nejakutnější potřebu a tu se snaží uspokojit. Chování dělníka je řízeno několika stavovými automaty, které reprezentují jeho potřeby. Vždy má tak uloženou potřebu kterou aktuálně uspokojuje a stav v kterém se nachází proces uspokojení potřeby. Každá potřeba má funkci `satisfyNeed (int state)`, která je volána z třídy `Worker` se stavem v kterém se nachází její uspokojování. Návratovou hodnotou této funkce je -1 pro neúspěšné uspokojení potřeby, 0 pro úspěšné uspokojení potřeby a nebo kladné číslo vyjadřující stav v kterém se nachází uspokojovací proces.



Obrázek 4.1: Model dělníka

4.2.2 Pohyb dělníka

Pohyb dělníka je implementován pomocí nástroje NavMesh, který je součástí Unity. Na objekt dělníka, se připojí komponent NavMeshAgent. Tomuto komponentu se poté nastaví parametry, jako jsou výška, poloměr nebo rychlost agenta. Kromě připojení komponentu NavMeshAgent, je třeba vygenerovat síť, kterou bude tento komponent využívat k navigaci. Vygenerovat tuto síť je možné pouze v editoru scény v Unity a není možné ji generovat za běhu. K implementaci dynamických překážek slouží komponent NavMeshObstacle. Tvar překážky je omezený a může být pouze koule, nebo kvádr.

Jakmile je vygenerovaná síť, stačí agentovi zadat bod do kterého se má přesunout. Pokud tento bod není dostupný, přesune se agent co nejblíže mu síť umožňuje.

4.3 Implementace mapy

4.3.1 Ukládání informací o mapě

Jak již bylo zmíněno v návrhu, je mapa rozdělena na čtverce. Každý z těchto čtverců, je reprezentován instancí objektu `Quadrant`, která si o něm uchovává různé informace. Mezi tyto informace patří poloha na mapě, nejmenší vzdálenosti od jednotlivých typů budov a surovin, nebezpečí požáru či povodně nebo jaká budova je na něm umístěna. Tyto informace jsou pak průběžně aktualizovány. Aktualizace nejsou okamžité a šíří se postupně od místa události po celé mapě. Informace se šíří po spirále a rychlost šíření lze nastavit.

4.3.2 Generátor terénu

Terén je generován ve dvou fázích. V první fázi se vygeneruje řeka a ve druhé zdroje surovin.

4.3.2.1 Generování řeky

Řeka je náhodně generována ze zadaných parametrů, těmi jsou:

- Množina dvojic bodů určující segmenty, kterými má řeka protékat
- Hloubka
- Šířka
- Detail
- Klikatost

Generování řeky pak probíhá tak, že se v n cyklech každý segment řeky rozdělí na dva nové a bod v kterém se rozdělili se posune o k násobek normály původního úseku. Vektor normály, není normalizován a jeho velikost zůstává rovna velikosti původního úseku. Kde n je rovno parametru Detail. Hodnotu k získáme jako náhodné číslo z intervalu $(-l, l)$, kde se l v každém cyklu mění podle funkce:

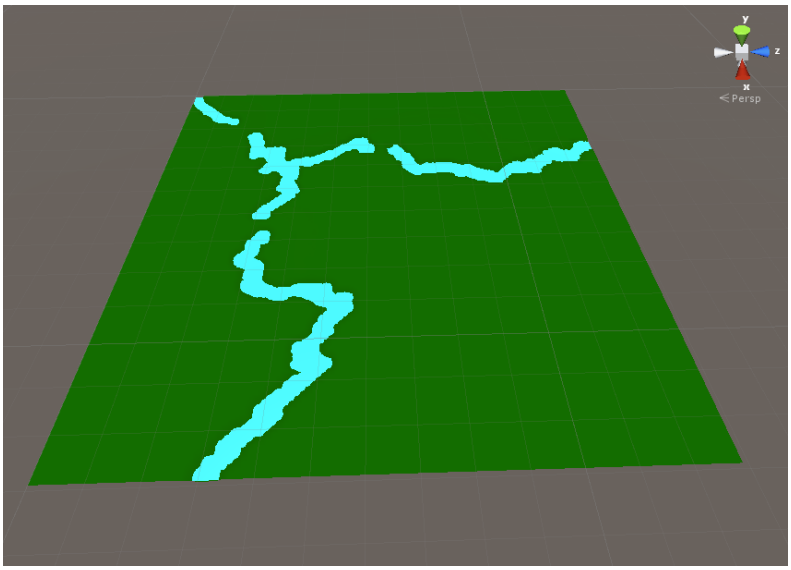
- $F(i) = K/i^2$
  - i – číslo probíhajícího cyklu, indexované od 1
  - K – Klikatost

Po vygenerování výsledných segmentů řeky, se na základě šířky a hloubky řeky předpočítá matice, v které jsou uloženy výškové souřadnice koryta řeky. Šířka řeky udává šířku a výšku matice. Hodnoty v matici se pak řídí touto funkcí:

- $A[i, j] = ((\cos(\Pi(1 + D/D_s)) + 1)/2) * R_s/T_{mh}$

- A – matice výšek
- i – vertikální pozice v matici
- j – horizontální pozice v matici
- R_s – šířka řeky
- $D = |\vec{s}|, \vec{s} = \{i - R_s/2, j - R_s/2\}$
- $D_s = |\vec{t}|, \vec{t} = \{R_s, R_s\}$
- T_{mh} – maximální výška terénu

Tato matice poslouží jako "štětec", který se aplikuje v každém bodě obsaženém ve vygenerovaných segmentech. Tím se vytvoří koryto řeky. Nyní již stačí jen přidat objekty, reprezentující hladinu vody a řeka je hotová.



Obrázek 4.2: Náhodně vygenerovaná řeka

4.3.2.2 Generování surovin

Ke generování zdroje každé ze tří surovin získávaných těžbou jsou potřeba dva parametry. Na kolika oddělených místech se má vyskytovat n a jaké přibližné množství m by na každém z nich mělo být.

Poté se provede n cyklů. V každém se:

1. Náhodně vybere bod na mapě okolo kterého budou suroviny generovány.
2. Náhodně se vybere výsledný počet surovin z intervalu $(m^{(}\sqrt{m}, m)$.
    a) V cyklu se generují náhodné body na jednotkové kružnici, jejichž souřadnice se pak vynásobí postupně se zvětšujícím poloměrem.

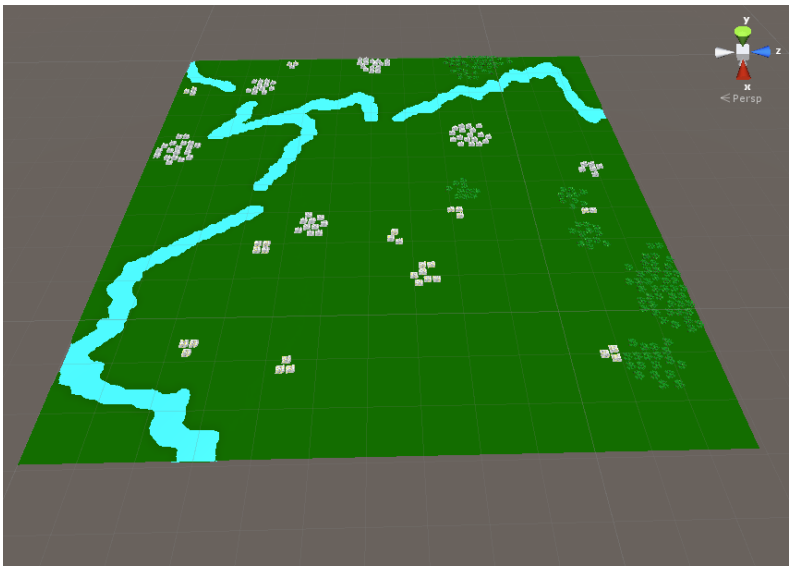
b) Poté se náhodně rozhodne, zda bude surovina umístěna, pravděpodobnost na umístění suroviny se s rostoucím poloměrem zmenšuje.

c) Pokud je rozhodnuto o umístění suroviny na vygenerovaných souřadnicích, pokusí se algoritmus o umístění suroviny, to může selhat, pokud je již mimo terén, nebo je místo již obsazené.

d) Pokud se umístění povede, je započítáno.

e) Poté se cyklus opakuje dokud není umístěné požadované množství surovin, nebo počet pokusů o umístění nepřesáhne $10m$, tato konstanta byla stanovena na základě odhadu a osvědčila se.

3. Po umístění dostatečného množství surovin pro danou oblast se pokračuje v hlavním cyklu



Obrázek 4.3: Náhodně vygenerované suroviny

4.4 Výkonnost

Při konfiguraci notebooku Acer Aspire V3-571G

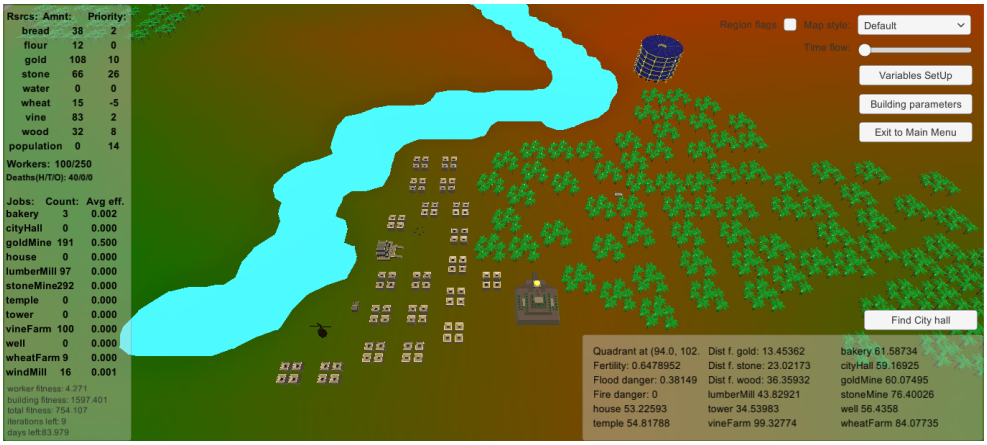
- Intel Core i5-3210M 2,5GHz
- NVIDIA GeForece GT640M, 2048MB
- 6 GB DDR3

Je model schopný vizualizovat při 100× zýšené rychlosti běhu

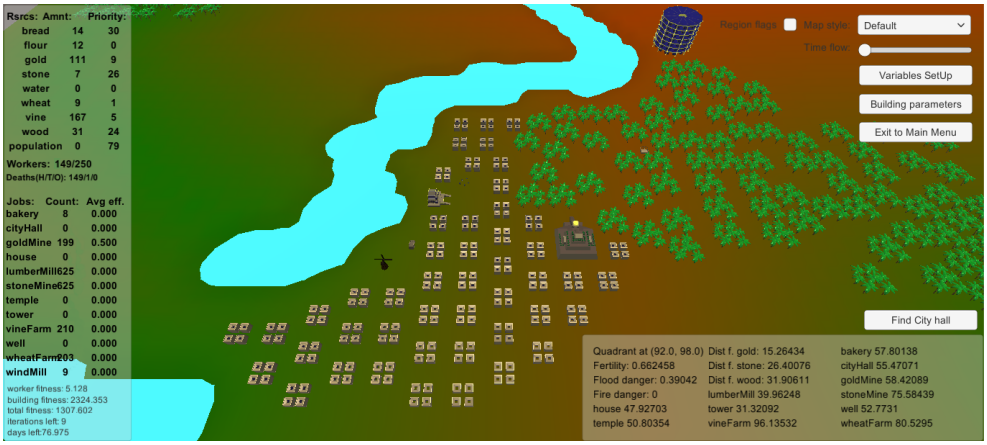
- 100 dělníků při 10-15 fps a 256,5 tisících trojúhelníků ve scéně viz 4.4
- 150 dělníků při 5-15 fps a 270 tisících trojúhelníků ve scéně 4.5
- 250 dělníků při 5-10 fps a 272 tisících trojúhelníků ve scéně 4.6

V případě normální rychlosti(1×)

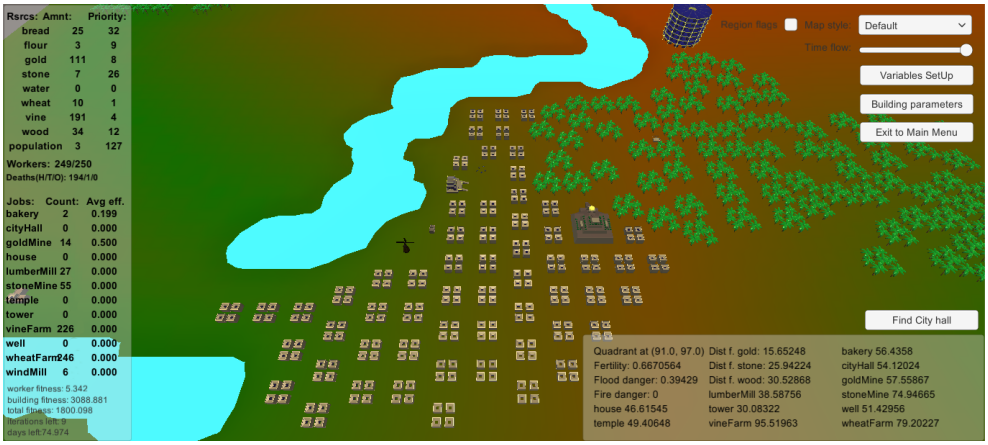
- 100 dělníků při 35-45 fps a 256,5 tisících trojúhelníků ve scéně viz 4.4
- 150 dělníků při 30-40 fps a 270 tisících trojúhelníků ve scéně 4.5
- 250 dělníků při 25-35 fps a 272 tisících trojúhelníků ve scéně 4.6



Obrázek 4.4: Scéna obsahující 100 dělníků (256,5 tisíc trojúhelníků)



Obrázek 4.5: Scéna obsahující 150 dělníků (270 tisíc trojúhelníků)



Obrázek 4.6: Scéna obsahující 250 dělníků (272 tisíc trojúhelníků)

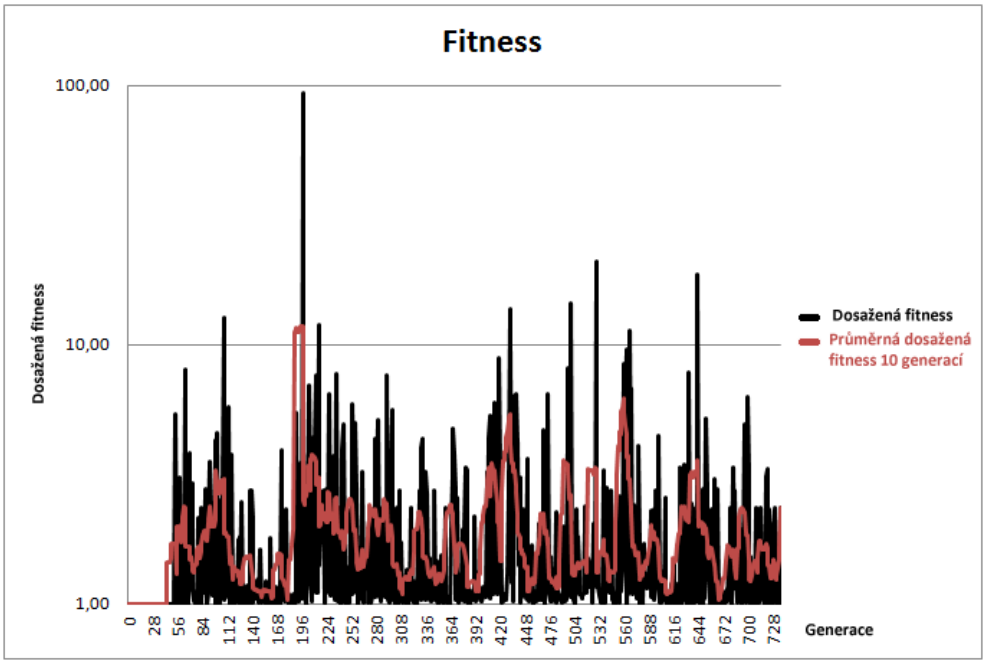
Výkonnost modelu by mohla být i větší, je však omezena API Unity 5, které není thread-safe (negarantuje bezpečnou manipulaci s daty z více vláken najednou).

4.5 Výsledky aplikace evolučních algoritmů

Aplikací jednotlivých verzí evolučních algoritmů, se potvrdilo, že prohledávaný prostor (156 rozměrný) je příliš velký, než aby byl za daných výkonnostních podmínek prohledávatelný s větším úspěchem. V této podkapitole jsou uvedeny výsledky, ke kterým se dopracovaly jednotlivé verze. V každé verzi byla vždy jedna iterace dlouhá až 100 dní (v případě vymření populace dělníků byla ukončena). Těchto iterací proběhlo pro každou verzi v naprosté většině případů více než 350, v případě první více než 700.

4.5.1 První verze

- Fitness dělníka: $F_d(f) = h_v(T_{lv}/T_{va} + T_{lj}/T_{ja})/2$
  - f – fenotyp
  - h_v – výška věže
  - T_{va} – průměrný čas bez vody
  - T_{ja} – průměrný čas bez jídla
  - T_{lv} – časový limit bez vody
  - T_{lj} – časový limit bez jídla
- Fintess budov: $Fb(f) = h_v((U_h + U_z)/C_p)$
  - f – fenotyp
  - h_v – výška věže
  - U_h – úmrtnost hladem
  - U_z – úmrtnost žízní
  - C_p – celková populace
- Celková fitness: $F(f) = hv(F_b(f) + F_d(f))$
- Inicializace proběhla nastavením odhadnutých vhodných hodnot parametrů.
- Ke šlechtění je vybráno nejlepších pět jedinců z každé části genotypu. Hodnoty jejich fenotypu se zprůměrují. K náhodně vybraným parametrům, se přičte náhodně vygenerované číslo. Poté se náhodně vybrané parametry vynásobí náhodným číslem ze zvoleného intervalu.
- Z grafu 4.7 je vidět, že ani po více než 700 generacích se fitness nějak zvlášť dlouhodobě nezlepšila. Takto nastavený evoluční algoritmus k uspokojivému řešení nekonverguje.



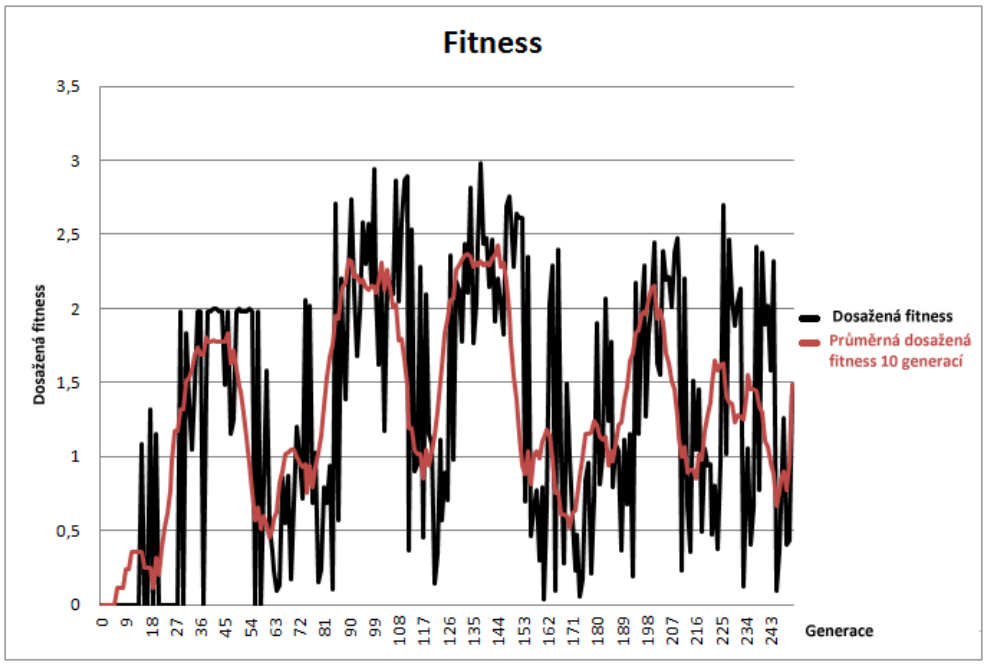
Obrázek 4.7: Graf vývoje fitness první verze

4.5.2 Druhá verze

- Fitness dělníka, budov i celková jsou rovny výšce věže.
- Inicializační operátor měl podobu náhodné inicializace.
- Šlechtění probíhá stejně jako v první verzi.
- Inicializace proběhla nastavením odhadnutých vhodných hodnot parametrů.
- Z grafu 4.8 vývoje fitness založené pouze na výšce věže nepozorujeme rostoucí vývoj, opět je pouze velice chaotický.

4.5.3 Třetí verze

- Fitness dělníka je stejná jako v první verzi
- Fitness budov se počítá pro každou budovu zvlášť. Fitness bodovy uspokojujici hlad:
  - $F_h(f) = h_v(U_h/C_p)$
    - f – fenotyp
    - h_v – výška věže



Obrázek 4.8: Graf vývoje fitness druhé verze

  - U_h – úmrtnost hladem
  - C_p – celková populace

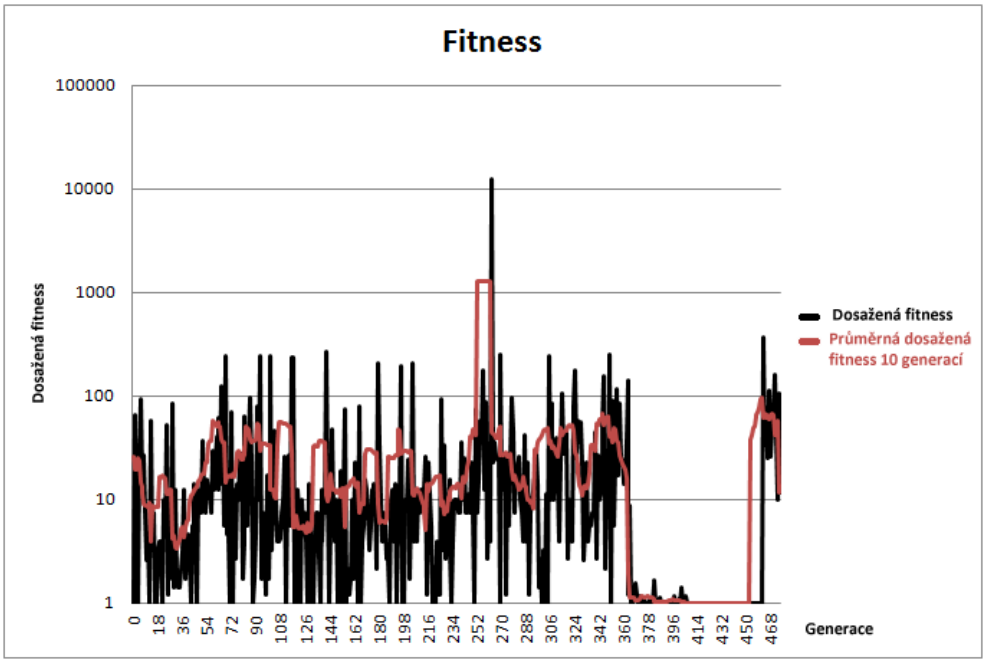
- Fitness funkce pro budovy uspokojující žízeň:
  - $F_z(f) = h_v(U_z/C_p)$
    - f – fenotyp
    - h_v – výška věže
    - U_z – úmrtnost hladem
    - C_p – celková populace

- Fitness ostatních budov:
  - $F_o(f) = h_v$
    - h_v – výška věže

- Celková fitness:
  - $F(f) = ((F_o(f)O + F_h(f)H + F_z(f))/(O + H + Z) + F_d(f))$
    - f – fenotyp
    - h_v – výška věže
    - $F_o(f)$ – fitness ostatní budov

  - $F_h(f)$ – fitness budov uspokojujících hlad
  - $F_z(f)$ – fitnes budov uspokojujících žízeň
  - $F_d(f)$ – fitnes dělníků
  - O – počet ostatní budov
  - H – počet budov uspokojujících hlad
  - Z – počet budov uspokojujících žízeň

- Inicializační operátor měl podobu náhodné inicializace.
- Šlechtění probíhá stejně jako v první verzi s ohledem na více typů fitness u budov.
- I v tomto případě není vývoj grafu 4.9 fitness rostoucí. Velký propad u několika posledních generací, lze přisuzovat příliš velkým změnám fenotypů. Je tedy možné usoudit, že ani toto není vhodný způsob šlechtění.



Obrázek 4.9: Graf vývoje fitness třetí verze

4.5.4 Čtvrtá verze

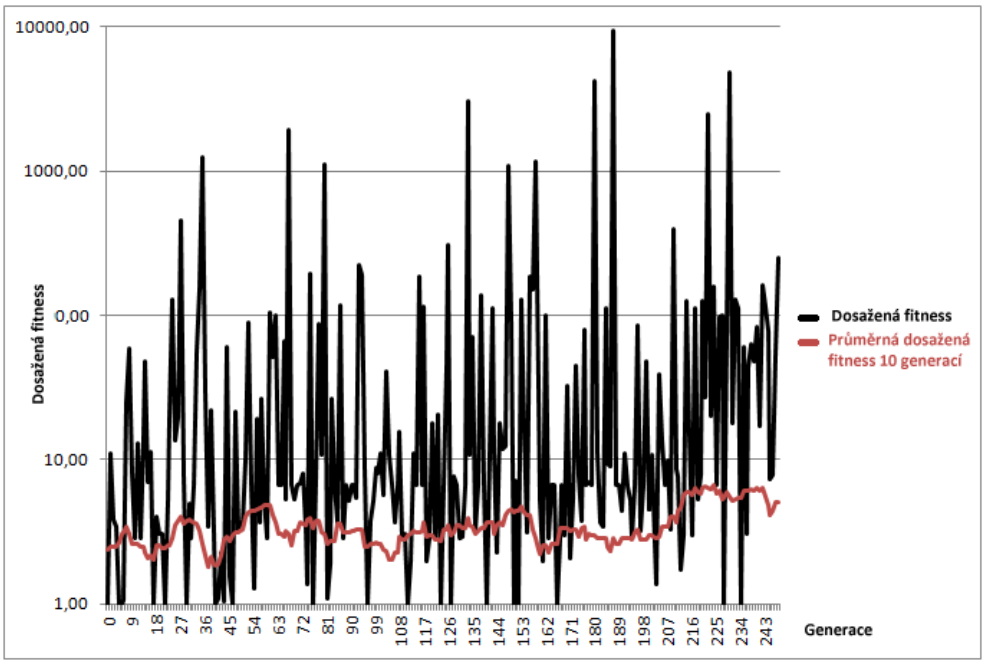
- Fitness se počítá jako ve třetí verzi.
- Inicializace proběhla nastavením odhadnutých vhodných hodnot parametrů.

- Operátor selekce má podobu turnaje, kdy je vylosováno náhodné množství fenotypů a ten s nejlepší fitness je vybrán k dalšímu šlechtění.

- Mutace probíhá přičtením náhodného čísla k náhodně vybraným parametrům. Dvě čísla jsou lineárně generována z intervalu $(0, 1)$ a poté transformována pomocí polární formy Box-Mullerovy transofrmace. Tato transformace má na vstupu dvě čísla a stejně tak i na výstupu. Proto je pak s pravděpodobností $p = 0,5$ vybráno první číslo. Toto číslo je poté vyděleno konstantou stanovenou pro danou skupinu iterací z intervalu $\langle, 4\rangle$.

- K zmenšení prohledávaného prostoru, byly z mutace vynechány některé parametry budov. Pro tento případ se předpokládá, že lepší nastavení, než 0 by mělo zanedbatelný pozitivní vliv na model a případná změna hodnoty by spíše zpomalila proces šlechtění. Tím se zmenšil prozkoumávaný prostor ze 156 na 75 rozměrů.

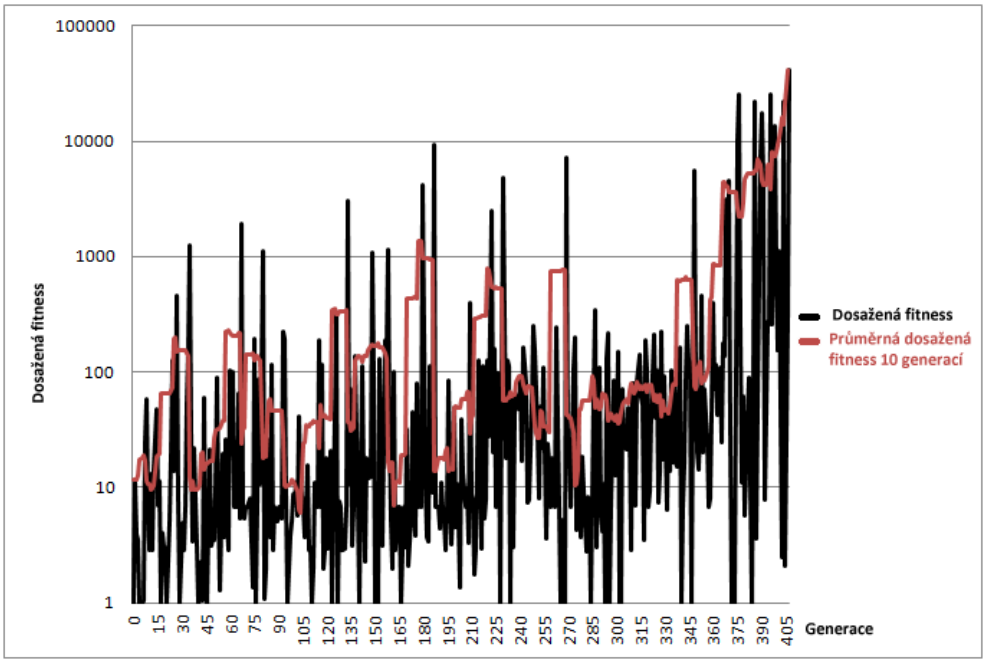
- Graf 4.10 čtvrté verze je první, kde lze pozorovat míré zlepšení. Fitness sice stále kolísá, což je z velké části způsobeno turnajovou selekcí, která dává prostor pro evoluci i fenotypům s nižší fitness. Jisté zlepšení je však z grafu vidět.



Obrázek 4.10: Graf vývoje fitness čtvrté verze

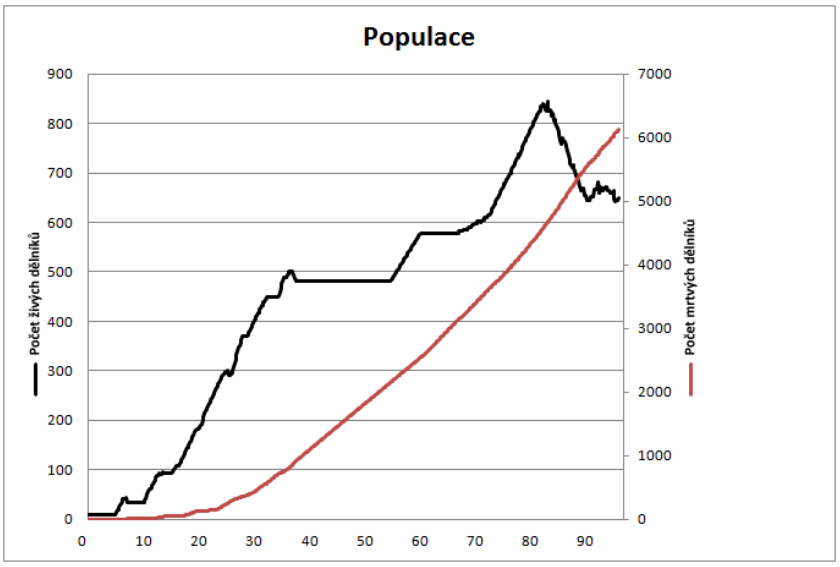
4.5.5 Pátá vezre

- Pátá verze využila dat získaných ze čtvrté verze.
- Od čtrvté se liší pouze tím, že se pokouší zvýšit pravděpodobnost růstu fitness upravenou verzí turnajové selekce ze čtvrté verze. V páté verzi selekce probíhá výběrem deseti nejlepších fenotypů z nichž je následně náhodně jeden vybrán k dalšímu šlechtění. Zvýší se tím riziko uváznutí v lokálním optimu. Vzhledem k prohledávanému prostoru, bude v těchto podmínkách jakýkoliv posun k lepšímu úspěchem.
- Zmenšení turnaje na 10 nejlepších jedinců se na grafu 4.11 vývoje fitness pozitivně projevilo. Natolik pozitivně, že výpočet jedné generace se stal časově velmi náročným pro velkou populaci dělníků. Populace čítala v některých bodech i přes 1000 jedinců, typicky však okolo 500.

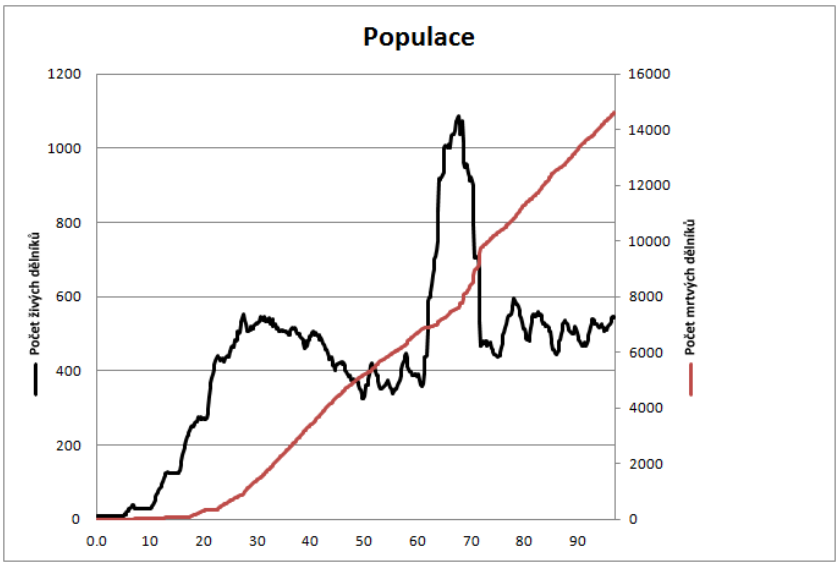


Obrázek 4.11: Graf vývoje fitness páté verze, včetně dat čtvrté verze

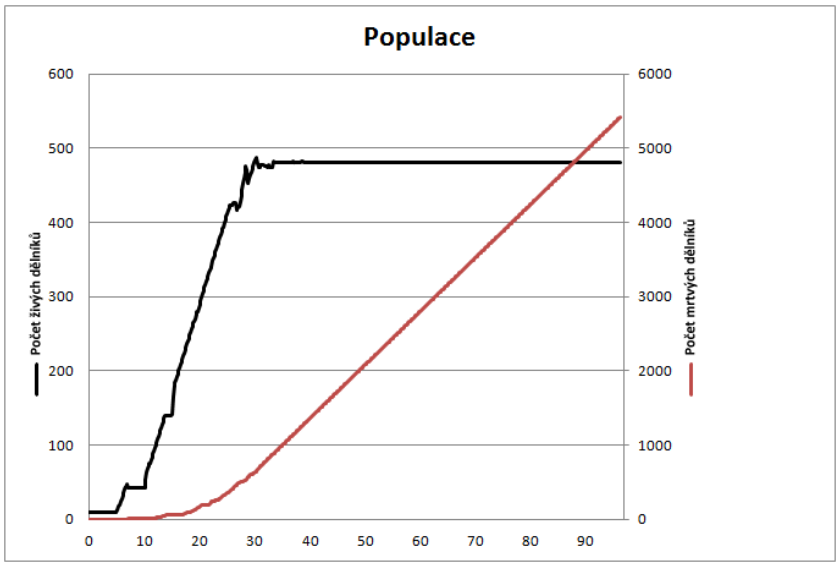
Z páté verze jsou uvedeny grafy vývoje populací vybraných iterací. Záznamů stavu populace připadá 6 na jeden den.



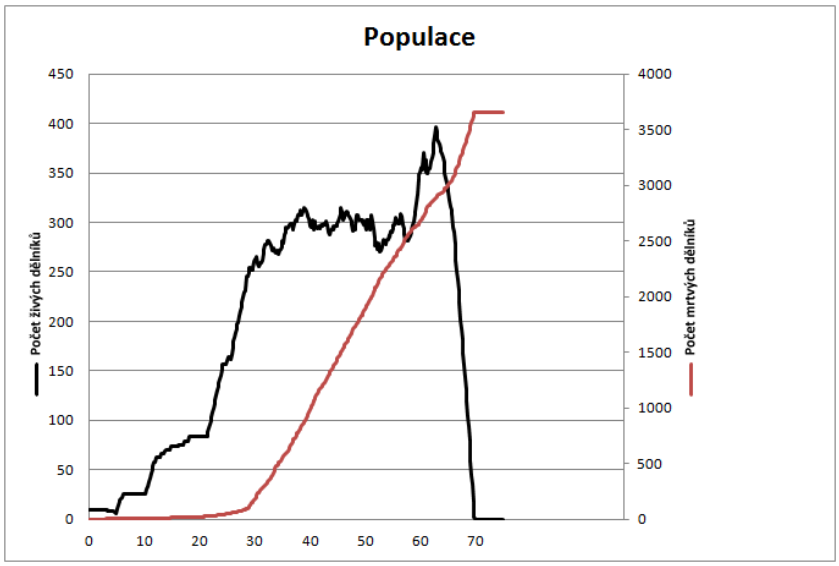
Obrázek 4.12: Graf patří iterace s nejlepší fitness
Ggraf patřící iteraci, která dosáhla nejlepší fitness a výšky věže 10,52 pater.



Obrázek 4.13: Graf iterace s nejlepší fitness
Graf vývoje populace iterace, která dosáhla nejvyššího počtu dělníků v jednom okamžiku. Celková fitness této iterace je hluboce pod průměrem, i přes velmi dobrou výšku věže 10,49 pater. Maximální dosažená výška věže je 10,55 pater.



Obrázek 4.14: Graf iterace s vemli dobrou fitness a průměrnou věží
Tento graf patří iteraci, jejíž fitness patří mezi nejlepší, nicméně výška věže je pouze 4,71 patra. Za vysokou fitness zde stojí životní podmínky dělníků. Zhruba od 40. dne došlo k úplnému vyrovnání porodnosti a úmrtnosti a velikost populace se již téměř nezměnila.



Obrázek 4.15: Graf iterace s vysokou věží a špatnou fitness
Tento graf patří iteraci, jejíž fitness patřila mezi slabší, ale výškou věže vynikala 9,25 pater. Z grafu je patrný velký populační růst, krátké kolísání a následný celkový úpadek populace.

4.6 Výsledky testů

V průběhu implementace byla každá nová funkcionalita podrobena testům zaměřeným nejenom na správnou funkčnost, ale i bezchybnost. Testy byly zaměřeny nejen na běžné situace, ale i krajní. Chyby odhalené těmito testy, nebyly nijak závažné a byly vždy ihned napraveny.

K odhalování skrytých chyb byl model spuštěn po delší dobu (i 20 hodin v kuse). Tyto testy přispěly k odhalení mnoha chyb týkajících se nejen špatné práce s pamětí, ale i logických chyb, které měly za následek například špatnou správu surovin.

Tyto dlouhodobé testy také prokázaly schopnost modelu v nastavených podmínkách budovat město. Tato schopnost se samozřejmě do značné míry odvíjela od podoby fenotypu.

Dále byl model podroben testování výkonu. K tomuto účelu byl použit nástroj Profiler obsažený v Unity, který přehledně zobrazuje využití výkonu počítače. S jeho pomocí se podařilo identifikovat časově náročné úkony a rozložit jejich výpočet na více částí a tím zajistit plynulejší běh modelu. Všechny testy byly několikrát zopakovány k potvrzení své správnosti.

Vzhledem k účelu modelu, který má být primárně dynamickým prostředím ve hře a nikoliv samostatnou aplikací, byly testy uživately velmi omezené a přispěly pouze k drobným úpravám grafického rozhraní.

Závěr

Zhodnocení práce

V této podkapitole jsou zhodnoceny jednotlivé části výsledné implementace. Uvedeny jsou, jak klady ,tak zápory výsledné práce.

Klady

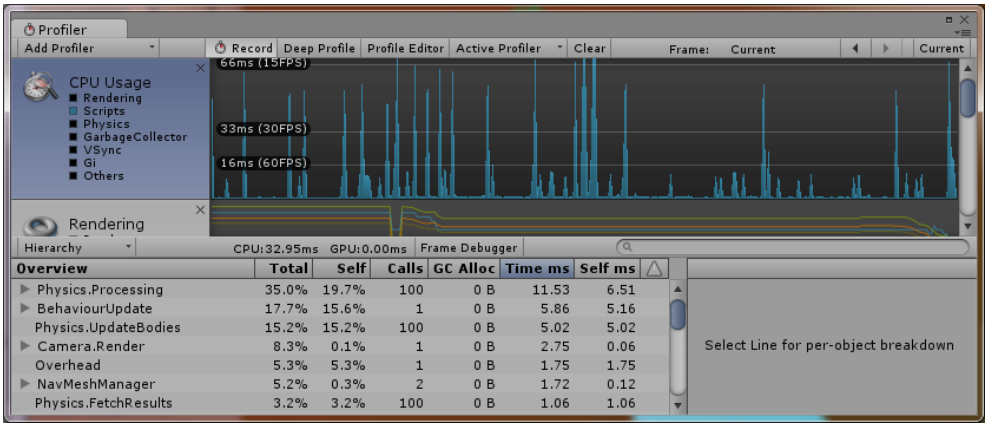
Za vydařenou část práce lze považovat strukturu dělníka, kterou se podařilo navrhnout velmi snadno rozšiřitelnou. K rozšíření chování dělníka stačí pouze vytvořit skript popisující nové chování a dodržující několik jednoduchých pravidel. Následně se pouze přidá jako další komponent a dělník je rozšířen o nové vlastnosti.

Další povedenou částí práce je rozhodně generátor terénu, který z několika málo parametrů generuje pokaždé jinou krajinu. Ta může obsahovat řeku, lesy a hory.

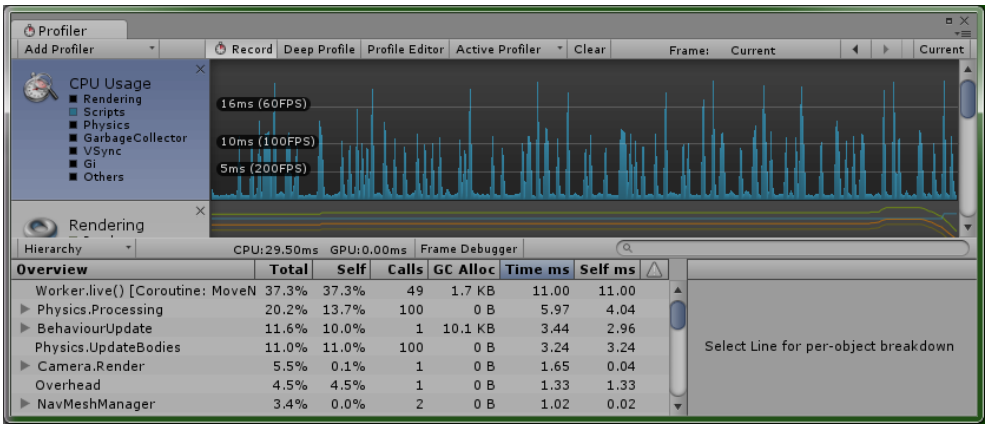
Optimalizace je částí, která není zařaditelná přímo do záporných částí, nicméně je na ní stále co zlepšovat. Je nutno brát v úvahu, že API Unity není thread-safe, což velmi snižuje možnosti optimalizace zvlášť u multi-agentního modelu, který k využití více vláken přímo vybízí. I přesto se podařilo dosáhnout slušných zlepšení, jak je vidět na záznamech z Profileru před optimalizací 4.16 a po optimalizaci 4.17 (oba záznamy jsou pořízeny za podobných podmínek), a proto je optimalizace zařazena do kladů.

Zápory

Méně zdařilou částí práce je rozšiřitelnost o další budovy a suroviny. I přes poměrně snadnou rozšiřitelnost v této verzi práce, je zde jistě prostor pro zlepšení a zjednodušení procesu zakomponování nových budov, nebo surovin do modelu.



Obrázek 4.16: Profiler před optimalizací



Obrázek 4.17: Profiler po optimalizaci

Další méně povedenou částí je nastavení některých parametrů, které nebyly předmětem šlechtění. Především pak časových limitů pro ukojení vitálních potřeb v poměru k času od kterého se může agent rozmnožit. Došlo tak k tomu, že šlechtěním model dospěl k tomu, že je výhodnější nechat dělníka rozmnožit a nechat zemřít, než ho nechat najíst. Takové chování vzhledem k nastavení fitness funkce, zvýhodňující přežívání dělníků, je překvapivé.

Vyhodnocení splnění zadání

1. Analyzujte problematiky modelů pro samoorganizující se rozvoj urbanistické scény.

   - Tento bod je splněn v kapitolách Rešerše 1 a Analýza 2. V kapitole Rešerše jsou popsány vybrané modely a použitelné metody z oblasti umělé inteligence. V kapitole Analýza jsou pak ty samé modely a metody zhodnoceny.

2. Definujte na základě požadavků zadavatele soubor pravidel (model) prostředí, podle kterých se bude město rozvíjet.

   - Tomuto bodu se věnuje především kapitola Návrh 3, v které jsou rozebrány funkční a nefunkční požadavky, soubor surovin, budov a jednotek, ale také pojetí času, hledání nejvhodnějšího nastavení parametrů nebo generátor náhodného terénu. Částečně je pak toto téma zahrnuto i v Úvodu . Bod je považován za splněný.

3. Pomocí metod softwarového inženýrství navrhněte prototyp aplikace simulující samoorg. rozvoj urbanistické scény s ohledem na jeho využití pro strategickou hru a to včetně vhodného rozhraní.

   - Stejně jako předchozímu bodu je tomuto věnován prostor především v kapitole Návrh 3, kde jsou podrobně rozepsány principy fungování dělníků, budov, souboru surovin a mnoho dalšího. Bod lze považovat za splněný.

4. Implementujte prototyp vizualizace modelu města pomocí engine Unity.

   - Vybrané části výsledné implementace jsou popsány v kapitole Implementace 4. Samotná implementace se nachází na přiloženém disku. I přesto, že implementace neobsahuje všechny části z návrhu, je model funkční a tím pádem lze tento bod považovat za splněný.

5. Implementovaný prototyp podrobte vzhledem k účelu aplikace vhodným testům.

   - Podoba a výsledky testů jsou popsány v kapitole Implementace 4, konkrétně v její poslední podkapitole Výsledky testů 4.6. Poslední bod, lze považovat za splněný.

Návrhy na zlepšení

Hlavním bodem ke zlepšení je především zlepšení modulárnosti modelu co do rozšíření o nové budovy a suroviny. Nejlépe pak dosáhnout alespoň takové modulárnosti, jakou disponují potřeby dělníků.

Jistě by se hodilo přidat více modelů budov a prostředí, mezi kterými by se náhodně volilo při umisťování do mapy.

Dalším bodem pro zlepšení jsou animace dělníků, kterou v tuto chvíli mají pouze pro chůzi.

Zpřesnit by šlo měření vzdáleností tak, aby vypočítaná vzdálenost přesněji odpovídala té, kterou musí dělník urazit. Pro tyto účely by šel nejspíše použít nástroj Unity NavMesh.

Možnosti dalšího rozšíření práce

Možností rozšíření práce je mnoho. Zmíněny jsou pro představu pouze některé. Kromě zapracování modelu do hry, se nabízí rozšíření o další potřeby dělníků, nové typy budov, surovin i jednotek.

Potřeby dělníků by mohly být rozšířeny nejen o další způsoby uspokojení, ale i o uplně nové potřeby, jako třeba bavit se. Rozšíření surovin a budov je úzce propojené s uspokojováním potřeb dělníků. Mohla by přibýt třeba hospoda, divadlo, nebo přístav. S těmito budovami by se nutně přidaly charakteristické suroviny, v případě hospody pivo, nebo v případě přístavu ryby.

Literatura

[1] Strategy video game: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 3. 3. 2016 v 06:05. [vid. 2016-05-13]. Anglická verze. Dostupné z: `https://en.wikipedia.org/wiki/Strategy_video_game`.

[2] Invasion (1972)(Magnavox)(US): *Archive.org* [online]. Brewster Kahle. [vid. 2016-05-13]. Dostupné z: `https://archive.org/stream/Invasion_1972_Magnavox_US#page/n0/mode/1up`.

[3] Válečná mlha: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 19. 10. 2015 v 13:58. [vid. 2016-05-13]. Česká verze. Dostupné z: `https://cs.wikipedia.org/wiki/Válečná_mlha`.

[4] Design Technologies for Self-Organizing Cities: *Cityminded.org* [online]. [vid. 2016-05-13]. Dostupné z: `http://cityminded.org/design-technologies-for-self-organizing-cities-7100`.

[5] Dirk Helbing. *Social Self-Organization.* Berlín: Springer-Verlag Berlin Heidelberg, 2012. ISBN 978-3-642-24003-4.

[6] MCV March 16th: *Yudu.com* [online]. [vid. 2016-05-13]. Dostupné z: `http://content.yudu.com/A1w07v/MCV160312/resources/46.htm`.

[7] Jiří Bittner. *Umělá inteligence pro hry.* [přednáška][online]. Praha: ČVUT, 18. listopadu 2015. Dostupné z: `https://cent.felk.cvut.cz/predmety/39PHA/data/prednasky/07-ai.pdf`.

[8] Melanie Mitchell. *An Introduction to Genetic Algorithms.* Cambridge, Massachusetts: Massachusetts Institute of Technology, 1996. ISBN 0-262-13316-4.

[9] Introduction to genetic algorithms: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html.

[10] Ing. Tomáš Řehořek. *Evoluční výpočetní techniky, genetický lagoritmus, genetické programování.* [přednáška][online]. Praha: ČVUT, LS 2015/16. Dostupné z: https://edux.fit.cvut.cz/courses/BI-ZUM/_media/lectures/05-evolution-v4.0.pdf.

[11] Simple Reflex Agents: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/sra.html.

[12] Model based reflex agents: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/mbra.html.

[13] Goal based agents: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/gba.html.

[14] Utility based agents: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/uba.html.

[15] Learning agents: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/learning.html.

[16] Types of Environments: *Imperial College London* [online]. [vid. 2016-05-13]. Dostupné z: https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/environ.html.

[17] Michal Hapala. *CryENGINE etc..* [přednáška][online]. Praha: ČVUT, ZS 2015/16. Dostupné z: https://cent.felk.cvut.cz/predmety/39PHA/data/prednasky/09-cryengine.pdf.

[18] Public relations: *Unity Technologies* [online]. [vid. 2016-05-13]. Dostupné z: https://unity3d.com/public-relations.

[19] Self-organization: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 14. 4. 2016 v 18:33. [vid. 2016-05-13]. Anglická verze. Dostupné z: https://en.wikipedia.org/wiki/Self-organization.

[20] Animating Virtual Characters using Physics-Based Simulation: *Thomas Geijtenbeek* [online]. [vid. 2016-05-13]. Dostupné z: http://www.goatstream.com/research/thesis/index.html.

[21] Ian Millington. Decision trees. *Artificial inteligence for games*. San Francisco, Kalifornie: Morgan Kaufmann publishers, 2006. 342 – 356. ISBN 978-0-12-497782-2.

[22] Ian Millington. State machines. *Artificial inteligence for games*. San Francisco, Kalifornie: Morgan Kaufmann publishers, 2006. 357 – 382. ISBN 978-0-12-497782-2.

[23] Ian Millington. Fuzzy logic. *Artificial inteligence for games. San Francisco*, Kalifornie: Morgan Kaufmann publishers, 2006. 383 – 407. ISBN 978-0-12-497782-2.

[24] Generating Gaussian Random Numbers: *California Institute of Technology* [online]. [vid. 2016-05-13]. Dostupné z: `http://www.design.caltech.edu/erik/Misc/Gaussian.html`.

[25] KÜHNOVÁ, Jitka . *Modely dravec–kořist a jejich počítačové simulace*. Brno: MU 2007. Diplomová práce, Masarykova Univerzita, Přírodovědecká Fakulta. [online]. [vid. 2016-05-13]. Dostupné z: `http://is.muni.cz/th/77952/prif_m/diplomka.pdf`.

[26] Zónový model města: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 21. 11. 2013 v 17:19. [vid. 2016-05-13]. Anglická verze. Dostupné z: `https://cs.wikipedia.org/wiki/Zónový_model_města`.

[27] Benoit Mandelbrot. *The fractal geometry of nature*. New York: Times Books, 1977. ISBN 978-0716711865.

[28] Illinois Institute of Technology. *Examples of differing city structures based on road network strategies* [fotografie]. Procedural City Modeling [online]. Dostupné z: `https://ccl.northwestern.edu/papers/ProceduralCityMod.pdf`. Formát: 1261x421.

Příloha A

Seznam použitých zkratek

FPS Frames per second, počet snímků za sekundu

MMORPG Massively multiplayer online role-playing game, Hra více hráčů o velkém množství lidí s RPG prvky

Příloha B

Uživatelský manuál

B.1 Nastavení programu

Po spuštění programu babylon.exe, lze nastavit rozlišení, kvalitu a zda má být model spuštěn v okně. Po nastavení požadovaných hodnot a spuštění se uživatel dostane do hlavního menu.

B.2 Hlavní menu

V hlavním menu má uživatel několik možností.

- Může vybrat jaké parametry budou použity pro běh modelu. Na výběr má ze tří možností.
  - Use best setUp – k běhu se vybere náhodně vybrané nastavení parametrů ze zadaného počtu
  - Use default setUp – k běhu se použijí defaultní hodnoty parametrů
  - Use user defined – k běhu se použijí uživatelem nastavené hodnoty parametrů
- Parametry modelu lze nastavit v menu otevřených pomocí tlačítek Variables setUp B.3 a Buildings parameters B.4.
- Uživatel může nastavit parametr, kterým se vydělí každé číslo vygenerované pro účely mutace. Čím větší číslo uživatel zvolí, tím méně se budou parametry měnit.
- Dále může uživatel nastavit pravděpodobnost mutace parametrů posuvníkem Randomize probability. Ten nabývá hodnot od 0 (uplně vlevo) po 1 (uplně vpravo).
- V poli Iterations může uživatel nastavit kolikrát se má model spustit s daným nastavením.

- V poli Iterations length se naství kolik dní má maximálně jedna iterace běžet.
- V poli Iteration time scale lze nastavit jakou rychlostí má simulace probíhat. Tento parametr může nabývat hodnot od 0 (čas stojí) po 100.
- Pole Count of best used setUp nastavuje velikost turnaje pro turnajovou selekci.
- Tlačítkem Play se spustí simulace.
- Tlačítkem Quit se program ukončí.

B.3 Variables SetUp

Nastavení parametrů v menu Variables setUp

- Production rate – množství které budova vyprodukuje z jedné jednotky zpracovávané suroviny
- Production period – kolikrát děnně budova produkuje
- Priority multiplier – kolika násobek spotřebovávané suroviny si budova do budoucna vyžádá za každou jednotku zpracované suroviny
- Efficiency needed – jakou musí mít budova efektivitu, aby mohla být postavena další
- Info spread speed – jakou rychlostí, ve čtvercích za snímek, se šíří informace po mapě
- Camera speed – rychlost pohybu kamery
- Zoom speed – rychlost zoomování kamery
- Rotation speed – rychlost rotování kamery
- Days without food – parametr určující jak dlouho vydrží dělník bez jídla než zemře
- Days without water – parametr určující jak dlouho vydrží dělník bez vody/vína než zemře
- Vine or water – dělníci budou preferovat více víno/vodu
- Work or water – dělníci budou raději pracovat a pít víno, nebo se spokojí s vodou
- Pray or build tower – dělníci k uspokojení duchovna radějí upřednostní modlitbu, nebo stavbu věže

- Reproduce growth loss – parametr kterým se násobí touha dělníků množit se
- Life conditions to reproduce – jak dobře se musí mít dělníci, aby se chtěli množit, čím více vlevo, tím menší nároky
- Hunger – rychlost růstu hladu
- Thirst – rychlost růstu žízně
- Religion – rychlost růstu potřeby duchovna
- Reproduce – rychlost růstu potřeby množit se
- Wander – rychlost růstu potřeby prozkoumávat
- Rest – rychlost růstu potřeby odpočívat
- Work – rychlost růstu potřeby pracovat
- Pro uložení změn a opuštění nastavení slouží tlačítko Apply
- Pro opuštění nastavení bez uložení změn slouží tlačítko Cancel

B.4 Buildings parameters

Nastavení parametrů v menu Buildings parameters

- Distance from point – Důležitost vzdálenosti od bodu vyžádání budovy
- Fertility – důležitost úrodnosti půdy
- Fire danger – důležitost nebezpečí požáru
- Flood danger – důležitost nebezpečí potopy
- Distance from stone – důležitost vzdálenosti od zdroje kamene
- Distance from gold – důležitost vzdálenosti od zdroje zlata
- Distance from forest – důležitost vzdálenosti od zdroje dřeva
- Distance from water – důležitost vzdálenosti od vody
- Distance from farm – důležitost vzdálenosti od obilné farmy
- Distance from Mill – důležitost vzdálenosti od mlýna
- Distance from Same – důležitost vzdálenosti od budovy stejného typu
- Distance from Well – důležitost vzdálenosti od studny
- Pro uložení změn a opuštění nastavení slouží tlačítko Apply
- Pro opuštění nastavení bez uložení změn slouží tlačítko Cancel

B.5 V průběhu simulace

V průběhu simulace jsou stále dostupná nástavení Variables setUp B.3 a Buildings setUp B.4, v kterých lze i za běhu měnit chování modelu.

Navíc zde lze měnit rychlost plynutí času posuvníkem v pravém horním rohu v rozsahu od 0 (čas stojí) uplně vlevo po 100 (čas běží 100 násobnou rychlostí) uplně vpravo. Normální rychlost se nachází uprostřed.

Dále lze zobrazit vlajky označující středy znalostních čtverců zaškrtnutím zaškrtávacího políčka Region flags.

V nabídce Map style, lze zvolit podbarvení terénu na základě zvoleného parametru.

Tlačítko Find cityHall přesune kameru nad radnici.

Tlačítkem Exit to Main Menu se uživatel přesune zpět do menu.

V levém přehledu jsou vypsány suroviny, jejich počet ve skladu a jejich priorita vyjadřující požadovaný počet suroviny k vytěžení. Dále je zde uveden počet živých dělníků. Počet mrtvých dělníků se nachází na dalším řádku ve formátu počet zemřelých hladem/počet zemřelých žízní/počet zemřelých stářím. Pod údaji o stavech dělníků se nachazí výpis budov. U každé budovy je vypsán celkový počet pracovních pozic v dané budově. Posledních pět řádků je věnováno fitness, počtu zbývajících iterací a informaci o zbývajícím času právě probíhající iterace uvedené ve dnech.

Jakmile uplyne nastavený počet dní, nebo zemřou všchni dělníci automaticky se spustí další iterace. Po skončení poslední iterace je automaticky uživatel přesunut do hlavního menu.

Informace o nastavení dělníků a jejich fitness se uloží pouze pokud daná iterace doběhla až do konce a nebyla přerušena uživatelem.

B.5.1 Pohyb po mapě

- Uživatel může pohybovat kamerou přesunem kurzoru ke kraji obrazovky. Kamera kuzor následuje. Kamera se však bude držet v mezích určených mapou.
- Kameru lze rotovat okolo svislé osy otáčením kolečka myši.
- Kamerou lze i zoomovat, pomocí tlačítek + a - na numerické klávesnici. Rozsah zoomu je omezený.

Příloha C

Obsah přiloženého CD