Swarm Simulation
C# zápočtový program

Tomáš Karella

17. července 2017

1 Téma

Cílem zápočtového programu bylo implementovat framework pro evoluci robotického swarmu, jenž umožňuje variabilní tvorbu map, robotů a ostatních entit prostředí(paliva, interaktivních překážek...). Na základě vytvořeného prostředí spustí simulaci akcí jednotlivých robotů a ohodnocuje jejich chování. Poskytuje negrafickou multithread simulaci a vizuální vezi pro sledování korektosti simulace a vyvinutého chování.

2 Členění programu:

2.1 SwarmSimFramework

Vlastní framework, definice rozhraní, vlastní kód simulace, příklady scénářů

2.2 Vedlejší projekty

- SwarmSimVisu - Visualizace průběhu simulace, prohlížení vygenerovaných chování.
- SimpleNetworking - Umožňuje provádět simulaci mapy na vzdáleném stroji přes TCP protokol.
- Intersection2D - Implementace jednoduchých průsečíků v 2D prostoru (přímky, úsečky, kružnice).

3 SwarnSimFramework:

3.1 Úvod:

Map je centrální třídou projektu, zajištuje vlastní průběh simulace. Uchovává jednotlivé entity(, roboty, překážky, palivo, minerály), volá vyhodnocení akcí pro aktivní entity(roboty) včetně počítání kolizí a interakcí s mapou(přesuny pasivních entit, vysátí paliva, atd..). Potomci třídy entita reprezentuje objekt v mapě, všechny jsou odděny od stejného předka. V celém projektu se vyskytují 2 tvary entit, konkrétně se jedná o úsečku(sensory, efektory) dále o kruh(roboti, překážky, minerály). Robot zastává pozici aktivní entity pohybující se na mapě a interagující s ostatními entitami. Ke komunikaci, pohybu a změnám se v mapě používá robot efektory a sensory. Sensory se používají ke čtení informací z mapy, tedy vrací vektor čísel representující rozdílné vlasnosti mapy(vzdálenostní sensory, rádiové etc.), zatímco efektory mají opačný účel, tedy ovlivňovat mapu a entity a přijímají vektor čísel jako nastavení konkrétního efektory. Sensor a Efektor jsou implementací rozhraní ISensor a IEffector(viz. další kapitola). Pro simulování chování robotů slouží tzv. "mozek", v programu IRobotBrain, který má za úkol z vektoru ze všech sensorů vytvořit vektor pro všechny efektory a tímto způsobem řídit chování robota.

Vývin jednotlivých mozků je řízen přes experiment, což je pojem implementován v projektu různými způsoby dle požadavků uživatele. (MultiThreadExperiment, Experiment). Jejich společným cílem je nastavení parametrů dané mapy(počet překážek, druhy robotů..), iterace přes simulační kroky, průběh generací mozků, změnu mozků(differenciální evoluce, mutace..). Následně ukládání mezivýsledků a zobrazování postupu daného experimentu.

3.2 Hlavní třídy:

- Sensors - sensory, které čtou data z mapy
- Efektory - interakce s mapou a ostatními entitami
- Entities - Reprezentace jednotlivých entit, které se vyskytují na mapě. Všechny entity jsou odděděny od abstraktního předka **Entity**.
- Experiments - jednotlivé parciální evoluce pro řešení daného úkolu, které počítají s visualizací
- Map - Reprezentace 2D prostředí, kde se všechny entity pohybují, zajištuje kontrolu kolizí a celý průběh simulací
- MultiThreadExperiment - jednotlivé parciální experimenty, optimalizované pro běh na více vláknech bez GUI.
- RobotBrains - Reprezentace jednotlivých mozků implementující interface IRobotBrain
- Robots - konkrétní reprezentace robotů

3.3 Map

Reprezentace 2D prostředí simulace. Mapa je daná obdélníkem o dané velikosti při konstrukci. Během konstrukce také dostává všechny entity ve výchozích pozicích, co se budou v prostředí vyskytovat, také vytvoří jejich klony, aby později bylo možné vrátit mapu do počátečního stavu. Existují 4 základní typy entit v mapě. Jedná se o Robots - aktivní entity(IRobotEntity), na které je při každém kroku mapy, zavolána nejdříve PrepareMove() a dále Move() v náhodném pořadí, aby žádný robot nebyl upřednostěn. PasiveEntities pasivní entity, buď překážky nebo jiné nezpracované materiály. (CircleEntity), FuelEntities- palivo vyskytující se v mapě, pokud je spotřebováno je odebráno z tohoto seznamu. Jako poslední RadioEntities, což je vrstva rádiových signálů, které se počítají jen pro speciální kolize.

MakeStep() je metoda provádějící jeden krok simulace. Mapa charakterizují 4 krajní body A,B,C,D, také aktuální cyklus(počet zavolaných MakeStep()).

- Kolize:
  - Pro CircleEntity vrací bool, zda s něčím koliduje.
  - pro LineEntity vrací průsečík s nejbližším objektem mimo fuel na něj je speciální metoda.
  - pro CircleEntity reprezentující rádiový sensor, vrací slovní všech průsečíku s rádiovými signály.
  - pro CircleEntity existuje metoda CollisionColor, která vrací všechny průsečíky v dosahu CircleEntity.
- SceneMap - konkrétní mapy pro jednotlivé experimenty, zatím jsou dispozici dvě vzorové MineralScene a WoodScene
- Intersection - struktura pro jednotlivé druhy průsečíků z kolizí

3.4 Rozhraní

- IEffector - definuje efektor, který ovlivňuje pohyb a interakce robota s mapou.
  - k jeho použití slouží funkce Effect(float[] settings, RobotEntity robot, Map.map map). Settings určuje, jakým způsobem ovlivňuje robota a danou mapu. Před 1. použitím efektoru je nutné robota připojit, pomocí funkce ConnectToRobot(RobotEntity robot), která nastaví normalizační funkce(= rozsahy a transformace hodnot přicházející od robota)

- ISensor - definuje sensor, který čte prostředí simulace
  - k jeho použití slouží funkce float[] Count(RobotEntity robot, Map.Map map), která dle pozice robota vrátí informace, přečtené z mapy. Druh informací se liší konkrétními implementacemi. Před první použitím jiného robota je nutné analogicky jako u efektoru robot připojit pomocí funkce ConnectToRobot(RobotEntity robot).

- IRobotBrain - definuje mozek robot, tzn. jeho chování. Slouží k transformaci vektoru přicházejího ze sensorů na vektor vstupující do efektorů. K tomuto účelu slouží fce float[] Decide(float[] readValues). Dále každý mozek lze ohodnotit hodnotou Fitness, dle jeho úspěšnosti v simulaci. Každý mozek má vstupní a výstupní velikost (IoDimension), rozsahy hodnot pro výstup a vstup (InOutBounds), převodní funkci z interní hodnot počítání vstupu na hodnoty výstupní (Activation func), umí vytvořit svou čistou kopii (GetCleanCopy), případně se (de)serializovat (z)do json formátu.

- IExperiment - definuje průběh experimentu, ale je vhodný pro visualizační řešení. Obsahuje mapu na které je simulace prováděná. Každé volání MakeStep() provede nejmenší krok simulaci(jeden pohyb každé entity). Experiment musí být inicializován metodou Init(). Pokud experiment dosáhl svého cíle, FinnishedGeneration je nastaven na true.

3.5 Efektory a sensory:

3.5.1 Effectors:

- obsahuje konkrétní implementace efektorů, všechny třídy jsou odděděny od IEffector. Jedná se o efektory určené pro vzorové scénáře. Mohou být rozšířené skrz IEffector.

- MineralRefactor - slouží k přeměně minerálů (RawMaterialEnitity) na palivo. Refaktoruje entitu na vrcholu kontejneru robota.
- Picker - implementovaný jako LineEntity, slouží ke zvedání entit, které se protínají s jeho úsečkou. Dále umí na úsečku pokládat entity z vrcholu zásobníku.
- RadioTransmitter - umí vysílat rádiové různé rádiové signály dle nastavení Effect
- TwoWheelMotor - pohybuje s robotem, dle nastavení rychlostí koleček. Fyzikální model, lze najít, zde `http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html`.
- Weapon - dle nastavení může působit poškození robotům, protínající úsečku jeho působnosti.(LineEntity)
- WoodRefactor - slouží k přeměně RawMaterialEnitity, pokud protínají úsečku jeho působnosti(LineEnitity), přímo na mapě. Přeměněná entita tedy nemusí být v kontejneru.

3.5.2 Sensors:

- obsahuje konkrétní implementace sensorů, všechny třídy jsou odděděny od ISensor. Jedná se o sensory pro vzorové scénáře. Mohou být rozšířené skrz ISensor.

- FuelLInSensor - Sensor, který vrací vzdálenost od Fuel, pokud úsečka (LineEntity) nějaké na mapě protíná.
- LineTypeSensor - Sensor, který vrací vzdálenost od libovolné Enity(mimo fuel, rádiové signály) a jeho typ(EntityColor). Pokud nějakou na mapě protíná(LineEntity).
- LocatorSensor - Sensor, který vrací aktuální polohu robota a jeho orientaci vzhledem ke středu robota.
- MemoryStick - Sensor a Efektor v jednom, slouží k zapisování float do paměti. Pokud k němu přistupuji jako k sensoru vrací uložené hodnoty, pokud jako k effektoru, tak ukládá zapisované hodnoty.
- RadioSensor - Sensor, který vrací přečtené signály z okolí a průměr z jejich umístění. Implementován jako CircleEntity.

- TouchSensor - Sensor, který vrací jen binární hodnotu, zda protíná nějakou entitu nebo nikoliv. Implementován jako CircleEntity.
- TypeCircleSensor - Sensor, který vrací binární hodnotu pro každý druh entity(Entity Color), která říká, zda je daná entita v jeho okolí či nikoliv.

3.6 Entity

3.6.1 abstract class Entity

Reprezentuje společného předka a implementuje zakladní společné vlastnosti a metody pro všechna entity pasivní i nepasivní. Definuje vlastnost Color určující účel entit v mapě.

- ObstacleColor
- RawMaterialColor
- FuelColor
- RobotColor
- WoodColor

Dále jsou od Entity odděleny základní dva tvary entit abstraktní třídy CircleEntity a LineEntity, které přidávají konkrétní implementace pohybových funkcí a přidávají některé další vlastnosti.

3.6.2 CircleEntity:

Přepisuje metody MoveTo, RotateRadians pro pohybování kruhu. Přidává vhodné konstruktory.

3.6.3 LineEntity:

Přepisuje metody MoveTo, RotateRadians pro pohybování úsečkou. Přidává vhodné konstruktory.

3.6.4 RobotEntity

Potomek třídy CircleEntity, který tvoří základ pro jednotlivé roboty. Uchovává konkrétní instance efektorů a sensorů, zajišťuje komunikaci mezi nimi a mozkem(i převody jednotlivých rozsahů. Přidává další vlastnosti jako životy, množství paliva, číslo týmu, kontejner(možnost přesouvat a uchovávat ostatní instance třídy Entity).
Některé důležitější metody:

- List<CircleEntity> ContainerList() - robot může mít kontejner na CircleEntities, dané kapacity při vytváření robota, tata metody vratí celý jeho obsah
- PrepareMove(Map.Map map) - na dané mapě provede výpočet na všech sensorech a dané hodnoty předá mozku na zpracování, uloží vstup pro efektory z mozku.
- Move(Map.Map map) - spustí všechny efektory na základě vektoru vypočítaného v předchozí metodě.
- Metody spojené s kontejnerem - PushContainer, PopContainer, PeekContainer

3.6.5 Ostatní CircleEntity:

- FuelEntity - pasivní entita, která reprezentuje nádobu s palivem
- ObstacleEntity - pasivní entita, reprezentující překážky
- RadioEntity - pasivní entita, reprezentující rádiový signál s danou informací
- RawMaterialEntity - pasivní entita, reprezentující nezpracovaný materiál (strom, minerál)
- WoodEntity - pasivní entita, reprezentující zpracovaný materiál vytěžené dřevo

3.6.6 Příklady robotů:

- ScoutCuttorRobot - robot, který je určen pro scénář těžení stromů, obstarává kácení
- ScoutCuttorRobotWithMemory - stejný jako předchozí jen má navíc paměťový slot
- WoodWorkerRobot - robot ze scénáře těžení stromů, obstarává přesun pokáceného dřeva
- WoodWorkerRobotMem - stejný jako předchozí jen má navíc paměťový slot

3.7 Experiment:

Experimenty jsou určeny pro nastavení vývoje mozků. Jedná se o počet iterací jedné simulace mapy, velikost populací, algoritmus, který vytváří nové generace, ohodnocení fitness pro jednotlivé mozky. Jejich hlavním cílem poskytovat třídu, kterou používá GUI nebo konzole a v ní běží všechny simulace. Tomuto konceptu jsou v programu věnovány dvě třídy Experiment(GUI) a MultiThreadExperiment(výkon).

3.7.1 Experiments:

Základem experimentů je abstraktní třída Experimet<T>, kde T je potomek IRobotBrain typ mozku, který chceme vyvíjet. Obsahuje spoustu proměnných pro nastavení prostředí a evoluce. Viz. komentáře u dané třídy. Třída je uzpůsobena na jednotlivé kroky evoluce, aby po nich mohla přijít na řadu visualizace. Slouží výhradně k testování prostředí a výsledných mozků z experimentů.

Jednotlivé experimenty:

- TestingExperiment - debugovací experiment
- WalkingExperiment - debug. experiment
- TestingMap - experiment, který slouží pro nahrání nejlepších mozků a nastavení libovolného prostředí
- WoodCuttingExperimet - Experimenty vzorového scénáře WoodScene - dva druhy robotů (WoodCuttor, WoodWorker) mají za úkol pokácet a odvézt, co nejvíce dřeva na místo označené radiovým signálem.
  - WoodCuttingExperimentWalking - experiment pro evoluci nově vygenerovaných mozků a optimalizovaný na maximum objevených stromů (WoodCuttor)
  - WoodCuttingExperimentCutting - experiment pro evoluci už chodících mozků optimalizovaný na maximum pokácených stromů (WoodCuttor).
  - WoodCuttingExperimentWorkerWalking - experiment pro evoluci nově vygenerovaných mozků a optimalizovaný na maximum objevených zprac. stromů (WoodWorker)
  - WoodCuttingExperimentPickUp- experiment pro evoluci nově vygenerovaných mozků a optimalizovaný na maximum odnosených zprac. stromů (WoodWorker)

3.8 MultiThread

Základem MT experimentů je bstract class MultiThreadExperiment<T>, kde T je potomek IRobotBrain druh mozku, který vyvíjí. Tato třída obsahuje základní nastavení evoluce. (velikost populace, jméno, počet iterací atd..). Před spuštěním fce Run() je potřeba připravit Mapu a modely mozků, robotů pomocí přetížení abstraktní metody Init(). Funkce Run() -

pouští jednotlivé členy aktuální populace každou na jiném vlákně, jejich ohodnocení je implementována pomocí abstraktní metody CountFitness(map). Takto pokračuje napříč všemi generacemi až do poslední. Během běhu serializuje nejlepší mozky, graf(,pokud PC obsahuje GNUplot, tak i vykresluje), všechny mozky(ve zvolených generacích) .
Složky Mineral Scene a WoodScene obsahují vzorové příklady experimentů pro scénáře WoodScene a MineralScene.

3.9 RobotBrains

Třídy definující chování robotů. Základním principem je funkce, která přijme vektor float hodnot a z něj vytvoří jiný vektor float. Vstupní hodnoty předává robot ze sensorů a výstupní hodnoty jsou použity pro nastavení efektorů. Projekt obsahuje 3 základní mozky:

- FixedBrain - mozek, který ignoruje vstup a vrací daný výstup
- Perceptron - základní prvek neuronových sítí (vážený součet) lib. vstup a jeden výstup
- SingleLayerNeuronNetwork - neuronová síť tvořená z perceptronů.

Pro SingleLayredNetwork je připravený evoluční algoritmus Differenciální evoluce, definovaná dle `https://en.wikipedia.org/wiki/Differential_evolution`.

3.10 Externí knihovny, NuGet

- Intersection2D - implementace jednoduchých průsečíků mezi kruhem, přímkou
- MathNet.Numerics - pokročilé matematické funkce, používané v evolučních algoritmech, normální rozdělení apod.
- Newtonsoft.Json - serializace do jsonu
- System.Numerics - reprezentace Vektorú

3.11 Support třídy

- ActivationFuncs - funkce pro převod hodnot ze sensorů do efektorů
- GNUPlot - knihovna pro ovládání programu GNUPLOT
- RandomNumber - statická třída pro volání náhodných tříd
- SupportClasses - ostatní pomocné třídy

4 Ostatní projekty:

4.1 SwarmSimVisu:

Motivací pro tento projekt je sledování, ladění chyb a v neposlední řadě také pozování vyvinutých mozků a chování dokončených experimentů. Pro debugovací účely umí pouštět jednotlivé potomky třídy Experiment, kde lze sledovat průběh experimentů. Na kontrolu vyvinutých mozků je k dispozici Experiment "Testing Brain", kde mohu připravit experiment simulující průběh mapy, kde se pohybují mnou zvolené entity s nahraným serializovaným mozkem. Vykreslování probíhá přes třídu MapCanvas, kde je použita externí třída D2dControl.D2dControl. Centrální třídou je MainWindow, které spouští jednotlivé Experimenty. Instance třídy InfoWindow slouží pro krátké informativní zprávy. Pro sestavení vlastního "Testing Brain"experimentu jsou k dispozici dvě okna BrainSelectionWindow(nastavení globální parametrů simulace), BrainRobotConnectionWindow(připravení mozků a robotů).

4.2 SimpleNetworking:

Jedná se o projekt, který umožňuje spouštět simulaci jednotlivých generačních cyklů na vzdálených počítačích skrze TCP. Oproti normálním konvencím se jedná o jednoho clienta, který pouští na serverch simulaci a přijímá jejich výsledek. K tomuto jsou připraveny dvě třídy ClientTcpCommunicator a ServerTcpCommunicator, kde lze nastavit, zda se má odesílat mozek k ohodnocení či vygenerovat zcela nový. Po navázání spojení s dostupnými servery se odesílá mapa, dále mozek(nebo se generuje), pak běží výpočet na serveru a zpět je odeslána fitness mozku(ů) či celý serializovaný mozek. Pro pomocné třídy a funkce nad TCP slouží statická třída TcpControl.

4.3 InterSection2d:

Jedná se jednoduché průsečíky kruhu, přímek, úseček v 2D prostoru. Projekt cílí na rychlost, neboť se jedná o nejvíce volané třídy z celého projektu.