

Swarm - dokumentace

Tomáš Karella

Obsah

1	Úvod	3
1.1	SwarmSimFramework	3
1.2	Další části:	3
2	Uživatelská dokumentace	4
3	Požadavky	4
3.1	Spuštění	4
3.2	Výstupy:	4
3.3	GUI	6
4	Programátorská dokumentace	13
4.1	Úvod:	13
4.2	Hlavní třídy:	13
4.3	Map	14
4.4	Rozhraní	15
4.5	Efektory a sensory:	16
4.5.1	Effectors:	16
4.5.2	Sensors:	16
4.6	Entity	17
4.6.1	abstract class Entity	17
4.6.2	CircleEntity:	17
4.6.3	LineEntity:	17
4.6.4	RobotEntity	18
4.6.5	Ostatní CircleEntity:	18
4.7	MultiThread	19
4.8	RobotBrains	19
4.9	Externí knihovny, NuGet	19
4.10	Support třídy	19
5	Ostatní projekty:	20
5.1	SwarmSimVisu:	20
5.2	InterSection2d:	20

1 Úvod

Cílem Swarm 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.

1.1 SwarmSimFramework

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

1.2 Další části:

- SwarmSimVisu - Visualizace průběhu simulace, prohlížení vygenerovaných chování.
- Intersection2D - Implementace jednoduchých průsečíků v 2D prostoru (přímky, úsečky, kružnice).

2 Uživatelská dokumentace

3 Požadavky

Aplikace podporuje operační systém Windows. Pro její spuštění je potřeba mít nainstalovaný .NET Framework alespoň verze 4.7. Instalace verze 4.7 se nachází ve složce `\bin\dep`.

3.1 Spuštění

Spuštění hlavní konzolové aplikace je možné pouze s následujícími parametry:

- *-de konfigurační_soubor.expe* - spustí diferenciální evoluci s danou konfigurací
 - bez dalších parametrů vygeneruje první generaci náhodně
 - očekává jména souboru s serializovanou první generací, za každého robota jeden soubor
- *-es konfigurační_soubor.es* - spustí evoluční strategie s danou konfigurací
 - první generace jsou nastaveny v konfiguračním souboru
- *-stats nastavení_mapy.stats* - spustí 100 simulací a na standartní výstup vypíše průměrné charakteristiky mapy

3.2 Výstupy:

Výstupy evolučních algoritmů se kvůli paralelizaci liší.

-de

V místě běhu si program vytvoří pracovní složku (její název je uveden v konfiguračním souboru) a po skončení optimalizace přidá do místa běhu serializovanou poslední populaci, kdy se vygeneruje pro každý druh robota jeden soubor *jméno_experimentuBrain[pořadí_roboty].json*.

Pracovní složka obsahuje soubory s serializovanými nejlepšími jedinci, soubory mají název *[číslo_generace]bestbrain[pořadí_roboty].json*. Dále soubor s výpisem fitness pro vytváření grafů, každý řádek má podobu *[číslo_generace];fitness*

Na standartní výstup jsou vypisovány průběžné statistiky fitness, očekávaný čas ukončení, apod.

-es

Také v místě běhu vytvoří pracovní složku s názvem dle konfiguračního souboru. Ovšem jejich struktura je odlišná, protože každý jedinec běží v jiném vlákne. Každý jedinec má v pracovní složce svou vlastní složku s číslem vlákna a soubor s výpisem fitness pro graf ve stejném formátu jako u -de.

Jednotlivé podsložky, pak obsahují serializovaného jedince z každé generace a 20 nejlepších jedinců. Nejlepší jedinci pak mají název *best/pořadí_robota/-název_robota* a chování z generací *[číslo_generace]-název_robota*.

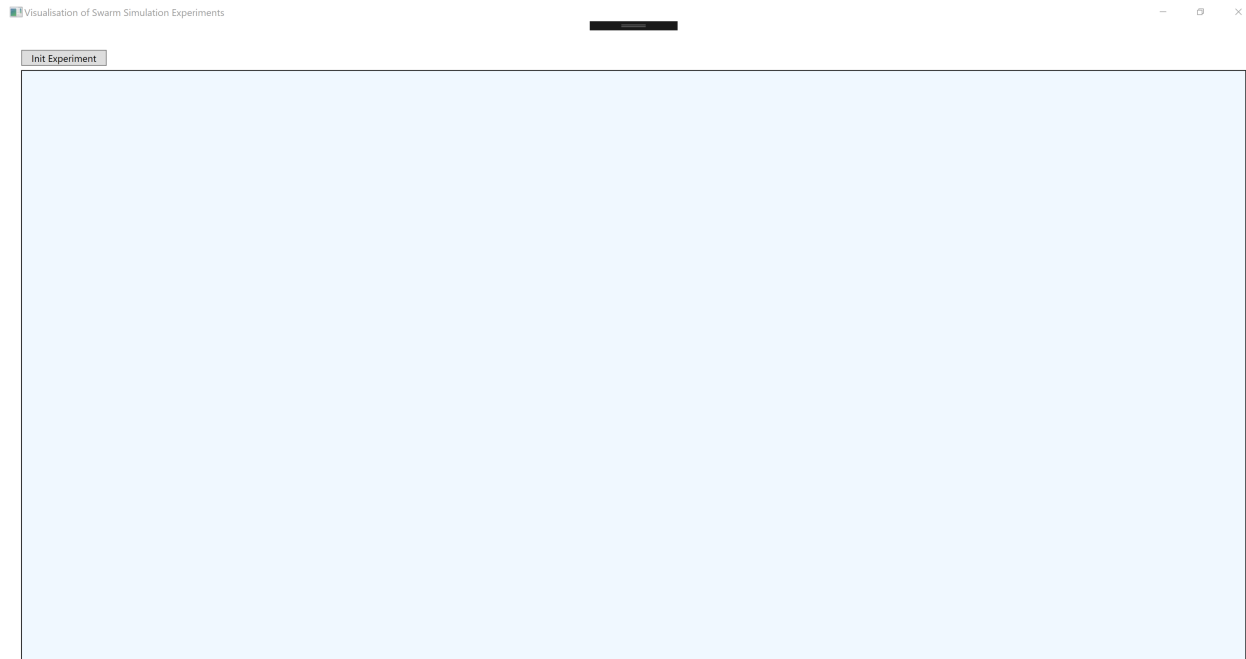
Na standartní výstup jsou vypisovány průběžné informace o aktuální populaci a číslu iterace jednotlivých vláken.

-stats

Vypisuje pouze na standartní výstup průměrné charakteristiky ze 100 simulací daných konfiguračním souborem. Každý řádek pak odpovídá jedné charakteristice a za : je uvedena jejich hodnota.

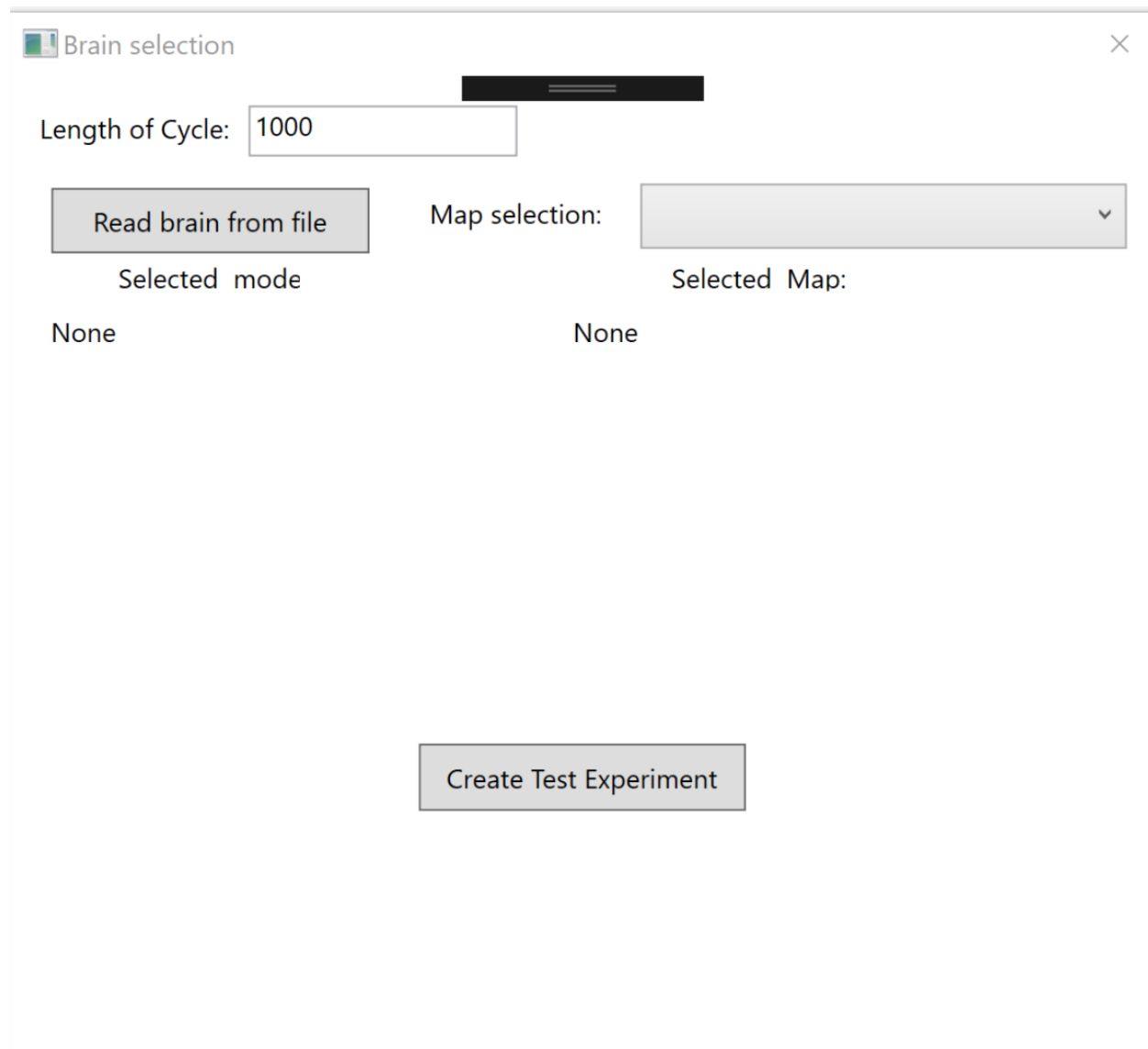
3.3 GUI

Program poskytující vizualizaci chování jednotlivých robotů se jmenuje *SwarmSim Visu*. Po jeho spuštění se objeví úvodní okno.



Obrázek 1: GUI - Úvodní obrazovka

Tlačítkem `init` začneme přípravu experimentu, v následujícím nastavení vybereme požadovanou mapu a roboty, které si chceme prohlédnout. V *Length of Cycle* můžeme zvolit délku simulace (počet iterací mapy).



Brain selection

Length of Cycle: 1000

Read brain from file

Map selection:

Selected mode

Selected Map:

None

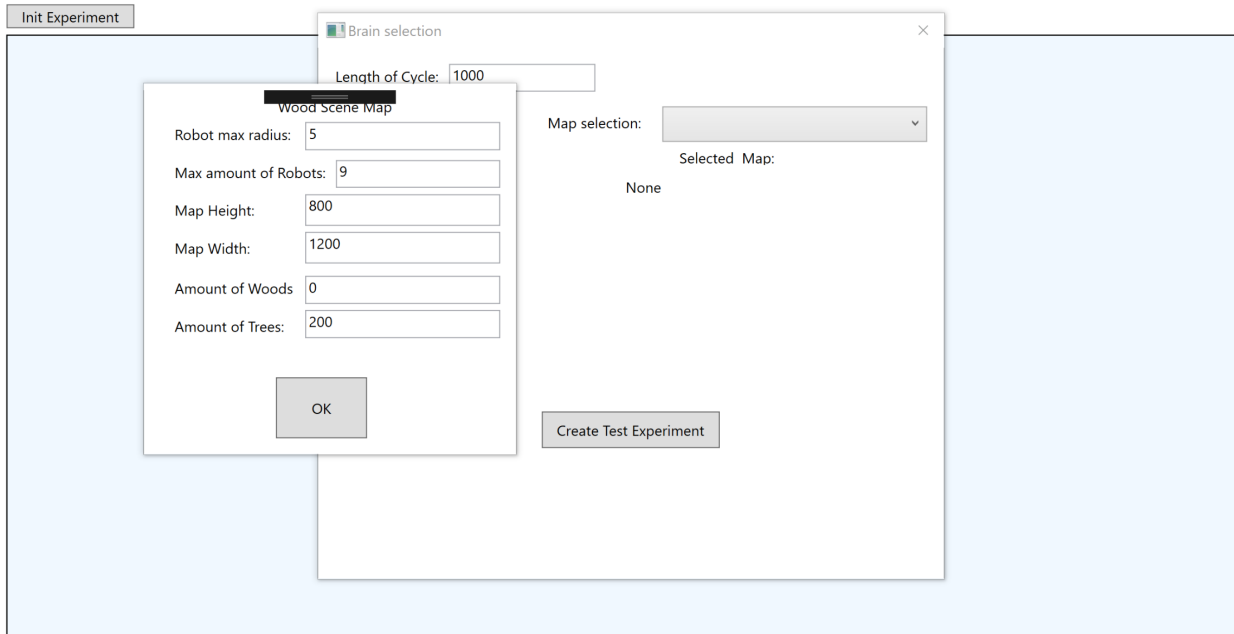
None

Create Test Experiment

Obrázek 2: GUI - vytvoření experimentu

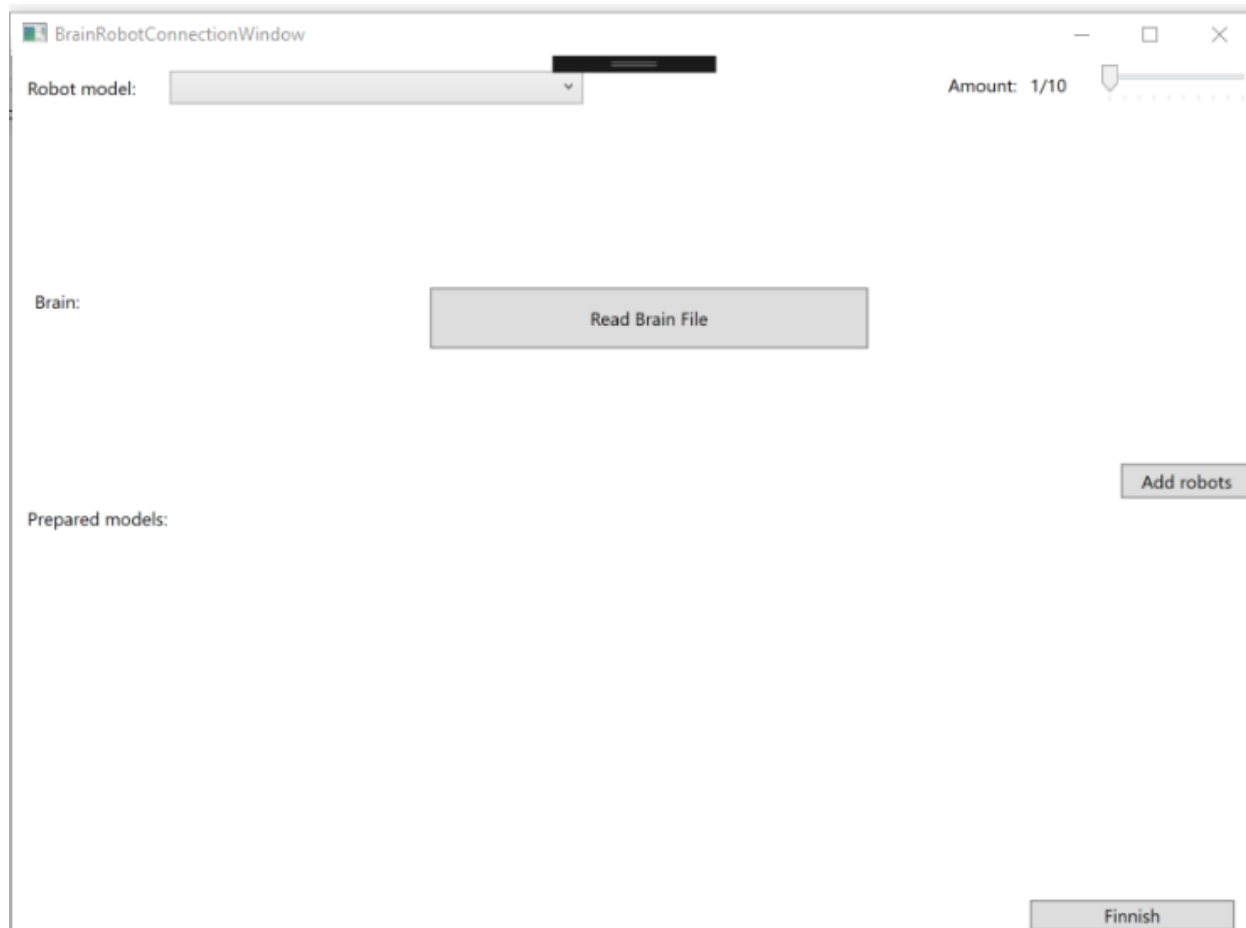
Na vybrané mapě můžeme nastavit počty jednotlivých objektů na mapě. Ostatní hodnoty jsou pouze informativní.

Visualisation of Swarm Simulation Experiments



Obrázek 3: GUI - nastavení mapy

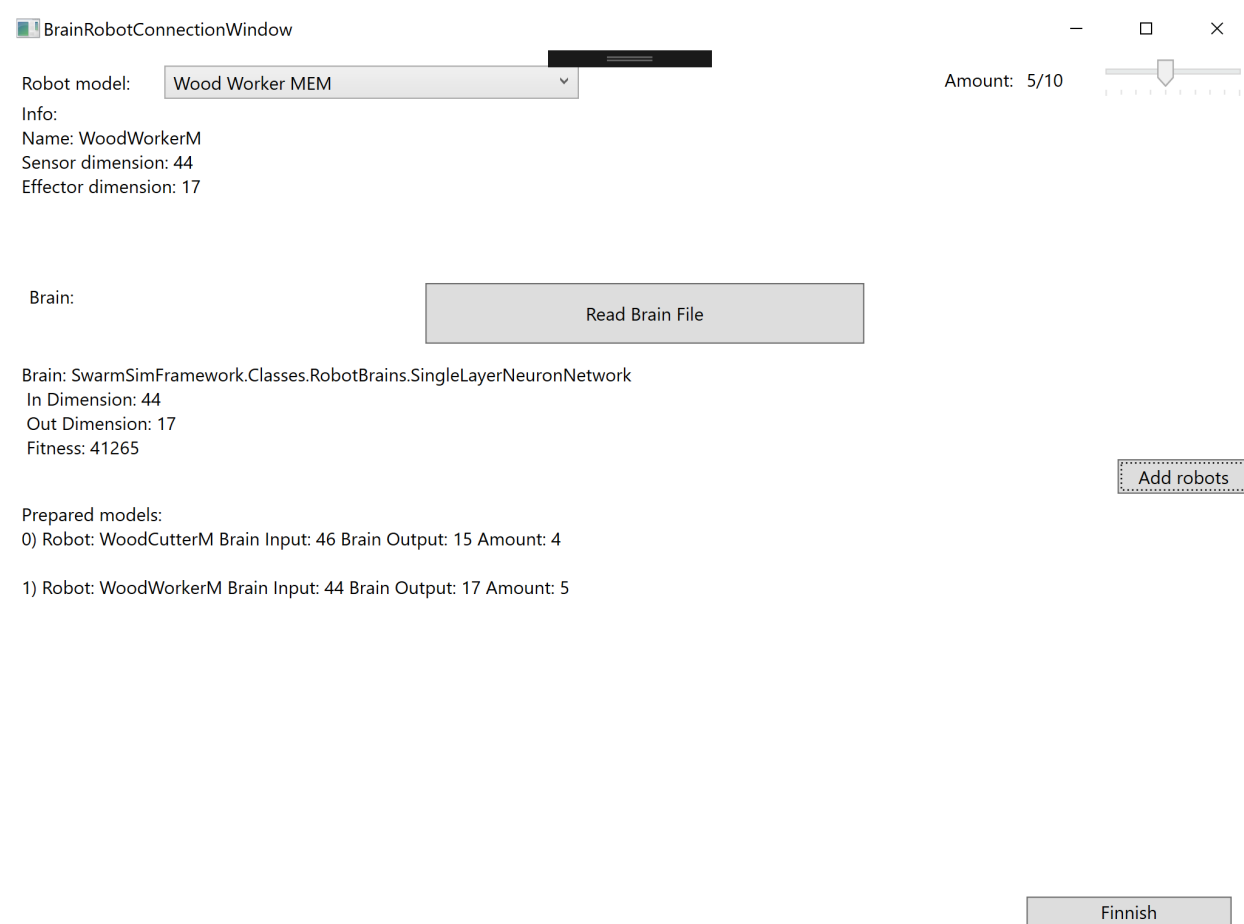
Dále je potřeba zvolit roboty a propojit je s robotickým mozkem (neuronovou sítí). K čemuž slouží tlačítko *Read brain from file*.



Obrázek 4: GUI - přidání robotů s ovládáním

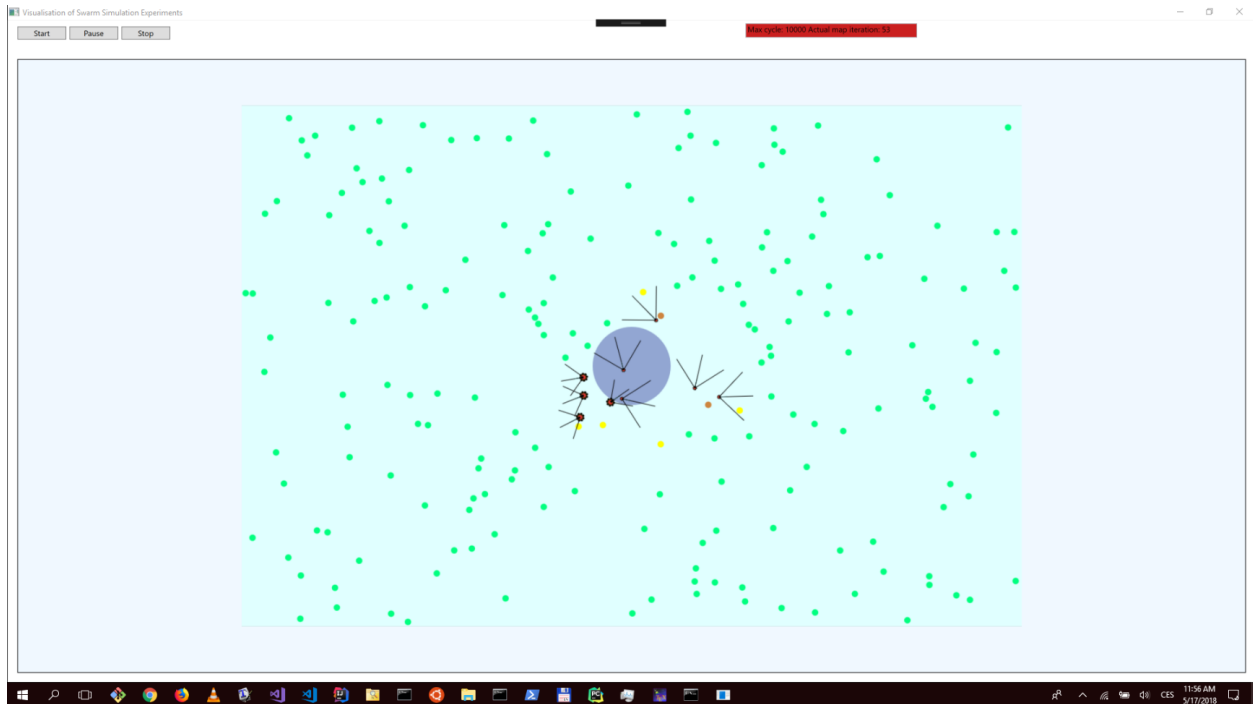
V tomto okně můžeme nastavit druh robota *Robot model* a jejich počet v mapě. Tyto zvolené roboty musíme spojit s kompatibilním chováním pomocí tlačítka *Read Brain File*, které otevře okno pro výběr soubor a je potřeba zvolit soubor s jedním serializovaným mozkem. Pokud je mozek kompatibilní k robotovi, objeví se jeho detaily po nápisem *Brain*. Pak jen stačí přidat je do simulace tlačítkem *Add robots* (pozor na limit počtu robotů v simulaci je kontrolován až na konci propojování).

Vyplněné okno pro projování vypadá následovně. Projování s roboty je potřeba potvrdit tlačítkem *Finnish*.



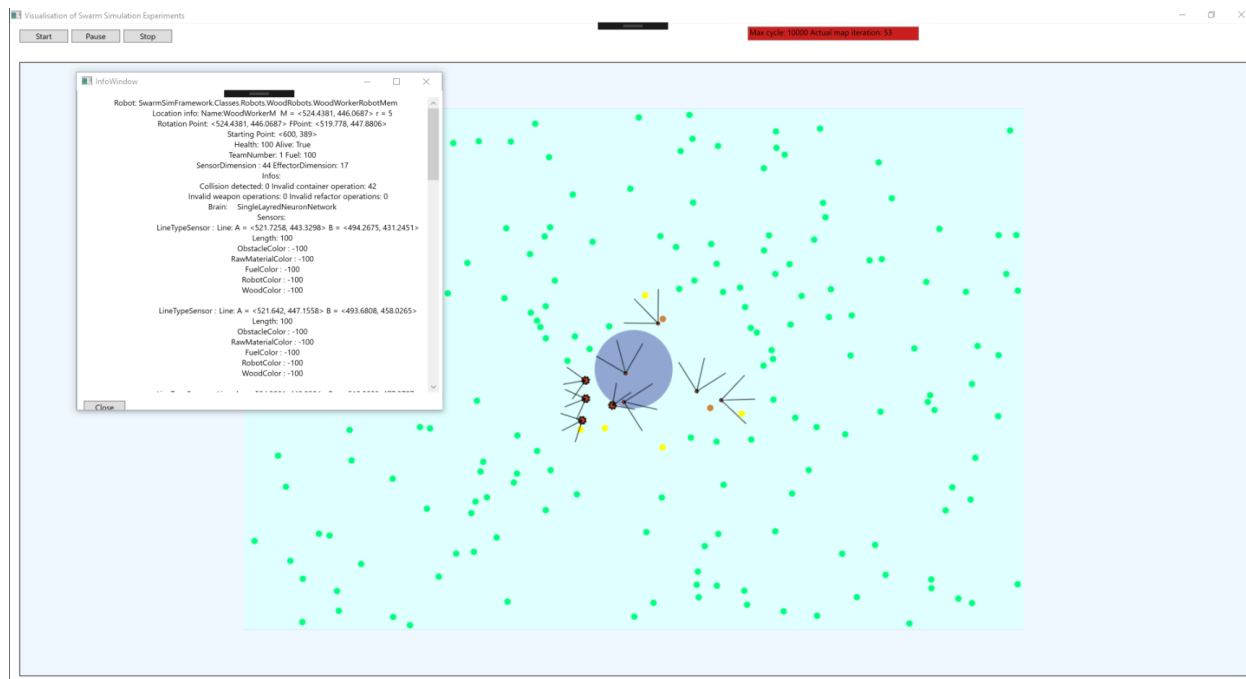
Obrázek 5: GUI - vyplněné přidání robotů s ovládáním

Nyní je vše připraveno pro pozorování simulaci, po stisknutí tlačítka *Create Test Experiment* se inicializuje námi specifikovaný experiment. V levém horním rohu se nacházejí ovládací tlačítka *Run* spustí simulaci, *Pause* pozastaví simulaci, *Stop* ji úplně zastaví.



Obrázek 6: GUI - začátek experimentu

Při pozastavené simulaci si můžeme pravým kliknutím na robota o něm zobrazit podrobné údaje.



Obrázek 7: GUI - dodatečné informace

4 Programátorská dokumentace

4.1 Úvod:

Map je centrální třídou projektu, zajišťuje 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 reprezentující rozdílné vlastnosti 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 (diferenciální evoluce, mutace..). Následně ukládání mezivýsledků a zobrazování postupu daného experimentu.

4.2 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 odděněny od abstraktního předka **Entity**.
- Experiments - jednotlivé parciální evoluce pro řešení daného úkolu, které počítají s vizualizací
- Map - Reprezentace 2D prostředí, kde se všechny entity pohybují, zajišťuje kontrolu kolizí a celý průběh simulací. Také obsahují jednotlivé části evolučních algoritmů pro vícevláknový běh evolučních strategií.
- MultiThreadExperiment - jednotlivé parciální experimenty, optimalizované pro běh na více vláknech bez GUI.
- RobotBrains - Reprezentace jednotlivých mozků implementující interface IRobotBrain
- Robots - konkrétní reprezentace robotů

4.3 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. PassiveEntities 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 MineralScene, WoodScene, CompetitiveScene
- Intersection - struktura pro jednotlivé druhy průsečíků z kolizí

4.4 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 robota připojit pomocí funkce `ConnectToRobot(RobotEntity robot)`.
- **IRobotBrain** - definuje mozek robot, tzn. jeho chování. Slouží k transformaci vektoru přicházejícího ze sensorů na vektor vstupující do efektorů. K tomuto účelu slouží funkce `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 vizualizační 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, `FinishedGeneration` je nastaven na `true`.

4.5 Efektory a sensory:

4.5.1 Effectors:

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

- MineralRefactor - slouží k přeměně minerálů (RawMaterialEntity) 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://rosum.sourceforge.net/papers/DiffSteer/DiffSteer.html>.
- Weapon - dle nastavení může působit poškození robotům, protínající úsečku jeho působnosti.(LineEntity)
- WoodRefactor - slouží k přeměně RawMaterialEntity, pokud protínají úsečku jeho působnosti(LineEntity), přímo na mapě. Přeměněná entita tedy nemusí být v kontejneru.

4.5.2 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 Entity(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 efektoru, 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.

4.6 Entity

4.6.1 abstract class Entity

Reprezentuje společného předka a implementuje základní 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.

4.6.2 CircleEntity:

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

4.6.3 LineEntity:

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

4.6.4 RobotEntity

Potomek třídy CircleEntity, který tvoří základ pro jednotlivé roboty. Uchovává konkrétní instance efektorů a sensorů, zajišťuje komunikaci mezi nimi a mozkiem(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, tato metoda vrátí 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 efektor z mozku.
- Move(Map.Map map) - spustí všechny efektor na základě vektoru vypočítaného v předchozí metodě.
- Metody spojené s kontejnerem - PushContainer, PopContainer, PeekContainer

4.6.5 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

4.7 MultiThread

Základem MT experimentů je abstract 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 vlákne, 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 `MineralScene` a `WoodScene` obsahují vzorové příklady experimentů pro scénáře `WoodScene` a `MineralScene`.

4.8 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 `SingleLayerNeuronNetwork` je připravený evoluční algoritmus `Diferenciální evoluce`, definovaná dle https://en.wikipedia.org/wiki/Differential_evolution.

4.9 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 Vektorů

4.10 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

5 Ostatní projekty:

5.1 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).

5.2 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.