Evolution

Prokop Divín

July 14, 2023

Contents

1 Uživatelská část

1.1 O programu

Program přečte předpřipravenou složku kde je nadefinované prostředí a zvířata v něm. Následně toto prostředí a chování zvířat v něm odsimuluje. Pokud simulace skončí vygeneruje program textový soubor, kde je zaznamenaný průběh simulace (vytištěná ASCI mříž znázorňující stav simulace po každém dnu) a csv soubor, kde je záznam s průměrnými hodnotami zvířat.

1.2 Pravidla

Zvíře musí během dne jíst jídlo nebo ostatní zvířata, pokud je na konci dne jeho hlad větší než snědlo jídla tak zemře. Pokud snědlo dost jídla, to znamená několikrát více než je jeho hlad, tak se rozmnoží. Pokud se rozmnoží vytvoří se nové zvíře, kterému se změní jeho vlastnosti(výška, síla smyslů=dohled, pohyblivost) o "mutovací" parametr.

Vlastnosti zvířete ovlivňují jeho hlad, to jak moc záleží na vybrané rovnici hladu.

1.3 Ovládání

Program je konzolová aplikace, která bere dva parametry. První parametr je cesta k souboru ze vstupními daty. Druhý je název soubory s výstupními daty. Z názvu souboru pro vstupní data se vytvoří dva soubory. Soubor s příponou ".csv", kde jsou zaznamenané statistiky z jednotlivých dnů simulace a soubor s příponou "_log.txt", kde je vytisknutý stav simulace po každém dnu.

1.4 Soubor s parametry

Soubor s parametry je textový soubor co udává definici prostředí. Pokud v něm chceme použít komentář, pak cokoliv po " #" až do konce řádku nebude aplikace vnímat.

V souboru můžou být definice tří věcí a to hlavičky, jídla, a zvířete. Více definicí jídla znamená více druhů jídla, a u zvířat obdobně.

1.4.1 definice hlavičky

Toto je příklad zapsání definice hlavičky v souboru. Kde každému parametru patří komentář vysvětlující význam.

```
*head  # označuje začátek definice hlavičky
days 50  # počet dnů simulace, totéž jako počet generací

map_hight 20  # výška mapy
map_width 25  # šířka mapy
nutritions_divider 2 #- udává kolik  zvířeti zbude z jídla po odečtení jeho
                     #  hladu (živiny co zbyly zvířeti se tímto číslem vydělí)
calculator 2 3 4 5 # 2- číslo funkce, která bude počítat hlad
                   # 3- číslo které určuje jednotky parametru "size" u zvířete
                   # 4- číslo které určuje jednotky parametru "sense" u zvířete
```

```
                    # 5- číslo které určuje jednotky parametru "dexterity" zvířete
---------  # ukončuje definici hlavičky
```

calculator x

Můžeme si vybrat ze čtyř funkcí, kde size, dexterity, sense jsou vlastnosti zvířete, kterému funkce počítá hlad.

-funkce 0 je inspirovaná vzorcem pro kinetickou energii.

$$hlad = (\frac{size}{size_r})^3 * (\frac{dexterity}{dexterity_r})^2 + \frac{sense}{sense_r}$$

-funkce 1 nezvýhodňuje žádný z parametrů.

$$hlad = \frac{size}{size_r} + \frac{dexterity}{dexterity_r} + \frac{size}{size_r}$$

-funkce 2 zdražuje cenu za sílu smyslů, jinak je jako funkce 0.

$$hlad = (\frac{size}{size_r})^3 * (\frac{dexterity}{dexterity_r})^2 + (\frac{sense}{sense_r})^2$$

-funkce 3 odpovídá tomu, váha zvířete není závislá na 3 mocnině jeho výšky ale na 2.

$$hlad = (\frac{size}{size_r})^2 * (\frac{dexterity}{dexterity_r})^2 + \frac{sense}{sense_r}$$

Pokud zadáme tedy parametr:

```
calculator 1 2 3 4
```

a bude se počítat hlad zvířete s vlastnostmi $size = 4, sense = 6, dexterity = 8$ pak výpočet hladu bude vypadat následovně.

$$hlad = \frac{4}{2} + \frac{6}{3} + \frac{8}{4} = 6$$

1.4.2 definice jídla

Toto je příklad zapsání definice jídla v souboru. Kde každému parametru patří komentář vysvětlující význam.

```
*plant    #musí být první, označuje začátek definice

name p1   # jméno pod jakým uvidíme zvíře v simulaci, musí mít právě dva znaky
  # kvůli tisku mapy
size 1.01  # pokud zvířete chce rostlinu sníst musí mít větší výšku.

count 25   # počáteční počet rostlin
nourishment 3  # je kolik živin poskytne jeden kus teto rostliny
changer 2 0.85 25 10  # 2..každé dva dny se změní maximální počet jídla,
              #  do kterého se doplňuje jeho počet na konci dne
              # 0.85 ...krát se sníží
```

```
           # 25 je maximální počet, kterého může dosáhnout
           # 10 ...je miniální počet kterého může dosáhnout
       # při překročení jedné ze dvou  hranic se jeho počet už
           # nemění

----- # musí končit  řádkou s alespoň jedním "-"
```

1.4.3 definice zvířete

```
*species  # musí být první, označuje začátek definice

name a1 # jméno pod jakým uvidíme zvíře v simulaci, musí mít právě dva znaky
        # kvůli tisku mapy

size 3.5   # výška zvířete
sense 4.2    # jak velký má přehled o okolí. Měří se stejně jako dexterity
dexterity 5   # a je o kolik polí se pohne za den
reproduction 1.2   # kolikrát víc živin potřebuje aby se rozmnožil (než je jeho hlad)
mutation 0.1   # a je náhodný prvek,číslo od 0-1.
            # Určuje jak moc se budou měnit statistiky při rozmnožení
            # k dané vlastnosti zvířete se přičte  nebo odečte jeho násobek

food 1.2 2 # 1.2 kolikrát se násobí živiny ze zvířat, když ho sní
   #  je kolikrát se násobí živiny z rostlin
count 10    # počet jedinců tohoto druhu  na začátku simulace

stop_eat 1.5  # maximalní počet jídla, které může zvíře sníst se vypočítá takto:
               #  hunger*reproduction*stop_eat....hunger*reproduction*1.5

------------  # musí končit řádkou s alespoň jedním "-"
```

1.5 Soubor s průběžnými daty

Je csv soubor, kde je pro každý den záznam. Jeden záznam je jeden řádek. Pro každou rostlinu tady najdem počet jedinců na začátku dne. Pro každý druh zvířete tady najdem, počet jedinců na konci dne, průměrnou hodnotu parametrů "Size", "Sense", "Dexterity".

Zde je příklad:

	day	0_plant_count	1_plant_count	0_species_size	0_species_sense	0_species_dexterity	0_species_count
2	0	25	25	3.5	4.2	5	10
3	1	25	25	3.475	4.23	4.96429	14
4	2	21	17	3.50875	4.2021	4.7975	20
5	3	21	17	3.48906	4.13963	5.00938	16
6	4	17	11	3.49708	4.2595	5.07083	12
7	5	17	11	3.72721	4.31865	4.92542	12
8	6	14	7	3.75725	4.3827	4.86875	8
9	7	14	7	4.03384	4.3555	5.76263	4
10	8	11	4	4.30558	4.5045	5.929	6
11	9	11	4	4.1503	4.3659	6.13067	6
12	10	9	4	4.3197	4.49064	6.48862	8
13	11	9	4	4.26997	4.29005	6.43372	12
14	12	9	4	4.61014	4.48989	7.12085	10
15	13	9	4	4.82125	4.71173	7.05958	14
16	14	9	4	5.12621	4.87376	6.83857	12
17	15	9	4	5.21468	4.9351	6.82532	14
18	16	9	4	5.37487	5.53864	7.09091	8
19	17	9	4	4.94518	5.36708	7.60793	8
20	18	9	4	4.98336	5.65118	7.8844	10
21	19	9	4	5.22188	5.70415	8.07101	12
22	20	9	4	5.04958	5.10996	8.13047	8
23	21	9	4	4.89117	4.44473	8.35658	8
24	22	9	4	5.5108	4.42686	9.18796	4
25	23	9	4	5.43254	4.37878	9.45323	6
26	24	9	4	5.45832	4.84624	10.3985	6
27	25	9	4	5.1959	5.10143	9.0722	2
28	26	9	4	5.1959	5.10143	9.0722	2
29	27	9	4				0
30	28	9	4				0
31	29	9	4				0
32	30	9	4				0
33							

2 Programátorská část

2.1 Rozmnožování

Jestliže nějaké zvíře sní dostatek jídla, to znamená jeho "*hlad*" $*$ "*reproduce*", kde "*hlad*" se vypočítá pomocí funkce zvolené v parametru "calculator" a "reproduce" je číslo zvolené pomocí stejnojmenného parametru, tak se rozmnoží. Rozmnožování probíhá tak že se od daného jedince vytvoří kopie a jeho parametry "size", "sense", "dexterity" se změní a to tak, že se od každého náhodně odečte, nebo přičte jejich jejich procentuální část určená parametrem "mutation". Nový jedinec se umístí na náhodné místo na mapě. Z tohoto důvodu jsou všechny tři vlastnosti reprezentované desetinným číslem. Kdyby nebyly nemohli by se postupně měnit při malé míře mutace a tak by zůstávaly stejné. Krom "size" se při používání vlastnosti zaokrouhlují.

Původně se měli vždy křížit dva jedinci zprůměrovat jejich vlastnosti a pak dále postupovat stejně. Ale to by zpomalovalo vývoj simulace, což jde zařídit i zmenšením parametru "mutation". Také by byl potřeba větší prostředí, které se hůře nastavuje.

2.2 Pohyb

Pohyb každého zvířete je rozdělen do tří částí kde se ve třech kolech zavolá na každé instanci nějakého zvířete funkce **MakeStep**, která vrátí souřadnice na které pole se má zvíře pohnout. Počet kol se může změnit změnou hodnoty statické proměnné **rounds** ve třídě **Animal**.

Při každém tomto kole může zvíře sníst nějaké jídlo nebo se k němu přiblížit a urazit zhruba třetinu maximální vzdálenosti kterou může ujít. Pokud není **dexterity** dělitelné třemi, pak o zbytek navíc se pohne v jeho posledním tahu.

2.3 Třídy

2.3.1 Simulation

Tato třída řídí běh simulace má za úkol její načtení spuštění a uložení výsledků. Tyto úlohy jenom řídí, jejich provedení mají na starosti další třídy.

2.3.2 ImputReader

Má za úkol načíst vstupní soubor a předat parametry simulace získané z něj. O kontrolu parametrů a jejich parsování se stará třída Chacker a její potomci. Pro každou sekci vstupního souboru je jeden potomek třídy Chacker. U každé sekce se kontroluje počet, typ parametrů a typ jejich hodnot.

2.3.3 Entities

V souborech ImputEntities.cs a Map.cs jsou třídy pro reprezentaci věcí v prostředí a pomocné struktury. Nějaké z nich tady budou zmíněny. K tomu aby jídlo a zvířata mohla být uložena ve stejné datové struktuře musí mít společného předka. Tímto předkem je třída **Item**.

Třída **Food** je potomkem třídy **Item**, představuje jednu instanci jídla na mapě má v sobě odkaz na instanci třídy **PlantSpecies**, která je pro každý druh zvířete jen jedna a uchovává informace o tomto druhu jídla. Proměnná **count** ve třídě PlantSpecies, je počet na jaký se jídlo doplní na konci dne. Proměnná **count** se může v průběhu času měnit, parametry k tomu jsou uloženy ve struktuře **changer**.

Třída **Animal** je také potomkem třídy **item** a představuje jednu instanci zvířete na mapě. Má v sobě odkaz na instanci třídy **AnimalSpecies**, která uchovává informace o druhu zvířete.

2.4 API

Hlavním datovou strukturou je dvojrozměrné pole nullable instancí třídy **Item**, jmenující se **map**. Tento objekt reprezentuje dvojrozměrnou mapu prostředí ve kterém se bude odehrávat simulace. Poloha jednotlivých položek je určena souřadnicemi. Jednotlivé položky ukazují na objekt reprezentující zvíře, nebo jídlo. Dvě věci nemohou být ve stejné poloze pokud na poloze nic není je zde null.

Objekt **map** je ve třídě Map.cs.

2.4.1 Map

Třída dědí od **MapLoader**, kde jsou funkce pro načítání parametrů a počáteční inicializaci Mapy.

Tato třída je jádrem programu. Má v sobě hlavní datovou strukturu **map**,a uchovává všechny informace načtené ze vstupního souboru. Všechny informace o druzích jídla jsou uloženy v listu **plants** a všechny informace o druzích zvířat jsou v listu **species**, kde jsou instance třídy **AnimalSpecies**. Pokud chce instance třidy **Animal** zjistit informace o svém druhu, pak má v sobě odkaz na příslušnou instanci. Tyto informace jsou v programu oddělené, protože k jedné instanci **AnimalSpecies** může odpovídat mnoho instancí **Animal**. Informace ve **AnimalSpecies** jsou společné pro všechna zvířata daného druhu.

Obdobně je to s třídou **Food**, kde také každá její instance má v sobě ukazatel na instanci třídy **PlantSpecies**. Tento odkaz je jedinou datovou položkou třídy **Food**. Bylo by možné se rovnou odkazovat ze všech míst na mapě, kde má být jídlo rovnou na instanci třídy s informacemi o druhu. Tímto řešením by se ušetřila jedna referencea třída **Food** by nemusela existovat, Stávající řešení je zvoleno, protože nabízí lepší možnost budoucího rozšíření funkcí rostlin na mapě.

Na rozdíl od jídla se vlastnosti každého jedince stejného druhu můžou měnit, proto je zde třída **Animal** nezbytná, aby se nemusely do každé instance zvířete kopírovat vlastnosti patřící jejímu druhu.

2.4.2 Funkce Day

Odsimuluje jeden den simulace. Začne zavoláním funkce **moveAnimal**, která pohne se všemi zvířaty

Potom co pohne se všemi zvířaty, tak se pro každé zvíře vypočítá hlad a rozhodne se jestli přežije, zemře nebo se rozmnoží. Na úplný konec se upraví maximální počet pro všechna jídla a pak se do tohoto počtu každý druh jídla dorovná

2.4.3 moveAnimal, a pohyb zvířete

Tato funkce třikrát po sobě projde souřadnice všech zvířat a zavolá na nich funkci **MakeStep** a pohne s nimi na souřadnice získáné z této funkce. Pokud nějaké zvíře sní jiné, tak se přemístí na jeho místo. To znamená, že tyto souřadnice budou do nového průchodu zařazeny dvakrát, ale tento problém řeší to že se souřadnice ukládají do HashSetu .

Funkce **MakeStep** je součástí třídy **Animal** a vrací souřadnice, na které se zvíře může pohnout. Všechny podmínky zda je tento tah možný se kontrolují v této funkci. Takto jsem to udělal protože jsem chtěl nechat co největší volnost při vytváření nového typu zvířete s jiným způsobem pohybu nebo prohlížení okolí. To znamená, že třídě **Animal** je předaná reference na hlavní vektor map s mapou prostředí.

Ve funkci **MakeStep** pomocí funkce **SearchAround** prohledá okolí ale jen do velikosti proměnné **sense** a najde největší a nejbližší zdroj zdroj jídla, který může sníst. Pokud je jídlo v dosahu vrátí jeho souřadnice. Pokud není v dosahu tak zavolá funkci **GoCloser** která vrátí souřadnice prázdného pole které je blíž. Pokud v okolí není jídlo, tak pomocí funkce **RandomEmptyBox** vrátí souřadnice prázdného místa kde se může pohnout a které jsou co nejdál. Ke generování souřadnic které zvíře prozkoumává jsem využil iteratoru.

2.4.4 Statistic

Tato třída chovává data a manipuluje s daty pro uložení. V programu je jediná její instance a to v třídě Simulation, která jí dá povel k uložení dat.

2.5 Testy

Všechny testy jsou ve složce tests. V programu hraje velkou roli náhoda. Náhodně se rozhoduje umístění nových zvířat, jídla a změna parametrů při rozmnožování. Proto se Generátor náhodných čísel inicializuje vždy číslem 43 aby byly testy zopakovatelné.

2.5.1 Neplatné vstupy

Tyto testy jsou ve složce errorinputs, mají za úkol otestovat, že program na špatný vstup nespadne a vhodně odpoví. Po každém z těchto testů by měl program skončit s chybovou hláškou a kódem 0. Testy testují, špatné nastavení parametru případně chybějící parametry.

2.5.2 Platné vstupy

Na všechny tyto vstupy má program po nějaké době skončit a vytvořit textový soubor se zaznamenaným průběhem simulace a csv soubor s výsledky. Při větších rozměrech mapy doporučuji otevírat textový soubor v editoru který automaticky nezalamuje příliš dlouhé řádky. Tyto testy také testují funkčnost komentářů a dalších věcí které by neměli mít na doběhnutí programu vliv. Jako je například rozdílné pořadí sekcí.

Test **race.txt a race2.txt**

U těchto testů jsou dva druhy zvířat a dva druhy jídla. Jedno můžou jíst jakákoliv zvířata a na druhé musí jedno zvíře alespoň trochu vyrůst, druhý druh je dost velký od začátku. Počet jídla se postupně v průběhu simulace zmenšuje.

Ve výsledcích by měl být vidět trend, kdy se vyšší zvíře zmenšuje a nižší trochu vyroste a poté zůstává stejně velké. v prvním testu druhé zvíře přežije. Zatím co v druhém testu ne, protože je zde požadavek pro minimální výšku i na druhý druh rostliny.

Test **test4.txt, test1.txt, test2.txt, test3.txt**

Tyto testy jsou identické až na druh calculátoru. Můžeme tedy vidět vliv výpočtu hladu na prostředí.

Test **big.txt**

V tomto testu je mapa 100x100, 3 druhy rostlin a 5 druhů zvířat. Jeho doběhnutí může chvíli trvat.

2.6 Kam dál

Dále by se dalo zlepšit zpracování výsledků, a to například tak že se podpoří grafickým zpracováním. K tomu může posloužit csv soubor. Dalším způsobem rozšíření by mohlo být přidání grafického rozhraní nebo nového druhu zvířete s jiným druhem pohybu.

3 Závěr

Při programová mě překvapil rozsah, očekával jsem, že bude program kratší. Přesto, že je program hotov, dalo by se do něj přidat ještě další věci. Jako je například grafické zpracování výsledků nebo přidání zadávání vstupních parametrů rovnou z uživatelského rozhraní. Dále by se dalo více experimentovat s funkcí pro pohyb zvířat a celkově ze způsobem simulování dne. Toto nebylo součástí zadání, je to jen podnět k budoucím pracím, nebo k plánu na dlouhé zimní večery. Další věcí, která by se dala přidat je udělat pohyb zvířat vícevláknově, což byl i původní plán, ale místo více vláknového programování jsem jako technologii z pokročilého c# použil nullable typy a iterator.