СУ “Св. Климент Охридски”,
ФМИ – Софтуерно инженерство
Курсов проект по Обектно-ориентирано програмиране

ECOSYSTEM

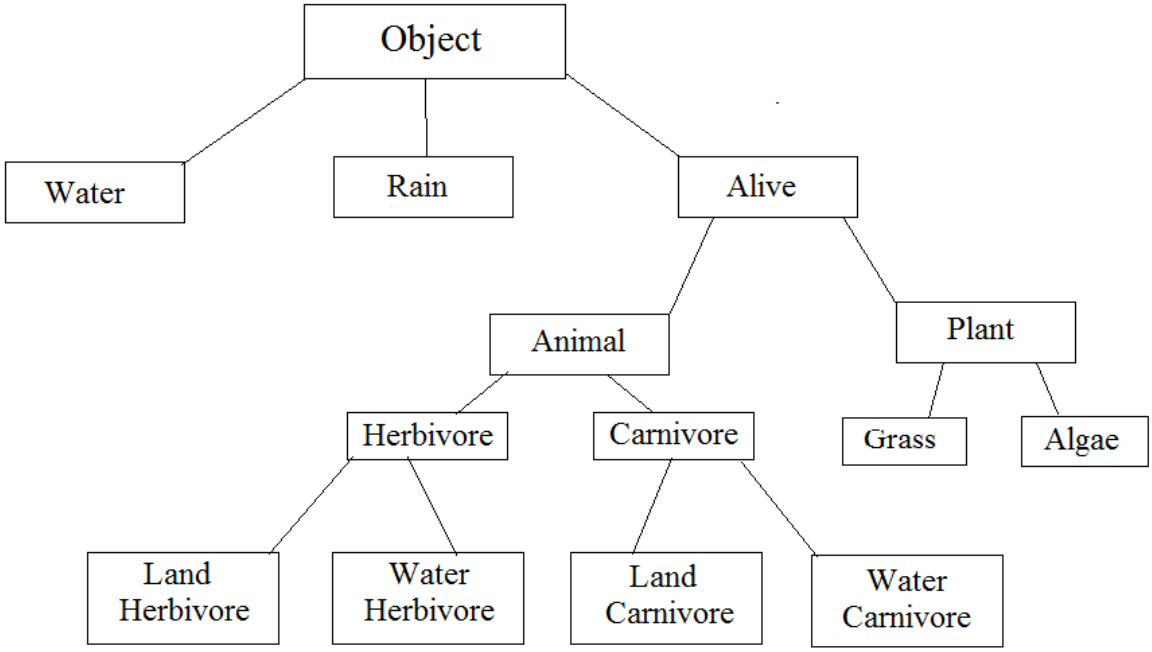
Мария Тодорова Чолакова,
ФN№61934

Съдържание

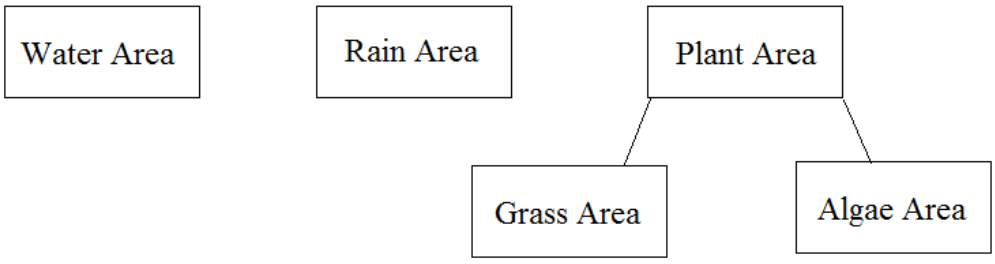
1.Въведение

Програмата представлява груба симулация на екосистема. В нея са включени различни класове за различните елементи на екосистемата – животни, растения, вода, дъжд. Симулацията се извършва на „ходове“ като един ход завършва, когато всеки един от компонентите на системата извърши действията си. Входните данни се четат от файл, а на конзолата се извеждат съобщения, за да се проследят процесите в екосистемата.

2.Йерархия на класовете



И отделно



В програмата също така има клас Engine, който управлява ходовете, h. и cpp. файлове за функциите, които четат входните данни, и h. файл с ексепшъните.

3. Описание на програмния код

3.1. Класът Object

В класа Object е дефинирана структурата Coordinate, която всички обекти надолу в йерархията притежават. Също така в Object се пазят размерите на полето (редовете и колоните) като статични променливи. Конструкторът на Object е protected, защото няма да създаваме обекти от този клас, а от неговите наследници.

3.2. Класовете Water и Rain

Обектите от класовете Water и Rain са съответно ъглите на водните и дъждовните зони. Всъщност те нямат никаква функционалност, но са създадени за по-прегледен вид, т.е. горният ляв и долният десен ъгъл на Water Area е от тип Water, а горният ляв и долният десен ъгъл на Rain Area е от тип Rain. Конструкторът на Rain не приема стойности, защото те се генерират на случаен принцип в конструктора на Rain Area.

3.3. Класът Alive

Обектите от класа Alive имат нова характеристика – жизнени точки. Поради това имат и нова функционалност – намаляване на жизнените точки, проверка дали обектът е все още „жив“ и търсене на Manhattan distance до друг „жив“ обект. Последната се използва при търсенето на жертвата на животните (във функцията FindTarget). Функцията CheckIfAlive() е виртуална, защото при растенията се проверява само дали жизнените точки са <= 0, а при животните се проверяват и годините.

3.4. Класът Plant

Това е абстрактен клас, който съдържа само чисто виртуалната функция TakeRain()=0, имплементирана в децата на Plant- Grass и Algae.

3.5. Класовете Grass и Algae

В тях е включена само функцията TakeRain().

3.6. Класът Animal

Абстрактният клас Animal наследява публично Alive, който наследява публично Object. Така всяко животно взима променливата за координатите от дядо си – класа Object, а променливата за жизнените точки от баща си – класа Alive. Но обектите от класа Animal имат нови характеристики и нова функционалност. Новите променливи – възраст, продължителност на живота, количество храна, което изяждат, пол, брой полета, които могат да изминат – са protected, за да се виждат от децата на Animal (за да не пишем сетъри и гетъри за всяка от тях). Освен тях обаче животните имат променлива target от тип пойнтер към Alive и променлива location от тип habitat(enum). Функциите за движението, стареенето, размножаването, проверка дали са във вода или на суша и дали са живи са

еднакви за всички животни, затова са имплементирани тук. Но функцията за храненето е чисто виртуална и се имплементира по различен начин в Herbivore и Carnivore.

3.7. Класът Herbivore
Той наследява Animal и има две функции – за намиране на най-близкото растение и за хранене. Функцията FindTarget приема като параметър вектора от растителните зони. По този вектор тръгва един цикъл, който за всяка зона проверява дали координатите на животното попадат в нея. Ако да – target приема растението, което е на същите тези координати, и цикълът се прекратява. Ако не, т.е. ако животното не е върху растение, се търси най-близката тревна площ и след като тя бъде открита target приема най-близката до животното точка от тази площ. Има вариант, при който всички растения от тревната площ да са „мъртви“, но зоната да съществува (например когато тревната площ е само 4 точки и животното е изяло всички от тях, или завали два-три пъти дъжд над едни и същи водорасли). Тогава координатите на животното ще попадат в някоя от тревните площи, но то пак няма да е върху растение. Затова се проверява дали target-а е „жив“ и ако животното е върху растение, но това растение не е живо, се търси нова най-близка тревна площ в масива от растителни зони, като настоящата се пропуска. „Мъртвата“ зона не може да се премахне от масива с растителни зони, което би улеснило нещата, защото ако завали над трева нейният капацитет се пълни на максимум и тази зона пак ще може да се използва от растителноядните. Функцията Eat() проверява дали на животното му липсват жизнени точки и само в този случай го кара да яде.

3.8. Класът Carnivore
Той наследява Animal и има три функции – за намиране на най-близкото растителноядно, за хранене и за биене. Функцията FindTarget приема като параметър вектор от всички тревопасни животни –сухоземните и водните. По този вектор тръгва един цикъл, който за всяко тревопасно проверява дали координатите на хищника съвпадат с неговите. Ако да – target приема растителноядното, което е на същите тези координати, и цикълът се прекратява. Ако не, т.е. ако хищникът не е върху животно, се търси най-близкото такова. Функцията Fight приема друг хищник и намаля жизнените точки и на двата хищника.

3.9. Класовете Land Herbivore, Water Herbivore, Land Carnivore, Water Carnivore
В тях няма нищо ново освен, че в конструктора им се дава стойност на променливата location от тип habitat. За сухоземните имаме location= land, а за водните- location = water.

3.10. Класът Rain Area
В конструктора на Rain Area на случаен принцип се генерират координатите на горния ляв и долния десен ъгъл на зоната. Те са съобразени с размерите на полето.

3.11. Класът Water Area
В този клас са имплементирани две булеви функции. Първата проверява дали дадени координати принадлежат на зоната. Тя се използва в класа Animal във функцията, която проверява къде се намира животното, и при четенето на животните от файла за определяне дали са водни или сухоземни. Втората функция проверява дали входните данни за зоната са коректни и хвърля ексепшън, ако не е така.

3.12. Класът Plant Area
Този клас, освен ъглите на зоната, има като променлива и вектор от растения(растителни точки от координатната система). Той също има функции за проверка на входа и дали дадени координати се намират в зоната. Освен това има и функция за проверка дали тревната площ се засича някъде с дъждовна зона. А функцията WaterOverlappedArea приема параметър дъждовна зона, за която по-рано е установено, че се засича с растителната, и за тези точки – растения, които принадлежат и на двете зони, вика тяхната функция TakeRain().

3.13. Класът Grass Area
Той наследява Plant Area и в конструктора му векторът от растителни точки се пълни с обекти от тип Grass. Този клас има и булева функция, която проверява дали тревната зона се засича с водна зона. Тази проверка е необходима при входа на данните и ако той не е правилен се хвърля ексепшън.

3.14. Класът Algae Area
Той наследява Plant Area и в конструктора му векторът от растителни точки се пълни с обекти от тип Algae. Този клас има и булева функция, която проверява дали зоната с водорасли е изцяло във вода. Тази проверка е необходима при входа на данните и ако той не е правилен се хвърля ексепшън.

3.15. Класът Engine
Този клас съдържа всички елементи на екосистемата, като те са сложени във вектори. Имаме вектори за водните зони, дъждовните зони, растителните зони; вектори за сухоземните тревопасни, водните тревопасни, сухоземните хищници и водните хищници. Във функциите HerbTurns() и CarnTurns() се обхождат масивите от животни и се викат техните функции за движение, хранене, растене. Във функцията HandleHerbivoreMeeting() се проверява дали две тревопасни, които са на една и съща точка, са от различен пол и ако да – те се размножават. Функцията HandleCarnivoreMeeting() също проверява дали два хищника са с

еднакви координати, но тогава те или се размножават, или се бият (зависи от пола им). Накрая на всеки ход се премахват умрелите животни(ако има такива) от съответните вектори. И ако всички вектори са празни, т.е. няма оцелели животни, симулацията приключва преждевременно.

4.Използвани технологии

Езикът, на който е написан проектът, е C++.

Използваното IDE е CodeBlocks.