Homework #2: Virtual World at FaceDuck

CSE4065: 소프트웨어 공학 @ Hnaynag Univ.

배지운 2015004584

1. 명세

명세에서는 주어진 “Virtual World”를 완성시키기 위해 필요한 주요 구현과 제약을 설명하고 있었다. Rabbit, Fox, Gnat과 같은 Actor들이 규칙에 맞게 행동하도록 정의해야 하며, 주어진 skeleton의 구조를 변경하지 않으면서 높은 재사용성과 캡슐화를 위한 디자인과 설계를 진행해야 한다.

1. 디자인 및 구현

Actor들을 상속하는 Actionable 추상 클래스를 정의했다. 추가 구현한 Bear, Hunter도 Actionable을 상속받는다.

* 1. Actionable

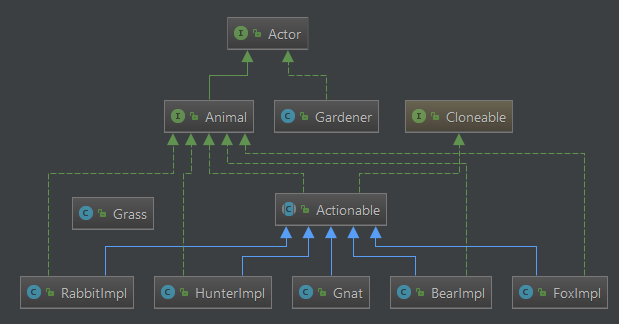
Actionable은 Actor들을 일반화 한 추상 클래스로 필요한 대부분의 기능을 일반화 하여 작성했다. Actionable은 행동을 할 때마다 시야에 들어온 다른 객체들을 바탕으로 다음 행동을 추론한다. Actionable은 act함수가 실행되면 propagation을 호출하여 이전 기억들을 토대로 다른 객체들의 현재 위치를 추론하고 주어진 시야에 있는 객체들을 behold를 통해 업데이트 한다. forgetfulness를 설정하여 기억이 얼마나 오래가는 지 결정한다. 시야 안의 객체들은 judge를 통해 수치화 된다. 이렇게 저장된 기억을 토대로 evaluate를 통해 각 위치의 선호도를 평가한다. 이렇게 평가된 위치들을 기반으로 decide는 가장 좋은 위치와 그에 따른 행동을 결정한다. Actionable은 이렇게 Actor의 의사 결정을 일반화 하여 처리함으로 각각 클래스에 해당하는 AI 클래스가 별도로 필요하지 않다. 하지만 명세에 각 AI 클래스를 구현하라는 조건이 있었으므로 형식적으로 AI 클래스의 act에 Actionable의 decide를 호출하도록 변경해 두었다.

Figure 1 Actionable과 자식 클래스

* propagation은 다른 객체들의 다음 위치를 예상한다. 사방으로 움직이거나 정지해 있다고 가정한다.
* behold는 시야에 들어온 다른 객체들의 위치를 기억한다.
* Judge는 인식한 객체를 수치화 한다. 기본적으로 먹을 수 있는 객체에 높은 점수를 준다.
* evaluate는 해당 지역이 얼마나 좋은 지 평가한다. 덮어써 평가 방식을 변경할 수 있다.
* decide는 기억을 토대로 최선의 행동을 결정한다.
* getForgetfulness를 덮어써 객체의 기억력이 얼마나 오래 가는지 결정한다.

또한 Actor가 할 수 있는 breed, move, eat과 같은 행동들을 정의하여 각각의 Actor에서 구현하지 않아도 된다. decide가 Command를 결정하는 과정에서 실행 가능한지를 isPossible을 통해 확인함으로 무결성을 보장함으로 각 행동은 단순히 객체를 world에서 지우고 생성한다.

Actionable은 Cloneable을 상속받아 clone을 통해 energy의 절반을 나눠 주면서 자식을 생산할 수 있다.

* + 1. Gnat

1의 시야와 대기 시간을 가지고 임의의 위치로 이동한다. judge로 모든 객체를 0으로 평가합니다.

* + 1. Rabbit

3의 시야와 2의 대기시간을 갖는다. 먹을 수 있는 Grass에 높은 점수를 주고 energy가 적을수록 더 높은 점수를 부여한다. 먹힐 수 있는 Fox, Bear, Hunter에게는 적은 점수를 주고 배가 부를수록 더 낮은 점수를 준다.

* + 1. Fox

5의 시야와 3의 대기시간을 갖는다. 먹을 수 있는 Rabbit에 높은 점수를 주고 energy가 breedLimit보다 작을 때 더 높은 점수를 준다. 먹힐 수 있는 Bear, Hunter에게 적은 점수를 주고 배가 부를수록 더 낮은 점수를 준다.

* + 1. Bear

7의 시야와 5의 대기 시간을 갖는다. 먹을 수 있는 Rabbit과 Fox에게 energy가 없는 정도에 따라 높은 점수를 준다. Hunter와는 경쟁 관계로 maxEnergy의 절반 이상일때는 높은 점수를 주어 사냥하려 하지만 아니라면 낮은 점수를 주어 피해 다닌다.

* + 1. Hunter

6의 시야와 4의 대기 시간을 갖는다. 먹을 수 있는 Rabbit과 Fox에게 각각 먹어서 획득 할 수 있는 energy만큼 점수를 준다. Bear와 경쟁 관계로 maxEnergy의 절반 이상일때는 높은 점수를 주어 사냥하려 하지만 아니라면 낮은 점수를 주어 피해 다닌다. Hunter는 사냥을 통해 부상을 얻어 자식을 낳을 수 없다.

* 1. Pair

두 개의 서로 다른 혹은 같은 값을 하나의 변수에 담기 위해 만들어진 클래스다. skeleton의 Location은 Integer 두개를 갖지만, 종종 다양한 타입이 필요하여 만들었다. equlas와 hashCode를 지원한다.

* 1. Utility

유용한 함수들을 모아둔 클래스로 모든 함수가 static으로 선언되어 있다. 싱글톤 패턴을 고려해 보았으나 의미상 적절하지 않아 static method로 설정하였다. 사방이 막혀 있는지 판단하는 isClosed, 인접한 타일 중 하나를 임의로 선택하는 randomAdjacent 등이 구현되어 있다.

* 1. Actors

모든 Actor를 표현할 수 있는 열거형이다. toString을 지원하여 쉽게 문자열로 표현할 수 있고 recognize를 통해 임의의 오브젝트를 어떤 형태의 Actor인지 쉽게 표현할 수 있다.

* 1. Action

모든 Command를 표현할 수 있는 열거형이다. command를 통해 Command로 쉽게 변환할 수 있다.

* 1. Recognizable

Actionable에서 judge에 사용할 수 있는 기본 척도인 값을 표현한 열거형이다. EDIBLE, COGNATION, IRRELEVANT, NEMESIS, UNKNOWN이 있고 각각 100, -5, -25, -100, 0의 값을 가진다.

1. 향상된 기능

기본 구조 외에 추가적으로 많은 기능들을 구현하였다. 모든 향상된 기능들은 @Custom Improve라는 주석으로 표시되어 있다.

* 1. Util

skeleton.util 패키지 안의 구현을 쉽게 사용할 수 있도록 조금 변경하였다. skeleton변경 없이 custom.util에서 해결할 수 있으나, 같은 작업을 위해 두 개 이상의 클래스가 간섭하는 것이 좋지 않은 디자인 같아서 변경하였다. 변경 사항은 크지 않으며 기능적 확장이 주를 이룬다. 해당 변경을 통해 20%이상의 불필요하고 의미가 맞지 않는 코드를 줄일 수 있었다.

* + 1. Direction

Location과 같이 사용하는 경우가 많은데 매번 Direction에 따라 x, y좌표를 따로 지정해 주는 것은 좋지 않은 디자인이라고 판단하여 열거형 자체가 x, y값을 가지도록 변경했다. 또한 STAY 상태를 추가하여 5가지 방향으로 변경했다. 이렇게 함으로써 Iteration을 좀 더 수월하게 실행하거나 불필요한 코드들을 생략할 수 있게 되었다.

* + 1. Location

Direction을 개선함으로써 Location과 Direction을 받는 생성자의 코드를 반절 이상 줄일 수 있었다. 또한 다른 Location과 같은 Location인지 판단하는 equals를 덮어씌워 구현했다. x, y 값이 모두 같으면 참을 반환한다.

* 1. LogUI

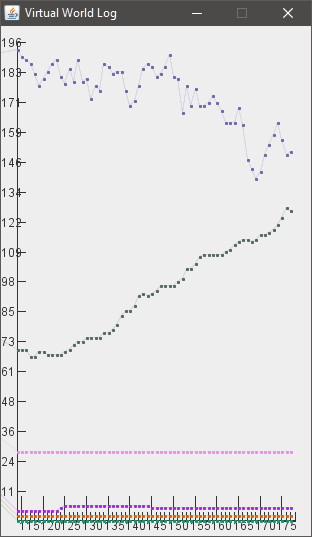
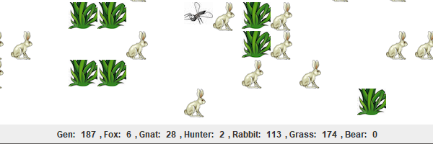
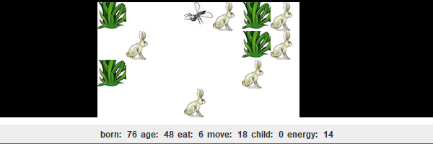
World의 Actor들의 추세를 가시적으로 쉽게 확인할 수 있는 로그 창을 만들었다. Custom.UI.LogUI에 작성되어 있으며 JPanel을 상속받아 구현하였다. 초기화 될 때 정보를 수집할 Actor들을 정한다. 로그를 수집하기 위해 World에 현재가 몇 세대인지 반환하는 getGeneration과 현재 Actor들의 숫자를 반환하는 getCount를 추가하였다. WorldImpl에는 Actor들의 숫자를 파악하기 위해 add와 deletion에 해당 Actor를 추적하는 코드를 삽입하였고 세대를 측정하기 위해 step마다 증가하도록 설정했다. WorldPanel에는 생성자 부분에 로그를 기록하고 그래프를 업데이트 하기 위해 람다를 등록하는 stepConsumer를 만들고 WorldUI에 setStepConsumer를 통해 매 step마다 world에서 값을 가져오고 LogUI에 값을 전달하는 람다 함수를 설정했다.

Figure 2 LogUI

그래프 사이즈는 196 \* 64로 설정하였고, 최근 64세대만 표시한다.

오른쪽에 보이는 그래프의 위에서부터 각각 Grass, Rabbit, Gnat, Fox, Hunter, Bear를 나타낸다. 시각적은 그래프로 표현되어 쉽게 추이를 확인할 수 있고 실험을 통해 적절한 안정 상태를 만드는데 사용될 수 있다.

* 1. Information Panel

하단에 간단한 정보를 보여줄 수 있는 라벨들을 추가했다. 기본 상태에서는 현재 세대 정보가 각 Actor의 개체 수와 함께 나타난다. Actionable객체를 선택하게 되면 해당 객체의 상세 정보가 나온다. born은 객체가 생성될 당시의 세대이고, age는 생성된 이후로 지난 세대 수, eat, move, child는 각각 어떤 행동을 했는지 나타내며 현재 energy도 표시된다. 빈 공간이나 Actionable이 아닌 객체를 클릭하면 기본 상태로 돌아갈 수 있다. 구현을 위해 WorldUI에 새로운 라벨을 추가하였고, WorldPanel에 trackConsumer를 만들어 선택한 객체를 추적하면서 정보를 받아올 수 있도록 구현하였다.

* 1. Actionable Sight

Figure 3 하단 패널

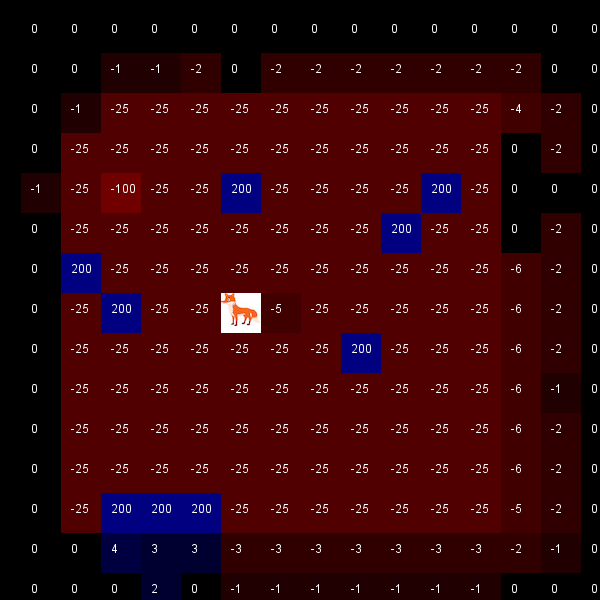
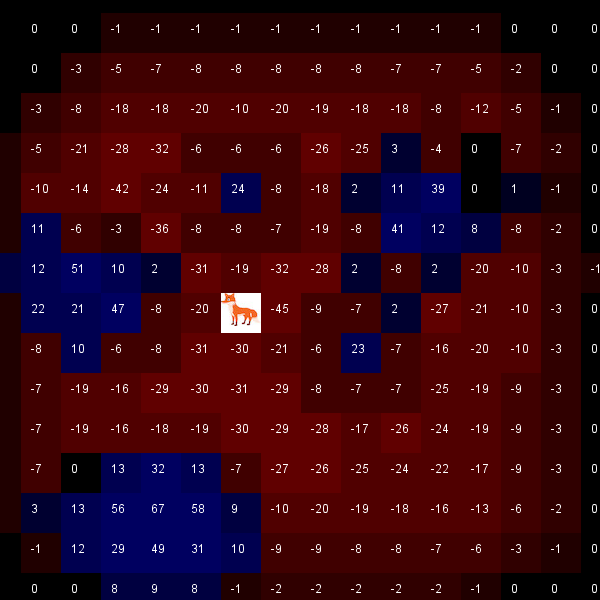
Actionable의 입장에서 잘 판단하고 있는지 확인 하기 위해 만든 Visual모드 입니다. VIEWRANGE, WEIGHTS, EVALUATED 3가지 모드를 지원하며 SightMode에 작성되어 있습니다. Actionable 객체를 선택하게 되면 먼저 VIEWRANGE모드가 되고 해당 객체가 볼 수 있는 범위만큼만 볼 수 있게 됩니다. WEIGHTS모드는 해당 객체가 기억하고 있는 memory를 보여줍니다. Actionable 객체는 behold를 통해 memory를 계속 업데이트 합니다. 마지막으로 EVALUATED모드는 각 지역을 실제로 Actionable의 evaluate를 통해 평가한 값을 보여줍니다. 이 값이 높은 쪽으로 객체가 움직이게 됩니다.

Figure 4 WEIGHTS 모드

Figure 5 EVALUATED 모드

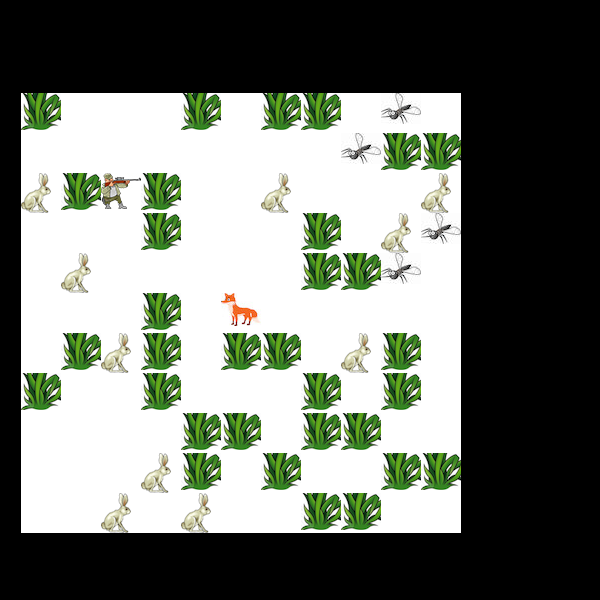
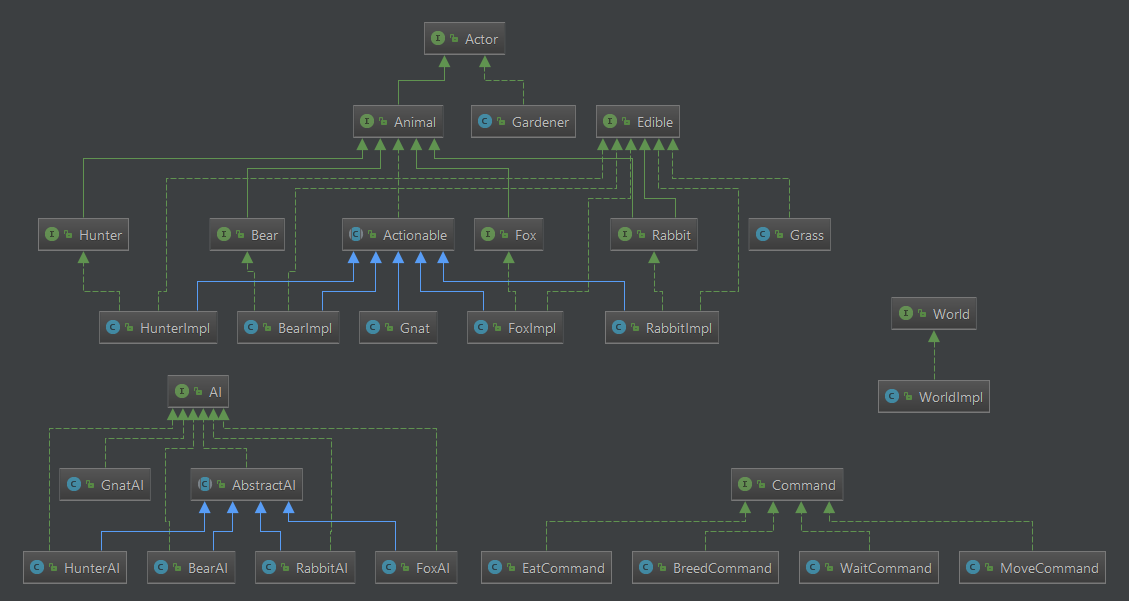


Figure 6 VIEWRANGE 모드

WorldPanel에 mouseListener를 추가하고 paintComponent를 수정하여 실제 그림이 그려진 후에 fillRect를 활용하여 원하는 기능을 구현할 수 있었다.

Figure 7 Class Diagram