**CS 499 Senior Project**

**Project Assignment**

**Life Simulation Life Form Behavior Rules**

**Source code location version**

**\*all locations are in blue under the original requirement**

**Notes**

Terms shown in all caps inside of angle brackets (<>) are the XML tag names in the sample simulation data file.

**Plant**

Plants grow from seeds in a hemispherical form, that is, it will be as high as the radius of its circular area on the ground.

* File: PlantLife.Java
  + Functions/Lines:
    - PlantLife() - 50
* File: ActorState.Java
  + Functions/Lines:
    - evolve\_a\_plant() - 532

A plant seed will begin growing 10 seconds of simulation time after it is ejected from its’ parent. In this first growth it will start with a radius of 1/100 of a DU.

* File: PlantLife.Java
  + Functions/Lines:
    - PlantLife() - 51
    - increment\_germination\_timer() - (purpose of function)
* File: ActorState.Java
  + Functions/Lines:
    - evolve\_a\_plant() - 501-506

A plant grows at a rate defined in a simulation data file (<GROWTH\_RATE>) as a fraction of its’ maximum size.

* File: ActorState.Java
  + Functions/Lines:
    - evolve\_a\_plant() - 508-533

The maximum size of a plant is defined in a simulation data file (<MAX\_SIZE>).

The plant continues to grow until it reaches its maximum size after which every hour of simulation time it will produce a seedpod which will pop spreading a random number of seeds. The maximum number of seeds that can be produced is defined in a simulation data file (<MAX\_SEED\_NUMBER>). The seeds will be distributed in a circular region around the plant. The maximum distance a seed can be thrown is defined in a simulation data file (<MAX\_SEED\_CAST\_DISTANCE>).

* File: PlantLife.Java
  + Functions/Lines: isGrowing() - (purpose of function)
* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 165-167
    - evolve\_a\_plant() - 508, 534-571
    - find\_seed\_coords() - 673-697

A percentage of the seeds distributed will germinate and grow. The percentage of viable seeds is defined in a simulation data file (<SEED\_VIABILITY>).

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 168, 551
    - find\_seed\_coords() - 699-727

The number of plants present when the simulation starts is defined in a simulation data file (<INITIAL\_PLANT\_COUNT>).

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 163

A plant will be defined in the data file with an X and Y coordinate (<X\_POS> and <Y\_POS>) and a diameter <P\_DIAMETER>.

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 170-179

**Grazer**

Grazers feed on plants.

When food is available the Grazer is able to eat enough to gain a number of EU per minute of simulation time. The number of EUs gained is defined in a simulation data file (<ENERGY\_INPUT>).

After 10 minutes of simulation time the Grazer will have eaten all food with in its’ reach (5 DU) and will have to move.

* File: Herbivore.java
  + Functions/Lines:
    - think() – 126 – 129
    - eat() – 223 – 250

A Grazer will expend a number of EU each time it moves 5 DU. The amount of energy expended is defined in a simulation data file (<ENERGY\_OUTPUT>). The energy output is the same whether the Grazer is looking for food or fleeing from a Predator.

* File: Herbivore.java
  + Functions/Lines:
    - moveToFood() – 298
    - flee() - 219

A Grazer can see any edible plant that is within its’ direct line of sight (not blocked by other obstructions) and within 150 DU of it.

* File: Herbivore.java
  + Functions/Lines:
    - moveToFood() – 336
    - getClosestActor() – 370 - 402
    - actorsInRange – 421 - 447
    - visibleActors() – 449 - 511

If a Grazer’s energy level drops below 25 it will be unable to move more than 10 DU because of weakness.

* File: Herbivore.java
  + Functions/Lines:
    - getMoveRange() – 252 - 260

If a Grazer’s energy level drops to zero it dies and disappears from the simulation.

* File: Herbivore.java
  + Functions/Lines:
    - spendEnergy() – 404 - 410
* File: ActorState.java
  + Functions/Lines:
    - Actorstate\_evolve() – 445

If a Grazer’s energy level rises above a level as defined in the data file (<ENERGY\_TO\_REPRODUCE> it will reproduce and produce two Grazer organisms each of which will have half of the original Grazer’s energy level.

* File: Herbivore.java
  + Functions/Lines:
    - eat() – 246 - 248
* File: ActorState.java
  + Functions/Lines:
    - Actorestate\_evolve() – 435 - 440

A Grazer will see a Predator if it is within 25 DU and not hidden by an obstruction. It will attempt to escape by running or hiding behind an obstruction.

* File: Herbivore.java
  + Functions/Lines:
    - think() – 123, 137
    - checkSurroundings() - 141 - 156
    - flee() – 158 - 221
    - getClosestActor() – 370 - 402
    - actorsInRange – 421 - 447
    - visibleActors() – 449 - 511

A Grazer can run faster than a Predator but only for a time defined in a simulation data file for a given number of simulation minutes (<MAINTAIN\_SPEED>), it will then slow to 75% of its’ maximum speed <MAX\_SPEED>.

* File: Herbivore.java
  + Functions/Lines:
    - flee() – 163 - 170

The initial number of grazers, their energy levels, and locations will be defined in a simulation data file.

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 186

A grazer will be defined in the data file with an X and Y coordinate (<X\_POS> and <Y\_POS>) and an initial energy level <G\_ENERGY\_LEVEL>.

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() – 193 - 203

**Predator**

The Predators feed on the Grazers.

* File: Predator.java
  + Functions/Lines:
    - think() - 588

Predators shall have a limited set of genetic traits as defined below which will determine its reactions to other Predators and to Grazers.

* File: Predator.java
  + Functions/Lines:
    - Variables & constants - 234–270
    - parse\_genotype() - 277–390

**Aggression**

Dominant Gene = A

Recessive Gene = a

Genotypes

Homozygous Dominant = AA – Predators with AA genotype are most aggressive. They will attempt to kill and eat other Predators unless seeking a mate.

Heterozygous Dominant = Aa – Predators with Aa genotype are moderately aggressive. They will attack other Predators only when hungry and no other food is available.

Homozygous Recessive = aa – Predators with aa genotype are non-aggressive. They will always flee when approached or attacked by other Predators. They will only approach another Predator when driven by the mating urge.

* File: Predator.java
  + Functions/Lines:
    - think() - 600–620

**Strength**

Dominant Gene = S

Recessive Gene = s

Genotypes

Homozygous Dominant = SS – Predators with SS genotype are the strongest and will succeed in killing and eating Grazers 95% of the time if they are caught. If they attack another Predator they will succeed in killing and eating that Predator 75% of the time if its’ strength is Ss and 95% of the time if its’ strength is ss. If two Predators with genotype SS fight there is a 50% chance one or the other will kill and eat and 50% chance they will disengage and go their own way. If one does kill and eat the other there is an even chance as to which will succeed.

Heterozygous Dominant = Ss – Predators with Ss genotype are moderately strong and will succeed in killing and eating Grazers 75% of the time if they are caught. If they attack another Predator they will succeed in killing and eating that Predator 25% of the time if its’ strength is SS and 75% of the time if its strength is ss. If two Predators with genotype Ss fight there is a 50% chance one or the other will kill and eat and 50% chance they will disengage and go their own way. If one does kill and eat the other there is an even chance as to which will succeed.

Homozygous Recessive = ss – Predators with ss genotype are weakest and will succeed in killing and eating Grazers 50% of the time if they are caught. If they attack another Predator they will succeed in killing and eating that Predator 5% of the time if its’ strength is SS and 25% of the time if its strength is Ss. If two Predators with genotype ss fight there is a 50% chance one or the other will kill and eat and 50% chance they will disengage and go their own way. If one does kill and eat the other there is an even chance as to which will succeed.

* File: Predator.java
  + Functions/Lines:
    - killChancePredator() - 144–210
    - killChanceHerbivore() - 218–230
    - nearest() - 537
    - attack() - 890–918
    - ignore() - 953–965

**Speed** - The maximum speeds a Predator can run (see tags below) and the times it can maintain that maximum speed (<MAINTAIN\_SPEED>) is defined in the data file. Speeds are given in DU per minute and times in minutes. After the maintain speed time is elapsed the Predator will slow at a rate of one DU per 15 seconds of simulation time till it comes to a stop.

Dominant Gene = F

Recessive Gene = f

Genotypes

Homozygous Dominant = FF – Predators with FF genotype are the fastest running (<MAX\_SPEED\_HOD>).

Heterozygous Dominant = Ff – Predators with Ff genotype are moderate speed runners (<MAX\_SPEED\_HED>).

Homozygous Recessive = ff – Predators with ff genotype are the slowest runners (<MAX\_SPEED\_HOR>).

* File: Predator.java
  + Functions/Lines:
    - Predator() - 428
    - move\_toward() - 786–838
* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 266–245

A Predator will expend a number of EU each time it moves 5 DU. The amount of energy expended is defined in a simulation data file (<ENERGY\_OUTPUT>). The energy output is the same whether the Predator is looking for food or chasing a Grazer.

* File: Predator.java
  + Functions/Lines:
    - Predator() - 427
    - move\_toward() - 827

When a Predator kills and eats a Grazer or another Predator it will gain 90% of that organism’s energy.

* File: Predator.java
  + Functions/Lines:
    - eat() - 924–940

A Predator will see a Grazer or another Predator that is within its’ direct line of sight (not blocked by other obstructions) and within 150 DU of it.

* File: Predator.java
  + Functions/Lines:
    - const. MAX\_VISIBILITY\_DISTANCE: 105
    - visible() - 659–688
    - intersects\_rock() - 702–777

A Predator will smell a Grazer or another Predator even if it is not within its’ direct line of sight and within 25 DU of it.

* File: Predator.java
  + Functions/Lines:
    - const. MAX\_SMELL\_DISTANCE: 25
    - visible() - 659–688

When a Predator’s energy level reaches a level defined in a simulation data file (<ENERGY\_TO\_REPRODUCE>) it will seek another predator and mate. Predators are bisexual, meaning they are both male and female but must mate with another Predator to reproduce. After mating each of the pair will give birth to a random number of new Predators up to a maximum number of offspring as defined in a simulation data file (<MAX\_OFFSPRING>). The gestation period <GESTATION> is also defined in simulation days in the simulation data file.

* File: Predator.java
  + Functions/Lines:
    - Predator() - 431–433
    - mate() - 971–981
    - reproduce\_with() - 987–993
* File: ActorState.java
  + Functions/Lines:
    - ActorState\_evolve() - 461–469

The genetic makeup of each offspring shall be determined by Mendel’s laws of genetics.

Each offspring will receive one gene for each trait from each parent. For example: if two predators each with genes Aa for aggression mate their offspring may get an AA, Aa, or aa combination. For each parent and for each trait there is a 50:50 chance of receiving either of the parent genes.

* File: Predator.java
  + Functions/Lines:
    - get\_baby\_genes() - 1045–1077

After giving birth a Predator and it’s offspring will move away from each other and not react to the other for one simulation hour.

* File: Predator.java
  + Functions/Lines:
    - nearest() - 537
    - ignore() - 953–964
    - give\_birth() - 1024, 1028–1035

Predator offspring are born with an energy level defined in a simulation data file. This does not diminish the parent’s energy level.

* File: Predator.java
  + Functions/Lines:
    - Predator() - 434
    - give\_birth() - 1015

Predator offspring have all the characteristics of an adult when born.

The initial number of predators, their energy levels, and locations will be defined in a simulation data file.

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 210, 221–249

A predator will be defined in the data file with an X and Y coordinate (<X\_POS> and <Y\_POS>), an initial energy level <P\_ENERGY\_LEVEL> and a char array giving the genotype <GENOTYPE>.

* File: ActorState.java
  + Functions/Lines:
    - ActorState\_create() - 221–249

**Obstacle**

An obstacle can be thought of as a round dome shaped rock.

An obstacle will be defined in the data file with an X and Y coordinate (<X\_POS> and <Y\_POS>), a diameter <O\_DIAMETER> and a height <O\_HEIGHT>.

* File: Rock.java
  + Functions/Lines:
    - Rock() - (purpose of function)

Issues while trying to create a stable world population:

Some of the obstacles we had while trying to get a stable population were:

* The wolves were way to aggressive
  + They ended up pushing the deer into the sides of the map where they would end up spending all of their energy running back and forth from the edge to a plant.
* The wolves didn't want to breed with each other, because they couldn't find enough energy via deer because the deer would get away most of the time.
  + This also included that the wolves couldn't move very far across the map without dying off, which made it very hard to mate since the middle of the map ends up being vacant because the deer (their food source) is pushed to the sides.
  + The deer also had this problem, they could never escape the corner/side they were being pushed into, because when it was time to move to a new plant, they died on the way there.
* To fix these issues we changed a couple things:
  + the energy to reproduce was raised so that both the deer and wolves would have stronger offspring. (the wolves would always die immediately after giving birth).
  + The amount of energy gained while eating was raised so the deer population would explode quicker
  + The amount of energy per distance unit was lowered for both the deer and the wolves so they could move longer distances without starving.
  + The “Alpha” wolves were more energy to start with so that they could almost immediately try to reproduce so that the deer had a little time to create a more stable population.
    - These wolves were also given less aggressive traits so they wouldn't pick off their own kind, but also stronger traits so they wouldn't be picked off by losing a fight to a deer.
  + We also lowered the amount of initial wolves to 2, so that they wouldn't overrun the deer immediately (but allowed them to breed quicker.)
  + We also doubled the amount of deer so the chances they would survive the initial onslaught were greater.
  + We also moved the rocks from their initial locations to help create more obstacles so the wolves wouldn't immediately spot the deer as easily.
* These changes that were made can make a stable population, but it also causes for a huge population of deer that slows down the simulation greatly.
  + We haven't been able to find a good balance where this doesn't happen.