DEVS-Based Modeling and Simulation for Predator-Prey

CSC8840

Anitha Subramanian

Spring 2020

PROBLEM STATEMENT AND OBJECTIVES OF THE PROJECT

Many of the most interesting dynamics in nature have to do with interactions between organisms. One of such interaction is Predator-Prey which are subtle, indirect and difficult to detect. Predator-Prey are the building blocks of ecosystem. Spices compete, evolve and disperse simply for the purpose of seeking resources to sustain their struggle for their very existence. This plays an important role in the ecology of populations, determining mortality of prey and birth of new predators. The scope of this project is to identify such kind of dependancies and interactions between the Predator and Prey.

This project presents a DEVS-based Predator-Prey simulation of a Sheep Grass System. The aim of this work is to study how the prey population changes with respect to predator and also to analyze the system behavior by tuning some of predator parameters. The representation in this project can be used to predict prey's growth over time. It also shows that using System Entity Structure to describe a Predator-Prey system is convenient and easy to be modeled in simulation environment such as DEVSJAVA.

The objectives of the project are to simulate following scenarios in the well behaving Predator-Prey model,

- Two different predators (with different move time and reproduce time but same life time) that eat the same prey and analyze the growth rate of prey population.
- Restrict the Predator movement directions to 180 degrees and analyze the growth rate of prey population.

GENERAL DESCRIPTION OF THE SYSTEM

The Predator-Prey model has grass as prey and sheep as predator. The initial populations and locations of sheep and grass can be set manually or generated randomly. The grass can grow in any random direction if that selected neighboring place is not occupied by anything. When there is no neighboring free place for the grass to reproduce, then the grass will not reproduce.

The sheep is in constantly moving state. The sheep will search for the grass in neighboring places and move to the place with grass to eat. If there is no grass in the neighboring places, the sheep will move to the empty place with the condition that there should not be any other sheep in it. The sheep should move within specified time units. The sheep will eat the grass immediately after it moves to that grass location. If the sheep doesn't eat grass for specified time units, it will die.

Similar to grass, the sheep will reproduce in any one of the neighboring places if it is free. The sheep will reproduce for every specified time units. Both the sheep and grass need to know the neighboring places to move and reproduce respectively.

GOALS OF THE MODELING AND SIMULATION STUDY

This is a realistic model of a Sheep Grass system. In “real life” the sheep grass model would be characterized by randomly occurring events. The results observed from the simulation of this project could be diverse. Some important aspects that can be measured are,

- Two different predators (with different move time and reproduce time but same life time) that eat the same prey and analyze the growth rate of prey population.

  - Definitely there would be decrease in growth of prey population if there two predators that eats the same prey in the land even if they have different move time and reproduce time.

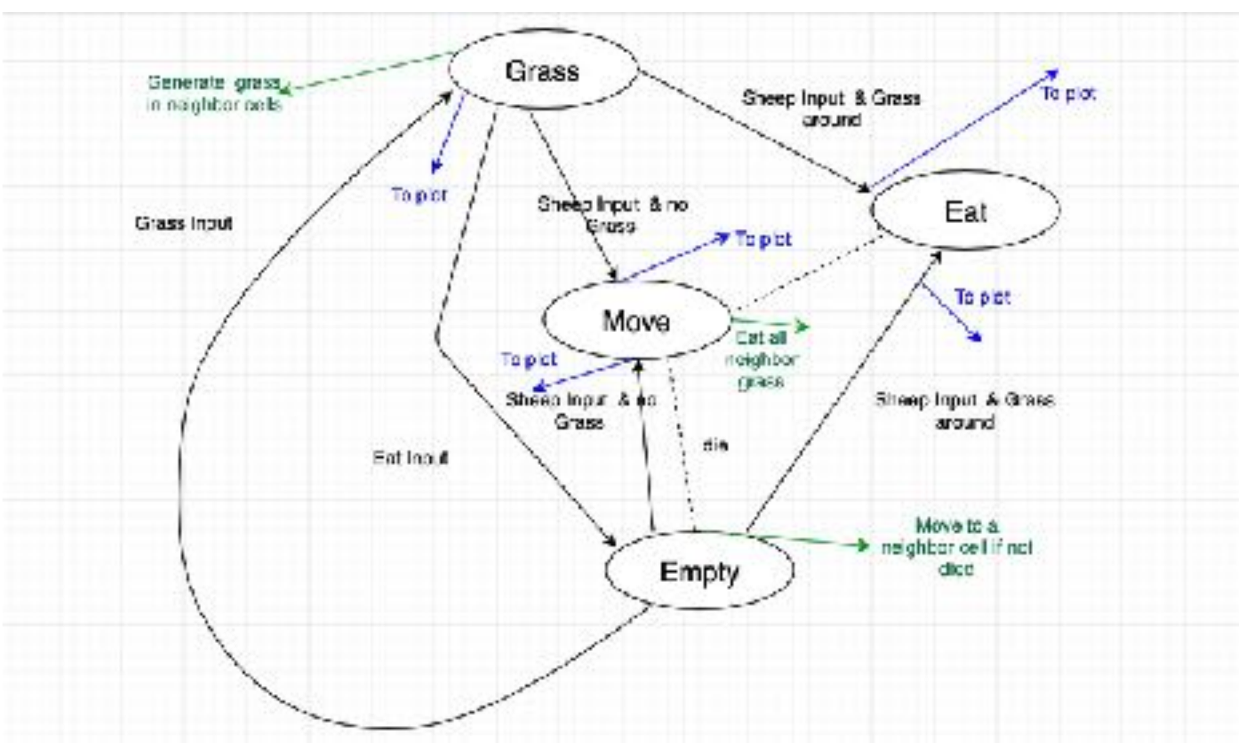
- Restrict the Predator movement directions to 180 degrees and analyze the growth rate of prey population.

  - If the Predator movement is restricted, there will be increase in growth of prey population.

THE SIMULATION MODEL

MODEL STRUCTURE:

The state diagram for the Sheep grass model is shown below.



The green lines depicts the output messages to the cell space. A sheep cell's attribute (sheep life time and sheep reproduce time) is contained in the message when the sheep moves from one cell to another. Two variables sheeplifetime and sheepreproducetime can be used to keep track of the life time and reproduce time.

EXPLANATION OF OPERATION:

Grass - Modeled behaviors:

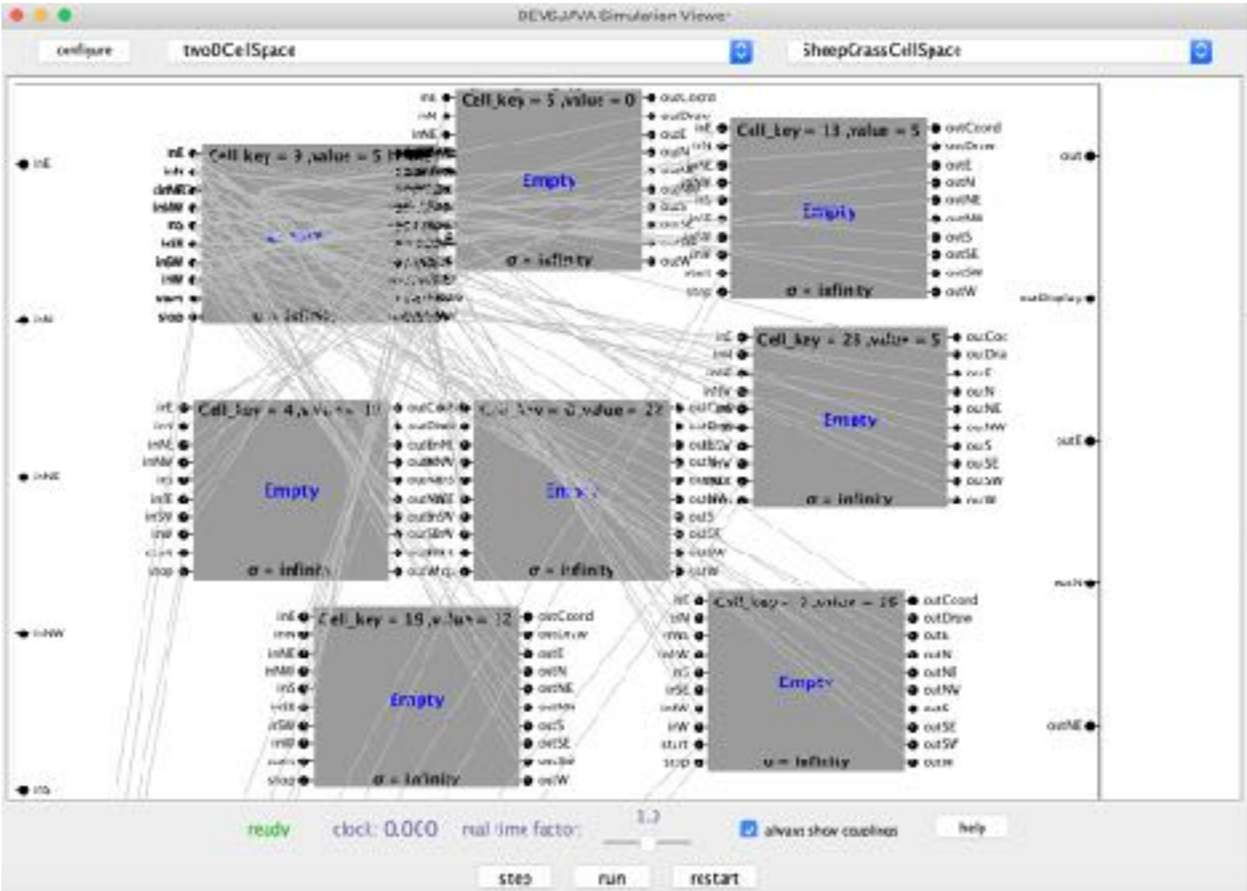
- ✦ Initialization - Hold in grass in random locations of the land.
- ✦ For the phase Grass, hold in grass in the specified x-coordinates and y-coordinates.
- ✦ For the phase Grassreproduce, hold in grass in the specified locations. Get the random neighbor cell out of 8, make the grass reproduce in that cell if it is empty. If it is not empty, don't reproduce.
- ✦ Get the random directions for the grass through the input ports.

Sheep - Modeled behaviors:

- ✦ Initialization - Hold in sheep in random locations of the land.
- ✦ If the phase is Sheep move, check for the grass or empty cell in the neighboring cells and then make the sheep move.
- ✦ For the phase of Sheep or Sheepreproduce, check for the sheeplife time, sheepmove time and sheep reproduce time.
    - ✓ If there is enough life time for the sheep to reproduce (ie.,life time is more than moving time of sheep) and enough reproduce time (ie., reproduce time is more than moving time of sheep), switch to sheepmove phase for sheepmovetime. Adjust the sigma time accordingly.
    - ✓ If there is not enough life time for the sheep to move or reproduce, make the sheep die.
    - ✓ If there is more sheep reproduce time than its life time and move time, switch to sheepreproduce phase. Adjust the sigma time accordingly.
    - ✓ Update the sheep move time, sheep life time, sheep reproduce time.
    - ✦ Get the random direction for the sheep reproduce or sheep move through input ports.

ILLUSTRATIVE MODEL CODE:

The Sheep Grass Model



DEVS Java code for External transition function of a Sheep Grass model:

DEVS Java code for Internal transition function of a Sheep Grass model:

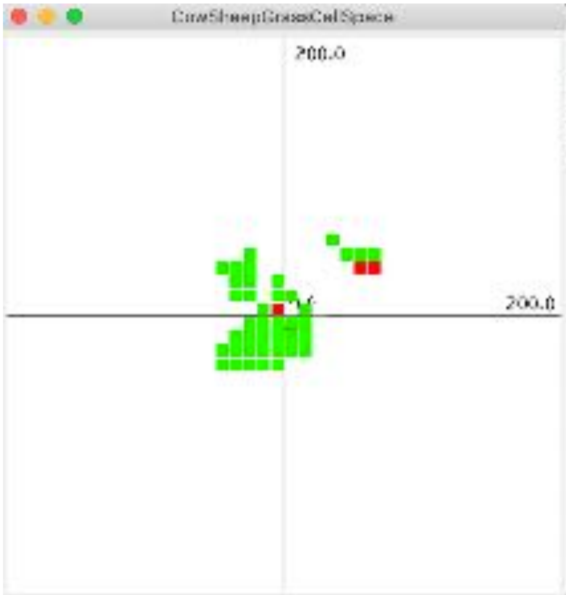
```
public void deltint()
{

    if(phaseIs("Grass"))
        holdIn("Grassreproduce",Grassreproducetime);
    else if(phaseIs("Grassreproduce"))
    {
        System.out.println("grass is producing at" +" "+x_pos+" "+y_pos);
        grasscount-1;
        grasscount=keepgrasscount(grasscount);
        System.out.println("Grass count : "+grasscount);

        holdIn("Grassreproduce",Grassreproducetime);
    }
    else if(phaseIs("Sheep") || phaseIs("Sheepreproduce"))
    {
        if(Sheepmovetime<Sheeplifetime && Sheepmovetime<Sheepreproducetime)
         {
             System.out.println("It is coming to sheep state or sheep reproduce state");
             //System.out.println("Sigma now is"+sigma);
             holdIn("Sheepmove",Sheepmovetime);
             Sheepreproducetime=Sheepreproducetime-sigma;
             Sheeplifetime=Sheeplifetime-sigma;
         }
         else if(Sheeplifetime<=Sheepmovetime && Sheeplifetime<=Sheepreproducetime)
             holdIn("Sheepdie",Sheeplifetime);
         else if(Sheepreproducetime<Sheepmovetime && Sheepreproducetime<Sheeplifetime)
         {
             holdIn("Sheepreproduce",Sheepreproducetime);
             Sheepmovetime=Sheepmovetime-sigma;
             Sheeplifetime=Sheeplifetime-sigma;

         }
    }
    else if(phaseIs("Sheepmove"))
    {
        System.out.println("Sheep is moving");
        SheepGrassCellSpace.cellphase[xcoord][ycoord]="Empty";
           passivateIn("Empty");

    }
    else if(phaseIs("Sheepdie"))
         passivateIn("Empty");
    else if(phaseIs("Empty"))
        holdIn("Empty",3.4);
```

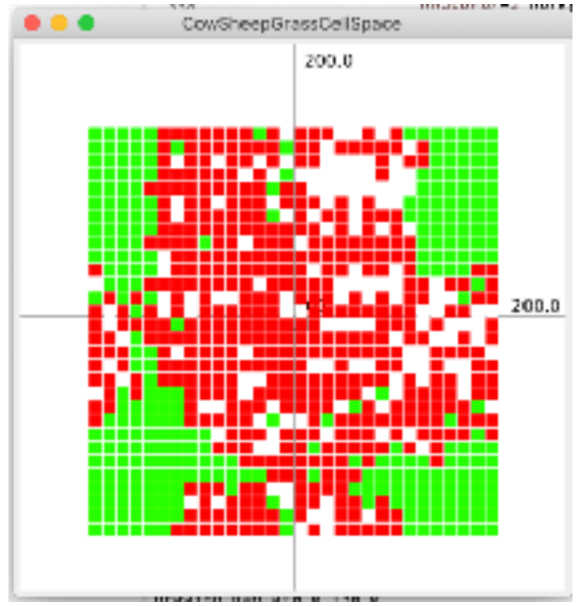
sheepMoveT, sheepLifeT, sheepReproduceT, grassReproduceT are the variables used to store the allotted time values for the Sheep move time, Sheep Life time, Sheep Reproduce time and Grass Reproduce time respectively. The dummy variables Sheepmovetime, Sheeplifetime, Sheepreproducetime, Grassreproducetime are initialized with allotted time values and then used for tracking the time values under certain conditions inside the functions (ie. values updated based on sigma value after it reached particular phase).

SIMULATIONS:

The below snapshots shows the simulation of Sheep Grass model. Green represents grass and red represents sheep. Blue represents Cow.

The snapshot on the left shows the initial setup of the Sheep grass model with few grass and 3 sheep in the land. The snapshot on the right shows the stable oscillation of sheep and grass after some time.





EXPERIMENTAL DESIGN:

The following experimental frames can be developed for this Sheep Grass model.

1. Cow Sheep Grass experiment

- This experiment can be made to analyze the growth rate of prey population when there is a single predator that eats the prey vs two predators that eat the same prey. This can be achieved by adding one more predator to the system with different parameters. So I have taken Cow as my other predator.

Initial conditions - Some cells with grass , few sheep and few cows.

Input generators - Phase received from the input ports such as Sheep move, Cow reproduce, etc.,

Simulator - Simulates the movement and reproduction of two predators by eating the same prey in the land. Two predators cannot eat the same prey. They can eat only the prey is available without any predator on it.

Time for Predators - The reproduce time, move time are different for both the predators except the life time. This allows us to identify the affected growth of prey population clearly.

Expected result - The growth of prey population will be less over the period of time because of more number of predators even if they have different parameters.

Code Snippet for Cow Sheep Grass experiment:

2. Sheep Direction Restriction test

- This experiment also made to analyze the growth rate of prey population by restricting sheep direction to 180 degrees instead of 360 degrees. ie., Restricting sheep move and sheep reproduce actions to only 5 directions instead of 8.

Initial conditions - Some cells with grass and few sheep. Change the functions getdirectionsheepreproduce, getdirectionsheep with directions limiting to 5.

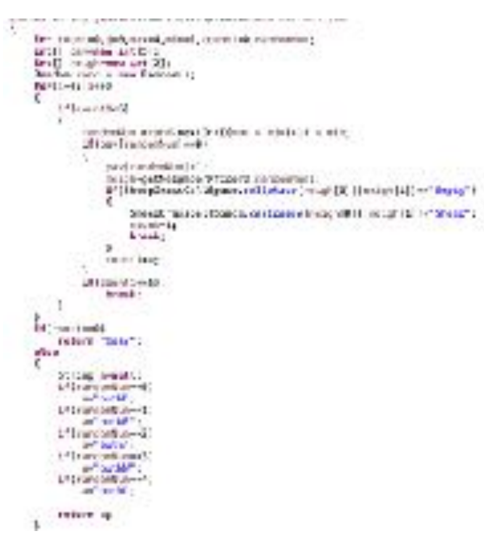
Input Generator - Phase received from the input ports such as Sheep move, Grass reproduce, etc.,

Simulator - Simulates the restricting movement of sheep to particular directions.

Grass direction - This would be all the 8 directions unlike sheep.

Expected result - The growth of prey population will be more as there is a restricted movement of predator in the land. Also there will be less population of predators since the reproduce direction also restricted.

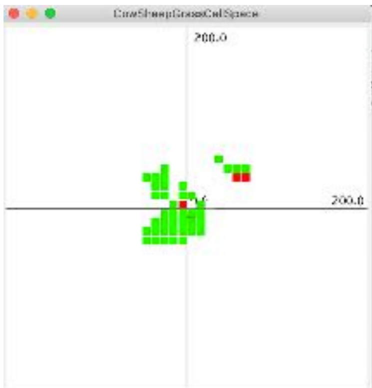
Code Snippet for Sheep Direction Restriction test:



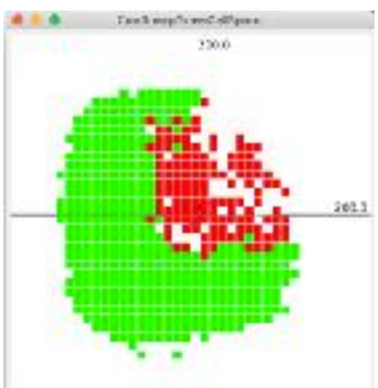
SIMULATION RESULTS & ANALYSIS:

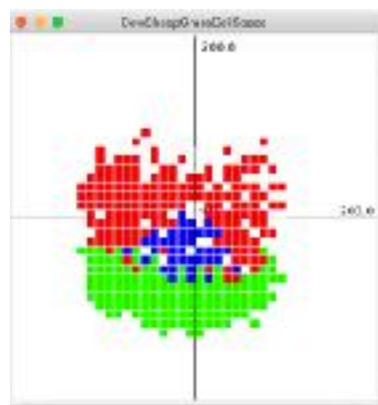
Cow Sheep Grass Experiment:

1. Initial state - Sheep Grass System Vs Cow Sheep Grass System

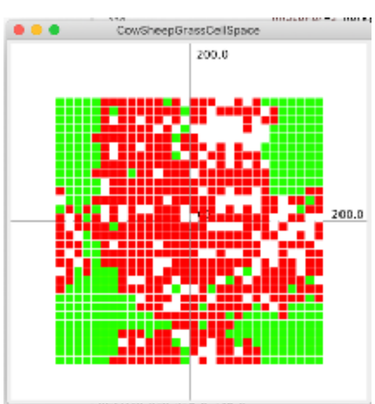


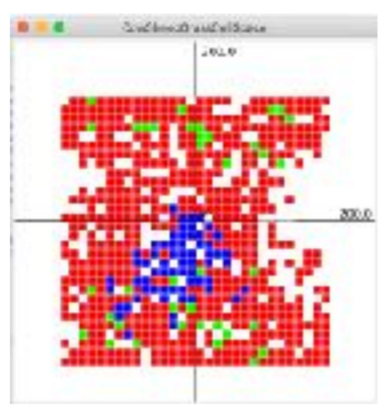
2. After time t - Sheep Grass System Vs Cow Sheep Grass System





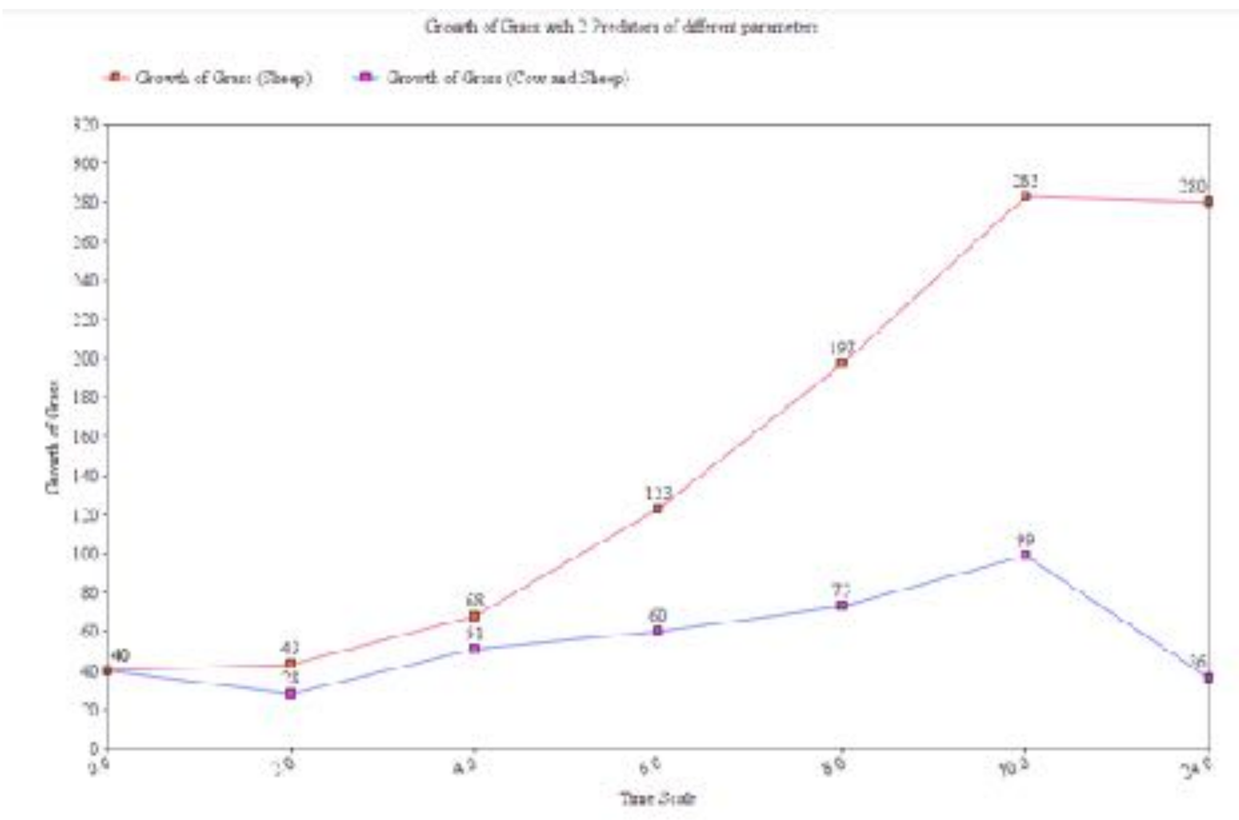
3. At the end - Sheep Grass System Vs Cow Sheep Grass System





Time value (e)	Growth of Grass (sheep)	Growth of Grass(Cow and Sheep)
Initial	40	40
2.0	43	28
4.0	68	51
6.0	123	60
8.0	197	73
10.0	283	99
Towards end	280	36

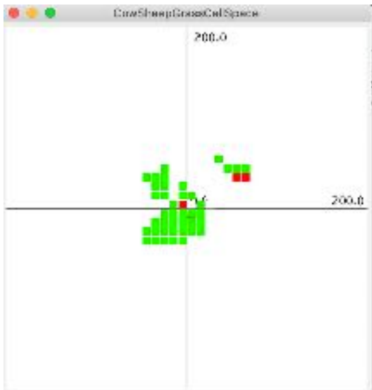
From the above simulation results data, We can see that the growth rate of prey population is affected when there are two predators with different parameters that eats the same prey in the system compared to single predator in the system.



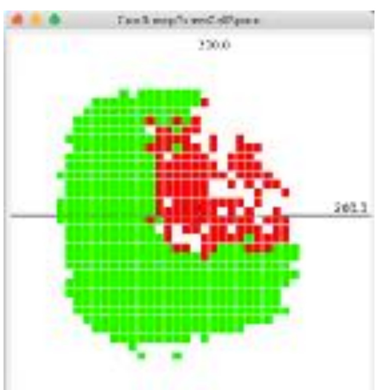
From the above graph, we can see that there is a fall in growth of grass with 2 predators of different characteristics.

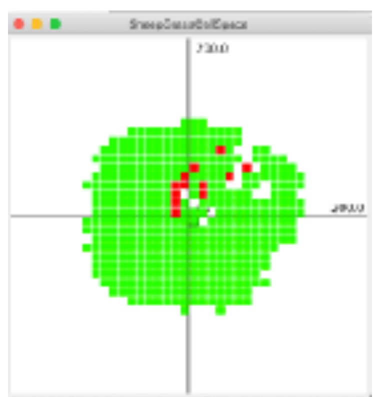
Sheep Direction Restriction test:

1. Initial State - Sheep with 8 directions Vs Sheep with 5 directions

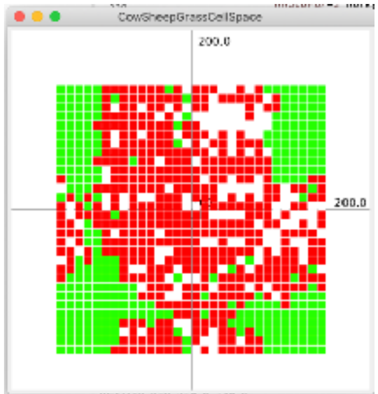


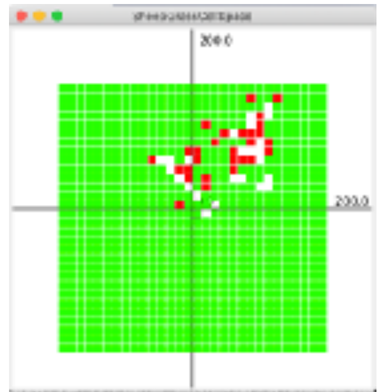
2. After time t - Sheep with 8 directions Vs Sheep with 5 directions





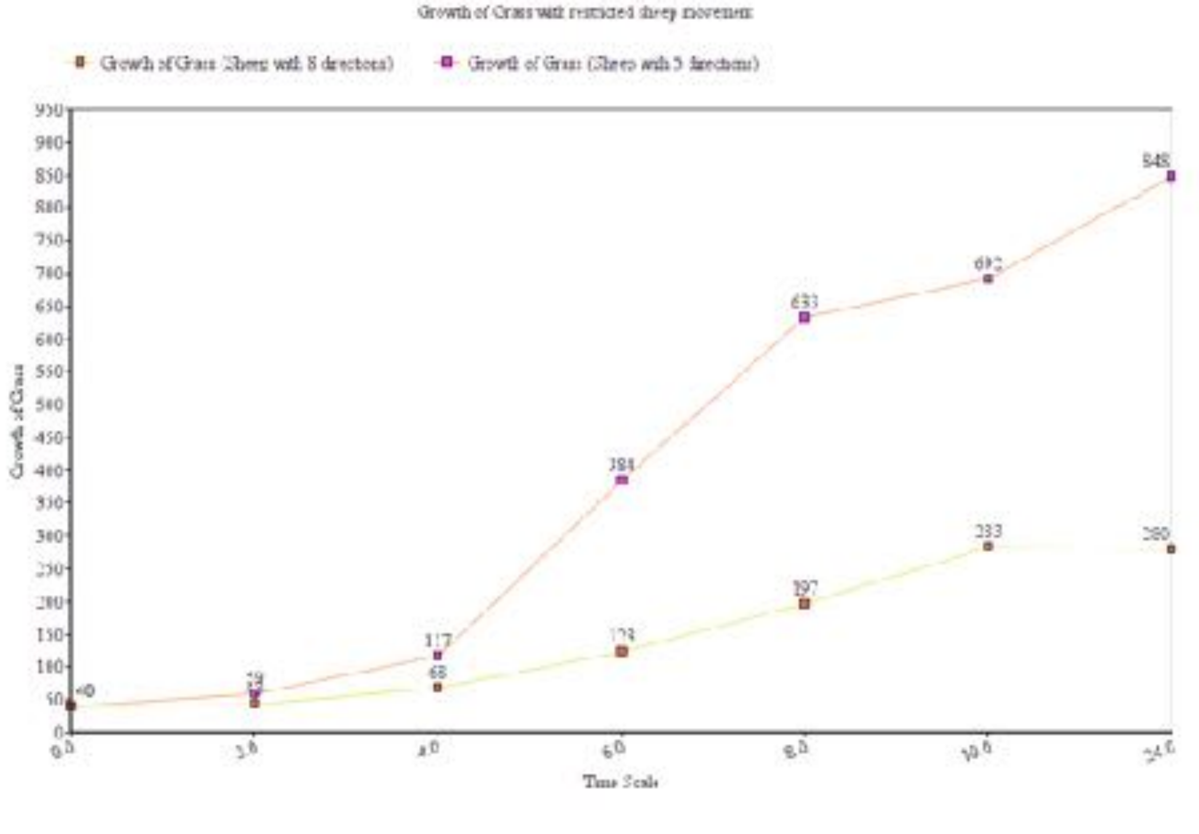
3. At the end - Sheep with 8 directions Vs Sheep with 5 directions





Time Value (e)	Growth of Grass (Sheep with 8 directions)	Growth of Grass (Sheep with 5 directions)
Initial	40	40
2.0	43	59
4.0	68	117
6.0	123	384
8.0	197	633
10.0	283	692
Towards end	280	848

From the above simulations data, we could see that restricting the direction of predator to limited directions results in high growth rate of prey population and also very less predator reproduce.



From the graph, we can see that the growth of prey is more with the restricted movement of predator. If it is not restricted, the growth of prey will be less as seen from the graph.

CONCLUSION:

We could observe from these simulation experiments that the growth of prey population depends on predator and its parameters. When we add one more predator that eats the same prey to the system, there is a less growth rate of Prey population. So the predator population is indirectly proportional to prey population. *Higher the Predator lesser the Prey*. Another simulation experiments shows that restricting the movement of predator results in high growth rate of prey population. *When there is a restriction on Predator, Prey population increases.*

FUTURE WORK:

This model is just implemented with single predator and prey. The future work may include adding different predator to the system like wolf and observe the system under the same above mentioned conditions.

REFERENCES:

1. CSC8840 - Our Class slides and documents from the Professor Dr.Xialion Hu.
2. http://www.scholarpedia.org/article/Predator-prey_model - For basic understanding of the Predator-Prey model.
3. Zeigler, B.P., Praehofer, H., and Kim, T.G. (2000), theory of modeling and simulation. 2ed, Academic Press, New York, NY.