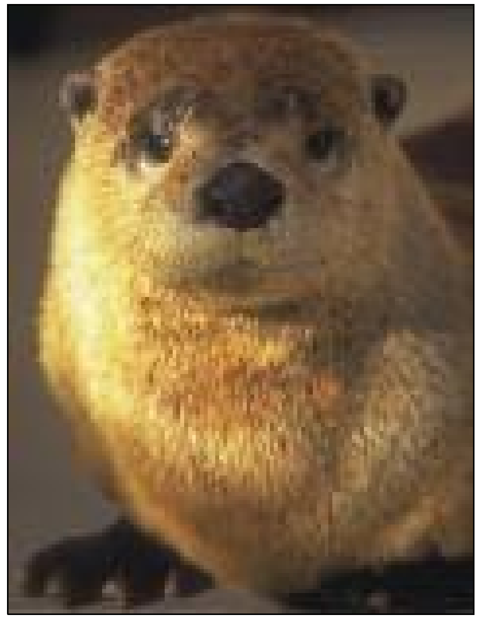
River Otters And Their Effect on Riverine Ecosystems



New Mexico
Supercomputing Challenge
Final Report
April 2, 2008

Team Number: 59
Los Alamos Middle School

Team Members:

Thomas Henderson
Johnny Jacobs
Steven McCrory
Jack Mockler
Simon Redman
Marie-Luise Schmidt

Executive Summary

We researched the North American River Otter and factors that affect them to put in our model. Then we narrowed the factors down to two or three. We used three programming languages to model the otter's success in their environment. We learned the programming languages and became better programmers. We hoped to obtain the same results from at least two of our models, but we did not because none of the models were completed. To the best of our knowledge, these are the first models that represent river otters in New Mexico.

Introduction

The object of our project is to create a model of the events that would occur if the New Mexico Department of Game and Fish reintroduced river otters into the Rio Grande environment. River otters were originally native to New Mexico until they became extinct for unknown reasons. They have been planning to do this for some time, and were planning to reintroduce the otters last fall but were delayed. If we could create an accurate model, then we might be able to advise the New Mexico Department of Game and Fish how, when, and where to release the otters. We might also tell them the minimum number of otters that can be released while still achieving success. Since otters are fairly expensive we could allow them to spend the least amount of money on this project, letting them have more money for other projects. This would also maximize the profit that otters give to New Mexico.

Problem Statement

Our overall mission is to answer the question: what is the feasibility of reintroducing river otters into New Mexico?

Figure 1: Possible Otter Introduction Site

Description of the problem

Our research problem is to help reintroduce river otters into the central Rio Grande. The native population of river otters is thought to be extinct. Critics believe that the otters will not survive in New Mexico, based on the extinction of the native otters. Because the river otters are extinct, the New Mexico department of game and fish is attempting to reintroduce otters for economic benefit. They also think the otters will eat invasive species of fish and crayfish. Many anglers and game fishers are opposed to the idea of reintroducing the otters, because the native fish populations might be harmed by them.

Our main purpose in this project is to hopefully help the Department of Game and Fish decide whether or not to reintroduce the otters. To reach this goal we first researched otters and their prey to get enough information to make a realistic model, possibly in several different languages. After this we made a model to represent the otters lives in the central Rio Grande. This will hopefully help determine whether river otters would survive in the central Rio Grande.

For our models we chose an agent based modeling system to model how the otters would survive in an ecosystem similar to a river based ecosystem. An agent based model is a model which simulates many objects, or agents that follow a prescribed set of rules that control their actions. The reason we chose an agent-based model over a mathematical model is that a mathematical model requires advanced mathematics. Our group is not that skilled in math, so probably could not easily learn the mathematics involved in a mathematic model. To prepare for the model we created a spreadsheet of all the possible agents and simplified to a few for the models. The spreadsheet is shown in Appendix A. In the spreadsheet we accounted for the following:

Look at all the factors or "agents" that impact the outcome

<u>Agents considered:</u>

Otters

Fish (both slow fish and fast fish)

Crayfish

Otter competitors (muskrats, raccoons, etc.)

Environment (pollution, vegetation, burrows)

Other Factors

Fish food (insects)

<u>Agent Characteristics</u>

Representation of reproduction

Calculate fish population as otter numbers grow

Determine otter range and population

Track otters age and health

Game fish populations

Invasive fish populations

Crayfish populations

The following programming languages were used:

1) Starlogo TNG
2) Netlogo
3) Java (Madkit)

Team Work

The otter team consists of six members, and working with a team on a project of this size was a new experience for all the team members. All team members were initially involved in the research step and all contributed papers and articles from the Internet and other sources, the most important of which are in the Bibliography.

Most of the team members expressed an interest in modeling and coding and wanted to opportunity to learn more about different languages and approaches. Four team members, Thomas, Johnny, Jack and Simon focused on modeling and computers, with Thomas and Johnny producing the StarLogo TNG models in this report. Jack focused on NetLogo and the NetLogo model in this report is his. Simon chose the Java/Madkit approach and his model is presented below.

All team members were involved in the grueling task of creating the final report and, without serious bodily injury to any of the team members or supporting helpers, managed to create this document.

Starlogo TNG Model 1

(program blocks can be found in Appendix B page 16)

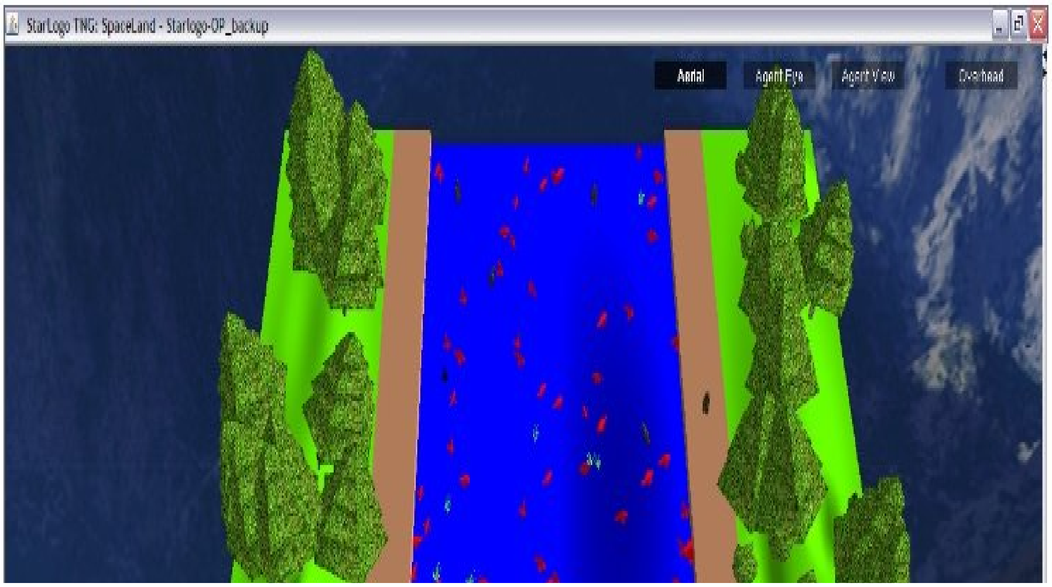


Figure 2: Visual Representation of Model

One of our computer models was done in Starlogo TNG. We find that Starlogo TNG provides more of a visual representation of what is going on than other programs such as Netlogo, or Java. Our agents are fast fish, slow fish, otters, hiding spots, bushes, and trees. The slow fish represent crayfish and other bottom feeders. The fast fish represent game fish such as trout and bass. The otters eat slow fish and fast fish. They can also breed with each other. The fast fish breed and the slow fish breed and hide in the hiding spots. Whenever an agent collides with another agent of the same type it has a fifty-percent chance of breeding. The otters only breed when they are on land. We make the slow fish hide by saying that it cannot move when it collides with a hiding spot. The slow fish also shrink and turn green when they hide. Our model represents a small stretch of a New Mexican river in which the fish and otters thrive together. Currently, when we run our model for about ten minutes, the fast fish reproduce until they reach about 100, the otters reach about 50. The slow fish however stay about the same as when they started due to the fact that when they hide they never come out from hiding. This is probably the most unrealistic factor in our model.

Some of the factors that we were not able to put in were the effects of time on the agents, environmental effects, anglers, and otter competitors. Before we can add these factors we will need more experience in Starlogo TNG. At this point in time, we have a working knowledge of approximately half the blocks in the program, but there are still many more to learn about.

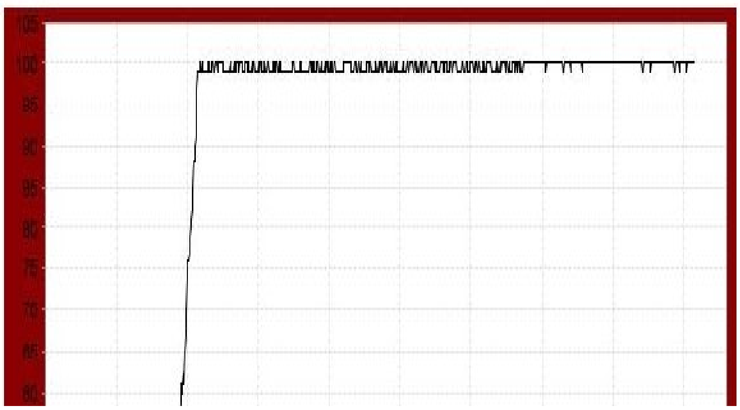


Figure 4: Number of Fast Fish vs. Time Setup

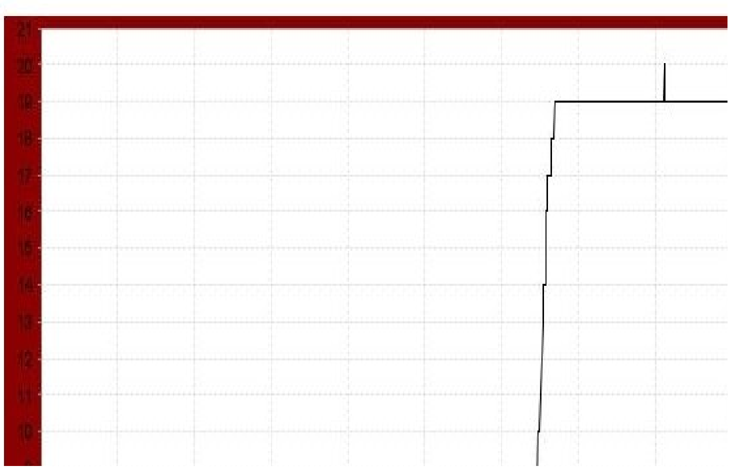


Figure 5: Number of Otters vs. Time Step

Starlogo TNG Model 2

In our model we have three agents. The first agent is the otters. The next is the fish that the otters feed on. Third, we have muskrats which compete with the otters for food. Since the different agents represent different animals, they interact differently with each other. The otters and muskrats both eat a fish when they collide with it. When any two agents of the same type collide they breed making more identical agents. When muskrats run into a certain color of ground, representing a river bank, they dig a burrow, which the otters use to breed.

Our model represents the real world in some ways but not others because of limitations in the program and our modeling experience. In the real world, there are usually one to two otters per mile of river. In our model we cannot show this because there are so many fish in one mile of river that the computer lags and freezes. Otters eat ten fish a day so if our model included time, we might be able to model a day of an otter's life, but we do not have enough modeling experience to include time. Otters also eat only certain sizes of fish. Fish reach a certain size when they are a certain age, so again we could only model this if we had enough programming experience to make time in our model. Otters prefer certain fish over others, such as slow fish, like carp, over fast fish, like trout, because slower fish are easier to catch. We don't have this in our model because it would take a long time for us to put in and not have many positive effects on our model.

If we continue this model next year, there is something we would like to do differently. We would like to change to a different programming language that has less glitches and can handle more intensive numbers of agents. If this programming language has improved in any extensive way by next year we might use it again.

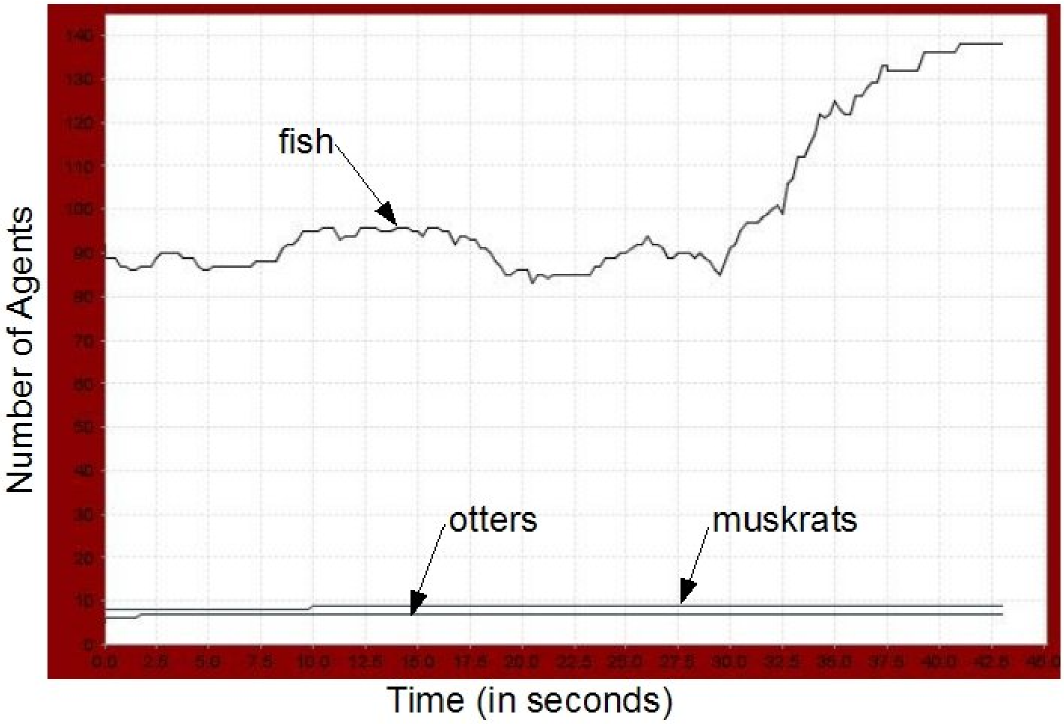


Figure 6: Starlogo TNG Model 2 Results

Netlogo Model

(code can be found in Appendix C, page 17)

In the Netlogo model there are three agents. These agents are carp, trout, and otters. There are also insects modeled as patches for the fish to eat. All of the agents age and eventually die. The agents gain energy from eating, and lose energy by moving, and reproducing. All of the agents reproduce one time per year, but only if they are a certain age. They can have a certain number of babies each year. The otters eat only when their energy drops below a set value called

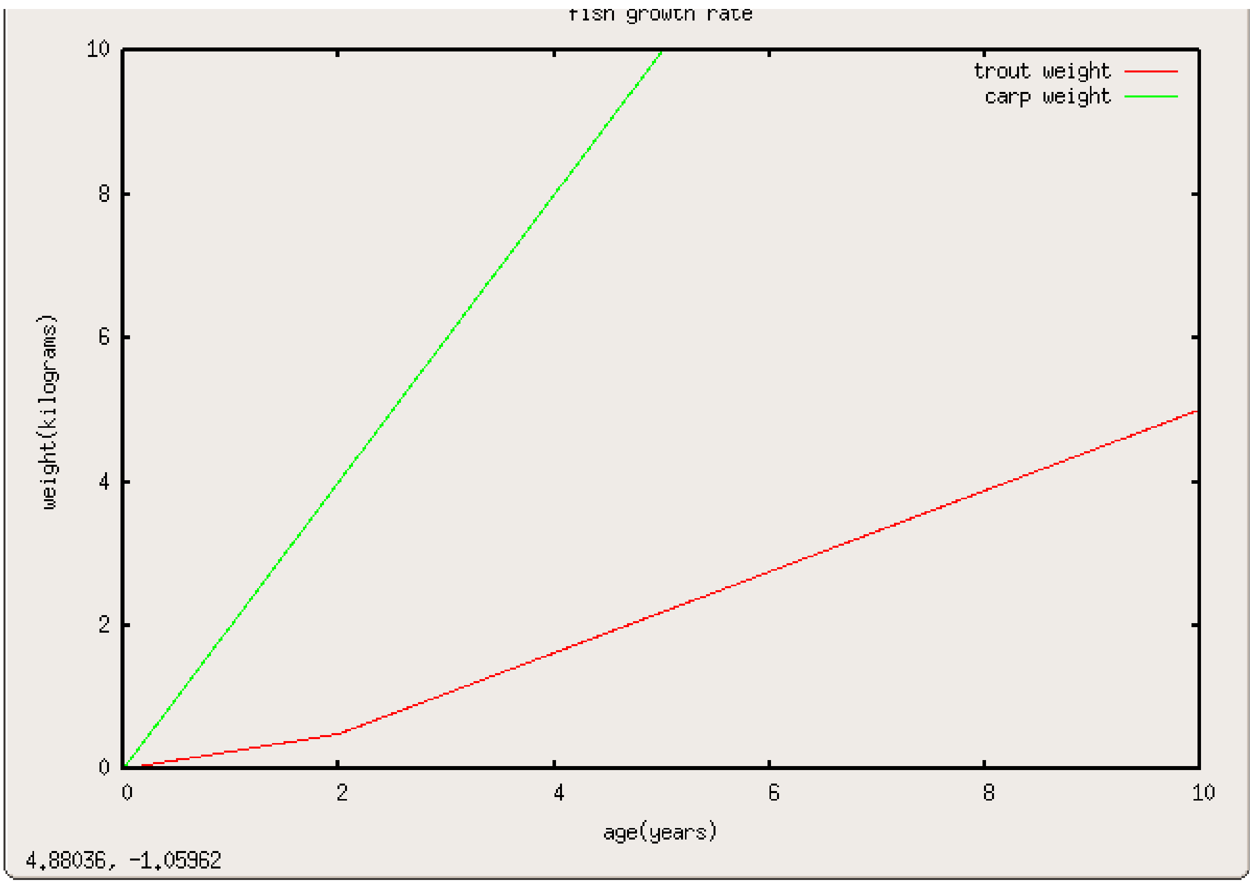
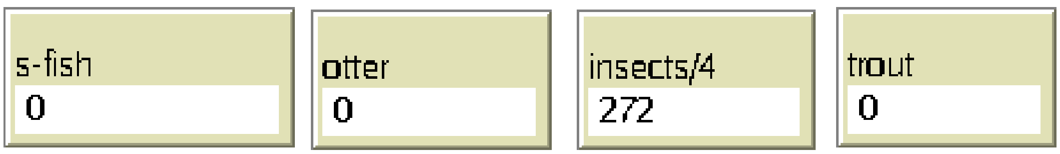


Figure 7: Fish Growth Rate in Netlogo Model

“idealhealth”. If their ideal health is reached they do not eat again until it drops below “idealhealth”. The fish gain weight as they grow older and give more energy to the otter who eats them the larger they are (Figure 7 shows the plot of fish size as they grow older). Fish eat insect patches and a larger fish will eat a smaller fish.

Netlogo Model Results

The results of the net logo model are that the fish will die out about four years after the otters are reintroduced, causing the otters to die as well. These results are almost certainly not accurate however because the model is not realistic. The otters do not breed the way they should and the number of fish has to get too high for our computer to handle for the model to reach an equilibrium. The model also does not have crayfish in it, which are one of the otters main sources of food. (Figure 8 shows a plot of the agent numbers as time passes in the model).



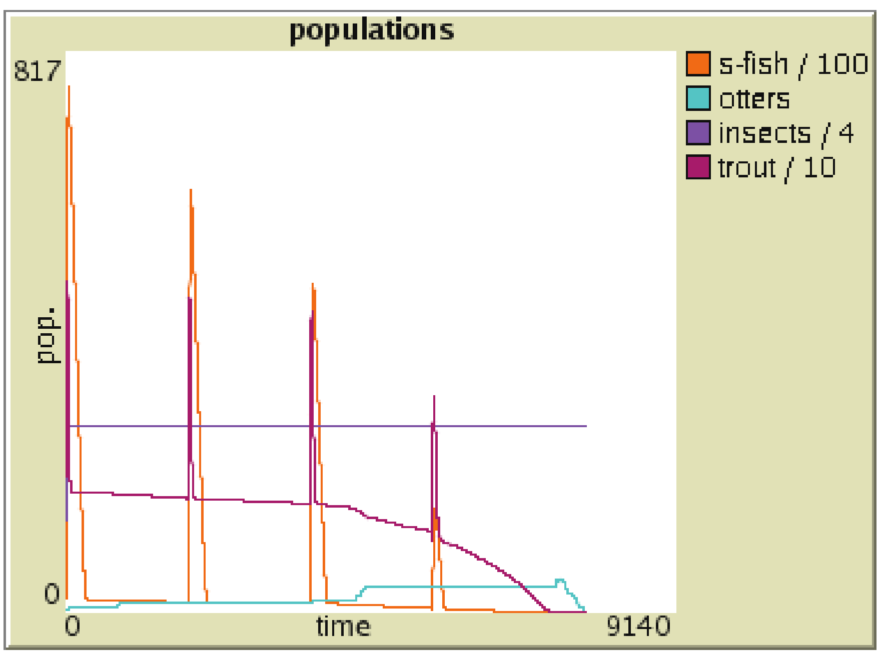


Figure 8: Netlogo Model Results (4 years)

Netlogo Model Future Work

If we work on this model in the future there are several things we would add to make it better. For one thing we would add crayfish into the model. This would make it more realistic and might help to stabilize the model. Another thing we would add would be a more accurate representation of otter reproduction. This would take some work, but would probably keep the otter population lower and let more fish survive to breed. One other thing we would add would be more realistic data. The more realistic the model gets, the better results it gives so it is always good to add as much data as possible. These are the three main things we would add to our model.

The Java Model

(code can be found in Appendix D, page 22)

The Java model has some accurate features, but, like all the other models, there are some problems. One problem is that the area in which the otters and fish live is approximately three screens long. This means that not everything in the model can be seen at once. However, it has some advantages. We have decided that one screen is equal to one mile, which allows us to model three miles of river.

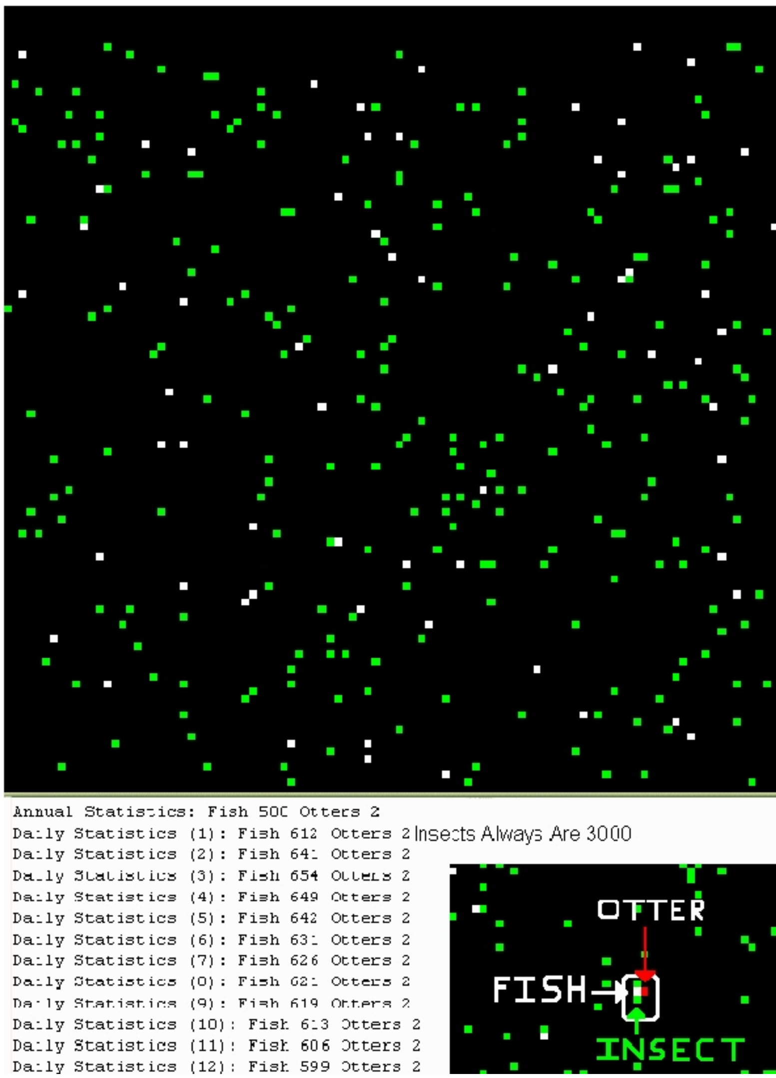


Figure 9: Java Model Screenshot

Another problem is that our computer lags when there are too many agents in the model. This means that it will be hard to accurately model an ecosystem. One thing that might fix these problems is scaling the model. One other problem is that the fish starve. Even if there are no predators the fish still die, or at least dwindle down to very low numbers from lack of food.

The model currently holds thirty otters, but does not have the computing power to hold enough fish. It outputs the statistics of all the populations, which makes it easier to see and graph.

Conclusion

We have developed models to represent a riverine ecosystem in New Mexico with river otters. To our knowledge we are the first ones to attempt such simulations. We divided the team into three sub-groups that each worked on different models. We did this to better validate our results. The three programming languages that we used were Starlogo TNG, Netlogo, and Java. Starlogo TNG had superior graphics but bogged down the computer and lagged. Netlogo was much harder to learn and had less sophisticated graphics, but was able to make a much more complicated model before it began to lag and hit glitches (which was not a problem in our Netlogo model). The Java language was the hardest of all to learn but had the potential to make a much more complicated model. We did not get very far in our Java model because it was so difficult to learn.

Once we were finished modeling they all needed more work, and we did not come to any definite conclusion about the otters based on our models. Our models show that the otters and /or the fish will die within a few years. However, the models are not entirely accurate. Judging by our research , we still believe they would be able to live in certain places in New Mexico, such as the central Rio Grande and possibly the Gila River. The models were successful to varying degrees. We are looking forward to continuing this project next year.

Bibliography

1. Greer, K. 1955. Yearly Food Habits of the River Otter in the Thompson Lakes region, northwestern Montana, as indicated by scat analyses. The American Midland Naturalist. 54(2) p. 299-313.
2. Hansen, Heidi. 2003. Food Habits of the North American River Otter. Unpublished manuscript. University of Wyoming. 7 pp.
3. Kroeger, Timm. 2005. Economic Benefits of reintroducing the River otter into rivers in New Mexico. Prepared for Amigos Bravos, 35 pp.
4. New Mexico Dept. of Game and Fish. 2006. Feasibility Study: Potential for restoration of river otters

in New Mexico. Review Draft. 59 pp.

5. Savage, M. and Klingel, J., 2003. The Case for River Otter Restoration in New Mexico. A Report to the River Otters Working Group. The Four Corners Institute, Santa Fe, NM. 10 pp.
6. Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.
7. Wilensky, U. (1997). NetLogo Wolf Sheep Predation model. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
8. Madkit Simulation is based on code by, and copyright to : Fabien Michel
9. Starlogo TNG, http://education.mit.edu/starlogo-tng, MIT Media Laboratory

Acknowledgments:

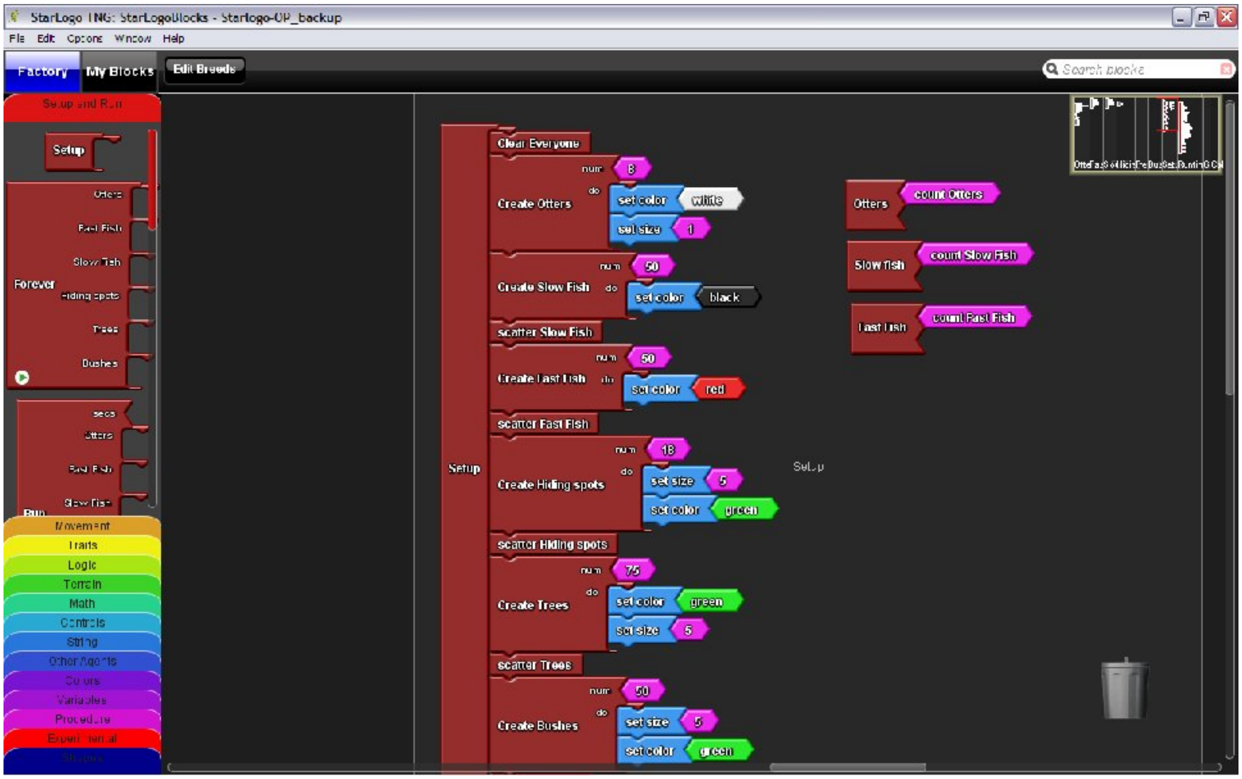
Our programming mentor: Rueben Guadiana
Our school sponsor: Mrs. Le-Ann Salazar
And to all of our wonderful parents who helped us to complete the challenge marathon

Appendix A. Agent Spreadsheet

	Otters	Slowfish	Ffish	Crayfish	Bottomfeeders	Tadpoles	Insects	Amphibians	Small mammals	Plants
Otters	otters reproduce, compete for food	preferred food for otters	occasional food for otters	occasional food for otters, exotic	preferred food for otters	no direct effect	occasional food for otters	occasional food for otters	occasional food for otters	occasional food for otters
Sfish		eat each other, compete for food, reproduce	eat each other	compete for food	compete for food	food for slowfish	food for slowfish	occasional food for slowfish	occasional food for slowfish	food for slowfish
Ffish			eat each other, reproduce	compete for food	compete for food	food for fastfish	food for fastfish	occasional food for fastfish	occasional food for fastfish	food for fastfish
Crayfish				reproduce, compete for food	compete for food	compete for food, crayfish eat tadpoles	food for crayfish	amphibians eat crayfish	sm mammals eat crayfish	food for crayfish
Bottomfeeders					reproduce, compete for food	food for bottomfeeders	food for bottomfeeders	compete for food	sm mammals eat bottomfeeders	food for bottomfeeders
Tadpoles						compete for food	food for tadpoles	tadpoles are amphibians	small mammals eat tadpoles	eat plants??
Insects							reproduce, compete for food	insects are food for amphibians	insects are food for small mammals	plants provide habitat for insects
Amphibians								reproduce, compete for food	small mammals eat amphibians	plants provide habitat for
Small mammals									reproduce, compete for food	plants provide habitat and food
Plants										compete, pollination
Mollusks										
Algae										
Bacteria										
Anglers										
Pollution										
Muskrats										
Beaver										
Nutrias										
Cows										

	Mollusks	Algae	Bacteria	Anglers	Pollution	**Muskrats**	Beavers	**Nutrias**	**Cows**
Otters	need research	no direct effect	bacteria kills	kills otters	pollution kills	for food and habitat with otters,	dens are used by otters	**Need research**	**Need research**
Sfish		food for slowfish	bacteria kills	kill slowfish	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Ffish		food for fastfish	bacteria kills	kill fastfish	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Crayfish		food for crayfish	bacteria kills	no direct effect	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Bottomfeeders		food for bottomfeeders	bacteria kills	occasionally kills bottomfeeders	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Tadpoles		food for tadpoles	bacteria kills	no direct effect	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Insects		food for insects	bacteria kills	no direct effect	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Amphibians		food for amphibians	bacteria kills	anglers eat frog legs	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Small mammals		food for small mammals	bacteria kills	no direct effect	pollution kills	**need research**	affect habitat	**Need research**	**Need research**
Plants		no direct effect	bacteria kills	anglers trample plants; plants impede anglers	pollution kills	**need research**	for houses (willow, cottonwood)	**Need research**	**Need research**
Mollusks	reproduce, compete,	7	bacteria kills	anglers trample?	pollution kills	**need research**	beavers eat mollusks?	**Need research**	**Need research**
Algae		no direct effect	bacteria kills	no direct effect	pollution kills	**need research**	no direct effect	**Need research**	**Need research**
Bacteria			no direct effect	no direct effect	pollution kills	**need research**	bacteria kills?	**Need research**	**Need research**
Anglers				compete	pollution hurts	**need research**	affect habitat	**Need research**	**Need research**
Pollution					no direct effect	pollution kills	pollution kills	**Need research**	**Need research**
Muskrats						reproduce, compete for food	compete for habitat, food?	**Need research**	**Need research**
Beaver							reproduce, compete for food	**Need research**	**Need research**
Nutrias								reproduce, compete for food	**Need research**
Cows									reproduce, compete for food

Appendix B: Starlogo TNG Model 1 Program Blocks



Appendix C: Netlogo Code

```
breed [otters otter]
breed [s-fishes s-fish]
breed [trouts trout]
turtles-own [energy age reptime weight]
globals [ year tmpage ]
patches-own [countdown]

to setup
  clear-all
  set year 1825
  ask patches [ set pcolor blue ]
  if insects? [
    ask patches [
      set countdown random insect-regrowth-time
      set pcolor one-of [black blue]
     ]
  ]
  set-default-shape otters "turtle"
  create-otters initial-number-otter
  [
    set color brown
    set size .5
    set label-color blue - 2
;;    set energy random (2 * otter-gain-from-food)
    set energy idealhealth
    set age 2
    set reptime random 1825
    setxy random-xcor random-ycor
  ]
  set-default-shape s-fishes "fish"
  create-s-fishes initial-number-s-fish
  [
    set color gray
    set size .3
    set label-color blue - 2
    set energy random (2 * s-fish-gain-from-food)
    set age random 8
    set weight age * 2
    if age > 5 [set weight 10]
    set reptime 1825
    setxy random-xcor random-ycor
  ]
  set-default-shape trouts "trout"
  create-trouts initial-number-trout
  [
  set color yellow
  set size .3
  set label-color blue - 2
  set energy random (2 * s-fish-gain-from-food)
  set age random 8
  set weight .25 * age
  if age >= 2 [set weight .6 * age + .22]
```

```
  set reptime 1825
  setxy random-xcor random-ycor
  ]
  display-labels
  update-plot
end

to go
  if not any? turtles[ stop ]
  ask s-fishes [
    move
    eat-baby
    eat-trout-baby
    if insects? [
      set energy energy - .1
      eat-insects
    ]
    s-fish-reproduce
    get-old
    death
  ]
  ask otters [
    move-otter
    set energy energy - 1.
    catch-s-fish
    catch-trout
    otter-reproduce
    get-old
    death
  ]
    ask trouts [
    move
    eat-baby
    eat-trout-baby
    if insects? [
      set energy energy - .1
      eat-insects
    ]
    trout-reproduce
    get-old
    death
  ]
  if insects? [ ask patches [ grow-insects ] ]
  tick
  update-plot
  display-labels
  show mean [energy] of otters
end

to move
rt random 90
lt random 90
fd random 5
end

to move-otter
```

```
rt random 90
lt random 90
fd random 7
end

to move-trout
rt random 90
lt random 90
fd random 9
end

to eat-insects
if energy < 100 [
  if pcolor = black [
    set pcolor blue
    set energy energy + s-fish-gain-from-food
    ]
]
end

to trout-reproduce
if random-float 100 < reproduce-s-fishes and age > 3 and reptime > year [
    set reptime 0
    hatch 5 [rt random-float 360 fd random 5 set age 0 set reptime 0]
    set energy (energy - 1)
  ]

end

to s-fish-reproduce
if random-float 100 <= reproduce-s-fishes and age > 1 and reptime > year [
    set reptime 0
    hatch 50 [ rt random-float 360 fd random 5 set age 0 set reptime 0]
set energy (energy - 3)
  ]

end


to otter-reproduce
if random-float 1 < reproduce-otter and age > 2 and reptime > year[
show reptime
show age
set reptime -1825
set energy (energy / 10)
hatch 2 [ rt random-float 360 fd 6 set age 0 set reptime 0 set energy idealhealth / 4 ]
]
end

to catch-s-fish
if energy < idealhealth [
  let prey one-of s-fishes-here
    if prey != nobody [
    if random 7 >= random 5 [
        ask prey [set tmpage weight]
        ask prey [ die ]
```

```
        set energy energy + otter-gain-from-food * tmpage
      ]
    ]
]
end

to catch-trout
if energy < idealhealth [
  let prey one-of trouts-here
    if prey != nobody [
      if random 7 >= random 9 [
        ask prey [set tmpage weight]
        ask prey [ die ]
        set energy energy + otter-gain-from-food * tmpage
      ]
    ]
]
end

to eat-baby
if age > 1 and random 100 < 10 [
  let prey one-of s-fishes-here
    if prey != nobody [
      ask prey [ set tmpage age]
      if tmpage < 1 [
        ask prey [ die ]
        set energy energy + s-fish-gain-from-food ]
      ]
    ]
end

to eat-trout-baby
if age > 1 and random 100 < 10 [
  let prey one-of trouts-here
    if prey != nobody [
      ask prey [ set tmpage age]
        if tmpage < 1 [
          ask prey [ die ]
          set energy energy + s-fish-gain-from-food ]
    ]
]
end

to death
if energy < 0 [ die ]
if age > 10 [ die ]
end

to grow-insects
if pcolor = blue [
ifelse countdown <= 0
[set pcolor black
set countdown insect-regrowth-time ]
[ set countdown countdown - 1 ]
]
end
```

```
to update-plot
set-current-plot "populations"
set-current-plot-pen "s-fish / 100"
plot count s-fishes / 100
set-current-plot-pen "otter"
plot count otters
set-current-plot-pen "trout / 10"
plot count trouts / 10
if insects? [
set-current-plot-pen "insects / 4"
plot count patches with [pcolor = black] / 4

]
end

to display-labels
ask turtles [ set label "" ]
if show-energy? [
ask otters [ set label round energy ]
if insects? [ ask s-fishes [ set label round energy ]  ask trouts [ set label round energy ] ]
]
if show-age? [
ask turtles [set label round age ]
]
end

to get-old
set age age + 5.48e-4
set reptime reptime + 1
end
```

Appendix D: Java/Madkit Code

```
/*
 * OtterLauncher.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import turtlekit.kernel.Launcher;

/** Otter simulation launcher

 @author Fabien MICHEL
 @version 1.1 6/12/2000 */

@SuppressWarnings("serial")
public class OtterLauncher extends Launcher {

	public static final String SIMULATION_NAME = "OTTER";

	/** The launcher.  We need this to add agents. */
	public static OtterLauncher otterLauncher;
	/**
	 *
	 */
	int numberOfprey = 500;  // 1000;
	int numberOfpredator = 2;
	int numberOfInsects = 3000; //This is constant, one dies, one is made.
	int predatorVision = 6;
	int preyVision = 5;

	public OtterLauncher() {
		setCyclePause(10);
		setSimulationName(SIMULATION_NAME);
		setWidth(1000);
		setHeight(100);
	}

	public void setNumberOfprey(int add) {
		numberOfprey = add;
	}
```

```
public int getNumberOfprey() {
        return numberOfprey;
}

public void setPredatorVision(int add) {
        predatorVision = add;
}
public void setpreyVision(int add) {
        preyVision = add;
}
public int getPredatorVision() {
        return predatorVision;
}
public int getPreyVision() {
        return preyVision;
}
public void setNumberOfpredator(int add) {
        numberOfpredator = add;
}

public int getNumberOfpredator() {
        return numberOfpredator;
}

public void setNumberOfInsects(int add) {
        numberOfInsects = add;
}

public int getNumberOfInsects() {
        return numberOfInsects;
}

public void addSimulationAgents() {
        otterLauncher = this;  // save the launcher.
        setCyclePause(10);

        for (int i = 0; i < numberOfprey; i++) {//add the prey with the method addTurtle
                Fish fish = new Fish(preyVision);
                addTurtle(fish);
        }
        for (int i = 0; i < numberOfpredator; i++) { //add the predator with the method addTurtle
                Otter otter = new Otter(predatorVision);
                addTurtle(otter);
        }

        for (int i = 0; i < numberOfpredator; i++) { //add the predator with the method addTurtle
                OtterCompetitor otter = new OtterCompetitor(predatorVision);
                addTurtle(otter);
        }
        for (int i = 0; i < numberOfInsects; i++) { //add the predator with the method addTurtle
                Insect insect = new Insect();
                addTurtle(insect);
        }
        // add a statistics agent
        StatisticsAgent statisticsAgent = new StatisticsAgent();
        addTurtle(statisticsAgent);
```

```
		addViewer(6); // we choose a default viewer with a cell size of 3

	}

	/** Adds a new otter.  */
	public void newOtter() {
		Otter otter = new Otter(predatorVision);
		addTurtle(otter);
	}
	/** Adds a new fish..  */
	public void newFish() {
		Fish fish = new Fish();
		addTurtle(fish);
	}
	/** Adds a new insect.  */
	public void newInsect() {
		Insect insect = new Insect();
		addTurtle(insect);
	}

}

/*
 * Otter.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 *
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import java.awt.Color;

import turtlekit.kernel.Turtle;

/**
 * A Otter
 *
 *
 */

@SuppressWarnings("serial")
public class Otter extends MyTurtle {
```

```
	public static final String OTTER_ROLE = "otter";

	/** We'll start chasing fish when we're half hungry.  Since we
	 * need to eat two fish per day and this is counted in cycles,
	 * half hungry is half a day.
	 */
	public static final int FISH_HEALTH_ADDER = DAY/10;

	/** The vision radius is used to decide how easily the
	 * otter can see the fish.
	 */
	protected int visionRadius;

	/** The health of the otter.  This is 0-1000 and decrease by one each cycle.
	 * a zero otter is a dead otter.
	 */
	protected int health;

	/** The constructor.  The vision radius is used to decide how easily the
	 * otter can see the fish.
	 * @param visionRadius
	 */
	public Otter(int visionRadius) {
		super("live");
		this.visionRadius = visionRadius;
	}

	public void setup() {
		// Random generator = new Random();
		playRole(OTTER_ROLE);
		randomHeading();
		setColor(Color.red);
		if (countTurtlesHere() > 0) {
			fd(10);
		}
		health = DAY;
	}

	public String live() {
		setHeading(towardsAFish(visionRadius));
		// once a year (cycleCount % YEAR), we'll bred.
		// For simplicity, if we're not hungry we'll have a 50% chance of producing an otter
		// (Males don't reproduce, some females have more than one pup, but some have none)
		// Only 2+ year old otters bred
		if ( cycleCount > YEAR*2 && health >= FISH_HEALTH_ADDER && cycleCount != 0  &&
(cycleCount % YEAR == 0) ) {
			//if ( random.nextBoolean() ) { // only works for 50%
			if ( random.nextInt(2) > 0 ) {  // pick an integer and a max/min
				OtterLauncher.otterLauncher.newOtter();
			}
		}
		health--;
		if ( health <= -WEEK*4 ) { // we can survive a week without food, if we haven't eaten in a week,
we're dead
			return null; // dead
```

```
		}
		move();
		cycleCount++;  // increment the cycle count, we survived another round
		return "live";
	}

	void move() {
		int speed = random.nextInt(5);
		if ( health < 25 ) {
			speed += 1; // hungry
		}
		else if ( health > 75 ) {
			speed = 1;  // why bother to move I'm full
		}
		//System.out.println("OTTER " + (speed + 1));
		if (countTurtlesAt(dx(), dy()) > 0) {
			turnRight(50);
		}
		if (countTurtlesAt(dx(), dy()) > 0) {
			turnLeft(100);
		}
		if (countTurtlesAt(dx(), dy()) == 0) {
			fd(speed + 1);
		}
		else {
			turnRight(50);
		}
	}

	/** Move towards a fish.  We'll only do this if we're hungry, that is, if health is down a fish. */
	double towardsAFish(int radius) {
		if ( health < FISH_HEALTH_ADDER ) {  // we won't chase fish unless we're down a fish (this
number could be adjusted. */
			for (int i = -radius; i <= radius; i++) {
				for (int j = -radius; j <= radius; j++) {
					if (!(i == 0 && j == 0)) {
						Turtle[] tur = turtlesAt(i, j);
						if (tur != null && tur.length > 0 &&
tur[0].isPlayingRole("fish")) { // instead of "instanceof". So fish can be another java class
							return towards(tur[0].xcor(), tur[0].ycor());
						}
					}
				}
			}
		}
		return Math.random() * 180;
	}

	/** Increase our heath, we caught a fish. */
	public void caughtFish() {
		// two fish per day for optimum health
		// we caught one for today, if we go through here twice, then we'll have enough.
		// HOWEVER, we won't start trying to catch fish unless we're half way hungry.
		health += FISH_HEALTH_ADDER;
	}
```

```
	/** Returns the health. */
	public int getHealth() {
		return health;
	}

}
* Fish.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 * Copyright (C) 2008 Simon Redman
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import java.awt.Color;
import java.util.Random;

import turtlekit.kernel.Turtle;

/** A Fish
 */

@SuppressWarnings("serial")
public class Fish extends MyTurtle {

	public static final String FISH_ROLE = "fish";

	/** We
	 * need to eat two bugs per day and this is counted in cycles,
	 */
	public static final int INSECT_HEALTH_ADDER = DAY/2;

	/** the health of the fish. This is zero-1000 and decreases one per cycle but goes
	 * up 100 each time an insect is eaten.
	 */
	protected int health;

	/** The vision radius is used to decide how easily the
	 * otter can see the fish.
	 */
	protected int visionRadius;

	/** The constructor.  The vision radius is used to decide how easily the
	 * fish can see the insects.
```

```
 * @param visionRadius
 */
public Fish(int visionRadius) {
	super("live");
	this.visionRadius = visionRadius;
}

public Fish() {
	//first behavior to do (here it is the only behavior of this turtle)
	super("live");
}



public void setup() {
	setRole(FISH_ROLE);
	randomHeading();
	setColor(Color.white);
	if (countTurtlesHere() > 0) {
		fd(1);
	}
	health = DAY;  // need to eat at least twice per week
}

/** The behavior, this is called every cycle. */
public String live() {
	// if we've been caught, return null which is our death.
	if (caught()) {
		return null;
	}

	health--;
	if ( health <= -WEEK*2 ) { // we can survive half a week without food, if we haven't eaten in half
a week, we're dead
		//return null; // dead
	}

	// twice a year, we'll bred.
	// For simplicity, if we're not hungry we'll have a chance to produce between 0 and 100 fish
	// (Males don't reproduce, some females have more than one pup, but some have none)
	// Only 2+ year old otters bred
	if ( health >= INSECT_HEALTH_ADDER && cycleCount != 0  && random.nextInt(500) ==
cycleCount %500 ) { // (cycleCount % (YEAR/6) == 0) ) {
		//if ( random.nextBoolean() ) { // only works for 50%
		int offspringCount = random.nextInt(10);
		for ( int counter = 0 ; counter < offspringCount ; counter++ ) {
			OtterLauncher.otterLauncher.newFish();
		}
	}

	turnRight(Math.random() * 60);
	turnLeft(Math.random() * 60);
	move();
	cycleCount++;  // increment the cycle count, we survived another round
	return "live";
}
```

```
	void move() {
		Random r = new Random(); // Default seed comes from system time.
		int speed = r.nextInt(2);
		// int randomizer = r.nextInt(1); // Could be used to futher randomize
		// speeds

		//System.out.println("FISH " + (speed + 2));
		for (int i = 0; i < 4; i++)
			if (countTurtlesAt(dx(), dy()) > 0) {
				if (Math.random() > .5)
					turnRight(Math.random() * 170);
				else
					turnLeft(Math.random() * 170);
			}

		// avoid being two on the same patch
		if (countTurtlesAt(dx(), dy()) == 0) {

		}
		fd(speed + 2);
	}

	/** test if I'm dead.  */
	boolean caught() {
		// this is a double loop. i and j are the coordinates on the screen, so incrementing
		// i scans along the X axis.  At each point on the X axis, j is incremented so
		// the inner loop is scanning the Y axis.  So the outer loop scans across, and at
		// each point the inner loop scans up and down, so the whole display is scanned
		// Ignore the if (true...) line
		// The "turtlesAt" finds all the turtles at the given X and Y (i,j)
		// but this returns an array, that may be null or have no elements
		// if it is not null and has at least one element, then we'll look through for an otter
		// (there are probably at least two agents here, the otter and the fish).  If we find an
		// otter then we're dead (return true).
		for (int i = -1; i <= 1; i++) {
			for (int j = -1; j <= 1; j++) {
				if (true || !(i == 0 && j == 0)) {
					Turtle[] tur = turtlesAt(i, j);
					Turtle otter = null;
					if (tur != null && tur.length > 1 ) {
						for ( int counter = 0 ; counter < tur.length ; counter++ ) {
							if
( tur[counter].isPlayingRole(Otter.OTTER_ROLE)) { // instead of "instanceof". So otter can be another java class
								otter = tur[counter];
							}
						}
						if ( otter != null ) {
							//System.out.println ("Otter? " + otter);
							// if it's really an otter, then we'll indicate that is
caught a fish.
							if (otter instanceof Otter &&
((Otter)otter).getHealth() < Otter.FISH_HEALTH_ADDER) {
								((Otter)otter).caughtFish();
								return true;
							}
```

```
                                                }
                                        }
                                }
                        }
                }

                //if (cpt > 3) {
                //      return true;
                //} else {
                //      return false;
                //}
                // if we escaped the otter then we're still alive!!! ^_^
   return false;
        }
        double towardsAIncect(int radius) {
                if ( health < INSECT_HEALTH_ADDER ) {  // we won't chase an insect unless we're down an
insect (this number could be adjusted. */
                        for (int i = -radius; i <= radius; i++) {
                                for (int j = -radius; j <= radius; j++) {
                                        if (!(i == 0 && j == 0)) {
                                                Turtle[] tur = turtlesAt(i, j);
                                                if (tur != null && tur.length > 0 &&
tur[0].isPlayingRole("fish")) { // instead of "instanceof". So fish can be another java class
                                                        return towards(tur[0].xcor(), tur[0].ycor());
                                                }
                                        }
                                }
                        }
                }
                return Math.random() * 180;
        }
        /** Hiding percentage.  The percentage depends on how hungry the fish is.  If it's full, it'll
         * hide, if it's hungry it'll be more obvious.
         *
         */
  public int hidingPercentage () {
        return INSECT_HEALTH_ADDER-health;
  }

  /** We caught an insect add 100 to the health. */
        public void caughtInsect() {
                health+= INSECT_HEALTH_ADDER;
        }
}
/*
 * Otter.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```java
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import java.util.Random;

import madkit.kernel.AgentAddress;
import turtlekit.kernel.Turtle;

/**
 * A Otter
 *
 * @author Fabien MICHEL
 * @version 1.1 17/10/2000
 */

@SuppressWarnings("serial")
public abstract class MyTurtle extends Turtle {

	/** The role we are playing.  We're assuming only one. */
	protected String role;

	/** This is the cycle count, it represents how long we've been alive. */
	protected int cycleCount;

	//OK, here're some totally arbitrary definitions, that may need
tweaking to get it right.
	// for now, one cycle = 30 mintes (but the otter only hunts during
daylight, so 12 hours.
	/** A Day represented in cycle counts  */
	public static final int DAY = 1*24; //30*24;

	/** A Week represented in cycle counts  */
	public static final int WEEK = DAY*7;

	/** A Year. */
	public static final int YEAR = DAY*365;

	/** Random number generator for this turtle. */
	Random random;

	/** Default constructor. */
	public MyTurtle() {
		super();
		random = new Random(); // Default seed comes from system time.
	}

	/** Required from superclass. */
	public MyTurtle(String initMethod) {
		super(initMethod);
		random = new Random(); // Default seed comes from system time.
	}
```

```
	/** returns the count of this species */
	public int getAgentCount() {
		AgentAddress[] agents = getAgentsWithRole("Turtlekit", "OTTER", role);
		return agents.length;
	}

	/** returns the count of this species */
	public int getAgentCount(String role) {
		AgentAddress[] agents = getAgentsWithRole("Turtlekit", "OTTER", role);
		return agents.length;
	}

	/** Assume that we are only playing one role. */
	public void setRole (String role ) {
		// I would override this but it's marked as "final" for no good reason.
		playRole(role);
		this.role = role;
	}

	/** Returns the number of DAYS this agent has been alive. */
	public int daysAlive () {
		return cycleCount/DAY;
	}

	/** Returns the number of YEARS this agent has been alive. */
	public int yearsAlive () {
		return cycleCount/YEAR;
	}

}
/*
 * OtterLauncher.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import turtlekit.kernel.Launcher;

/** Otter simulation launcher
```

```
 @author Fabien MICHEL
 @version 1.1 6/12/2000 */

@SuppressWarnings("serial")
public class OtterLauncher extends Launcher {

	public static final String SIMULATION_NAME = "OTTER";

	/** The launcher.  We need this to add agents. */
	public static OtterLauncher otterLauncher;
	/**
	 *
	 */
	int numberOfprey = 500;  // 1000;
	int numberOfpredator = 2;
	int numberOfInsects = 3000; //This is constant, one dies, one is made.
	int predatorVision = 6;
	int preyVision = 5;

	public OtterLauncher() {
		setCyclePause(10);
		setSimulationName(SIMULATION_NAME);
		setWidth(1000);
		setHeight(100);
	}

	public void setNumberOfprey(int add) {
		numberOfprey = add;
	}

	public int getNumberOfprey() {
		return numberOfprey;
	}

	public void setPredatorVision(int add) {
		predatorVision = add;
	}
	public void setpreyVision(int add) {
		preyVision = add;
	}
	public int getPredatorVision() {
		return predatorVision;
	}
	public int getPreyVision() {
		return preyVision;
	}
	public void setNumberOfpredator(int add) {
		numberOfpredator = add;
	}

	public int getNumberOfpredator() {
		return numberOfpredator;
	}

	public void setNumberOfInsects(int add) {
```

```
		numberOfInsects = add;
	}

	public int getNumberOfInsects() {
		return numberOfInsects;
	}

	public void addSimulationAgents() {
		otterLauncher = this;  // save the launcher.
		setCyclePause(10);

		for (int i = 0; i < numberOfprey; i++) {//add the prey with the method addTurtle
			Fish fish = new Fish(preyVision);
			addTurtle(fish);
		}
		for (int i = 0; i < numberOfpredator; i++) { //add the predator with the method addTurtle
			Otter otter = new Otter(predatorVision);
			addTurtle(otter);
		}

		for (int i = 0; i < numberOfpredator; i++) { //add the predator with the method addTurtle
			OtterCompetitor otter = new OtterCompetitor(predatorVision);
			addTurtle(otter);
		}
		for (int i = 0; i < numberOfInsects; i++) { //add the predator with the method addTurtle
			Insect insect = new Insect();
			addTurtle(insect);
		}
		// add a statistics agent
		StatisticsAgent statisticsAgent = new StatisticsAgent();
		addTurtle(statisticsAgent);

		addViewer(6); // we choose a default viewer with a cell size of 3

	}

	/** Adds a new otter.  */
	public void newOtter() {
		Otter otter = new Otter(predatorVision);
		addTurtle(otter);
	}
	/** Adds a new fish..  */
	public void newFish() {
		Fish fish = new Fish();
		addTurtle(fish);
	}
	/** Adds a new insect.  */
	public void newInsect() {
		Insect insect = new Insect();
		addTurtle(insect);
	}

}

/*
 * Otter.java -TurtleKit - A 'star logo' in MadKit
```

```
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 *
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import java.awt.Color;

import turtlekit.kernel.Turtle;

/**
 * A Otter
 *
 *
 */

@SuppressWarnings("serial")
public class Otter extends MyTurtle {

	public static final String OTTER_ROLE = "otter";

	/** We'll start chasing fish when we're half hungry.  Since we
	 * need to eat two fish per day and this is counted in cycles,
	 * half hungry is half a day.
	 */
	public static final int FISH_HEALTH_ADDER = DAY/10;

	/** The vision radius is used to decide how easily the
	 * otter can see the fish.
	 */
	protected int visionRadius;

	/** The health of the otter.  This is 0-1000 and decrease by one each cycle.
	 * a zero otter is a dead otter.
	 */
	protected int health;

	/** The constructor.  The vision radius is used to decide how easily the
	 * otter can see the fish.
	 * @param visionRadius
	 */
	public Otter(int visionRadius) {
		super("live");
```

```
		this.visionRadius = visionRadius;
	}

	public void setup() {
		// Random generator = new Random();
		playRole(OTTER_ROLE);
		randomHeading();
		setColor(Color.red);
		if (countTurtlesHere() > 0) {
			fd(10);
		}
		health = DAY;
	}

	public String live() {
		setHeading(towardsAFish(visionRadius));
		// once a year (cycleCount % YEAR), we'll bred.
		// For simplicity, if we're not hungry we'll have a 50% chance of producing an otter
		// (Males don't reproduce, some females have more than one pup, but some have none)
		// Only 2+ year old otters bred
		if ( cycleCount > YEAR*2 && health >= FISH_HEALTH_ADDER && cycleCount != 0  && 
(cycleCount % YEAR == 0) ) {
			//if ( random.nextBoolean() ) { // only works for 50%
			if ( random.nextInt(2) > 0 ) {  // pick an integer and a max/min
				OtterLauncher.otterLauncher.newOtter();
			}
		}
		health--;
		if ( health <= -WEEK*4 ) { // we can survive a week without food, if we haven't eaten in a week, 
we're dead
			return null; // dead
		}
		move();
		cycleCount++;  // increment the cycle count, we survived another round
		return "live";
	}

	void move() {
		int speed = random.nextInt(5);
		if ( health < 25 ) {
			speed += 1; // hungry
		}
		else if ( health > 75 ) {
			speed = 1;  // why bother to move I'm full
		}
		//System.out.println("OTTER " + (speed + 1));
		if (countTurtlesAt(dx(), dy()) > 0) {
			turnRight(50);
		}
		if (countTurtlesAt(dx(), dy()) > 0) {
			turnLeft(100);
		}
		if (countTurtlesAt(dx(), dy()) == 0) {
			fd(speed + 1);
		}
		else {
```

```
				turnRight(50);
			}
		}

		/** Move towards a fish.  We'll only do this if we're hungry, that is, if health is down a fish. */
		double towardsAFish(int radius) {
			if ( health < FISH_HEALTH_ADDER ) {  // we won't chase fish unless we're down a fish (this
number could be adjusted. */
				for (int i = -radius; i <= radius; i++) {
					for (int j = -radius; j <= radius; j++) {
						if (!(i == 0 && j == 0)) {
							Turtle[] tur = turtlesAt(i, j);
							if (tur != null && tur.length > 0 &&
tur[0].isPlayingRole("fish")) { // instead of "instanceof". So fish can be another java class
								return towards(tur[0].xcor(), tur[0].ycor());
							}
						}
					}
				}
			}
			return Math.random() * 180;
		}

		/** Increase our heath, we caught a fish. */
		public void caughtFish() {
			// two fish per day for optimum health
			// we caught one for today, if we go through here twice, then we'll have enough.
			// HOWEVER, we won't start trying to catch fish unless we're half way hungry.
			health += FISH_HEALTH_ADDER;
		}

		/** Returns the health. */
		public int getHealth() {
			return health;
		}

}
```

```
* Fish.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 * Copyright (C) 2008 Simon Redman
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;
```

```
import java.awt.Color;
import java.util.Random;

import turtlekit.kernel.Turtle;

/** A Fish
 */

@SuppressWarnings("serial")
public class Fish extends MyTurtle {

	public static final String FISH_ROLE = "fish";

	/** We
	 * need to eat two bugs per day and this is counted in cycles,
	 */
	public static final int INSECT_HEALTH_ADDER = DAY/2;

	/** the health of the fish. This is zero-1000 and decreases one per cycle but goes
	 * up 100 each time an insect is eaten.
	 */
	protected int health;

	/** The vision radius is used to decide how easily the
	 * otter can see the fish.
	 */
	protected int visionRadius;

	/** The constructor.  The vision radius is used to decide how easily the
	 * fish can see the insects.
	 * @param visionRadius
	 */
	public Fish(int visionRadius) {
		super("live");
		this.visionRadius = visionRadius;
	}

	public Fish() {
		//first behavior to do (here it is the only behavior of this turtle)
		super("live");
	}



	public void setup() {
		setRole(FISH_ROLE);
		randomHeading();
		setColor(Color.white);
		if (countTurtlesHere() > 0) {
			fd(1);
		}
		health = DAY;  // need to eat at least twice per week
	}

	/** The behavior, this is called every cycle. */
```

```
	public String live() {
		// if we've been caught, return null which is our death.
		if (caught()) {
			return null;
		}

		health--;
		if ( health <= -WEEK*2 ) { // we can survive half a week without food, if we haven't eaten in half
a week, we're dead
			//return null; // dead
		}

		// twice a year, we'll bred.
		// For simplicity, if we're not hungry we'll have a chance to produce between 0 and 100 fish
		// (Males don't reproduce, some females have more than one pup, but some have none)
		// Only 2+ year old otters bred
		if ( health >= INSECT_HEALTH_ADDER && cycleCount != 0  && random.nextInt(500) ==
cycleCount %500 ) { // (cycleCount % (YEAR/6) == 0) ) {
			//if ( random.nextBoolean() ) { // only works for 50%
			int offspringCount = random.nextInt(10);
			for ( int counter = 0 ; counter < offspringCount ; counter++ ) {
				OtterLauncher.otterLauncher.newFish();
			}
		}

		turnRight(Math.random() * 60);
		turnLeft(Math.random() * 60);
		move();
		cycleCount++;  // increment the cycle count, we survived another round
		return "live";
	}

	void move() {
		Random r = new Random(); // Default seed comes from system time.
		int speed = r.nextInt(2);
		// int randomizer = r.nextInt(1); // Could be used to futher randomize
		// speeds

		//System.out.println("FISH " + (speed + 2));
		for (int i = 0; i < 4; i++)
			if (countTurtlesAt(dx(), dy()) > 0) {
				if (Math.random() > .5)
					turnRight(Math.random() * 170);
				else
					turnLeft(Math.random() * 170);
			}

		// avoid being two on the same patch
		if (countTurtlesAt(dx(), dy()) == 0) {

		}
		fd(speed + 2);
	}

	/** test if I'm dead.  */
	boolean caught() {
```

```
		// this is a double loop. i and j are the coordinates on the screen, so incrementing
		// i scans along the X axis.  At each point on the X axis, j is incremented so
		// the inner loop is scanning the Y axis.  So the outer loop scans across, and at
		// each point the inner loop scans up and down, so the whole display is scanned
		// Ignore the if (true...) line
		// The "turtlesAt" finds all the turtles at the given X and Y (i,j)
		// but this returns an array, that may be null or have no elements
		// if it is not null and has at least one element, then we'll look through for an otter
		// (there are probably at least two agents here, the otter and the fish).  If we find an
		// otter then we're dead (return true).
		for (int i = -1; i <= 1; i++) {
			for (int j = -1; j <= 1; j++) {
				if (true || !(i == 0 && j == 0)) {
					Turtle[] tur = turtlesAt(i, j);
					Turtle otter = null;
					if (tur != null && tur.length > 1 ) {
						for ( int counter = 0 ; counter < tur.length ; counter++ ) {
							if
( tur[counter].isPlayingRole(Otter.OTTER_ROLE)) { // instead of "instanceof". So otter can be another java class
								otter = tur[counter];
							}
						}
						if ( otter != null ) {
							//System.out.println ("Otter? " + otter);
							// if it's really an otter, then we'll indicate that is
caught a fish.
							if (otter instanceof Otter &&
((Otter)otter).getHealth() < Otter.FISH_HEALTH_ADDER) {
								((Otter)otter).caughtFish();
								return true;
							}
						}
					}
				}
			}
		}

		//if (cpt > 3) {
		//	return true;
		//} else {
		//	return false;
		//}
		// if we escaped the otter then we're still alive!!! ^_^
  return false;
	}
	double towardsAIncect(int radius) {
		if ( health < INSECT_HEALTH_ADDER ) {  // we won't chase an insect unless we're down an
insect (this number could be adjusted. */
			for (int i = -radius; i <= radius; i++) {
				for (int j = -radius; j <= radius; j++) {
					if (!(i == 0 && j == 0)) {
						Turtle[] tur = turtlesAt(i, j);
						if (tur != null && tur.length > 0 &&
tur[0].isPlayingRole("fish")) { // instead of "instanceof". So fish can be another java class
							return towards(tur[0].xcor(), tur[0].ycor());
						}
```

```
                                        }
                                }
                        }
                }
                return Math.random() * 180;
        }
        /** Hiding percentage.  The percentage depends on how hungry the fish is.  If it's full, it'll
         * hide, if it's hungry it'll be more obvious.
         *
         */
  public int hidingPercentage () {
        return INSECT_HEALTH_ADDER-health;
  }

  /** We caught an insect add 100 to the health. */
        public void caughtInsect() {
                health+= INSECT_HEALTH_ADDER;
        }
}
/*
 * Otter.java -TurtleKit - A 'star logo' in MadKit
 * Copyright (C) 2000-2002 Fabien Michel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package turtlekit.simulations.otters;

import java.util.Random;

import madkit.kernel.AgentAddress;
import turtlekit.kernel.Turtle;

/**
 * A Otter
 *
 * @author Fabien MICHEL
 * @version 1.1 17/10/2000
 */

@SuppressWarnings("serial")
public abstract class MyTurtle extends Turtle {

      /** The role we are playing.  We're assuming only one. */
      protected String role;
```

```
	/** This is the cycle count, it represents how long we've been alive. */
	protected int cycleCount;

	//OK, here're some totally arbitrary definitions, that may need 
tweaking to get it right.
	// for now, one cycle = 30 mintes (but the otter only hunts during 
daylight, so 12 hours.
	/** A Day represented in cycle counts  */
	public static final int DAY = 1*24; //30*24;

	/** A Week represented in cycle counts  */
	public static final int WEEK = DAY*7;

	/** A Year. */
	public static final int YEAR = DAY*365;

	/** Random number generator for this turtle. */
	Random random;

	/** Default constructor. */
	public MyTurtle() {
		super();
		random = new Random(); // Default seed comes from system time.
	}

	/** Required from superclass. */
	public MyTurtle(String initMethod) {
		super(initMethod);
		random = new Random(); // Default seed comes from system time.
	}

	/** returns the count of this species */
	public int getAgentCount() {
		AgentAddress[] agents = getAgentsWithRole("Turtlekit", "OTTER", 
role);
		return agents.length;
	}

	/** returns the count of this species */
	public int getAgentCount(String role) {
		AgentAddress[] agents = getAgentsWithRole("Turtlekit", "OTTER", 
role);
		return agents.length;
	}

	/** Assume that we are only playing one role. */
	public void setRole (String role ) {
		// I would override this but it's marked as "final" for no good 
reason.
		playRole(role);
		this.role = role;
	}

	/** Returns the number of DAYS this agent has been alive. */
	public int daysAlive () {
		return cycleCount/DAY;
```

```
	}

	/** Returns the number of YEARS this agent has been alive. */
	public int yearsAlive () {
		return cycleCount/YEAR;
	}

}
```