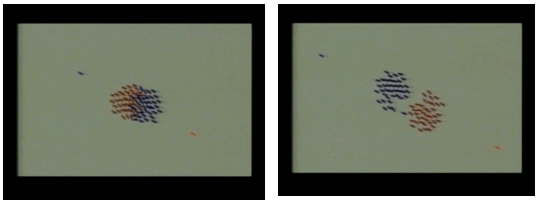
Introduction to Simulations

CS1316: Representing Structure and Behavior

Story

- What's a simulation? Why do we simulate?
  - Discrete vs. Continuous
  - Resources
- Building software to be modifiable: Software Engineering
  - Building models out of objects: aggregation, generalizing and specializing
- Continuous Simulations
  - Predatory-prey: Wolves and Deer
  - Changing our simulation
  - Creating *hungry* wolves
  - Other options: Hungry deer? Deer sex? Wolf sex?
- How do we compare simulations?
  - Creating text files

Wildebeests as Simulations



Simulations

- "A simulation is a representation of a system of objects in a real or fantasy world.
  The purpose of creating a computer simulation is to provide a framework in which to understand the simulated situation, for example, to understand the behavior of a waiting line, the workload of clerks, or the timeliness of service to customers.
  A computer simulation makes it possible to collect statistics about these situations, and to test out new ideas about their organization."
  - Adele Goldberg & David Robson, *Smalltalk-80: The Language and Its Implementation* (Addison-Wesley, 1989)

Simulations and Objects

- Object-oriented programming was invented, in part, to make simulations easier to build!
- The characteristics of objects make them *more* like real world objects, e.g.,
  - Each *thing* ***knows*** some stuff and ***knows how*** to do some stuff.
  - Objects get things done by asking each other to do things.
  - Your internals are private, unless you want to make them otherwise.

Continuous vs. Discrete Simulations

- Two main kinds of simulations in the world.
- Continuous: Each moment of time is simulated.
  - When every moment counts.
- Discrete: Skip to the important moments.
  - Want to simulate 100 years?

Resources

- Resources are points of *coordination* in a simulation.
  - Examples: A cashier, a library book, a parking space on a ferry, a jelly bean.
- Some resources are *fixed* and others are *produced* and *consumed.*
- Some resources are *renewable* and *shared*.
- Others are *coordinated*.
  - Example: For a surgeon to do a surgery, the patient must meet the surgeon at the *operating table* (the resource)

When an object has to wait...

- What happens if you (or your proxy object) need a resource and it's not available?
  - You wait in a *queue*
  - A list that is *first-in-first-out (FIFO)*

A simulation is an executed model

- Setting up a simulation is a process of *modeling* the world (real or fantasy) to be simulated.
- That model is realized in terms of *objects*.
- We want our model to:
  - Reflect the world.
  - Be easy to extend and change.
- Some of our modeling techniques:
  - Aggregation
  - Generalization and specialization

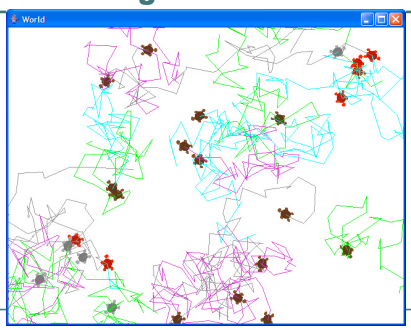
Aggregation

- Some objects are made up of other objects.
  - Cars have engines
  - People have livers and lungs
    - These internal things are objects, too!
    - Livers don't directly mess with the innards of lungs!
- We call this *aggregation*
  - Putting references to some objects inside of other objects.

Generalization and Specialization

- There are general and specialized forms of real world objects.
  - Cells are biological objects that have membranes and a nucleus and mitochondria and…
  - Blood, lung, and liver cells are all *cells* but have specialized functions.
- The superclass-subclass relationship is a way of *modeling* general forms of objects and specialized forms of objects

Making it concrete: Wolves eating deer



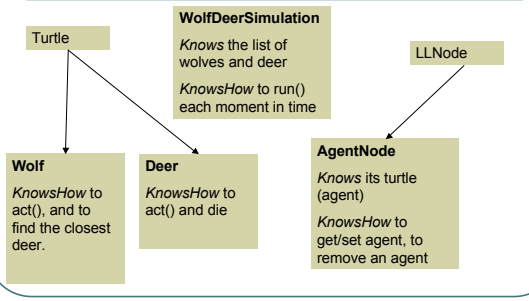
Running the simulation

```
Welcome to DrJava.
> WolfDeerSimulation wds = new WolfDeerSimulation()
> wds.run()
>>> Timestep: 0
Wolves left: 5
Deer left: 20
>>> Timestep: 1
Wolves left: 5
Deer left: 20
<SIGH!> A deer died...
>>> Timestep: 2
Wolves left: 5
Deer left: 19
>>> Timestep: 3
Wolves left: 5
Deer left: 19
<SIGH!> A deer died...
>>> Timestep: 4
Wolves left: 5
Deer left: 18
```

An Example Simulation

- The WolfDeerSimulation is a *continuous* simulation.
  - Each moment in time is simulated.
- It has no *resources*.
- It is a *predator-prey* simulation
  - A common real world (ecological) situation.
  - There are parameters to change to explore under what conditions predators and prey survive and in what numbers.

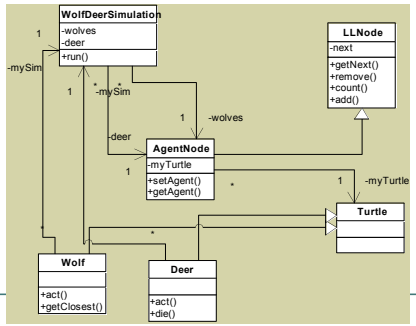
The Model of this Simulation



Complicated Set of Relationships in this Model

- Wolf and Deer are kinds of Turtle
  - Specializations of Turtle
- AgentNode is a kind of LLNode
- AgentNodes each have one Turtle (Wolf or Deer) inside it.
- WolfDeerSimulation has two AgentNodes for the lists of live wolves and deer.
- Each Wolf and Deer knows what simulation its in.

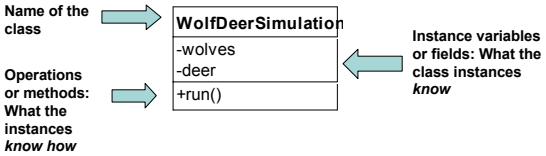
A UML Class Diagram



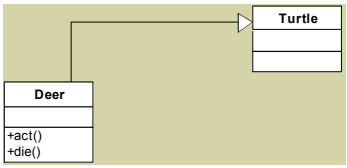
Unified Modeling Language (UML)

- This is a *UML class diagram.*
  - A graphical notation for describing the relationships between classes in a model.
- UML is a standard that describes several different kinds of diagrams.
  - Collaboration diagrams: How objects work together and how they call on one another.
  - Sequence diagrams: What the order of events are in an object system.

A class in a UML class diagram

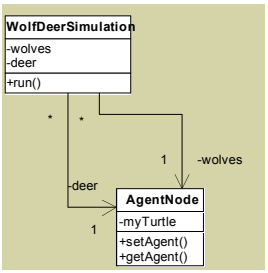


Generalization-specialization relationships



A Deer is a subclass of Turtle: It's a specialization of Turtle

Associations



WolfDeerSimulation has two AgentNodes in it: One to represent wolves and one to represent deer.

AgentNodes don't know their simulation

A Class Diagram describes the Model, without the Code