Programming Languages and Translators

COMS 4115

EcoMod

Ecosystem Modeling Language

White Paper

Vika Kanchakouskaya

June 11, 2007

Introduction

Computer modeling is becoming increasing important for understanding how our world works. Experimenting and interacting with virtual models enables us to get a deeper insight into our surroundings and predict effects of various circumstances on the environments we live in. EcoMod is a language that helps simulate simplified virtual models of ecosystems, which can be studied, observed and experimented with.

Value Proposition

EcoMod provides support for creating various ecosystems of plants, animals and physical environments. With EcoMod one can create a simplified system inhabited by living and non-living entities, watch the system develop, simulate interference of natural and artificial phenomena into the system, and test robustness of the components of the system. EcoMod makes it easy for specialists in various domains to simulate such ecosystems as Rainforest, Desert, Marine, Human, and Urban ecosystems, etc. Models created with EcoMod allow for observation of these systems in the time progression, and help understand development of its components, their responses to various conditions, and their reaction to stimuli of leaving and non-leaving elements of the environment. The language is especially valuable for scientists who are, trying to understand evolution, predict effects of climate change and global warming on the life on Earth, or determine impact of human beings on the environment.

EcoMod Users

EcoMod is simple enough to be adopted by scientists of various domains who don't have special training in Computer Science, but is powerful enough to simulate a wide range of ecosystems. EcoMod users are scientists willing to experiments and understand properties of different communities, their habitats, and interaction of their components. Biologist can simulate ecosystems of Taiga, Tundra, Savanna, and Desert with their unique flora and fauna. Ecologists, with the help of EcoMod models can explore the effects of deforestation on the habitat of Rainforests, and impact of Greenhouse effect and raising temperatures on various ecosystems around the world. Urban Planning specialists can explore the effects of urbanization on human and environmental health. Marine scientists can observe the depths unreachable by human beings.

Properties of the language

EcoMod is a high-level domain-specific language which makes ecosystem modeling easier than with a general purpose programming languages like C++ or Java. EcoMod is compiled into C++ code, so it is supported on any platform capable of running C++ programs. The constructs of the language such as **system**, **objects**, **qualities**, **states**, and **actions** make it simple to create a model of an ecosystem, and observe its development with progression of time.

System is defined as the simplified ecosystem being modeled.

System is inhabited by **objects.** Various components of an ecosystem, like animals, plants and other non-living elements can be introduces to the model by defining the corresponding objects.

Objects may possess certain qualities, which can be depicted with the help of **quality** language construct. Qualities are a set of significant attributes, such as age, sex, defense strategy, health, etc, which describe and help differentiate between the objects of an ecosystem.

The behavior of the objects is modeled via finite state machine representation. **States** in EcoMod are constructs that store certain past information about the object up to the present time, and changes between the states occur by means of transitions. E.g. for a mammal, the following states of development can be defined: embryonic, metamorphosis, regeneration, aging, and death, and transitions between these states occur when an animal reaches a certain age.

Actions represent a way of interaction between objects and their community as well as between each other. Actions can be originated from the environment, and from other members of the habitat. Actions may trigger transitions between states and affect qualities of objects.

……..

In conclusion and before we proceed with an example of EcoMod definittions, we would like to add that the first release of the language, depending on how well the progress goes, may have a limited support for the features described in this document. The initial version of the language may enable the users to build simple models inhabited by a small number of objects that a capable of limited interaction. However the language will be developed further in order to support complex and fully functional models of ecosystems.

Sample Definitions in EcoMod

```
// Ecosystem is defined with system keyword
// The objects comprising the habitat of the ecosystem
// may be  nested within the definition of the system
System AquariumEcoSystem
{

// System object has a built is associative array to keep track if its objects
// which can be referred to by keyword habitat
// The built in operator add(Object) takes an Object instance
// as a parameter and adds the Object to the habitat

  // Ecosystem objects can be defined with object keyword
  // nested Object definitions are automatically added to habitat
  Object Whale
  {

  // Each object has a set of qualities that have either numeric values
  // (to indicate the degree of quality) or string values
  // qualities are defined with quality keyword followed by a semicolon
  Qualities:
      age             = 0
      health          = good
      defStrategy     = prey
      hunger          = 0
      avgNumChildren = 1

  // States are defined with the help of keyword states
  // The name of the state set is listed to the left of the assignment operator
  // The possible states are enumerated on the right of the assignment operator
  // separated by spaces. The first state  is taken to be the initial
  // state for this state set
  States:
    development   =  metamorphosis regeneration aging death
    welfare       =  full hungry

   // Transitions must be defined for every pair of states that can be
  // transformed from one to another. transitions keyword followed
  // by a colon mist be used for that purpose. from, to and if keywords indicate
```

```
  // the direction and the condition for the transition. Transition declaration
  //  must start with the name of the state set
  Transitions:
     development from metamorphosis to regeneration if  age >= 2
     development from regeneration to aging if  age >= 7
     development from aging to death if  age >=12
     welfare from full to hungry if hunger  >=10
     welfare from hungry to full if hunger  < 10

  // Actions are similar to functions. The describe what the objects can
  // do and what can be done to the objects. They are defined with
  // actions keyword
   Actions:
     Eat
     {
        if hunger > 0
           hunger --
        fi
     }

     Regenerate
     {
        if development == regeneration
           for child in avgNumChildren
              add(Whale) // this copies the parent version of Fish object with all
                         // the default values of qualities and states
     }

    }

}
```