Mongoose

Bo Xu  Tester
Chris Haueter  System Integrator
Etan Zapinsky  System Architect
Mark Fischer  Language Guru
Michael Rubin  Project Manager
Overview

- agent-based, discrete-event simulations
- clean syntax, object-oriented
- minimal language boilerplate
- post-simulation analysis
- innovative syntactical features
Project Management

- fluid roles
- Google Hangout meetings
- Gantt charts
- private mailing list + Github issues
- *dictatorial fiat + whips* ;)}
agent A {
    int x
    create()
    print("On create A")
    x = 0
    action {
        x = x + 1
    }
    destroy {
    }
}

agent B {
    int x
    create()
    print("On create B")
    x = 0
    action {
        x = one.x + 1
    }
    destroy {
    }
}

environment {
    populate {
        A one = A()
        B two = B()
    }
    action {
        print("one.x:", one.x)
        print("two.x:", two.x)
    }
    terminate{
        2:(x > 3) {
            print("I should print on turn 4")
        }
        (false) {
            print("I should never print")
        }
    }
    analysis {
    }
}
Probabilistic features

```java
pif(.2) {
    ~int x = 3
    print("x:", x)
}
pelif (.5) {
pif (0.6) {
    ~string s = "dependent probs!"
    print(s)
}
~int y = 7
print("y:", y)
} pelse {
    ~boolean b = false
    print("b:", b)
}

~int x = ( 1:5 | 3:4 | 7:8 )
print(x)
if( ( 1:true | 1:false ) ){
    print(1)
} else {
    print(0)
}

string rand(~string x, ~string y) {
    return ( 1:x | 1:y )
}
print(rand('hi', 'bye'))

( 1:( 1:'a' | 1:'b' ) | 1:'c' )
```
Compiler Tools

- PLY (Python Lex-YACC)
Translator Architecture

Frontend
- Lexer
- Parser

Backend
- AST walker
- Mongoose

Source Program
- Tokenized Code
- Abstract Syntax Tree
- Processed Abstract Syntax Tree
- Running Program
Running a Mongoose program

- environment.populate()
- agent.create()
- agent.action()
- environment.action()
- terminate()
- analysis()
PIG: Pig is a two player game played with a single die. The object of the game is to accumulate 100 points. The player to roll first is randomly determined by coin flip. During each turn a player repeatedly rolls the die earning points equal to the face value on each roll until:

- A one is rolled, in which case the turn is over and the player forfeits all points earned so far on that turn.

- The player holds, takes the points earned so far and hands the dice over to the opponent.

Your job is to write a Python application that allows two computer players to play Pig. Your computer player should have two different possible modes of play. The beginner level computer strategy is simply to always hold after 3 rolls. The advanced level computer strategy is to halt after attaining 20 points on the turn.
(demos)
Mongoose Runtime Environment

- Python 2.7.2 (CPython)
- Unix only (for now)
- manage.sh
Development Environment

- pip + virtualenv + virtualenvwrapper
- git + github
- emacs / vim / sublime text 2
- cactus.py (markdown → HTML)
Future (v1.1) Features

- parallelizing (iid)
- state saving (stop/start programs)
- built-in aggregation decorators (@average)
- fix current bugs
Test Plan

- unit tests (assertions for doc and test)
- highly descriptive test names
- integration tests
- nose (automation)
- sniffer
Conclusions

● rely heavily on git

● consistent velocity → work gets done

● *state* is a bitch

● frontend ambiguity → backend complexity

● grow by accretion & integrate earlier
Special thanks to:
  ● Karan
  ● Professor Aho