Simulation: Fundamentals, Robots

Jeffery von Ronne

Department of Computer Science
Carnegie Mellon University

15-110 Unit 12A
November 14, 2011

Simulation

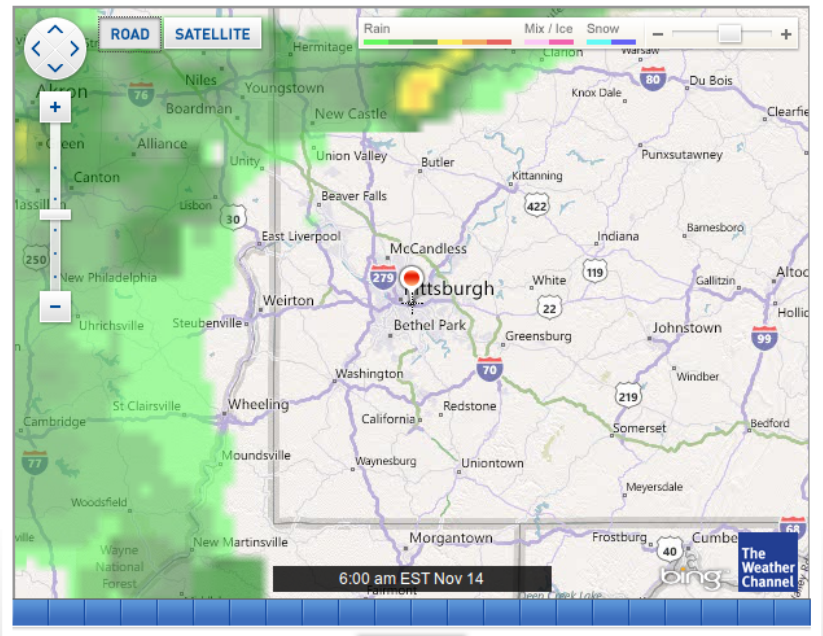
simulation The imitative representation of the functioning of one system or process by means of the functioning of another [i.e., a computer program]. (Meriam Webster)

- dynamic processes
- method of understanding and predicting the behavior of real processes
- based on mathematical models (ignore some details)
  - often include many interacting components
- distinct from but often connected to visualization

Computer Simulations

Applications of simulation include . . .

- Manhattan Project (Nuclear Detonation)
- Weather Forecasting
- Climate Modeling
- Traffic Flow Modeling
- Automobile Development
- Flight Simulators
- Chemical/Biological Processes



Approaches

- *time-stepped* vs. event-driven
- grid-based (like cellular automata, or weather forecasting)
  vs. *agent-based* (like the traffic modeling)
- *deterministic* vs. stochastic (pseudorandom numbers)

Examples

- Climate Modeling: `http://tinyurl.com/czjuhwx`
- Vehicle Traffic:
  `http://www.youtube.com/watch?v=ma0PGET3yNQz`
- Air Traffic: `http://tinyurl.com/bnn7zq`)

Approaches to Simulation

time-stepped The evolution of a system over time is modeled by updating the entire system's state to reflect the passing of each increment of time.

event-driven The process is represented as a chronological sequence of events. Events occur at instants in time and represent a change of state in the system.

grid-based The simulation associates state information with locations. Updates to each location occur based on its previous state and those of its immediate neighbors. These are also called stencil codes.

agent-based The simulation associates state information with individual objects (agents). Location is part of the state of the object

stochastic The simulation incorpoates (pseudo)-random numbers in the process of determining successive states of teh simulation.

deterministic The simulation is one that does not, and thus will always give the same result given the same inputs.

Predicting Halley's Arrival

1609 Johannes Kepler

- planets have elliptical orbits around the sun
- go faster near the sun

1686 Isaac Newton

1705 Edmund Halley

- the comets observed in 1531, in 1607, and 1682 were the same
- elliptical orbit like the planets, `http://tinyurl.com/6erguv`
- but the timing was slightly off
- orbit perturbed by attraction of Jupiter and Saturn

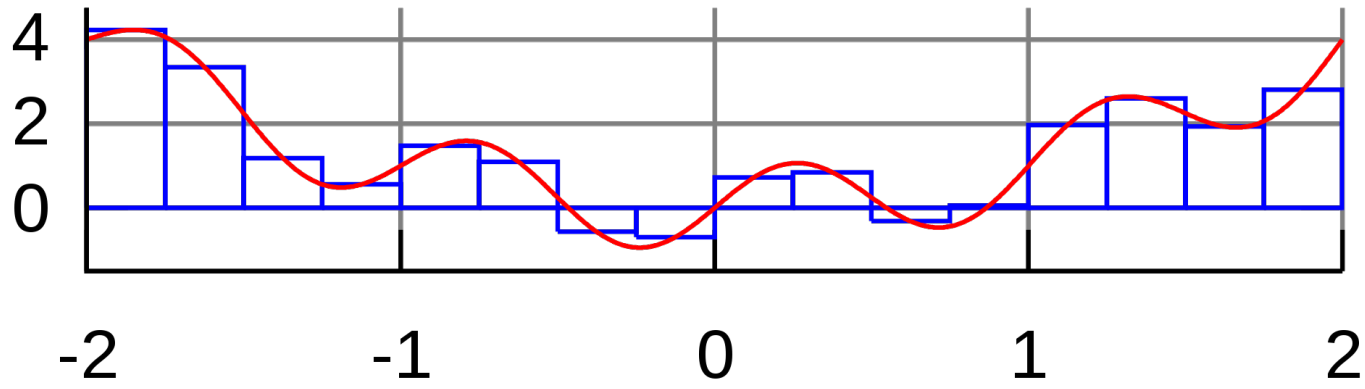
1758 Alexis Claude de Clairaut

- 5 months of calculations
- predicted arrival of Halley's Comet within a month

Clairaut's Calculation

Clairaut's Approach

- update by time steps
  - Numerical Integration of $x^2 + sin(2\pi x)$



- started with elliptical orbit
- for each time period, do the following
  1. calculate acceleration due to Jupiter and Saturn
  2. calculate change in velocity for time period
  3. use new velocity to get new elliptical orbit
  4. find new position based on current orbit
- This is a simulation of (part of) the solar system.

Limitations of Simulation

- inexact models
- imprecision in parameters / initial conditions
  - chaotic systems
- digital simulation vs. analog world
  - rounding errors (compounded through multiple iterations)

RubyLabs: Simulating a Rover

Starting Robot:

```
view_robot(:track => :on,:flag => [200,200])
```

Operations for `robot`:

`advance(t)`	Advance t time-steps
`turn(a)`	Turn clockwise by a degrees
`orient`	Rotate to approximate a circle
`heading`	the direction (in degrees) the robot is headed
`speed`	Return the robto's velocity in (meters per hour)
`track(:on)`	Turn on the tracking option
`track(:off)`	Turn off the tracking option

Robot Maneuvers: Zig Zag and Triangle

Zig Zag:

```
robot.turn(45)
4.times {
  robot.advance(5)
  robot.turn(90)
  robot.advance(5)
  robot.turn(-90)
  }
```

Triangle:

```
robot.turn(30)
3.times{
  robot.advance(10)
  robot.turn(180 - 60)
}
```

Robot maneuvers: Circle

Starting perpendicular to center...

Very Course Approximation:

```
12.times {
  robot.orient
  robot.advance(10)
}
```

A Smoother Approximation:

```
110.times {
  robot.orient
  robot.advance(1)
}
```