

CS 475/575: Simulation
Spring 2005

Slide set 1: Intro, Defs
M. Overstreet

CS 475/575 1

Class overview

⌘How to build simulations:

- ☒key issues:
 - ☒how to do quickly
 - ☒how to build the right model
 - graphics, virtual reality, simulation are combining for some applications
 - ☒how to know of it's correct
 - ☒how to interpret output correctly
 - ☒component reuse (DoD emphasis)

CS 475/575 2

Texts:

⌘Kelton, Sadowski, Sadowski, *Simulation with Arena*, available in bookstore

⌘Park, Leemis, *Discrete-Event Simulation: A First Course*, to be published by Prentice-Hall, not available in bookstores

- ☒ *The good news is it's free!*

CS 475/575 3

Since this is CS class:

- ☒ focus on computational issues
 - ☒ and how internals work
- ☒ focus on simulation programming languages
 - ☒ use right tool to solve problem
 - ☒ make new mistakes, not old ones
- ☒ speed, speed, speed:
 - ☒ build models fast
 - ☒ build fast models
 - efficient implementations
 - distributed/parallel simulation

CS 475/575 4

Class mechanics

- ⌘ Class material available at www.cs.odu.edu/~cmo.
- ☒ follow links
- ⌘ Slides are outline only! if you miss class, you will still need to get notes from someone.
- ⌘ See syllabus for prerequisites, tentative schedule, grading & lateness policies

CS 475/575 5

Simulation Languages

- ⌘ Dealing with concurrency long before other languages
 - ☒ OO stuff directly out of simulation community
- ⌘ Conflict between "quick & dirty" answer and "cpu intensive" solutions
 - ☒ quick development but slow execution
 - ☒ fast execution but slow expensive development
- ⌘ Languages:
 - ☒ arena: good for simple models
 - ☒ general purpose language (examples in C)

CS 475/575 6

Programming Assignments

- ⌘ 3 Arena assignments (CD comes with text)
 - who needs access here at ODU?
 - several arena homework problems
 - simple changes to existing code
- ⌘ 2 C/C++ assignments
 - study in efficient implementations
 - several c/c++ homework problems
- ⌘ Arena
 - really quick & easy to use: "no programming"
 - graphics, animation: lights, camera, action!
 - graphics, animation, easy integration with excel, visual basic

CS 475/575 7

Term Project

- ⌘ Part team, part individual
 - Shared data collection
 - Need data to parameterize model
 - Need data to validate model
 - Individual coding
 - Individual project report
 - Assigned after midterm

CS 475/575 8

Central problems in simulation

- ⌘ Build the right model
 - what to include; how much detail?
 - effects validity, speed, development time, reuse
- ⌘ How to get good performance?
 - basic data structure is events list
 - issues in distributed/parallel simulation
- ⌘ How do you KNOW you can believe output?
 - bad model/data/wrong assumptions?
 - bad code?
 - wrong interpretation of output?
- ⌘ How do you get others to believe output?
 - advantages & disadvantages of animation

CS 475/575 9

Tricky stuff

- ⌘ Bad RNGs
- ⌘ Incorrect statistical analysis
- ⌘ Parallel simulation

CS 475/575 10

What's simulation, at least in 475/575?

- ⌘ Use of models to replicate behavior of "real" system
- ⌘ Models can have very different objectives:
 - ☑ Scientific experimentation
 - ☑ understand behaviors of existing system
 - ☑ how does DNA work?
 - ☑ how did the "big bang" work?
 - ☑ Design a new system
 - ☑ optimize a new system, evaluate alternatives
 - ☑ find "surprises" (aka design flaws)
 - ☑ Enhance understanding (use simplified animations)
 - ☑ Develop skills (use scripted situations: land a 747 with 2 engines failed)

CS 475/575 11

What is sim (cont.)

- ⌘ Merging of
 - ☑ graphics/virtual reality/simulation
 - ☑ problems are human perception/speed of rendering
- ⌘ Much "scientific computing" is simulation
 - ☑ fluid flow problems; understanding pollution problems in Chesapeake Bay
 - ☑ problems are mostly numeric

CS 475/575 12

Different needs for different uses

- ⌘ Sometimes need complex, messy depictions-- realism important
 - ☑ for military training
 - ☑ learn to fly 747
- ⌘ Sometimes need abstract simplified models
 - ☑ use model to enhance user's understanding
 - ☑ extraneous details inhibit understanding
- ⌘ Sometimes performance crucial, sometimes not
- ⌘ Sometimes correctness crucial, sometimes not
- ⌘ Some simulations are only run once, some are used frequently

CS 475/575 13

What is simulation in 475?

- ⌘ Both textbooks take on simulation much too narrow
 - ☑ Implicitly assumes simulation is used for analysis or design
- ⌘ Ignores training, education, scientific experimentation, recreation, etc.

CS 475/575 14

Traditional uses of sim.

- ⌘ Simulation used when system of interest:
 - ☑ doesn't exist (sim used to design system)
 - ☑ is too expensive to use for "mere" studies
 - ☑ is too dangerous
 - ☑ is too slow
 - ☑ want to take quick look at alternatives to see which to pursue in detail
 - ☑ long period studies (say, a week or a decade)
 - ☑ is too fast
 - ☑ slow it down to see what's happening

CS 475/575 15

Reality: main "consumer" of sim is US Dept of Defense

⌘ Software example (old): specify, design, develop, validate software for Aegis cruiser:

- basic ftns
 - navigation
 - communication
 - weapons
 - sensors (situation awareness)

CS 475/575 16

Example: radar component

⌘ Design system so that, at any point, radar inputs to rest of system can be:

- from a REAL radar system
- from history (recorded sensings from previous field exercises)
- from simulation model (to experiment with different loadings, situations, whatever)
- from large training exercise where inputs come from other players (models, people, planes, ships)

CS 475/575 17

Uses

⌘ Such a systems could be used to:

- test effect (loadings) from newer radar system on other ship systems
- evaluate correctness of other ship systems which must respond to radar
- train personnel by simulating dangerous, expensive, but carefully scripted conditions (to meet training objectives)
- evaluate human factors: light loads, heavy loads, special circumstances

CS 475/575 18

Term: "discrete event"

- ⌘ Previous example, for most purposes, can be represented at a "discrete event" system:
 - ☒ at t1, first blip appears
 - ☒ at t2, pkt containing data is placed on network
 - ☒ how to determine how long it takes after t1 for t2?
 - ☒ at t3 pkt arrives at node i
 - ☒ at t4, second blip appears
 - ☒ . . .
- ⌘ Often all important behavior can be modeled as occurring at discrete points in time even though things change continuously in the real system

CS 475/575

19

Discrete event: more

- ⌘ How to determine how long till next important event?
 - ☒ hardware characteristics: speed of processor, size of data packet, bandwidth of network, load on network, etc. (e.g., basic physics)
 - ☒ model in detail, so that that time depends on a whole bunch to individual steps, each with its own timing characteristics
 - ☒ use timing data from a real system (study tape of real radar system)
 - ☒ random number (study data from real system to choose best fitting statistical distribution)

CS 475/575

20

Quick, brief history

- ⌘ A very old programming language is GPSS developed by Geoffrey Gordon at Bell Labs to do telephone network simulation
 - ☒ a call arrives at t1 with duration t2 (phone co had lots of data on how long calls are and when they come in)
 - ☒ call gets routed by switch X which takes t3 time to do this, etc.
- ⌘ Arena is very much like GPSS, but it wiggles (as it comes out of the box)

CS 475/575

21

"Grandparent" of OO is Simula-67

⌘ Created about 1963 by Ole-Johan Dahl and Kristen Nygaard at the Norwegian Computing Center. Based on Algol-60.

CS 475/575

22

Assignment

⌘ Read Leemis text, chapter 1

CS 475/575

23

Basic terms - 1

- ⌘ *System*: collection of interacting components usually with identifiable function or purpose.
- ⌘ A *model* is a representation of a system used to replicate some features of that system.
- ⌘ Each system must have some *boundary* between it and its *environment*.
- ⌘ *Entities* are objects of interest in the system
- ⌘ An *attribute* is a property of an entity.
- ⌘ The *state* of an entity is the set of attributes needed to describe the entity at any given time.

CS 475/575

24

More terms - 2

- ⌘ An *event* is an instantaneous occurrence that changes the state of some entity. Events are represented as occurring in 0 time.
- ⌘ An *activity* is a time period in which the state of an entity does not change (well, sorta').
- ☑ Activities start and stop with events.

CS 475/575

25

More terms - 3

- ⌘ A *discrete system* is one in which state changes occur at discrete points in time.
- ⌘ A *continuous system* is one in which some attribute is represented as changing continuously over time.
- ⌘ *Physical (or iconic) models*: often built to scare; can be used in wind tunnels, etc.
- ⌘ *Analytic models*: mathematical equations
- ⌘ *Monte Carlo simulation*: use of random numbers to approximate non-random behavior.

CS 475/575

26

Example: filling tank

Suppose we have tank with water flowing at some rate, maybe changing, and water removed at some other rate dependent on pressure, the amt of water in the tank would probably be represented as a continuous variable.

But what if we only need to react when the tank goes empty or overflows. Is this a continuous or a discrete system?

CS 475/575

27

Std discrete system: Bob's Bank

⌘ Study objectives:

- estimate average waiting time for customers
 - at drive-up window
 - who come into lobby and wait for tellers
 - who come into lobby and wait for loan officer
- estimate utilization of
 - drive-up window teller
 - lobby tellers
 - loan officer

⌘ (Might also want to study effect of on-line services, or air conditioning needs, or floor plan, or ATM reliability or ...)

CS 475/575

28

Bank example (cont)

⌘ What other information would you need to build a simulation to meet the study objectives?

Some customers arrive with deposit slips completed, some don't. Some are 21 years old while others are 80 years old. Do these matter? My answer is a question: how much money is in the budget for this simulation? More details->more effort->more money

CS 475/575

29

Main points

- ⌘ One can't just build a model of a system
- ⌘ Need to know study objectives
 - What data are needed from the simulation
 - Accuracy required

CS 475/575

30

Steps in simulation (one version)- 1

- ⌘ Determine objectives
 - outputs needed
- ⌘ Build conceptual model
 - how does system work?
- ⌘ Specify model
 - can be given to programmer
- ⌘ Data collection
 - for many uses of simulation, most time-consuming part

CS 475/575 31

Steps in simulation - 2

- ⌘ Code
 - in Arena or some other programming language
- ⌘ Verify
 - does code implement model?
- ⌘ Validate
 - does model output match system output?

These are hard issues-- will discuss later

CS 475/575 32

First example

Time between job random
 Execution time random

Objective:

- What is the average time (both waiting and running) a job spends in the CPU?
- What is the percentage of CPU idle time?

Approach:

- Generate 1000 arrival times
- Generate 1000 service times

CS 475/575 33

Look at pgm ssq1 from class web site

- ⌘ Written in C
 - ☑ Easy to read
- ⌘ Reads a file of arrival and service times

CS 475/575 34

Questions:

- ⌘ Why 1000 jobs?
- ⌘ Why time in seconds (and to thousands)?
- ⌘ What if we were to simulate Bob's Bank with a single queue and 1 teller?
- ⌘ What if we were to simulate Sue's Barber Shop? or Sue's Bakery? or Sue's Emergency Room? or Sue's CPU ...
- ⌘ And what about queueing theory?

CS 475/575 35

Classical repairman model

- ⌘ Used in lots of studies
- ⌘ Many automated tools (e.g. looms) that fail (maybe run out of thread)
- ⌘ A small number of repairmen fix broken tools
- ⌘ What's the optimal number of repairmen?
- ⌘ See conceptual model in Leemis text (pg. 6)
 - ☑ Point: in real world repairmen go to machines
 - ☑ In conceptual model, broken machines queue up in front of repairmen

CS 475/575 36

Reading assignment, etc.

- ⌘ Kelton, et al., ch. 1, 2 by Mon.
- ⌘ Install arena (see app. e in text)
- ⌘ Run model 03-01 by Wed.
 - See chap. 3 for instructions
 - Mail me a copy of the run report
 - If questions, ask in class on Mon.

CS 475/575

37

Bouncing ball: continuous simulation example

Model:
 Assume ball is projected at v meters/sec. at angle θ to the horizon.
 Initial velocity in x direction, v_x , is $v \cos \theta$.
 Initial velocity in y direction, v_y , is $v \sin \theta$.
 Both v_x and v_y change over time. (What about z ?)
 $v_x(t+\Delta t) = v_x(t) + a_x \Delta t$, $v_y(t+\Delta t) = v_y(t) + a_y \Delta t$
 Assume no acceleration in x direction but gravity operates in y direction
 ($a_y = -9.75$)
 So $x(t+\Delta t) = x(t) + v_x \Delta t$, and
 $y(t+\Delta t) = y(t) + (v_y(t+\Delta t) + v_y(t)) \Delta t / 2$
 When ball hits ground, velocity is lost, so assume:
 if $y(t+\Delta t) \leq 0$ then $v_y(t+\Delta t) = |0.9 * (v_y(t+\Delta t))|$
Is this model correct?

CS 475/575

38

1st prog assignment

- ⌘ Implement the bouncing ball simulation using the provided model
- ⌘ See class web site for details
- ⌘ Note: this code should be short. If it starts getting long, you're doing something wrong

CS 475/575

39
