

CS 475/575
Slide set 3

M. Overstreet
Old Dominion University
Spring 2005

1

Sim Prog Lang Overview

⌘ Best source for training/overviews: Winter Simulation Conference. Intro tutorials, workshops. All major vendors are there. Proceedings (1500 pp)

⌘ Lots of commercial simulation languages. See (Nance 95, A History of SPLs)

⌘ Leader in their development is US DoD; heavy consumer of simulation

CS 475/575 sp05

2

Just a few examples:

⌘ GPSS - very briefly & only due to its influence. No progs.

⌘ Simscript - history interest, code examples only. Also some derivatives.

⌘ Simula - history interest, fake code examples only

⌘ ModSim III (CACI) Modula-based object-oriented lang

⌘ Arena - typical of a dozen languages; Awesim (from Pritsker), SLX (Henricksen), Promodel (ProcessModel), VSE (Balci)

⌘ CSim19 - typical of extension langs, basically a package for C++. Has adv & disadv.

CS 475/575 sp05

3

SPL objectives

- ⌘ Improve productivity of developers
- ⌘ Improve quality of models (less errors)
- ⌘ Provide vocabulary for modeling
- ⌘ Quicker model construction
- ⌘ (Sometimes) more efficient model execution
- ⌘ (Make money for vendor)

CS 475/575 sp05

4

SPL Requirements (Nance 93)

- ⌘ "Generate random numbers, so as to represent the uncertainty associated with an inherently stochastic model,
- ⌘ Process transformers, to permit uniform random variates obtained through the generation of random numbers to be transformed to a variety of statistical distributions,
- ⌘ List processing capability, so that objects can be created, deleted, and manipulated as sets or as members, added to and removed from sets,
- ⌘ Statistical analysis routines, to provide the descriptive summary of model behavior so as to permit comparison with system behavior for validations purposes ...
- ⌘ Report generation, to finish an effective presentation of potentially large reams of data to assist in decision making ...
- ⌘ Timing executive or a time flow mechanism, to provide an explicit representation of time."

CS 475/575 sp05

5

Simscrip example

```
PREAMBLE
'' set defaults, defines global variables, requests some
'' forms of automatic data collection, model entities, etc.
DEFINE num.arrival, num.services, queue.size, and server.status
AS INTEGER VARIABLES.
EVENT NOTICES INCLUDE
  arrival
  end.service
  end.simulation.
PRIORITY ORDER is arrival, end.service and end.simulation.
TALLY Avg.queue.size as the AVERAGE of queue.size,
  Max.queue.size as the MAX of queue.size.
ACCUMLATE QueueHist( 0 to 10 by 1) AS THE HISTOGRAM of queue.size.
END
```

CS 475/575 sp05

6

Simscript cont.

```
ROUTINE INITIALIZE
'' Create and initialize a bunch of variables (which were
'' declared in the PREAMBLE), like:
LET num.arrivals = 0
LET num.services = 0.
LET queue.size = 0.
LET server.status = free
READ ( mean.arrival.time, mean.service.time, run.time )
CREATE the entities which the model needs
SCHEDULE arrival NOW.
SCHEDULE end.simulation IN run.time MINUTES
RETURN
END
```

CS 475/575 sp05

7

Simscript cont.

```
EVENT arrival SAVING EVENT NOTICE
LET num.arrivals = num.arrivals + 1
IF server.status = free THEN
  server.status = busy
  SCHEDULE end.service IN negexp.f(mean.service.time) minutes
ELSE
  queue.size = queue.size + 1
  SCHEDULE an arrival IN newexp.f( mean.arrival.time ) minutes
RETURN
END

EVENT end.service
LET num.services = num.services + 1
IF queue.size > 0 THEN
  queue.size = queue.size - 1
  SCHEDULE an end.service IN negexp.f( mean.service.time) minutes
ELSE
  server.status = idle.
RETURN
END

EVENT end.of.simulation
'' Write out the report.
RETURN
END
```

CS 475/575 sp05

8

Simscript - more

Many "interesting" features

for loops

for $i = 1$ to n , except when $x(i) < \text{min.value}$

for every job in queue.1, do

for every job in queue.1 do

for each job in queue.1 do

for all fish do

ragged arrays

left & right monitored vars

set ops (really list ops):

FILE	REMOVE FIRST	FOR EACH v FROM w IN set
FILE FIRST	REMOVE LAST	FOR EACH v AFTER w IN set
FILE LAST	REMOVE specific	IN REVERSE
FILE BEFORE	FOR EACH v IN set	FOR EACH v AFTER w IN set
FILE AFTER	FOR EACH v IN set	IN REVERSE [also IN REVERSE]

CS 475/575 sp05

9

GPSS example

```
SIMULATE
*
* ONE-LINE, SINGLE-SERVER QUEUEING MODEL
*
GENERATE      18,6      ARRIVALS EVERY 18 +- 6 MINUTES
QUEUE        JOEQ      GET IN LINE
SEIZE        JOE       GRAB THE BARBER
ADVANCE      15,3      HAIRCUT TAKES 15 +- 3 MINUTES
RELEASE      JOE       FREE THE BARBER
DEPART      JOEQ      EXIT LINE
TERMINATE    0         EXIT THE SHOP
*
* TIMER SEGMENT
*
GENERATE      ,,48000,1  RUN FOR 8 HRS (IN MINUTES)
TERMINATE    1         SHUT DOWN
START        1
END
```

CS 475/575 sp05

10

ModSim example

```
DEFINITION MODULE GroundCombatModule;
FROM SimMod IMPORT ProcessObj;
FROM RanMod IMPORT Obj;
FROM GeometryMod
IMPORT VectorType, VectorEqual, VectorAdd;
CONST
ScanInterval = 10;
AimInterval = 20;
MoveInterval = 60;
BurstCount = 3;
TanksPerBattalion = 40;

BattalionObject = OBJECT( ProcessObj );
location: VectorType;
ASK METHOD Init;
TELL METHOD MyStatusIs( In msg: ARRAY OF CHAR );
END OBJECT;
```

CS 475/575 sp05

11

ModSim cont.

```
TankObject = OBJECT( ProcessObj )
name: ARRAY[ 0 .. 20 ] OF CHAR;
canShoot: Boolean;
canMove: Boolean;
hq: BattalionObject;

ASK METHOD InitTank( IN boss: BattalionObject );
TELL METHOD Move( IN dest: VectorType );
TELL METHOD Acquire;
TELL METHOD Fire( IN target: TankObject );
TELL METHOD FiredUpon;

PRIVATE
ourDice: RandomObj;
speed: VectorType;
ASK METHOD SetCourse( IN dest: VectorType );
ASK METHOD CalculateSector;
TELL METHOD ScanSector: TankObject;
ASK METHOD DebugPrint( IN msg: ARRAY OF CHAR );
END OBJECT;

END MODULE
```

CS 475/575 sp05

12

ModSim cont

```
OBJECT BattalionObject;
ASK METHOD Init;
VAR
  i: integer;
  atank: TankObject;
BEGIN
  location.X := 13.45;
  location.Y := 55.35;
  FOR i := 1 TO TanksPerBattalion
    NEWOBJ( atank );
    ASK atank TO InitTank( SELF );
  END FOR;
END OBJECT;

OBJECT TankObject;
ASK METHOD InitTank( IN boss: BattalionObject );
BEGIN
  hg := boss;
  location := ASK boss location;
  NEWOBJ( ourDice );
  canShoot := TRUE;
  canMove := TRUE;
  Acquire; { Start looking around for targets }
END METHOD;
CS 475/575 sp05
```

13

ModSim cont.

```
ASK METHOD DebugPrint( IN msg: ARRAY OF CHAR );
BEGIN
  OUTPUT( "Tank status: ", msg, "." );
END METHOD;

TELL METHOD Move( IN dest: VectorType );
BEGIN
  { Set speed vector }
  SetCourse( dest );
  { Head in that direction }
  WHILE NOT VectorEqual( location, dest )
    WAIT DURATION MoveInterval;
    ON INTERRUPT
      IF NOT canMove
        TELL hg MyStatusIs( "immobile" );
      END IF;
    TERMINATE;
  END WHILE;
  VectorAdd( location, speed );
  CalculateSector;
END WHILE;
END METHOD;
CS 475/575 sp05
```

14

ModSim cont

```
TELL METHOD Acquire;
VAR
  theTank: TankObject;
BEGIN
  WHILE canShoot
    { continues as long as functional }
    WAIT DURATION ScanInterval;
  END WHILE;
  theTank := ScanSector;
  IF theTank <> NILOBJ
    WAIT FOR SELF Fire( theTank );
  END IF;
  END WHILE;
  TELL hg MyStatusIs( "lost gun" );
END METHOD;

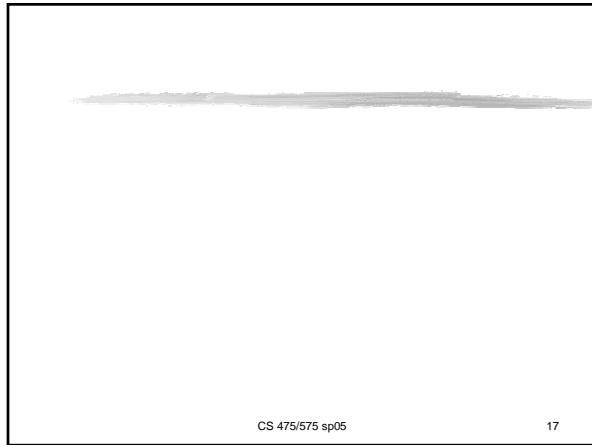
TELL METHOD Fire( IN target: TankObject );
VAR
  shots: INTEGER;
BEGIN
  FOR shots := 1 TO BurstCount
    WAIT DURATION AimInterval;
  END FOR;
  TELL target FiredUpon;
END METHOD;
CS 475/575 sp05
```

15

ModSim cont

```
TELL METHOD FiredUpon:
BEGIN
  { Pick a number from 1 to 6 }
  CASE ASK ourDice UniformInt( 1, 6 );
  WHEN 1: DebugPrint( "destroyed" );
    InterruptAll;
    DISPOSEOBJ( SELF );
  WHEN 2: canShoot := FALSE;
    DebugPrint( "hit and unable to fire" );
    Interrupt( "Fire" );
  WHEN 3: canMove := FALSE;
    DebugPrint( "hit and immobilized" );
    Interrupt( "Move" );
  OTHERWISE
    DebugPrint( "shot at and missed" );
  END CASE;
END METHOD;
END OBJECT;
END MODULE;
```

CS 475/575 sp05 16



CS 475/575 sp05

17

Reading assignment

- ⌘ Read Leemis, Chapter 5, pages 185-194 (though section 5.1.4)
- ⌘ Problems 5.1.3 and 5.1.4 will be due Feb. 16.

CS 475/575 sp05

18

Simula-like example - 1

```
PROCESS customer(i)
  WAIT shopping_time
  IF size( clerk_queue(i) ) < size( clerk_queue( 2 ) )
  THEN put i in clerk_queue( 1 )
  ELSE put i in clerk_queue( 2 )
  WAIT UNTIL service begins for customer( i )
  WAIT service_time
END customer PROCESS

PROCESS clerk(i)
  UNTIL end_simulation DO
    WAIT UNTIL clerk_queue(i) nonempty
    WAIT service_time
    REMOVE first FROM clerk_queue(i)
  END clerk PROCESS
```

CS 475/575 sp05

19

Simula-like example - 2

```
MAIN
  START clerk( 1 )
  START clerk( 2 )
  i = 1
  UNTIL end_simulation DO
    START customer( i )
    i = i + 1
    WAIT interarrival_time
  END UNTIL
END main
```

CS 475/575 sp05

20

Revised Simula - 1

```
PROCESS customer(i)
  WAIT shopping_time
  IF size( clerk_queue(i) ) < size( clerk_queue( 2 ) )
  THEN j = 1
  ELSE j = 2
  place i in clerk_queue(j)
  IF size( clerk_queue(j) ) = 1
  THEN ACTIVATE clerk(j)
  WAIT service_time
END customer PROCESS

PROCESS clerk(i)
  UNTIL end_simulation DO
    IF size( clerk_queue(i) ) = 0
    THEN PASSIVATE
    WAIT service_time
    j = first( clerk_queue(i) )
    REMOVE j FROM clerk_queue(i)
    ACTIVATE customer(j)
  END UNTIL
END clerk PROCESS
```

CS 475/575 sp05

21

Event Lists

Event lists, an abstract data type, are really **Priority Queues** where the priority is event time.

Basic ops:

```
InitQueue( q )
Insert( q, n )
n = DeleteMin( q )
```

Much studied

For many simulations, most CPU time is spent in list ops

What's the best implementation? Not obvious.

Early optimization, since inserts are *almost* monotonic in time, delete from head, insert from tail.

CS 475/575 sp05

22

Event lists - cont

Best implementation?

If linked list, ordered by priority (i.e., event time):

Init is $O(1)$

DeleteMin is $O(1)$

Insert is $O(n)$

If avg list length is n , &

Insert keys for new events are uniformly distributed over range of existing inserts
(Usually uniform assumption incorrect)

CS 475/575 sp05

23

Proposed data structures

Linked list	$O(n)$
Implicit heaps	$O(\log n)$
Leftist trees	$O(\log n)$
Two list	$O(n^{0.5})$
Henriksen	$O(n^{0.5})$
Binomial queues	$O(\log n)$
Pagodas	$O(\log n)$
Skew heaps	$O(\log n)$
Splay trees	$O(\log n)$
Pairing heap	$O(\log n)$

CS 475/575 sp05

24

Assignment!

⌘ Find 5 "recent" papers on event list implementation

- Performance
- Algorithms
- Code

⌘ Due Tuesday, Feb. 16. (Too soon: chg to 2/21!)

- Write a brief summary (a few sentences)
- Be ready to describe in class.
- Mail & bring hard copy to class

CS 475/575 sp05

25

Event lists - cont

What's safe to assume about insert distributions?

Very model dependent

e.g., in network simulations, some events are at picosecond, other at minutes level

How to choose?

Use whatever SPL provides

If expect list to be short, doesn't matter

If providing package for others to use in future

Study the literature

Do empirical study

CS 475/575 sp05

26

Pseudo code

```
{ create initial queue }
initialize( q );
for i = 1 to n do
    n = allocate;
    n^.time = randomInitTime;
    insert( q, n );
end for;
{ perform sequence of hold operations }
beginTime = processTime;
for i = 1 to m do
    n = deletMin( q );
    n^.time = n^.time + randomTimeIncrement;
    insert( q, n );
end for;
now = process_time;
avgHoldTime = ( beginTime - now )/m;
```

CS 475/575 sp05

27

Details

Vary size of n: 5, 10, 50, 100, 500, 1000, 5000, 10000, ...
too many so stop when you can make valid conclusion

Vary distribution of inputs:

Uniform(0.0, 2.0)	$2 * \text{random}$
Exponential(1.0)	$-\ln(\text{random})$
Uniform(0.9, 1.1)	$0.9 + 0.2 * \text{random}$
Bimodal	$0.95238 * \text{random} + \text{if random} > 0.1 \text{ then } 9.5238 \text{ else } 0.0$
Triangular	$1.5 * \text{random} ^ 0.5$

(To check your coding, compute actual mean of each distribution and compare to mean of you really compute)

CS 475/575 sp05

28

More details

Vary data structures:

- Linked list
- Two list
- Henriksen
- Splay trees

CS 475/575 sp05

29

Report contents

⌘ Write like an internal study/technical report.

⌘ Include:

- Purpose of study
- Definition of terms
- Methodology of study
- "Digested" results (graphs, etc.)
- Your personal assessment of results (text)
 - source code and prog output can be included in app.

CS 475/575 sp05

30

Assignment

1. Read "Lists Documentation" on class web site
2. Run "randtest.cpp". Be ready to report results.

CS 475/575 sp05

31

Announcements

- ⌘ Midterm next week
- ⌘ Take-home, due midnight, Feb. 28
- ⌘ Will be available Thurs.

CS 475/575 sp05

32

For your own protection:

- ⌘ Make sure you can run the provided code for the list structures.
 - Write a driver that:
 - Places 20 nodes in a list with random event times
 - Deletes them one at a time, displaying their event time
- ⌘ Due March 3.
- ⌘ Never assume free code works. It usually doesn't.

CS 475/575 sp05

33
