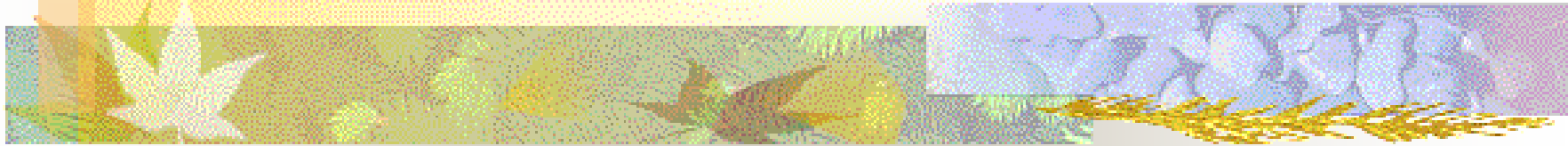


# ECE560: Computer Systems Performance Evaluation

## Lecture#4 Extra -- Simulation with AweSim



Instructor: Dr. Liudong Xing



# References on AweSim

1. A. Pritsker & J. O'Reilly, “Simulation with Visual Slam and AweSim (2<sup>nd</sup> Edition)”, Wiley, John & Sons, 1999
2. Some papers on AweSim available from course website
  - *Introduction to AweSim* by J. Jean O'Reilly and William R. Lilegdon
  - *AweSim: The Integrated Simulation System* by A. Alan B. Pritsker and J. Jean O'Reilly
3. *Website:*
  - A Collection of Modeling and Simulation Resources on the Internet: <http://www.idsia.ch/~andrea/simtools.html>



# AweSim Overview (I)

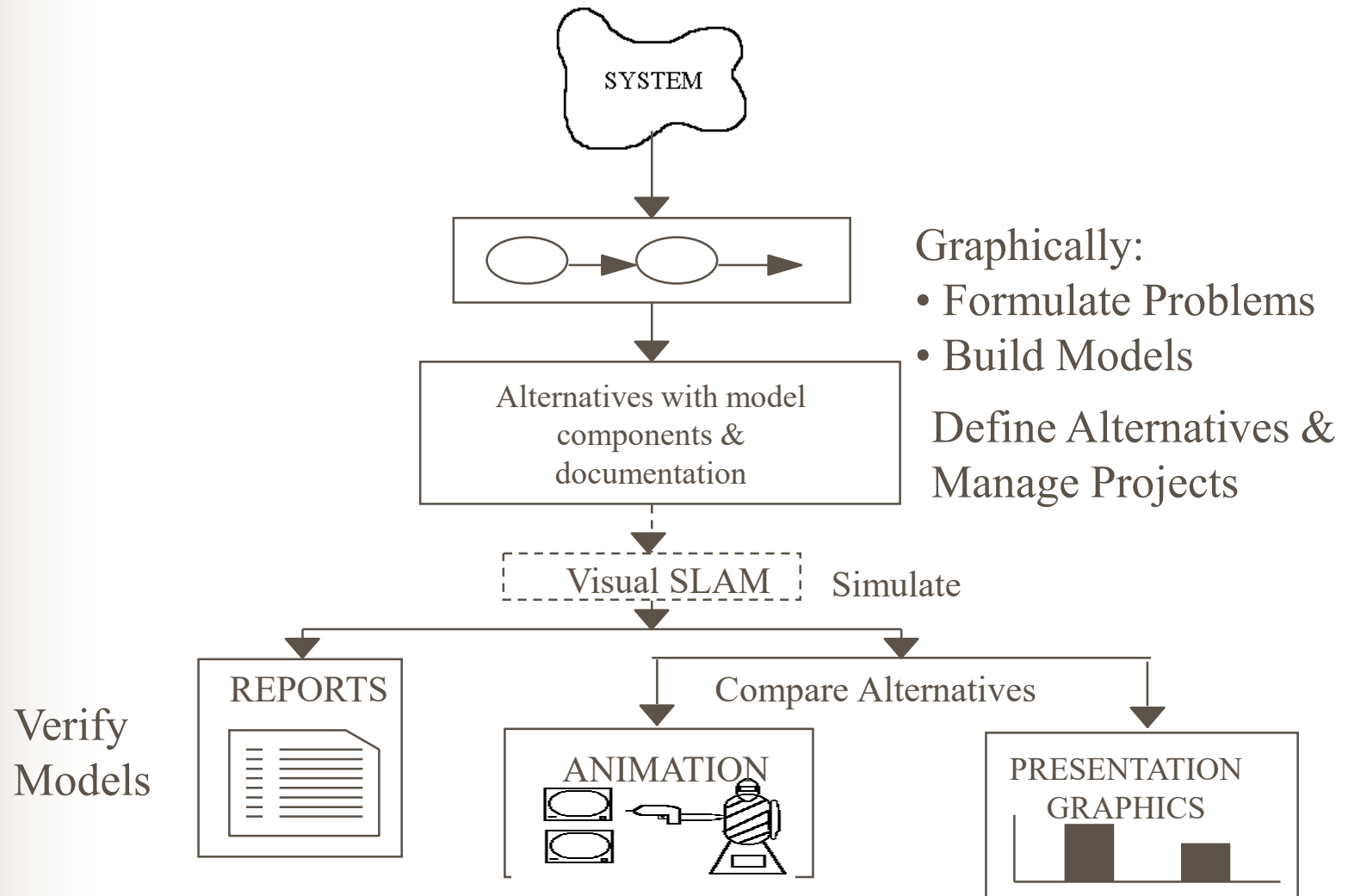
- A general-purpose simulation system, supporting
  - Model building
    - Building models of various system alternatives
    - Discrete models: define changes in system status that occur at event times, using C or Visual Basic
    - Continuous models: write equations defining the value of continuous variables at any point in time and define how often to update them
    - Network models: Define process interaction using predefined “events”
  - Analysis of models using simulation
  - The presentation of simulation results
    - Animation to visualize the dynamics, structure, and control logic of the model
    - Reports and graphs (bar charts, histograms, pie charts, plots) to display quantitative simulation performance measures



## AweSim Overview (II)

- Incorporates the *Visual SLAM* modeling technology
  - Visual simulation language for alternative modeling
  - AweSim provides a simulation problem-solving environment for Visual SLAM
  
- Based on Windows and designed to integrate easily with other Windows applications
  - Easily move input data from Excel worksheet to AweSim input tables
  - Graphical & textual information from AweSim database (output table) can be exported to other Windows package (Excel, Word) for additional analysis and documentation

# AweSim Overview (III)



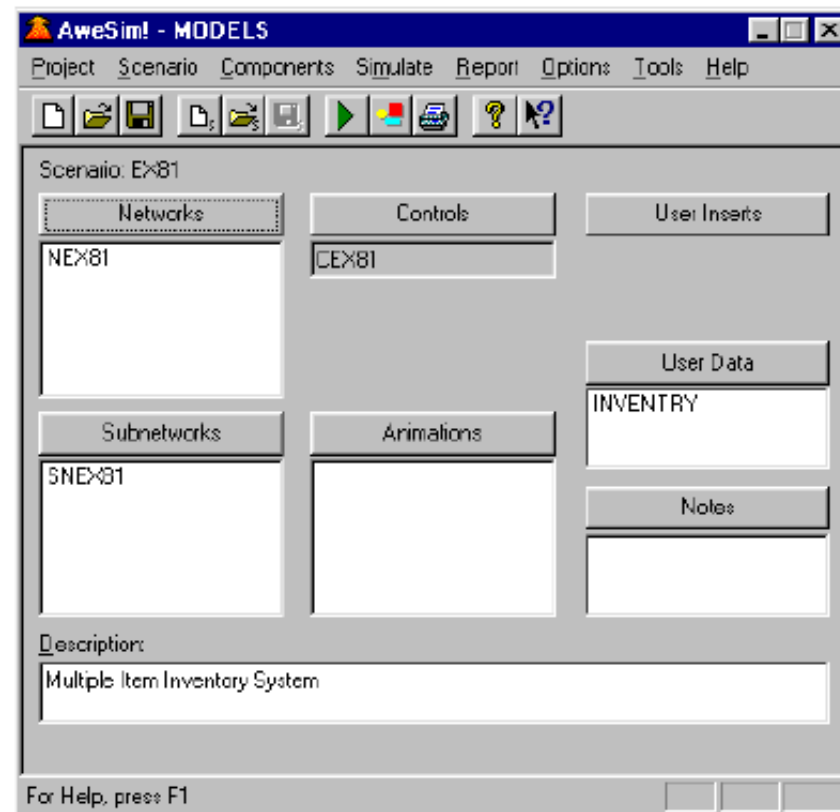


# Project Framework

- An AweSim project → an engineering task/problem
  - A project contains all information describing the problem you are solving
- A project consists of one or more scenarios
  - Each scenario represents a particular system alternative
  - People usually work with more than one scenario on a project
  - Each scenario includes network, subnetwork, control, user data, user inserts and animation components (Table 1.1 / Figure 1)
  - Component builders (software programs) are accessed via the Components menu items or via the component's pushbutton.



Figure 1: The AweSim Executive Window with the Illustration of Scenario Components



The current project under study is identified in the title bar.

The Current Scenario Box shows the model components associated with the scenario presently being analyzed.



# AweSim Basic Network Modeling

## Objectives

- Learn the function and syntax of basic AweSim network nodes

CREATE	AWAIT
TERMINATE	COLCT
GOON	ASSIGN
RESOURCE	FREE
ACTIVITY	

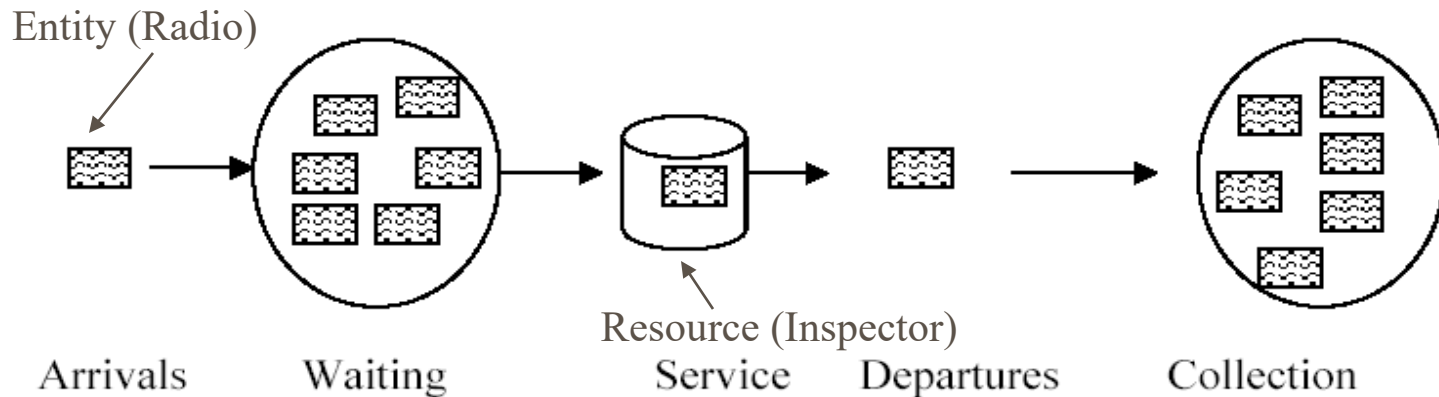
- Learn the function and syntax of basic AweSim control statements

GEN   LIMITS   INIT   NETWORK   FIN

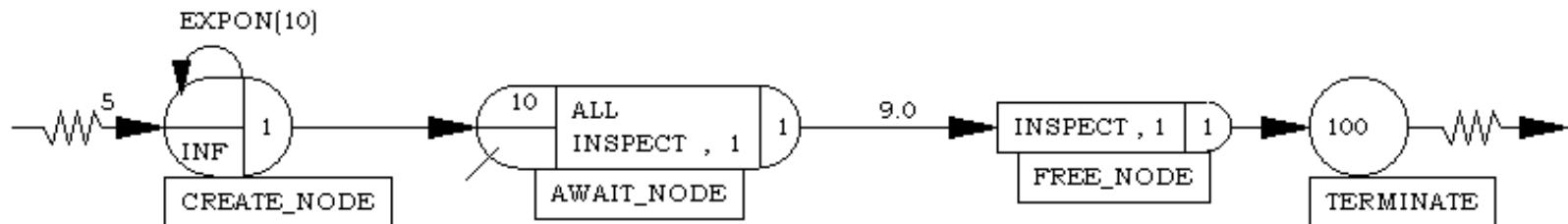
- Develop and run an AweSim model using these constructs within AweSim.



# Single Server Queueing System Example



1	INSPECT	1	10
---	---------	---	----



# CREATE Node

TF: time the first entity enters the system

TBC: time between creations of entities

MV: variable used to maintain mark time

MC: maximum number of entities to create

M: maximum number of branches an entity can be routed from node



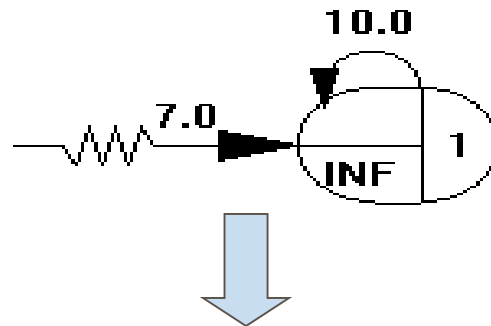
- Creates a new entity within the network at intervals defined by *TBC* (Time Between Creations)
- Can save the arrival time as an entity attribute

# CREATE Node Explanation

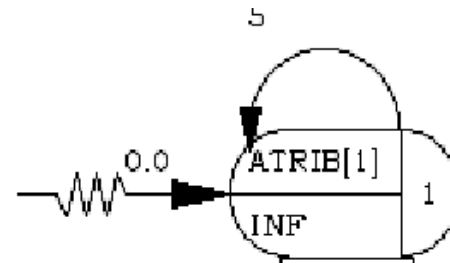
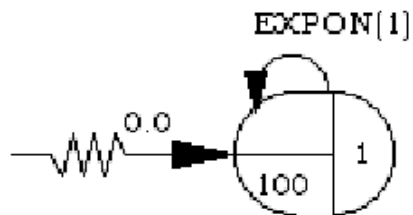
- The node is released initially at time  $TF$  and thereafter according to the specified time between creation  $TBC$ , up to a maximum of  $MC$  releases. At each release a maximum of  $M$  emanating activities are initiated. Time of creation is stored in variable  $MV$  if one is defined



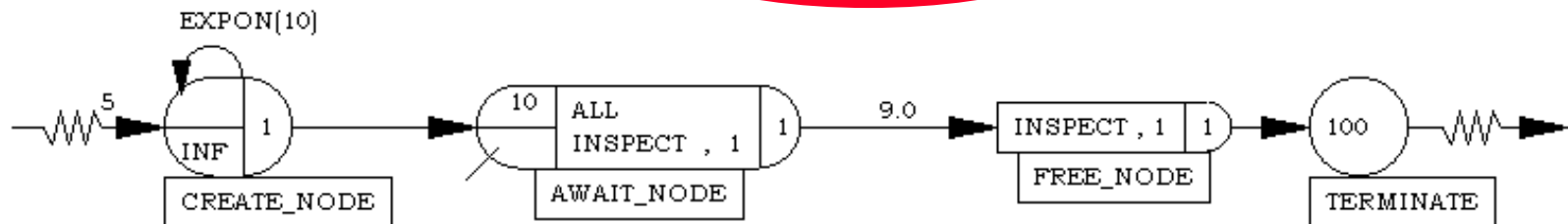
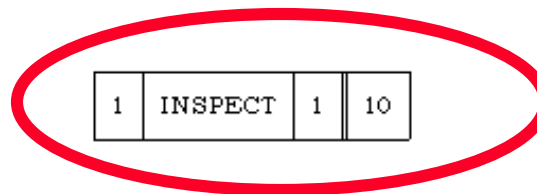
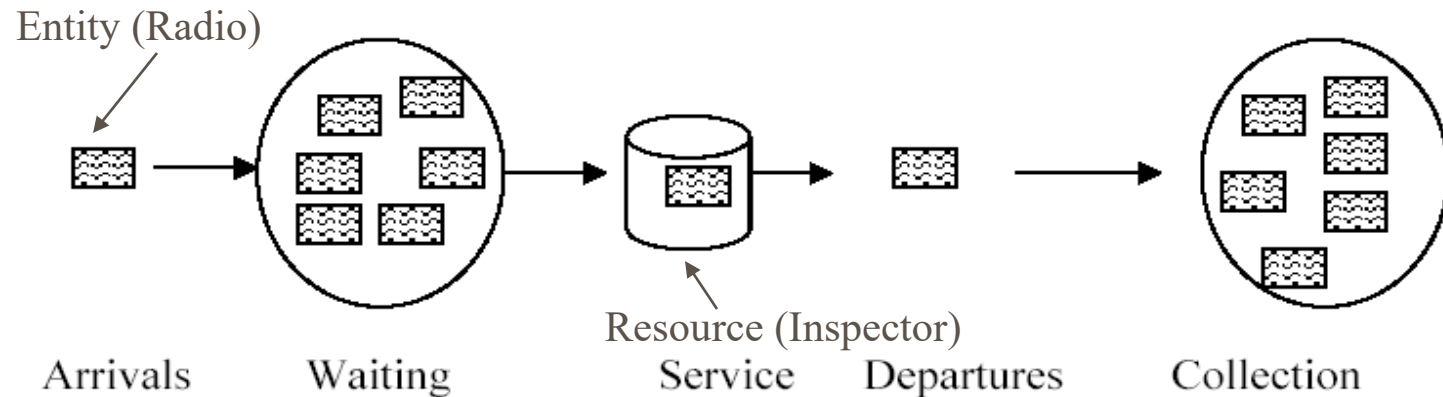
# CREATE Node Examples



Node Label:	<input type="text"/>	Save TNOW:	<input type="text"/>	F[x]	<input type="button" value="OK"/> <input type="button" value="Cancel"/>	
Time Between:	<input type="text" value="10.0"/>	F[x]	Max to Create:	<input type="text" value="INF"/>		F[x]
Time of First:	<input type="text" value="7.0"/>	F[x]	Max Branches to Take:	<input type="text" value="1"/>		



# Single Server Queueing System Example







# RESOURCE Block

RNUM: Resource number (e.g. 3)

RNUM	RLBL	CAP	IFL	Repeats
------	------	-----	-----	---------

RLBL: Resource label

CAP: Number of units of the resource initially available

IFL: File to poll for entities waiting for a resource

Repeats: additional files associated with the resource

- Can be identified by name (*RLBL*) or number (*RNUM*)
- Has no inputs or outputs as no entities flow through it
- Used by *AWAIT*, *PREEMT*, *FREE*, *ALTER* nodes to identify resource types associated with a node

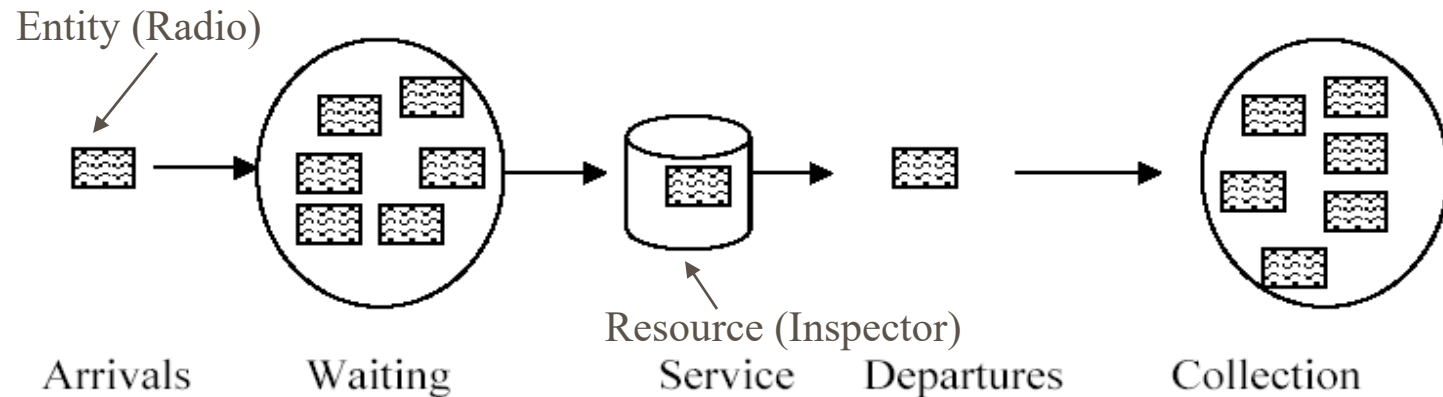


# RESOURCE BLOCK Example

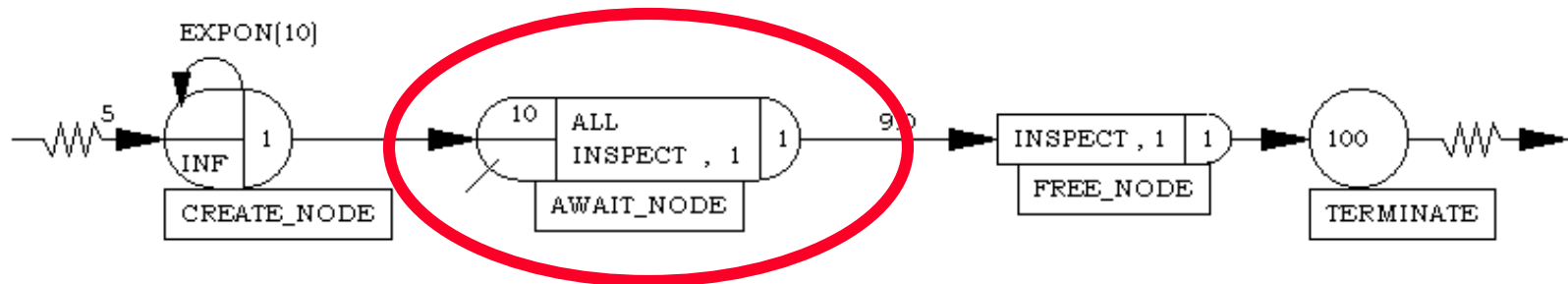
1	INSPECT	1	10
---	---------	---	----

Resource #:	1	Resource Files		OK	Cancel
Label:	INSPECT	File:			
Initial Capacity:	1	10	<u>C</u> hange		
			<u>I</u> nsert		
			<u>D</u> elte		
<div>RESOURCE Block</div> <div>CAP</div>					

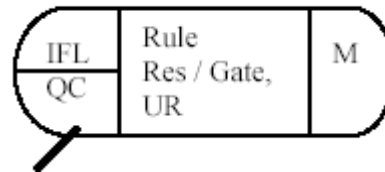
# Single Server Queueing System Example



1	INSPECT	1	10
---	---------	---	----

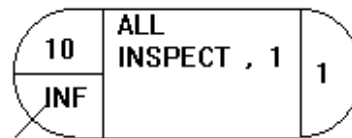


# AWAIT Node



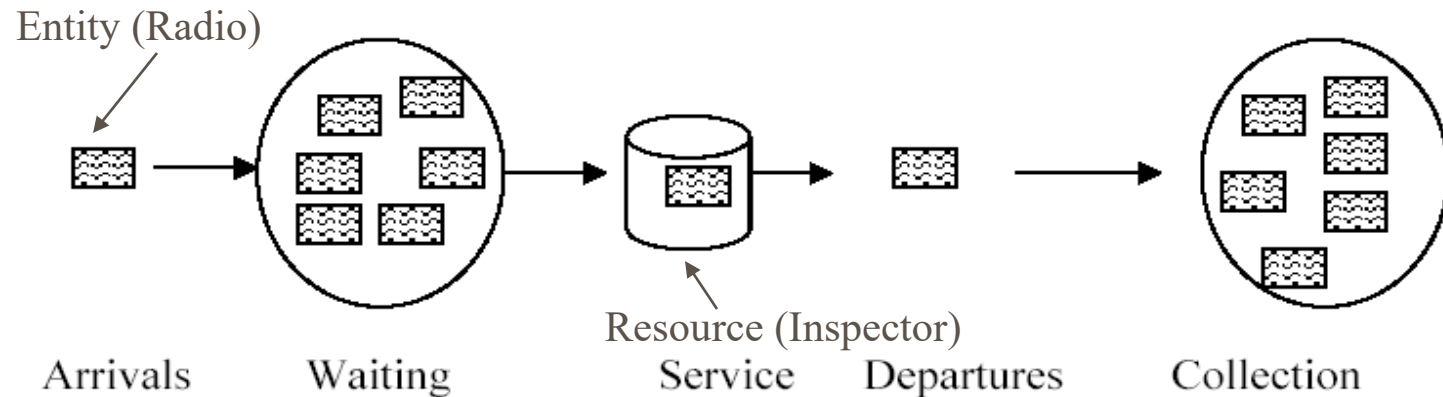
- Used to store entities waiting for *UR* units of resource to be available or gate to open (use resource or gate label names)
- Arriving entities are placed in file *IFL*
- *QC* specifies the queueing capacity of the node
- *Rule* specifies the resource allocation rule
- *M* specifies the maximum branches leaving entities can take

# AWAIT Node Example

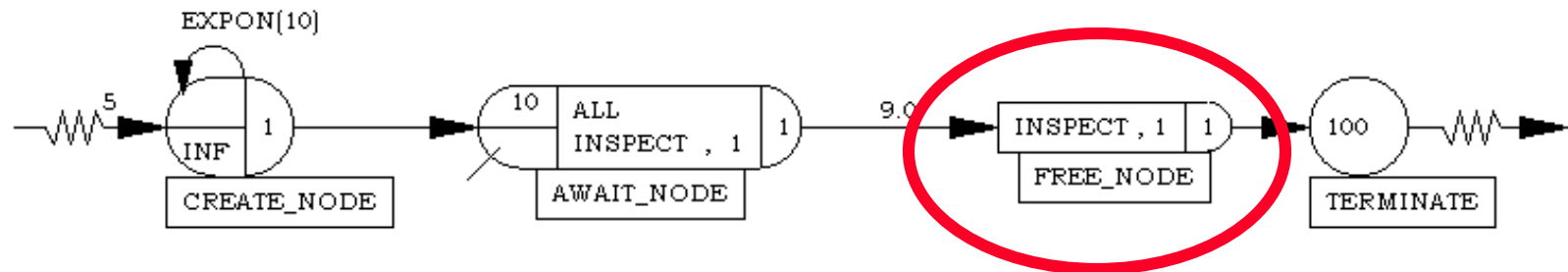


Node Label:	<input type="text"/>	File #:	<input type="text" value="10"/>	File Capacity:	<input type="text"/>
Allocations		<input type="text" value="INSPECT,1"/>		<input type="button" value="Change"/> <input type="button" value="Insert"/> <input type="button" value="Delete"/>	
Resource, Gate, or Group:	<input type="text"/>	Units of Res.:	<input type="text"/> F[x]	Max Branches to Take:	<input type="text" value="1"/>
			<input type="radio"/> None <input type="radio"/> Block <input type="radio"/> Balk: <input type="text"/>		
How to Allocate <input checked="" type="radio"/> All <input type="radio"/> One: <input type="text"/> F[x] <input type="radio"/> Alloc: <input type="text"/> F[x]			Label of this node. Used for branching and reports		
<input type="button" value="OK"/> <input type="button" value="Cancel"/>					
AWAIT Node					

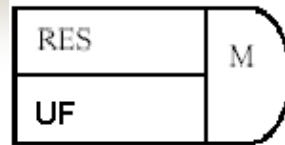
# Single Server Queueing System Example



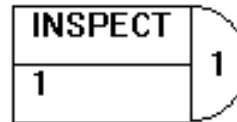
1	INSPECT	1	10
---	---------	---	----



# FREE Node



- Used to release resources previously allocated at an AWAIT node when an entity arrives at the node
- Also checks the list of AWAIT nodes to see if reallocation is possible
- Every entity arriving at a FREE node releases *UF* units of *RES* resource
- A maximum of *M* emanating activities can be initiated from the node
- Example:



Node Label:  Max Branches to Take:

Resources to Free

Resource:	<input type="text" value="INSPECT,1"/>	<input type="button" value="Change"/> <input type="button" value="Insert"/> <input type="button" value="Delete"/>
Units of Res.:	<input type="text" value="1"/> F[x]	
List of resources and amount to free		

FREE Node CAP



# ACTIVITY



- Branches are used to model activities which allow for the specification of time delays and routes for entities flowing in the network
- *DUR* specifies the duration of the activity using either explicit time or a distribution
- *CONDITION/PROBABILITY* specifies under what circumstance / probability a particular branch will be traversed by an entity
- *N* represents the number of parallel identical servers if the activity represents servers
- *A* is the activity number within the model
- Example:



Activity #:	1	End Node Label:		F[x]	OK	Cancel
Duration:	RNORM[18,0.5]	# of Servers:				
Condition:		Identifier:	"INSPECT ACTIVITY"			

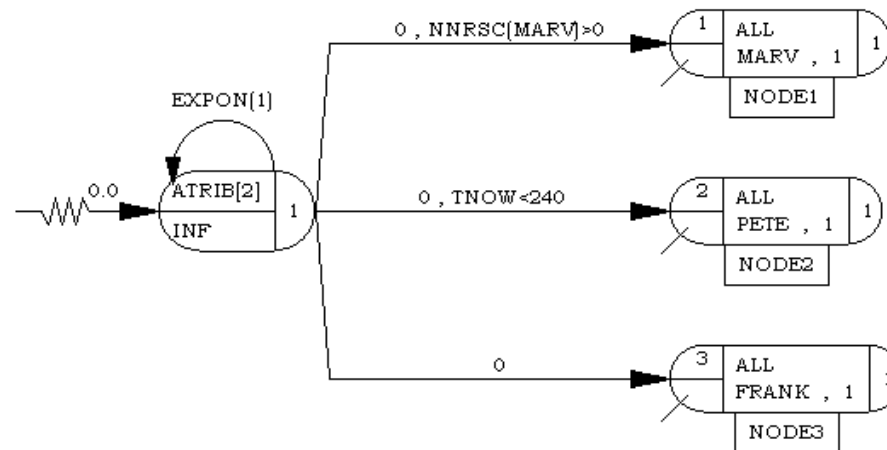
ACTIVITY

AWESIM

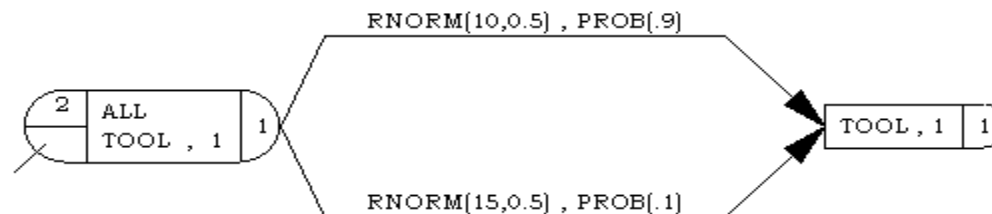
CAP

# More ACTIVITY Examples

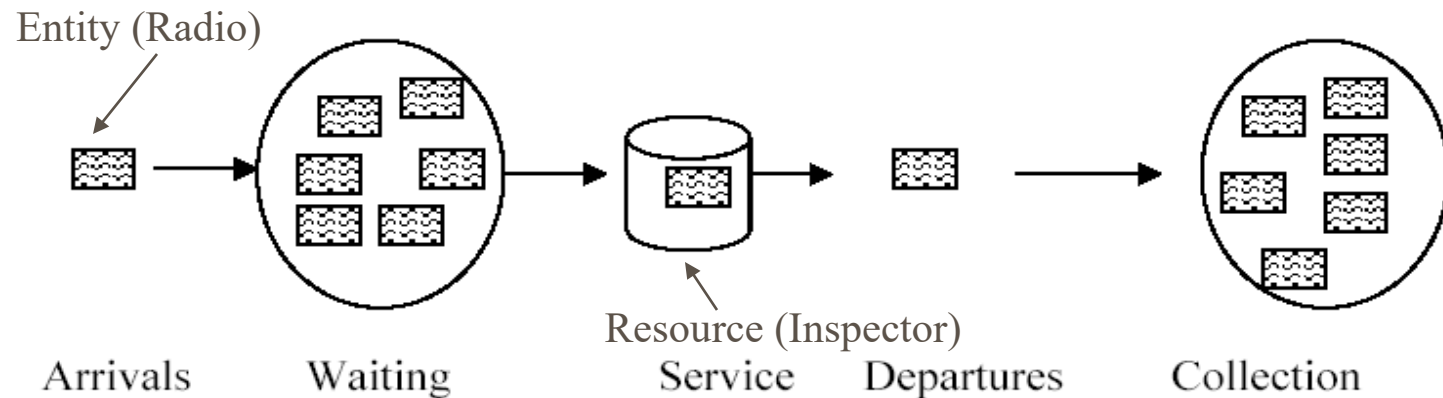
## ■ Conditional branching



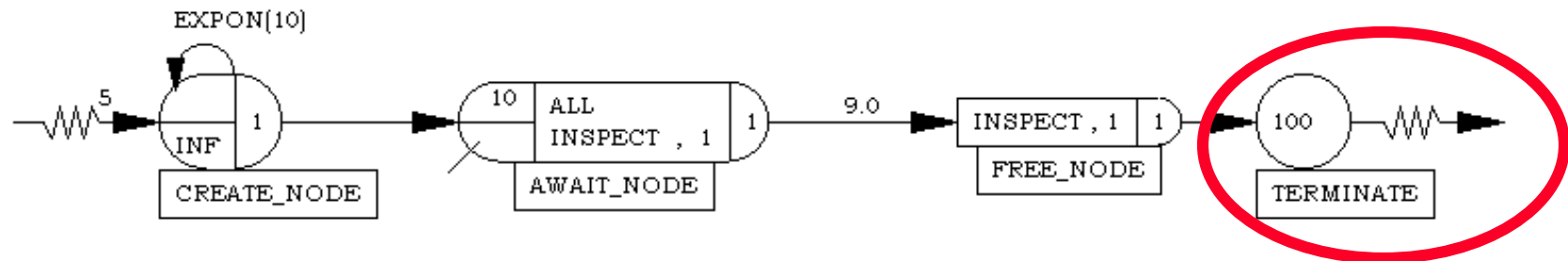
## ■ Probabilistic branching



# Single Server Queueing System Example



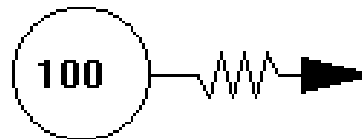
1	INSPECT	1	10
---	---------	---	----



# TERMINATE Node



- Used to destroy or delete entities from the network
- *TC* represents the number of entities to terminate to end the simulation
- Can associate a node label with the node
- Example:



Node Label:

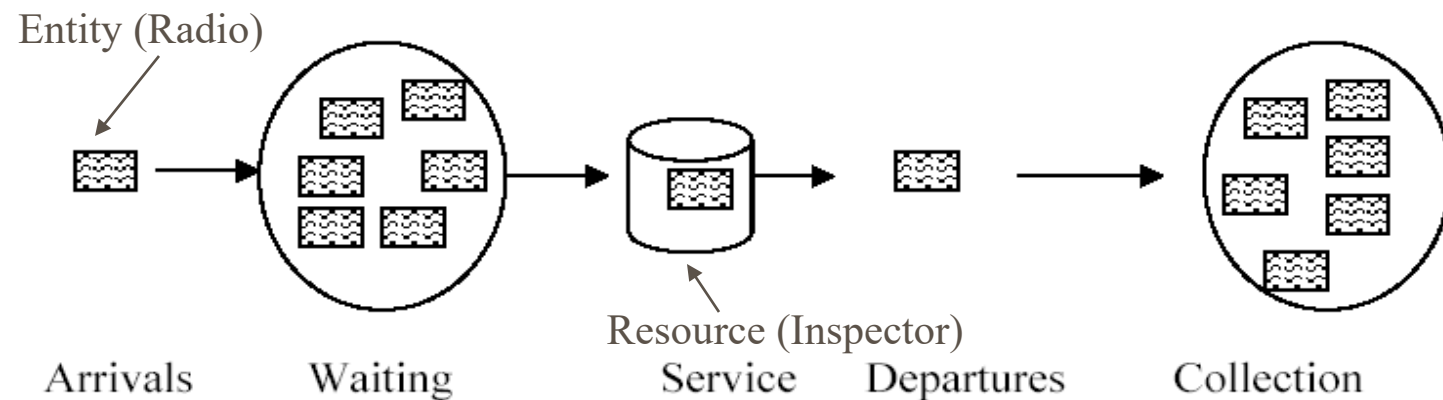
Term. Count:

TERMINATE Node

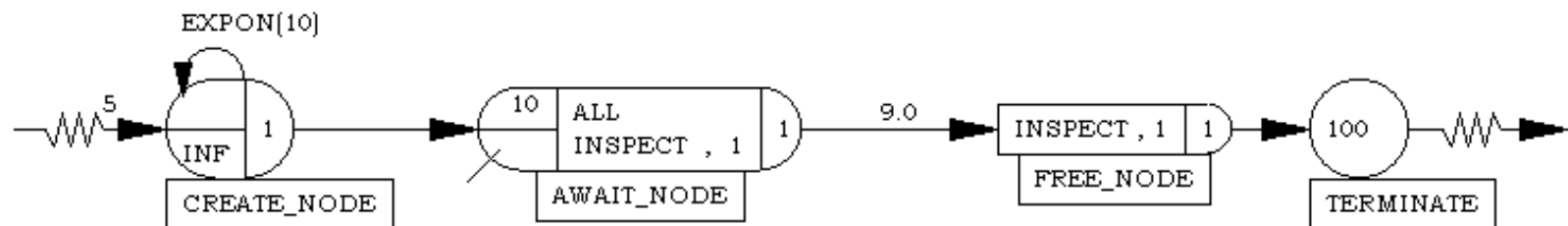
CAP

- Revisit “Radio Inspection Example”

# Revisit “Radio Inspection Example”



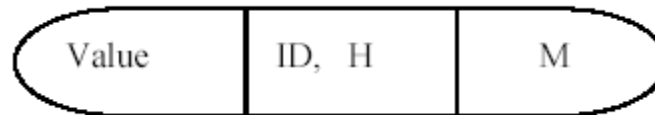
1	INSPECT	1	10
---	---------	---	----





# COLCT Node

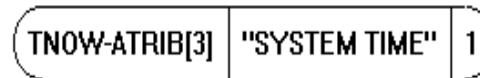
- To collect an observed statistics when an entity arrives at this point in the network



- *Value* represents the expression or variable to collect information on
- *ID* is used as the identifies for the collect node
- *H* is used to indicate that a histogram is to be built
  - Using 3 parameters: number of cells, low cell limit, cell width
- *M* indicates the maximum number of branches to take from the node



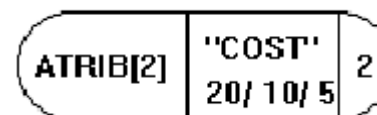
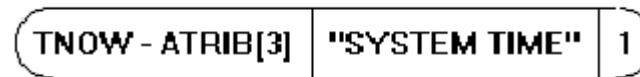
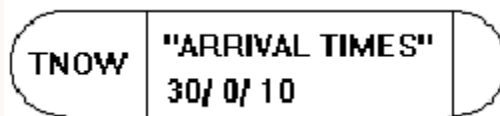
# COLCT Node Examples



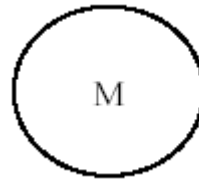
Node Label:	<input type="text"/>	<b>Histogram Information</b> # of Cells: <input type="text"/> Lower Limit: <input type="text"/> Cell Width: <input type="text"/>	OK	Cancel
COLCT #:	<input type="text"/>		Max Branches to Take: <input type="text" value="1"/>	
Value:	TNOW-ATRIB[3] <input type="button" value="F(x)"/>			
Identifier:	"SYSTEM TIME"			

COLCT Node CAP

More example:



# GOON Node



- Provides a continuation node where every entering entity passes directly through the node
- A maximum of  $M$  emanating activities are initiated
- A “placeholder” node used for branching or for separating activities
- Act like a simple router
- Examples:

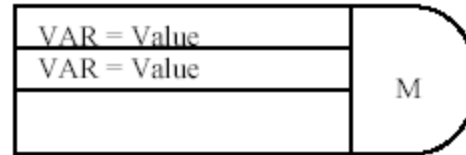


Node Label:

Max Branches to Take:

GOON Node

# ASSIGN Node



- Used as a method to assign values to entity attributes as they pass through the node
- Can also be used to assign values to system variables at each arrival of an entity to the node
- *VAR* defines Visual SLAW global or entity variable
- The type of *Value* (expression) must agree with the variable being assigned
- A maximum of *M* emanating activities are initiated



# ASSIGN Node – Left-Hand Side Variables

## Entity Variables

ATTRIB[I] Real valued attribute of current entity

LTRIB[I] Integer valued attribute of current entity

STRIB[I] String valued attribute of current entity

## Global Variables

XX[I] Real system or global array

LL[I] Integer system or global array

SZ[I] String system or global array

ARRAY[I,J] System doubly-subscripted array

SS[I] State variable I

DD[I] Derivative of SS[I]

STOPA Assignment to stop activities



# Basic System Variables

TNOW	Current time
NNACT(I)	Number of active entities in activity I at current time
NNCNT(I)	The number of entities that have completed activity I
NNQ(I)	Number of entities in file I at current time
NNRSC(RLBL)	Current number of units of resource type RLBL available
NRUSE(RLBL)	Current number of units of resource type RLBL in use
FIRSTARRIVE	Time of the first entity arrival at a COLCT node
LASTARRIVE	Time of the most recent arrival at a COLCT node

# ASSIGN Node Example

ATRI[2] = 7.0	1
ATRI[3] = ATRI[3]/XX[2]	
XX[1] = RNORM(4.0,2.0)	

Node Label:  Max Branches to Take:

Assignments

Variable:  F(x)

Equals:  F(x)

ATRI[2],7.0  
ATRI[3],ATRI[2]/XX[2]  
XX[1],RNORM(4.0,2.0)

ASSIGN Node CAP





# Agenda- AweSim Basic Network Modeling

## Objectives

- Learn the function and syntax of basic AweSim network nodes  
CREATE                      AWAIT  
TERMINATE                  COLCT  
GOON                        ASSIGN  
RESOURCE                    FREE  
ACTIVITY
- Learn the function and syntax of basic AweSim control statements  
GEN   LIMITS    INIT   NETWORK    FIN
- Develop and run an AweSim model using these constructs within AweSim.



# Control Statement

- Each scenario has an associated set of Visual SLAM control statements called a “control”
- The control provides the general information needed to simulate the model
  - How long the model should run
  - Initial conditions
  - Output options
  - File characteristics
  - etc

# GEN Control Statement

- The first statement in a set of Visual SLAM input
- Provides general information identifying the model
  - The number of executions
  - Certain global flags
- Input format
  - GEN, “NAME”, “PROJECT”, DATE, NNRNS, IXQT, IWARN, MXERR;
  - Example
    - GEN, “Pritsker”, “TV INSP. AND ADJUST”, 6/13/96,1,YES,YES;
    - GEN, “Pritsker”, “TV INSP. AND ADJUST”, 6/13/96;

GEN Control

Name: "Pritsker"

Project: "TV INSP. AND ADJUST"

Date: 6/13/96

# of runs: 1

Attempt Execution

☒ Yes ☐ No

Warn of Destroyed Entities

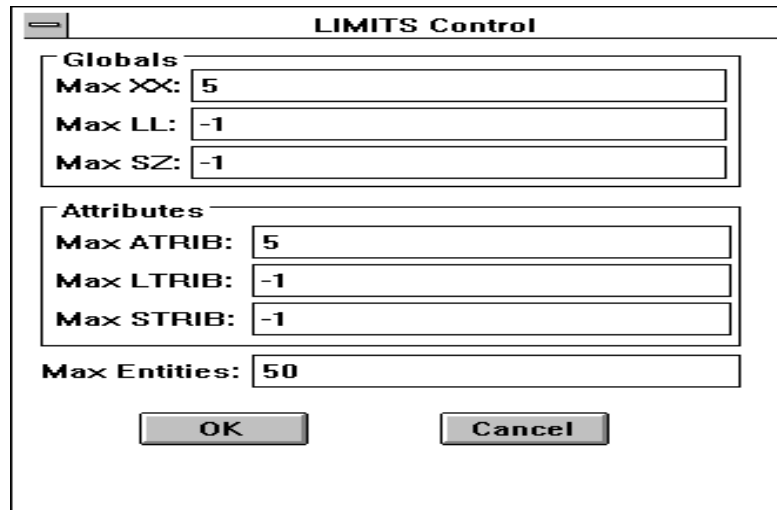
☒ Yes ☐ No

Max Errors:

OK Cancel

# LIMITS Control Statement

- Used to define the size of global variable arrays and the number of attributes defined for each entity
- The LIMITS statement must precede any use of the global arrays
- The limits for each array may be a constant or a global expression
- Input format
  - LIMITS,MXX,MLL,MSZ,MATRIB,MLTRIB,MSTRIB,MNTRY
  - Example: LIMITS,5,,,5,,,50;



The image shows a dialog box titled "LIMITS Control". It contains two main sections: "Globals" and "Attributes".

**Globals**

- Max XX: 5
- Max LL: -1
- Max SZ: -1

**Attributes**

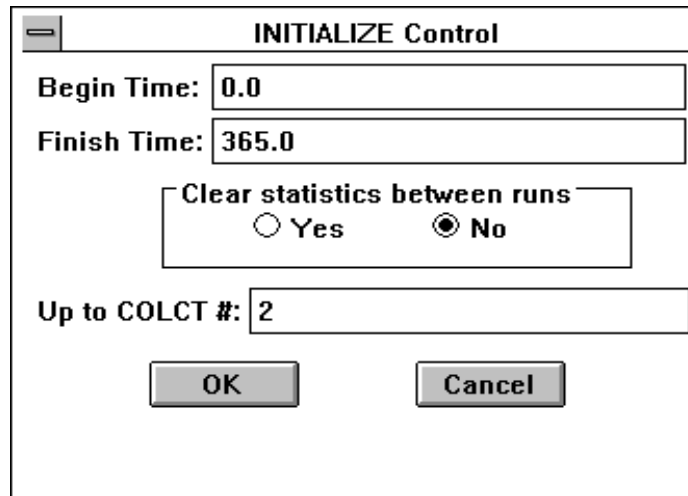
- Max ATRIB: 5
- Max LTRIB: -1
- Max STRIB: -1

Max Entities: 50

At the bottom, there are two buttons: "OK" and "Cancel".

# INITIALIZE Control Statement

- Used to specify the beginning and ending times for a simulation and to reset initialization options for clearing statistics
- Input format
  - INITIALIZE,TTBEG,TTFIN,JJCLR,NCCLR;
  - Example: **INITIALIZE,,365,NO,2;**



The screenshot shows a dialog box titled "INITIALIZE Control". It contains the following fields and controls:

- Begin Time:** A text box containing the value "0.0".
- Finish Time:** A text box containing the value "365.0".
- Clear statistics between runs:** A group box containing two radio buttons: "Yes" (unselected) and "No" (selected).
- Up to COLCT #:** A text box containing the value "2".
- Buttons:** "OK" and "Cancel" buttons at the bottom.





# NETWORK Control Statement

- Used to denote the beginning of a network description
- Input processing may take a significant amount of time for large models, it's convenient to bypass this processing when the network does not change from run to run
- Input formats
  - NETWORK, OPTION, FILE;
- Examples
  - NETWORK;
  - NETWORK,SAVE,"NET1.DAT";
  - NETWORK,LOAD,"NET1.DAT";



# FIN Control Statement

- Denotes the end to all Visual SLAM input statements; all remaining simulation runs will be executed without further data input by Visual SLAM
- Input format
  - FIN;



# Agenda- AweSim Basic Network Modeling

## Objectives

- Learn the function and syntax of basic AweSim network nodes

CREATE	AWAIT
TERMINATE	COLCT
GOON	ASSIGN
RESOURCE	FREE
ACTIVITY	

- Learn the function and syntax of basic AweSim control statements

GEN   LIMITS   INIT   NETWORK   FIN

- Develop and run an AweSim model using these constructs within AweSim – ask for Dr. Xing for software if you are interested in it!