INTRODUCTION TO SIMULA: A TUTORIAL

W.R. Franta
143 Shepherd Laboratory
100 Union Street
University of Minnesota
Minneapolis, Minnesota 55455

This tutorial first presents the essentials of the SIMULA language, next gives the particulars of the simulation support system which has been realized using the language (and has become synonymous with the language) and concludes with an example (sample model) program drawn from the realm of local networks.

SIMULA is a general purpose language based upon a few powerful concepts, particularly the class concept which supports the specification and execution of concurrent co-routines. More specifically co-routines are realized as individual instances or objects generated from templates, known as class declarations, giving the attributes (data objects) and executable behavior of each and every object. Different attributes, behavior pairs can be specified by different class declarations, and a number of objects generated according to a multitude of different class declarations may co-exist and communicate (via shared data objects) in the co-routine environment. This general capability and its attendant machinery constitute a flexible and powerful environment for the description and execution of concurrent programs.

The roots of the SIMULA programming language are embedded in the bedrock of simulation. The name, SIMULA, is, in fact, an abbreviation of simulation language and its designers intended it to fill a need for a very general, high level language that facilitated the description of systems, while also supporting general computation.

From its beginnings its design objectives dictated:

. that it be a dynamic language, as time is an indispensible attribute of system simulation,
. that it be based on a small number of sound, powerful concepts, "selected to provide the research worker [model builder] with a standarized approach to a wide variety of [modelling] problems and to make it easy to identify [in the model] the various components of the system,"
. that the basic concepts embedded in the language should make it easy to develop and understand models expressed in it, so that the language might also serve as a tool for human communication.

The SIMULA response to these objectives was the class concept, its support machinery for communication amongst objects,as well as for the generation and naming of objects, all embedded in an ALGOL oriented syntax and block structuring package.

Using the class concept it is possible to develop class specifications or declarations in a hierarchical fashion, using simple building blocks to realize complex class declarations. Using this hierarchical construction, five class declarations were constructed (by the language designers) to realize a flexible simulation system revolving around a process oriented view of systems. The five defined class declarations are easily altered (extended) to mold it to particular needs, including those dictated by the event world view of simulation or continuous system simulation.

The tutorial draws upon material provided in [1,2] to first:

1) demonstrate the class concept,
2) detail class declarations,
3) specify object generation and object communication
4) and clarify issues related to hierarchical construction and the duration of object existence.

Next the function and use of the five classes, named linkage, head, link, simset, and simulation, is detailed particularly as they are commonly used to support the generation of discrete event simulation programs based on the process world view. These notions and use patterns are made concrete by next describing as simple multi-access local network system.

Specifically we imagine a local network to consist of a collection of (computer related or intelligent) devices which are all connected to a single coaxial cable over which they may communicate via transmitted messages. We call the interconnecting elements bus interface units (BIUs). Since the single coaxial cable channel is shared, an "access" protocol is required whereby a BIU having a message to transmit can gain exclusive control of the channel for transmission of its message. In local networks the connected BIUs are considered equal and each BIU wishing to transmit a message independently decides when it "might" do so. Its decision is generally based on its percieved busy or idle "state" of the channel. A node observing a busy (in use) channel defers transmission, for overlapped use of the channel by two or more BIU transmissions causes the collection of overlapped messages (which are said to collide) to become garbled, necessitating their retransmission. The nodes initiating transmission of the colliding messages use random delays prior to their retransmission attempts to avoid additional collisions. Owing to the end-to-end signal propagation delay associated with the cable, two or more nodes can nearly simultaneously (incorrectly) perceive the channel state to be idle (i.e., not in use) and initiate transmissions, transmissions which will collide and require retransmission.

Additionally we imagine the access protocol obeyed by each BIU with a message to transmit to be:

1. If the channel is observed idle then transmit the message.
2. If the channel is observed busy delay a "random time" and then proceed from step 1.
3. If the transmission was involved in a collision delay a "random time" and then proceed from step 1.

Note that:

1. We do not discuss how collisions are detected in the actual system. In the model we determine collisions by keeping a count of the number of BIUs "simultaneously" using the channel. Whenever the count becomes greater than one, collision has occurred.
2. Expected message delay and "effective" channel utilization are assumed to be the measures of interest, with effective implying contributions only from channel usages that result in successful transmissions.

This system is sufficiently complex to require a rather sophisticated model program yet simple enough to allow a representation in less than 170 lines so that a walk through of the model program is reasonable.

Franta, W.R., The Process View of Simulation, Elsevier North-Holland, 1977.

Franta, W.R., SIMULA: Basic features and Simulation Support, in Digital Simulation Tools Handbook, McGraw-Hill, 1982.