The Simula project

The project was initiated in 1962 by Kristen Nygaard, who saw the need for a language for simulation modelling.

From 1963 he and I cooperated very closely on language development. Implementation issues were my responsibility and issues of economy and organisation were KN's. The work took place at the Norwegian Computing Center (NCC).

General purpose algorithmic capability would be needed. Strategic considerations indicated that our language must be based on a standard one. Algol 60 was chosen.

There were three stages of language development, here identified as:

- Simula 0 (1962-63),
- Simula 1 (1963-65), and
- Simula 67 (1966-67).
Algol 60

- Has block structure.
- Procedures are specialized blocks.
- Is recursive, textually and dynamically.
- Is orthogonal and conceptually economical.
- There is compiler guaranteed and run time efficient access security to declared quantities, including specified parameters.
- It was well known in Europe.
Simula 1, 1963-65

The project was funded by UNIVAC, also providing a 1107 computer for the NCC at half price.

This was a result of negotiations by KN.

The work was to take place at the NCC.
Simula 1, 1963

In order to escape the Algol straitjacket, a new run time system had to be developed, which catered to non-LIFO storage management.

Then an extension for our purposes of Algol itself became possible.
Simula 1, 1963-64

The concepts of “customers” and “stations” were unified as “processes”, able to operate in quasi-parallel over system time.

Processes were referenced by pointers (indirectly through elements of circular lists).

They would admit access from outside to their “attributes”, i.e. quantities declared in the outermost block.
Simula 1, 1963-64

For programming security the use of pointers had to be catered to by reference counts, later supplemented by a garbage collector.

Secure attribute access through unqualified pointers necessitated compiler conscious run time testing:

```
inspect <process reference> when <process type> do
begin <attributes accessible> end
```
Simula 1, 1964

The Algol language was extended by the following ad hoc mechanisms:

- Procedure-like activity declarations, giving rise to quasi-parallel processes;
- (Only parameters called by value were allowed), *time*, a function procedure returning the system time;
- Mechanisms for the explicit scheduling of processes in system time;
- Attribute accessing by inspect, and
- A built-in concept of circular lists containing process pointers.
Simula 1, 1964

The new mechanisms were only available within a block prefixed by the keyword **SIMULA**. Thereby backward compatibility to Algol 60 was established.

Programming security was improved by pre-initializing all variables. This was mandatory for pointer variables.
Simula 1, 1965

The following skeleton example is found in the language documentation:

```simula
SIMULA begin;
activity car;          .. traveling on a linear road
begin real X0; T0; V ; .. position at given time, velocity
   real procedure X;   .. the position now
       X := X0 + V (time - T0);
   procedure UpdateV (V new); real V new; .. change velocity
       begin X0 := X; T0 := time; V := V new end;
<car behaviour> end of car;
activity police;
begin ...; inspect <process reference> when car do
   if X <within city> and V >50 then UpdateV (50); ...
end of police;
... ... end of simulation model;
```
Simula 1, 1965

The *Car* example shows that the idea of “objects” in the modern sense was established already in Simula 1: containing variables (*X0; T0; V*), as well as associated procedures (*X; newV*).

A comment in the language manual notes that the variable attributes unfortunately could not be hidden from outside view in a subblock, because the procedures too would lose access.
Simula 1, 1965-1966

Our experience with customer applications showed that Simula 1 was indeed a useful tool for simulation modelling. But there was also some frustration:

- A simpler `class` concept of objects not dependent on simulation oriented mechanisms (but able to operate like co-routines) could be important in a general purpose setting.
- The same would hold true for list processing facilities based on simple object pointers.
- The `inspect` mechanism was clumsy at times.
- C.A.R. Hoare had proposed (in an Algol Bulletin) a scheme for “record handling” allowing direct access to record attributes, based on pointers qualified by record class and possibly by record subclasses.
- The Simula 1 compiler was based on an Algol compiler for the UNIVAC 1107, which was far from optimal for our kind of extensions.
Simula 67, 1966

A new project for the development of an improved version of Simula was established, funded by the NCC.

One of the conditions was, that the project must prove profitable within the typical life time of a programming language, estimated at 3 to 5 years.

A new Algol/Simula compiler was planned in cooperation with The Technical University of Trondheim.
Simula 67, 1966

A main problem for us was how to adapt the class/subclass concepts of Hoare for our purposes. The breakthrough came in January '67, just in time for writing our paper for the forthcoming IFIP Conference on Simulation Languages at Lysebu, Oslo, May '67 ("Class and Subclass Declarations", in Ed.: J. Buxton: Simulation Languages, North Holland Publ, 1968).

A mechanism of class “prefixing” was defined, today referred to as “inheritance”. The prefix of a class was itself a class, which could be separately compiled and reused.
Simula 67, 1967

The reuse of separately compiled classes invites to use the language for bottom up program design.

A last addition to our paper was the idea of “virtual procedure attributes”, whose bodies could be (re-) defined in subclasses, and thereby, to some extent, support top down design.

At the same time virtual procedures could be seen as substitutes for procedure parameters to classes.
Orthogonality dictated that class declarations should be legal in any kind of block head. So-called application languages could now be defined on top of Simula, in the form of a class intended as a prefix to an ordinary block. Such a class would typically contain locally declared classes.

An example is the predefined class \textit{SIMULATION}, containing a class \textit{process} and other mechanisms corresponding to those of Simula 1. In particular the declarator activity would now be replaced by \textit{process} class.

One restriction to orthogonality turned out to be necessary for reasons of principle and practice: Prefixing across block levels had to be forbidden. As a reasonable consequence \textit{process class} declarations were confined to the outermost block level of a \textit{SIMULATION} block.
The simulation application language (fragment)

```plaintext
class SIMULATION;
begin class process; .. process-objects operate in simulated time
  begin real evtime; ref(process) nextev : - none;
      ....... end of process;
  ref(process) current; .. pointer to the first in time list
  real procedure time; time := current.evtime;
  procedure hold(DeltaT); real DeltaT; .. suspend for the time DeltaT
    inspect current do
    begin evtime := evtime+DeltaT;
      if evtime nextev.evtime then
      begin ref(process)X :-current; current :-X.nextev;
        <insert X at new position in the time list>;
        resume(current) end
      end of hold;
    ....... end of SIMULATION
```
Language Distribution

Simula 67 compilers were completed during the years 1969 to 1971.

Simula 67 was well received for simulation purposes, but its distribution was hampered by the pricing policy of the government agency behind the NCC.

Lectures at NATO Summer Schools, as well as a chapter in a book*, contributed to promote it as a general purpose programming tool.

Cultural Impact 4

OO was a practical tool in the development of certain graphical applications, in which some objects would be pictorially represented for every user to see.

These pioneering efforts lead to several important operating systems, such as Macintosh Operating System, and then to Windows.