Discrete simulation–terminology

Simulation uses a standard terminology that is not necessarily used in other areas in the same way.

State: As simulation progresses, the values of the simulator's internal variables change. A snapshot of these values (of all the variables) is called the **state of the (simulated) system.**

Event: any action that causes an instantaneous change of the state of the system is an **event**. Actions in progress such as "being in the process of washing a car" are not events and must not be taken into consideration in the model of a system.

Clock: is a system variable that indicates the current time in the simulated system. It is commonly called Now and it changes every time a new event occurs (and only then). Note that it has nothing to do with real time ("wall clock time").

More vocabulary

A model is made of two main classes of objects: **customers** and **servers**.

Fancier simulators also have **processes** and **observers**, but they are optional with the exception of the **input process** which is responsible for bringing customers into the system.

Another component of a simulator is its output handling module which can be implemented in many ways (e.g. using asynchronous observers, using function calls or using a separate output process).

Customers

A customer is the object that makes simulation move (like fuel to a motor). When there are no customers in the system, the state of the system remains unchanged.

Let us look at the life of customer 997:

1. Customer 997 is created (**is born**) when it **arrives** into the system (this is an event).
2. Subsequently, 997 is assigned a state, which varies in time as a result of events caused by other objects in the system (e.g. servers). Typical states are: WAITING, (being) SERVED, EXITING, GONE.
3. When 997 is done with, it goes through the EXITING state (usually not explicit) to the GONE state, which usually is not explicit either (the customer is simply deleted).

Servers

Servers are the engines that drive the simulator. Together with the input process, they are the event producers.

Servers may be have queues which store the customers waiting to be served. At any moment in time, every server is in some specific state, such as IDLE, BUSY, BROKEN, etc.

A server changes its state as a result of an **event**. These events may be:

DONE: when the server finishes serving its current customer.

MELTDOWN: when the server malfunctions.

ARRIVAL: when a new customer arrives and the server is in the IDLE state, the server starts working and becomes BUSY.

END: end of simulation. This event is very tricky to handle; this topic will be covered later.

Event–driven simulator

Several sample programs, such as this, are shown in the lab.

```
EventQueue EQ ;
EQ.Insert( The first event ) ;
EQ.Insert( END ) ;
while( (E = EQ.Deletemin()) != END ) {
    Now = E→Time ;
    switch( E→Type ) {
        ...
        ...
    }
}
```