Simulation & Modeling

Process Oriented Simulation

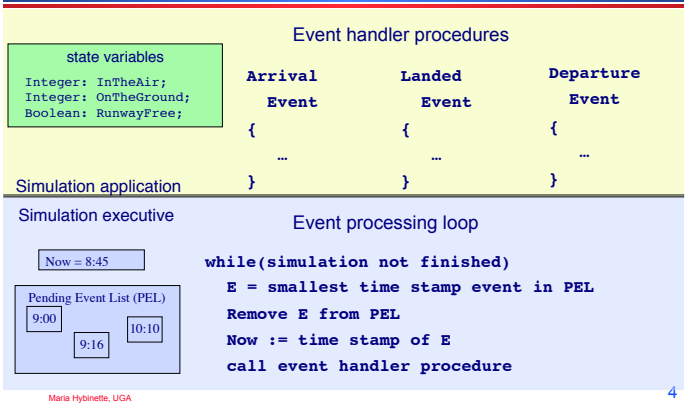
Review from Last Time

- Motivations to do simulations
- Modeling characteristics
- Time and event driven simulations

Today

- Event-Oriented Simulation (review)
- Process-oriented simulation:
  - » Fundamental concepts: Processes, resources
  - » Simulation primitives
  - » Example
  - » Implementation

Event-Oriented World View



Example: Event-Oriented Air traffic Simulation

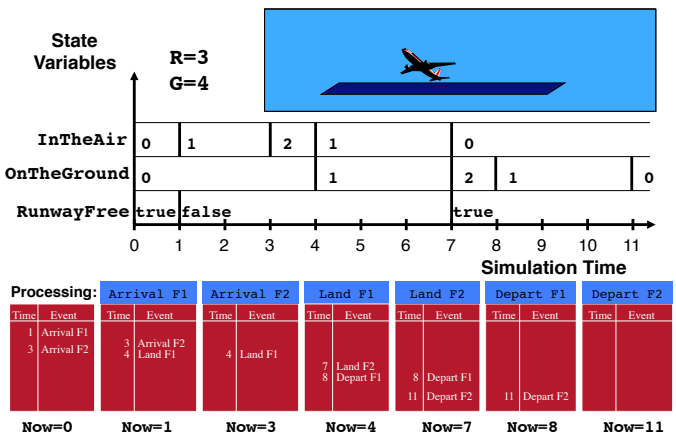
Now: current simulation time
InTheAir: number of aircraft landing or waiting to land
OnTheGround: number of landed aircraft
RunwayFree: Boolean, true if runway available

```
Arrival Event:
    InTheAir := InTheAir+1;
    if( RunwayFree )
      RunwayFree:=FALSE;
      Schedule Landed event @ Now + R;
```

```
Landed Event:
    InTheAir := InTheAir-1;
    OnTheGround := OnTheGround + 1;
    Schedule Departure event @ Now + G;
    if( InTheAir > 0 ) Schedule Landed event @ Now + R;
    else RunwayFree := True;
```

```
Departure Event:
    OnTheGround := OnTheGround - 1;
```


Execution Example



Event-Oriented World View

Event handler procedures

state variables
```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

```
Arrival Event {
   …
}

Landed Event {
   …
}

Departure Event {
   …
}
```

Simulation application

- Event-oriented simulation programs may be difficult to understand and modify:
  - » Program organized around state transitions
  - » Behavior of an aircraft distributed across multiple event handlers
  - » Flow of control among event handlers not obvious
  - » Suppose you want to model: Different aircrafts, airlines, pilots – imagine events for each segment (volume) of airspace

Process Oriented

- A simulation process models a specific entity with a well defined behavior.
  - » It describes the action performed of the process through out its lifetime.
    - – Models a specific entity with well defined behavior and it is encapsulated within the process.
    - – Example: an aircraft
- Event oriented view: lifetime of an event is a SINGLE instant in time.
- Process oriented view: lifetime is a time period of the 'process' or 'thread'

Event versus Process Oriented Views

Event Oriented View

state variables
```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

```
Arrival Event {
   …
}

Landed Event {
   …
}

Departure Event {
   …
}
```

Focus of model is on EVENTS and how they affect the state of the simulation.

Process Oriented View

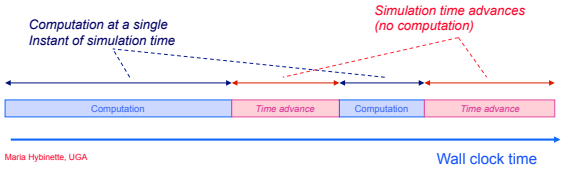
state variables
```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

```
Aircraft1 {
Arrive
Land
Depart
}

Aircraft2 {
Arrive
Land
Depart
}

AircraftN {
Arrive
Land
Depart
}
```

Entities modeled by processes.

Process Oriented Execution Model

- Focus simulation program around behavior of entities
  - » Aircraft: arrives, waits for runway, lands, departs
- Process-oriented simulation
  - » Process: *Thread of execution* describing entity behavior over time
  - » Resources: Shared resource used by entities (e.g., the runway)
- Execution: alternate between
  - » simulation computations at a single instant of simulation time, and
  - » advances in simulation time (no computation)



Simulation Primitives

Primitives needed to advance simulation time

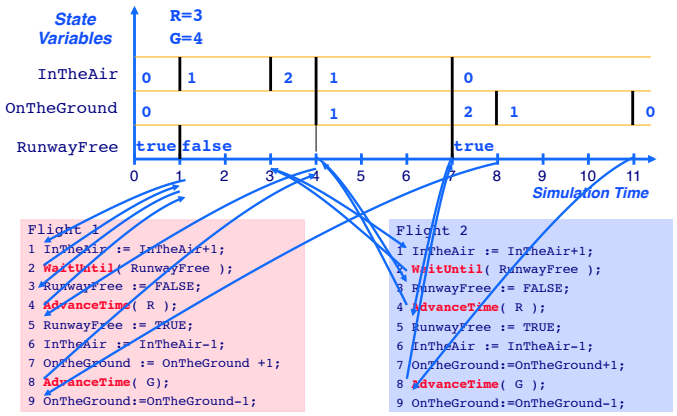
- `AdvanceTime(T)` : advance `T` units of simulation time
  - » Also called "hold"
  - » Example: `AdvanceTime(R)` to model using runway `R` units of simulation time
- `WaitUntil(p)` : simulation time advances until predicate `p` becomes true
  - » Predicate based on simulation variables that can be modified by other simulation processes
  - » Example: `WaitUntil(RunwayFree)` to wait until runway becomes available for landing
- Other combinations
  - » `WaitUntil(p,T)` : Wait up to `T` units of simulation time for predicate `p` to become true
  - » Not used in the air traffic example

Process Model Example: Aircraft

A new aircraft process is created with each Arrival event

```
/* simulate aircraft arrival, circling, and landing */
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;

1  InTheAir := InTheAir + 1;
2  WaitUntil( RunwayFree );        /* circle */
3  RunwayFree := FALSE;            /* land */
4  AdvanceTime( R );
5  RunwayFree := TRUE;
   /* simulate aircraft on the ground */
6  InTheAir := InTheAir - 1;
7  OnTheGround := OnTheGround + 1;
8  AdvanceTime( G );
   /* simulate aircraft departure */
9  OnTheGround := OnTheGround - 1;
```

Execution Example




```
Flight 1
1 InTheAir := InTheAir+1;
2 WaitUntil( RunwayFree );
3 RunwayFree := FALSE;
4 AdvanceTime( R );
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround := OnTheGround +1;
8 AdvanceTime( G);
9 OnTheGround:=OnTheGround-1;
```

```
Flight 2
1 InTheAir := InTheAir+1;
2 WaitUntil( RunwayFree );
3 RunwayFree := FALSE;
4 AdvanceTime( R );
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround:=OnTheGround+1;
8 AdvanceTime( G );
9 OnTheGround:=OnTheGround-1;
```

Implementation

Process-oriented simulations are built over event oriented simulation mechanisms (event list, event processing loop)

- Lifetime of a simulation process consists of a sequence of event computations.
- Event computation: computation occurring at an instant in simulation time
  - » Execution of code section ending with calling a primitive to advance simulation time
- Computation threads
  - » Typically implemented with co-routine (threading) mechanism
- Simulation primitives to advance time
  - » Schedule events
  - » Event handlers resume execution of processes

Aircraft Process

Identify computation associated with each simulation event

```
/* simulate aircraft arrival, circling, and landing */
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;

1  InTheAir := InTheAir + 1;
2  WaitUntil( RunwayFree );        /* circle */
3  RunwayFree := FALSE;            /* land */
4  AdvanceTime( R );
5  RunwayFree := TRUE;
   /* simulate aircraft on the ground */
6  InTheAir := InTheAir - 1;
7  OnTheGround := OnTheGround + 1;
8  AdvanceTime( G );
   /* simulate aircraft departure */
9  OnTheGround := OnTheGround - 1;
```

Aircraft Arrival

Aircraft Landing

Aircraft On The Ground

Aircraft Departs

Implementation: *AdvanceTime(T)*

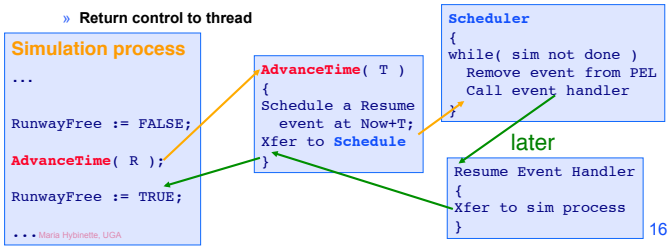
Causes simulation time in the process to advance by T units

Execute `AdvanceTime(T)`:

- » Schedule Resume event at time `Now+T`
- » Suspend execution of thread
- » Return execution to event scheduler program

Process Resume event:

- » Return control to thread



Implementation: *WaitUntil(p)*

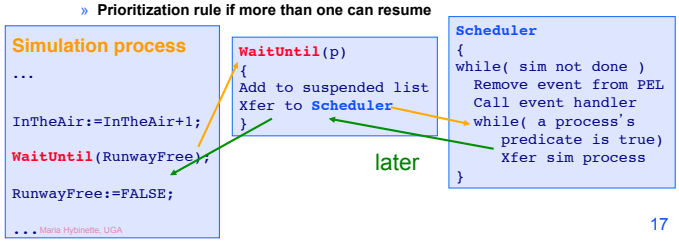
Suspend until predicate p evaluates to true

Execute `WaitUntil( p )`:

- » Suspend execution of thread, record waiting for p to become true
- » Return execution to event scheduler program

Main scheduler loop

- » For each suspended process, check if execution can resume
- » Prioritization rule if more than one can resume



Additional Notes

- Theoretically, both views are equivalent:
  - » Process-oriented simulations can be transformed to event-oriented simulations and vice versa
- Practically, runtime performance differs:
  - » Event-oriented views typically execute faster than process-oriented views

Summary

- Process-oriented simulation typically simplifies model development and modification
- Requires threading (e.g., co-routine) mechanism
- Additional complexity and computation overhead to suspend and resume simulation processes