Today

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

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;
if( InTheAir > 0 ) Schedule Landed event @ Now + R;
schedule RunwayFree := TRUE;
```

**Departure Event:**

```
OnTheGround := OnTheGround - 1;
```

Event-Oriented World View

- State variables:
  - Integer: InTheAir
  - Integer: OnTheGround
  - Boolean: RunwayFree

**Simulation executive**

```
Event processing loop
while(simulation not finished)
  E = smallest time stamp event in PEL
  E = smallest time stamp event in PEL
  Now := time stamp of E
  call event handler procedure
```

- Event handler procedures

```
Arrival Event
{
...
}
```

```
Landed Event
{
...
}
```

```
Departure Event
{
...
}
```

Execution Example

<table>
<thead>
<tr>
<th>State</th>
<th>Variables</th>
<th>R=3</th>
<th>G=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>InTheAir</td>
<td>0 1 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnTheGround</td>
<td>0 1 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunwayFree</td>
<td>true false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrival F1</td>
</tr>
<tr>
<td>3</td>
<td>Arrival F3</td>
</tr>
<tr>
<td>4</td>
<td>Landed F1</td>
</tr>
<tr>
<td>7</td>
<td>Depart F1</td>
</tr>
<tr>
<td>11</td>
<td>Depart F2</td>
</tr>
</tbody>
</table>

Simulation Time

- Execution Example
- Simulation execution
Event-Oriented World View

Event handler procedures

Simulation application

Event-Oriented World View

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 aircraft, airlines, pilots – imagine events for each segment (volume) of airspace

Event vs Process Oriented Views

Process Oriented View

Primitives needed to advance simulation time
- **AdvanceTime(T)**: advance T units of simulation time
  - Also called "hold"
  - Example: **AdvanceTime(R) to model 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

Event-Oriented Execution Model

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’

Process Model Example: Aircraft

A new aircraft process is created with each Arrival event

```
/* simulate aircraft arrival, circling, and landing */
int InTheAir;
int InTheGround;
int RunwayFree;

/* set initial values */
InTheAir := 0;
InTheGround := 0;
RunwayFree := TRUE;

/* simulate aircraft arrival */
{ InTheAir := InTheAir + 1;
  InTheGround := TRUE;
  RunwayFree := FALSE;
}
/* circle */
{ InTheAir := InTheAir - 1;
  /* wait for runway */
  WaitUntil(RunwayFree);
  /* land */
  InTheGround := InTheGround + 1;
  /* simulate aircraft on the ground */
  WaitUntil(InTheGround);
  /* simulate aircraft departure */
  InTheGround := InTheGround - 1;
  InTheAir := InTheAir - 1;
  RunwayFree := TRUE;
  /* simulate aircraft in flight */
  InTheAir := InTheAir + 1;
  /* repeat */
}
### Implementation

**AdvocateTime( T )**

Causes simulation time in the process to advance by T units

```
Execute AdvocateTime( 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

```
AdvanceTime( R );
RunwayFree := TRUE;
Resume Event Handler
```

**Implementation**

- 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

### 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
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