JSIMMOD
Introduction and Usage Ideas

NASUG Conference, March 15, 2002

Presented by:

Gregory Bradford

AirportTools

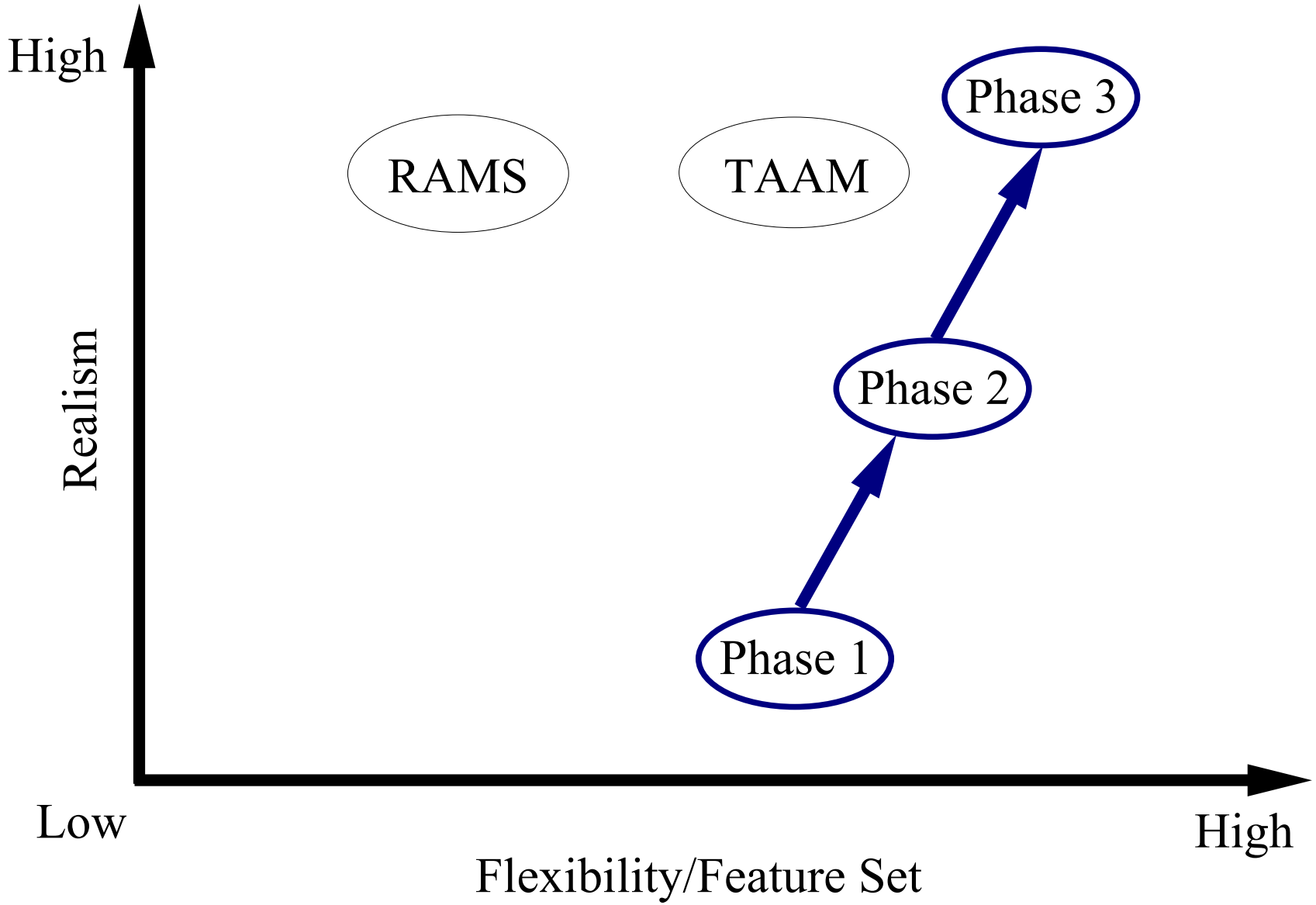
What is JSIMMOD?

JSIMMOD is an in progress re-implementation of SIMMOD. Thus, retaining core SIMMOD's functionalities and maintaining investments in SIMMOD expertise, applications, software and name recognition.

JSIMMOD is a new program with capabilities that will rival and surpass the capabilities of the other aviation simulation models.

JSIMMOD is a program that will be made widely available at no cost, with it's source code, so that, if it doesn't do what you need it to do, you will have the option of enhancing or fixing it.

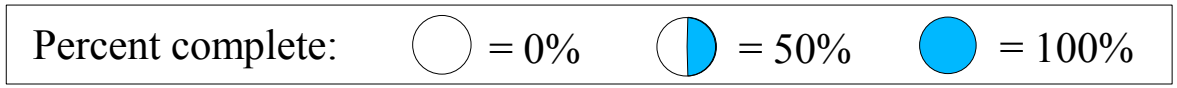
Where is JSIMMOD Going?



Please note: TAAM and RAMS are trademarks of Boeing and Eurocontrol respectively.

Phase 1 of JSIMMOD

- Incorporation of basic SIMMOD capabilities:
  - Basic airspace, including route traversal and separation logic.
  - Basic airfield logic including gate selections, taxipath planning and traversal, and departure queue logic.
  - Procedures and runway management.
  - Input and output file consistency.
- Advanced modeling and simulation capabilities:
  - Integration of Jython scripts and Jython plugin modules.
    - User defined logic via scripts written in Jython.
  - Integration of native Java plugin modules.
    - User defined logic via native plugins replacing internal logic.



Phase 2

- Add SIMMOD 3 concepts as needed.
- Incorporate broadcaster and listener capabilities:
  - Flights broadcast node and sector arrivals.
  - Controllers listen for node and sector arrivals and react.
  - Controllers broadcast flight instructions.
  - Flights listen for flight instructions and act upon them.
  - Dispatchers broadcast gate assignments.
- Implement unfinished optional SIMMOD logic as needed.
- Integrate with third party data, models, and tools as needed.

Phase 3

- Incorporate free flight concepts as necessary.
- Add, or integrate with:
  - Aviation safety models.
  - Aviation noise models.
  - Emissions models.
  - Human factors models.
  - Terminal models.
  - Land side models.

Recent Progress

- Generic Jython scripting and plugin functionality designed, implemented, and documented.
- Native Java plugin functionality fully designed, implemented, and documented.
- Arrival runway roll implementation just completed.
  - Linear deceleration methodology retained from SIMMOD.
  - Jython or native Java plugin enabled for customization purposes.
- Implemented foreign language localization capabilities.

Jython Examples Summary

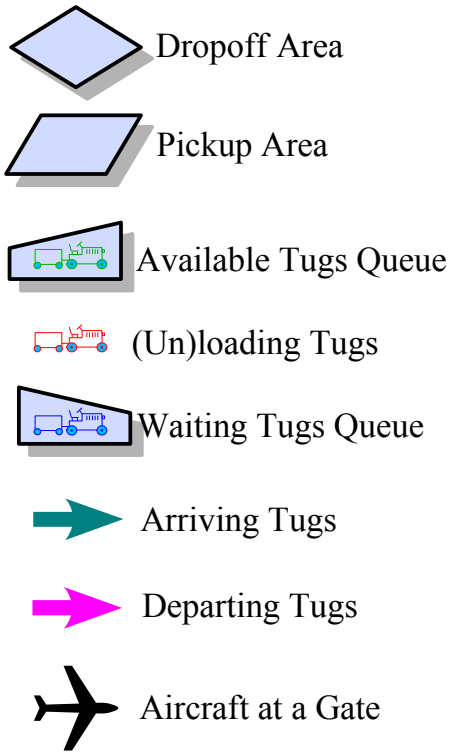
- Ground Support Equipment Model – A fully developed model which evaluates service levels provided by baggage tugs at a hypothetical airport.

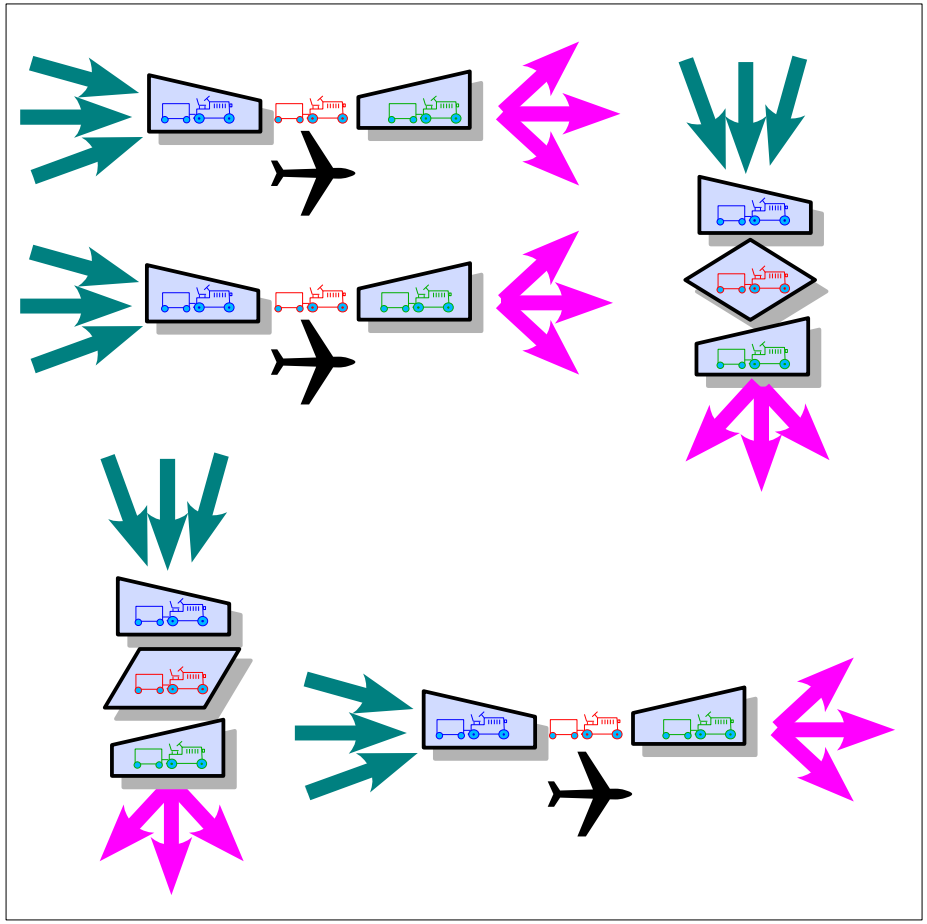
Utilizes JSIMMOD's internal scheduling mechanism and Jython scripts.

- JYTHON_SCRIPT Card – How to schedule a Jython script for execution at a certain time during JSIMMOD's execution.

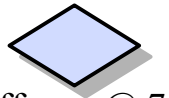
- A hypothetical DFW to LAX Flight – Shows some basic concepts of how a Flight's “lifecycle” might be affected by traffic levels and weather.

Modeling Ground Support Equipment With JSIMMOD

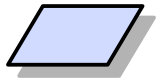




GSE Model Parameters

Dropoff Area: 

- ✔ Baggage dropoff area @ 7.5, 7.5 in a 10.0 by 10.0 grid.
- ✔ Active tug capacity = 1
- ✔ Waiting to dropoff area capacity = unlimited.
- ✔ Finished droping off area capacity = unlimited.

Pickup Area: 

- ✔ Baggage pickup area @ 2.5, 2.5 in a 10.0 by 10.0 grid.
- ✔ Active tug capacity = 1
- ✔ Waiting to pickup area capacity = unlimited..
- ✔ Finished picking up area capacity = Not Applicable.

Tugs: 

- ✔ 30 < Tug capacity in bags < 180 bags.
- ✔ 30 tugs total.
- ✔ 10 tugs initially assigned pickup area, 10 to dropoff area, and 10 to randomized gate locations.
- ✔ Tugs travel rate = 30 seconds/grid unit.
- ✔ At pickup or dropoff areas loading/unloading Tugs @ 1.5 seconds/bag.
- ✔ At aircraft tugs load/unload rate is 2.0 seconds/bag.

Aircraft: 

- ✔ Aircraft injected every 6 to 30 minutes.
- ✔ Aircraft are either an arrival or a departure.
- ✔ Aircraft are injected at randomized grid locations, which are considered “Gates”.
- ✔ 20 < Aircraft capacity in bags < 800 bags.
- ✔ Upon injection aircraft request tugs in sufficient numbers (and baggage carrying capacity) to load or unload the entire aircraft.

Statistics Collected

✈ Descriptions of Statistics Collected:

- ✔ Dropoff Delay – The time a tug spent waiting to dropoff baggage due to another tug currently utilizing the dropoff area.
- ✔ Pickup Delay – The time a tug spent waiting to pickup baggage due to another tug currently utilizing the pickup area.
- ✔ Aircraft Delay – The time a tug spent waiting to load or unload baggage due to another tug currently at the aircraft.
- ✔ Travel Time – The time a tug spent traveling to/from pickup, dropoff, and aircraft loading areas.
- ✔ Idle Time – The time a tug spent waiting to be activated.
- ✔ Aircraft Loading Time – The time spent by a tug loading it's baggage onto an aircraft.
- ✔ Aircraft Unloading Time – The time spent by a tug unloading baggage from an aircraft.
- ✔ Pickup Loading Time - The time spent by a tug loading baggage at the baggage pickup area.
- ✔ Dropoff Unloading Time – The time spent by a tug unloading baggage at the baggage dropoff area.

✈ Raw Data for Statistics:

- ✔ Found in a file called “tugs_logfile.csv” located in the Application's “Simulation” subdirectory.

Sample:

```
1.000000,Tug #11,pickup load time =,0.045833,will finish at,1.045833
1.045833,Tug #11,finishes pickup loading; traveling to,Aircraft #1
1.045833,Tug #12,activated to load at pickup
1.045833,Tug #11,begins travel to aircraft,travel time is,0.041908,arrival time is,1.087741
```

✈ Summary Statistical Data By Tug:

- ✔ **Found at the end of a file called “tugs_logfile.csv” located in the Application's “Simulation” subdirectory.**

Sample:

```
26.107601,Tug #8,Dropoff Delay,Pickup Delay,Aircraft Delay,Travel Time,Idle Time,Loading Aircraft Time.....
26.107601,Tug #8,0.308895,0.301075,1.158830,0.857479,22.816322,0.158333,0.221667,0.118750,0.166250,26.107601
```

Ground Support Equipment Output Statistics

	Dropoff Delay	Pickup Delay	Aircraft Delay	Travel Time	Idle Time	Aircraft Loading Time	Aircraft Unloading Time	Pickup Loading Time	Dropoff Unloading Time	Total Non-Idle Time
Tug #1	0.05	0.00	0.34	0.27	25.24	0.00	0.11	0.00	0.09	0.87
Tug #6	0.14	0.15	0.18	0.33	24.57	0.17	0.25	0.13	0.19	1.53
Tug #5	0.02	0.19	0.65	0.44	24.38	0.12	0.12	0.09	0.09	1.73
Tug #10	0.00	0.15	0.34	0.63	24.36	0.21	0.16	0.16	0.12	1.75
Tug #26	0.12	0.14	0.63	0.37	24.30	0.15	0.15	0.12	0.12	1.80
Tug #19	0.25	0.13	0.63	0.35	23.92	0.29	0.19	0.22	0.14	2.19
Tug #24	0.27	0.26	1.20	0.62	23.38	0.05	0.17	0.04	0.13	2.72
Tug #15	0.16	0.41	0.72	0.97	23.30	0.22	0.10	0.16	0.07	2.80
Tug #12	0.03	0.29	0.68	0.94	23.12	0.36	0.23	0.27	0.17	2.98
Tug #17	0.07	0.12	0.60	0.75	23.03	0.79	0.09	0.59	0.07	3.08
Tug #9	0.30	0.00	0.72	0.94	22.98	0.26	0.41	0.19	0.31	3.13
Tug #3	0.11	0.13	0.88	0.91	22.86	0.32	0.38	0.24	0.28	3.25
Tug #8	0.31	0.30	1.16	0.86	22.82	0.16	0.22	0.12	0.17	3.29
Tug #14	0.02	0.25	0.19	0.93	22.63	0.92	0.28	0.69	0.21	3.47
Tug #25	0.08	0.00	1.13	0.76	22.51	0.28	0.65	0.21	0.48	3.59
Tug #29	0.25	0.24	1.18	0.89	22.46	0.31	0.31	0.23	0.23	3.65
Tug #7	0.00	0.24	0.79	1.05	22.31	0.56	0.42	0.42	0.32	3.80
Tug #4	0.01	0.33	1.25	0.73	22.31	0.31	0.54	0.23	0.41	3.80
Tug #28	0.27	0.07	1.38	0.61	22.27	0.19	0.67	0.14	0.50	3.83
Tug #20	0.11	0.44	1.67	1.10	22.10	0.26	0.13	0.20	0.10	4.01
Tug #22	0.28	0.42	1.72	1.06	21.81	0.25	0.22	0.19	0.16	4.30
Tug #23	0.05	0.14	1.37	1.01	21.61	0.51	0.59	0.38	0.44	4.50
Tug #30	0.27	0.20	1.90	0.78	21.39	0.27	0.63	0.20	0.48	4.72
Tug #11	0.10	0.72	1.34	1.08	21.26	0.67	0.24	0.50	0.18	4.84
Tug #21	0.36	0.67	1.99	1.24	21.26	0.21	0.14	0.15	0.10	4.85
Tug #27	0.10	0.64	1.24	1.15	21.02	0.67	0.45	0.50	0.34	5.08
Tug #16	0.00	0.41	0.76	1.20	20.99	0.79	0.79	0.59	0.59	5.12
Tug #2	0.22	0.57	1.57	1.10	20.96	0.52	0.44	0.39	0.33	5.15
Tug #18	0.02	0.43	0.99	1.31	20.69	0.97	0.56	0.73	0.42	5.42
Tug #13	0.05	0.93	1.86	1.41	20.40	0.53	0.31	0.40	0.23	5.71
Sum:	4.04	8.97	31.02	25.78	676.25	11.30	9.94	8.48	7.46	106.98
Average:	0.13	0.30	1.03	0.86	22.54	0.38	0.33	0.28	0.25	3.57
Maximum:	0.36	0.93	1.99	1.41	25.24	0.97	0.79	0.73	0.59	5.71
Minimum:	0.00	0.00	0.18	0.27	20.40	0.00	0.09	0.00	0.07	0.87

All Values in Decimal Hours

JYTHON_SCRIPT Card Input

- The JYTHON_SCRIPT card is a new SIMU09 card.
- A JYTHON_SCRIPT card executes a Jython script at a certain time during the simulation.
- JYTHON_SCRIPT instances are generic higher-order objects within JSIMMOD, meaning they have the following attributes:
  - Can be scheduled and then rescheduled.
  - Run independently without a requirement for additional inputs.
  - Have full read/write access to JSIMMOD's internal data structures, including all SIMMOD cards.
  - Maintain their own internal data structures and variables.
  - Inherit built-in low-level reporting and logging of activities.

JYTHON_SCRIPT Card Input

- JYTHON_SCRIPT cards can be used for a limitless variety of purposes including:
  - Creation of an entire standalone Ground Support Equipment model.
  - Creation of "weather systems" that systematically move through the model reducing link capacities or intrail-separations.
  - Periodically polling departure queues to gather and report statistics.
  - Gathering internal simulation data, formatting the data, and forwarding that data on to a human factors model.
  - Implementation of "super-smart" DSDPaths that watch nearby traffic patterns and adjust their capacities to help prevent downstream congestion.

Jython Script Card Input

SIMU09 Format:

JYTHON_SCRIPT,Time,ReportingLevel,TraceLevel,UniqueId,JSName,UserData

(R)	**Time**	The time the Jython script should be run by JSIMMOD.
(I)	**Reporting Level**	The level of reporting which is desired for this object. User defined value(s).
(I)	**Trace Level**	Valid values are 0, 1, 200, 201.
(A)	**UniqueId**	The script's identifier.
(A)	**JSName**	The path and name of the Jython script to be run by JSIMMOD.
(A)	**UserData**	**A** free-form string field. Access is via the getUserData(String) method. Example call:

```
my_user_data = THIS.getUserData("JYTHON_SCRIPT_USERDATA")
```

SIMU09 Example:

JYTHON_SCRIPT,7.25,0,0,HeavySnow,c:\....\HeavySnowScript.py,Weather intensity is 9.5 runway is 17L

This record defines a Jython script called c:\....\HeavySnowScript.py to be run at 7.25 hours into the simulation. The ReportingLevel and the TraceLevel both equal zero. The script itself is uniquely identified by the identifier "HeavySnow". The analyst has supplied the following UserData: "Weather intensity is 9.5 runway is 17L".

Jython Script Card Input

- Pseudocode example inside the c:\....\HeavySnowScript.py Jython Script:

```
# Retrieve the internal Java object (An Entity object) which represents the Jython script to the scheduler.
my_script = THIS

# Retrieve the user data from the internal object
my_user_data = my_script.getUserData("JYTHON_SCRIPT_USERDATA")

# Split user data string up into an array
list_of_strings = split(my_user_data)

# Get the initial snow intensity value
snow_intensity = Float (  list_of_strings[4]  )

# Get the runway
rwy = list_of_strings[7]

# Get the model's scheduler
scheduler = my_script.getModel().getScheduler()

# Get a linear distribution from simulation library
min_snow  = 0.0
max_snow = 10.0
distribution = UniformRealDist ( min_snow , max_snow )
```

Continued

```
# Check the snow intensity
while   snow_intensity   >   3.2

  # Set runway procedures appropriately here
  if  snow_intensity < 7.2  then  rwy.setArrSep ( 2.5 )
  if  snow_intensity > 7.2  then  rwy.setArrSep ( 5.0 )
  if  snow_intensity > 9.2  then  rwy.setArrSep ( 7.5 )

  # reschedule 15 minutes into the future
  delta_time = 0.25

   # Reschedule script, then put it sleep until
   # scheduler wakes it up again
   scheduler.reschedule( my_script , delta_time )
   my_user_data.passivate()

   # get a new snow intensity
   snow_intensity = distribution.sample()
endwhile
```

DFW to LAX Example

The following example is part Jython and part pseudo code. The pseudo code is necessary since portions of the example are not implemented in JSIMMOD yet.

```
# Assume flight originates from a data file read by JSIMMOD earlier.
Flight = CURRENT_FLIGHT
# Set basic flight parameters
Flight.setOrigin( “DFW” )
Flight.setDestination( “LAX” )
Flight.setAircraftModel( “B757” )

# Determine the route to exit DFW area.
RTE_SET = JSIMMOD.getCard(“ROUTES”)
RTE#1 = RTE_SET.getObjectById (“17L_TO_CORNER_POST”)
RTE#2 = RTE_SET.getObjectById (“17R_TO_CORNER_POST”)

RTE_TO_FLY = RTE#1
If   RTE#1.getNumAcOnRoute()  >  RTE#2.getNumAcOnRoute()
      RTE_TO_FLY = RTE#2

Flight.addRoute ( RTE_TO_FLY )
```

DFW to LAX Example (cont.)

```
# Perform flight's lifecycle, i.e. Fly the route
Flight.body()

# Determine the route from DFW center to LAX center.
RTE#1 = RTE_SET.getObjectById ("CORNER_POST_N_LAX")
RTE#2 = RTE_SET.getObjectById ("CORNER_POST_S_LAX")

# Get hypothetical weather card
WEATHER_SET = JSIMMOD.getCard("WEATHER")
RTE_TO_FLY = RTE#1

If   WEATHER_SET.getMaxWeather( RTE#1 )  >=  WEATHER_SET.SEVERE
   RTE_TO_FLY = RTE#2

Flight.addRoute ( RTE_TO_FLY )

# Perform flight's lifecycle, i.e. Fly the route
Flight.body()
```

During the flight of the assigned route certain nodes and links may be "watched" by a gate dispatcher.

When the flight reaches XYZ node or a point on ABC link the dispatcher can send a message to the flight which indicates the assigned arrival gate.

Final Thoughts

- Development is accelerating since much of the core modeling and plugin development is now finished.

- Even as an alpha level product, JSIMMOD has flexibility and extensibility that far exceeds other leading models.

- Due to JSIMMOD's open object oriented design prototyping new aviation modeling concepts is much faster.

- Aviation simulation tools augmented by user-defined logic promise unprecedented levels of simulation capability.