An Introduction to Modeling and Simulation with DEVS

Gabriel A. Wainer
Department of Systems and Computer Engineering
Carleton University.
Ottawa, ON. Canada.
http://www.sce.carleton.ca/faculty/wainer

Outline

- Problem characterization
- DEVS formalism
- The CD++ tool
- Modeling complex systems using DEVS
- Examples of application

- FOCUS => Newcomers

Some of the slides here presented are part of Prof. B. Zeigler's collection (with permission!)

http://www.acims.arizona.edu

Motivation

- Analysis of complex natural/artificial real systems.

- Continuous systems analysis
  - Different mathematical formalisms
  - Simulation: solutions to particular problems under certain experimental conditions of interest

- Classical methods for continuous systems simulation
  - Based on numerical approximation
  - Require time discretization
  - Inefficient in terms of execution times
  - Complex composition; difficulties in integration, multiresolution models

Evolution in simulation technology

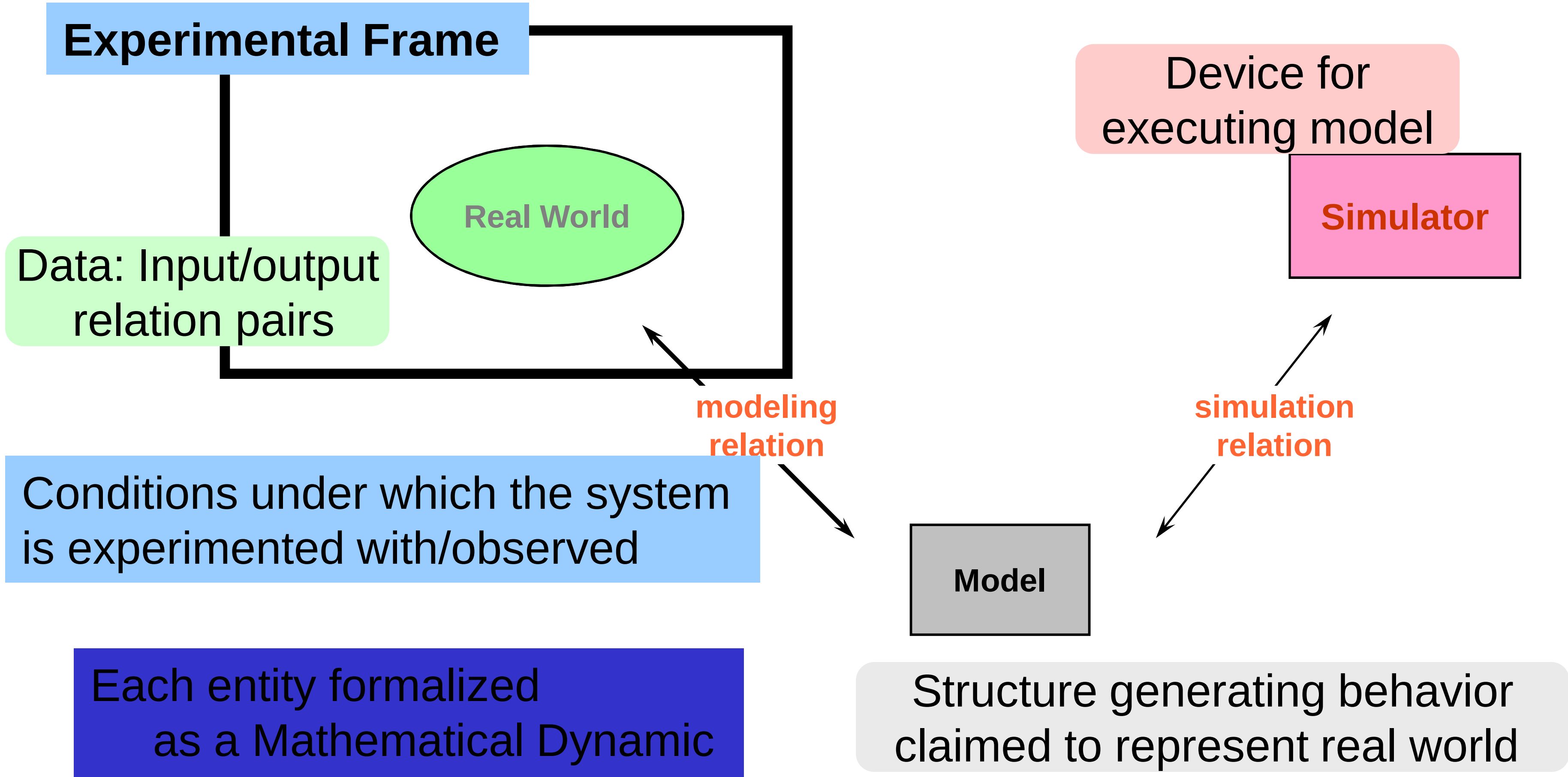
Carleton UNIVERSITY

- Reduced cost of modern computers
- Enhanced tools
- Statistical packages; application libraries
- Ease to use, flexibility
- Ease of analysis tasks
- Parallel/Distributed systems
- Enhanced visualization tools
- Standards (graphics, runtime support, distributed software)

Discrete-Event M&S

- Based on programming languages (difficult to test, maintain, verify).
- Beginning '70s: research on M&S methodologies
- Improvement of development task
- Focus in reuse, ease of modeling, development cost reductions

Separation of concerns in DEVS



Each entity formalized
as a Mathematical Dynamic
System
(mathematical manipulations
to prove system properties)

Current needs

- **Interoperability:**
  - computer-based and non-computer-based systems
    - support a wide range of models and simulations
  - hybrid interoperability
- **Reuse:**
  - model and simulation reuse (computer-based and otherwise)
    - centralized and distributed data and model repositories
- **Performance:**
  - Computational (local to each simulation)
  - Communication (among multiple simulations)

Current practices

- Ad-hoc techniques, ignorance of previous recommendations for software engineering.
- Tendency to encapsulate models/simulators/experimental frames into tightly coupled packages, (written in programming languages such as Fortran, C/C++, Java).
- Difficulties: testing, maintainability of the applications, integration, software reuse.
- Relatively few examples of storing previously developed simulation infrastructure commodities such that they can be adapted to developing interoperability test requirements

DEVS M&S methodology

Carleton UNIVERSITY

- DEVS can be used to solve the previously mentioned issues:
  - Interoperability and reuse
  - Hybrid systems definition
  - Engineering-based approach
  - Facilities for automated tasks
  - Reduced life cycles
  - High performance/distributed simulation

The DEVS M&S Framework

Carleton UNIVERSITY

- DEVS = Discrete Event System Specification
- Formal M&S framework
- Separates Modeling from Simulation
- Supports full range of *dynamic system* representation capability
- Supports hierarchical, modular model development
- Provides Well Defined Coupling of Components
- Supports
  - Hierarchical Construction
  - Stand Alone Testing
  - Repository Reuse

(Zeigler, 1976/84/90/00)

A Layered view on M&S

Carleton UNIVERSITY

Applications

Models

Simulators (single/multi CPU/RT)

Middleware/OS (Corba/HLA/P2P; Windows/Linux/RTOS…)

Hardware (PCs/Clusters of PC/HW boards…)

A Layered view on M&S

Applications

Models

Simulators (single/multi CPU/RT)

Middleware/OS (Corba/HLA/P2P; Windows/Linux/RTOS…)

Hardware (PCs/Clusters of PC/HW boards…)

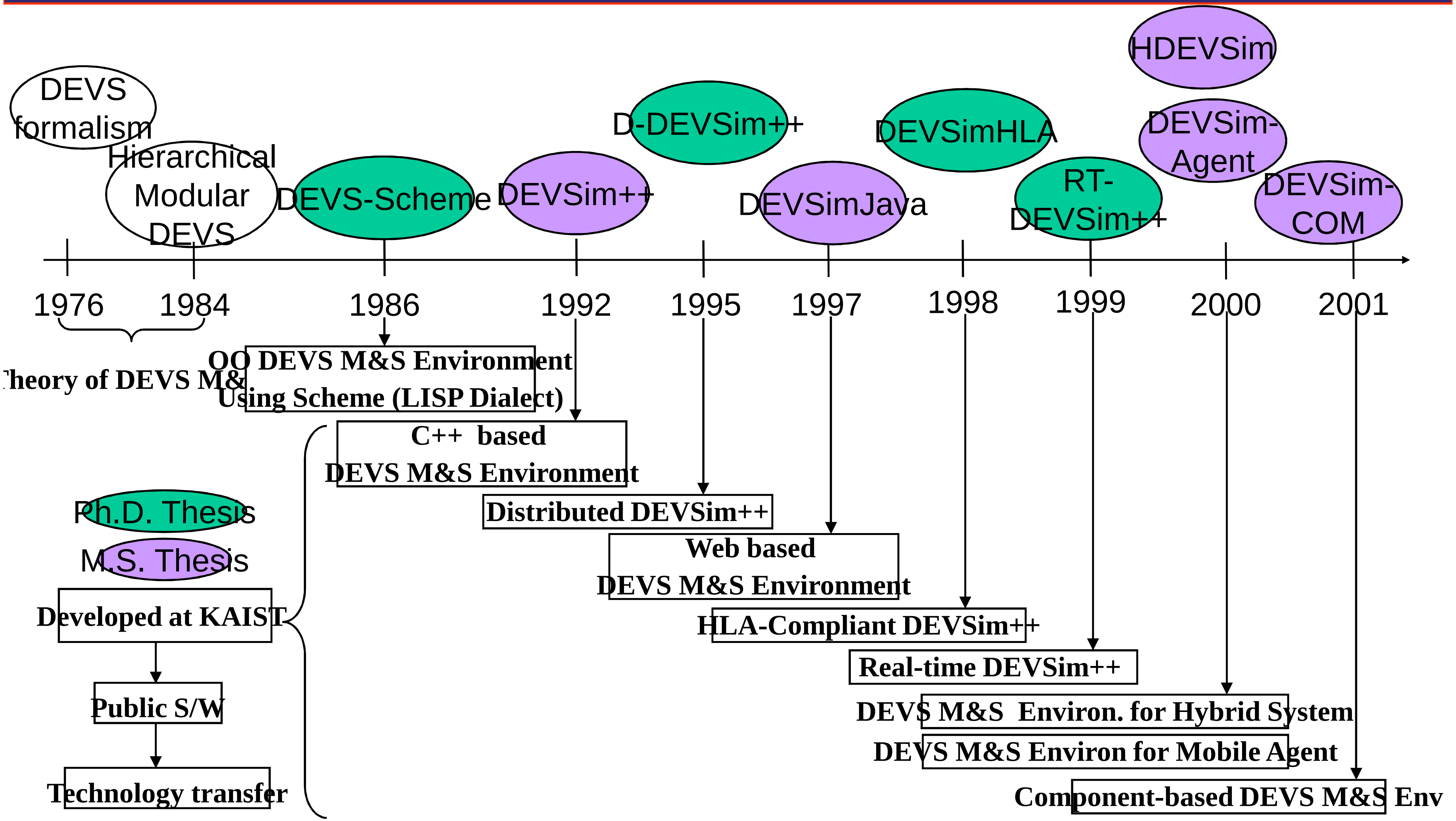
Advantages of DEVS

- Models/Simulators/EF: distinct entities with their own software representations.
- Simulators can perform single host, distributed and real-time execution as needed (DEVS simulators over various **middleware** such as MPI, HLA, CORBA, etc.).
- Experimental frames appropriate to a model distinctly identified; easier for potential users of a model to uncover objectives and assumptions that went into its creation.
- Models/ frames developed systematically for interoperability
- Repositories of models and frames created and maintained (components for constructing new models). Models/frames stored in repositories with information to enable reuse.

DEVS Toolkits

- **ADEVS** (University of Arizona)
- **CD++** (Carleton University)
- **DEVS/HLA** (ACIMS)
- **DEVSJAVA** (ACIMS)
- **GALATEA** (USB – Venezuela)
- **GDEVS** (Aix-Marseille III, France)
- **James** (University of Rostock, Germany)
- **JDEVS** (Université de Corse - France)
- **PowerDEVS** (University of Rosario, Argentina)
- **SimBeams** (University of Linz – Austria)
- **SmallDEVS** (University of Brno, Czech Republic)
- **VLE** (Université du Litoral - France)
- New efforts in China, France, Portugal, Spain, Russia.

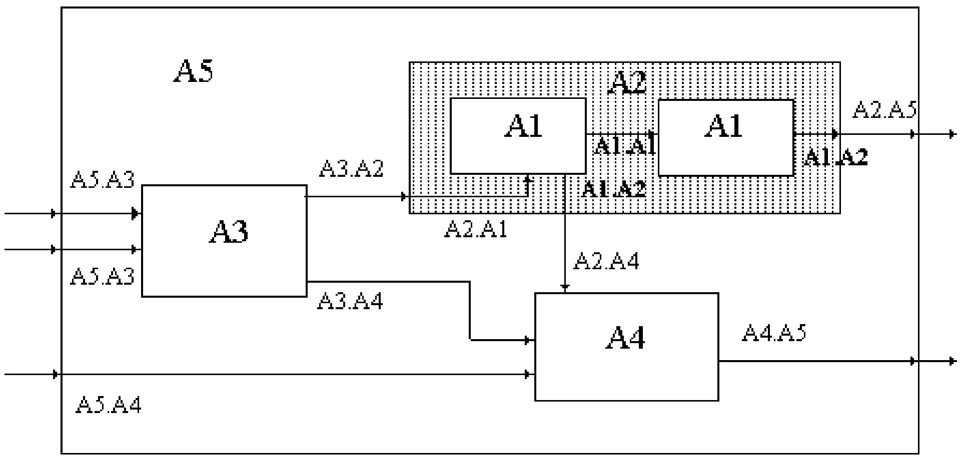
KAIST



DEVS formalism
Hierarchical Modular DEVS
DEVS-Scheme
DEVSim++
D-DEVSim++
DEVSimJava
DEVSimHLA
RT-DEVSim++
HDEVSim
DEVSim-Agent
DEVSim-COM
1976
1984
1986
1992
1995
1997
1998
1999
2000
2001
Theory of DEVS M&S
OO DEVS M&S Environment Using Scheme (LISP Dialect)
C++ based DEVS M&S Environment
Distributed DEVSim++
Web based DEVS M&S Environment
HLA-Compliant DEVSim++
Real-time DEVSim++
DEVS M&S Environ. for Hybrid System
DEVS M&S Environ for Mobile Agent
Component-based DEVS M&S Env
Ph.D. Thesis
M.S. Thesis
Developed at KAIST
Public S/W
Technology transfer

DEVS Formalism (cont.)

- Discrete-Event formalism: time advances using a continuous time base.
- Basic models that can be coupled to build complex simulations.
- Abstract simulation mechanism



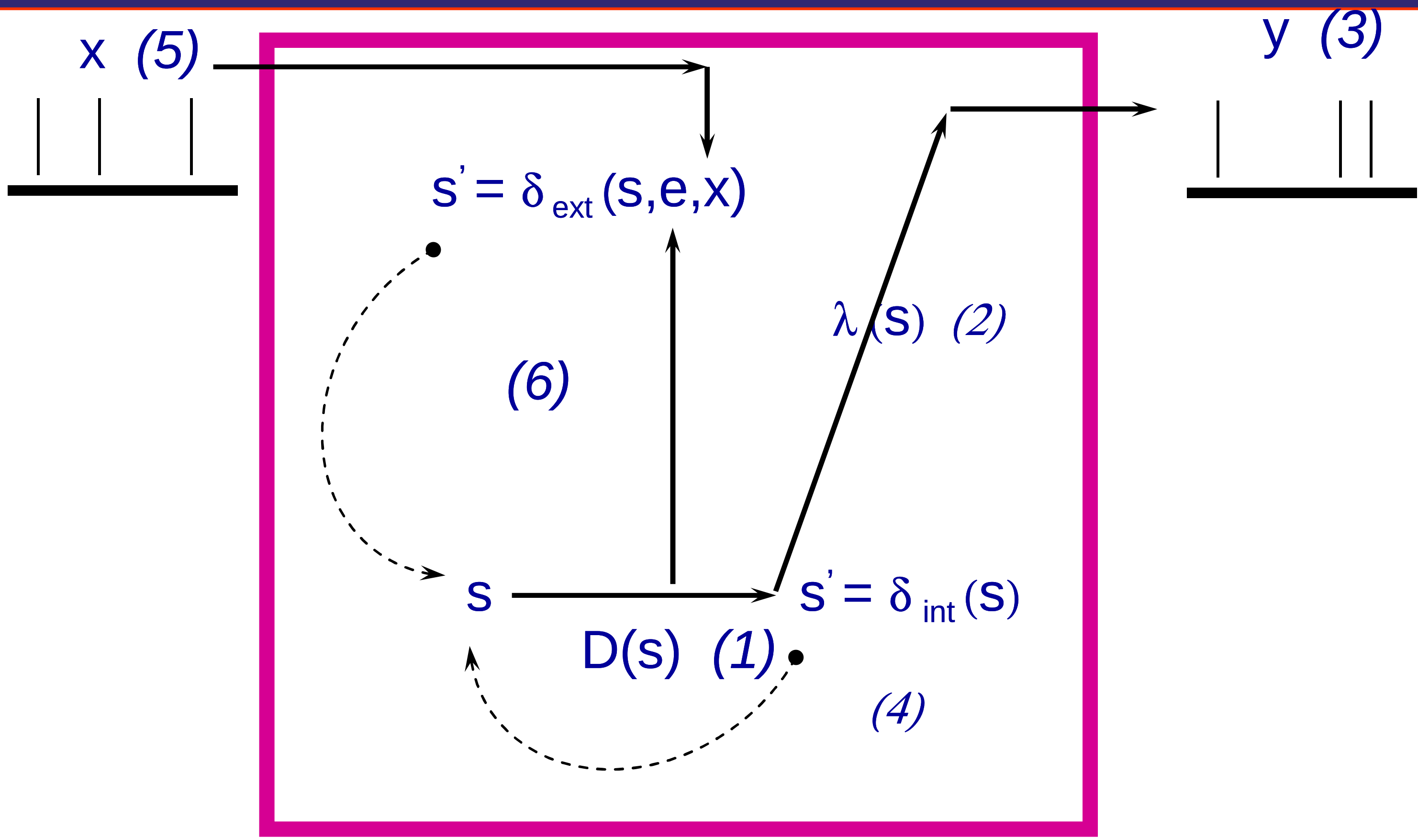
Atomic Models:

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >.$$

Coupled Models:

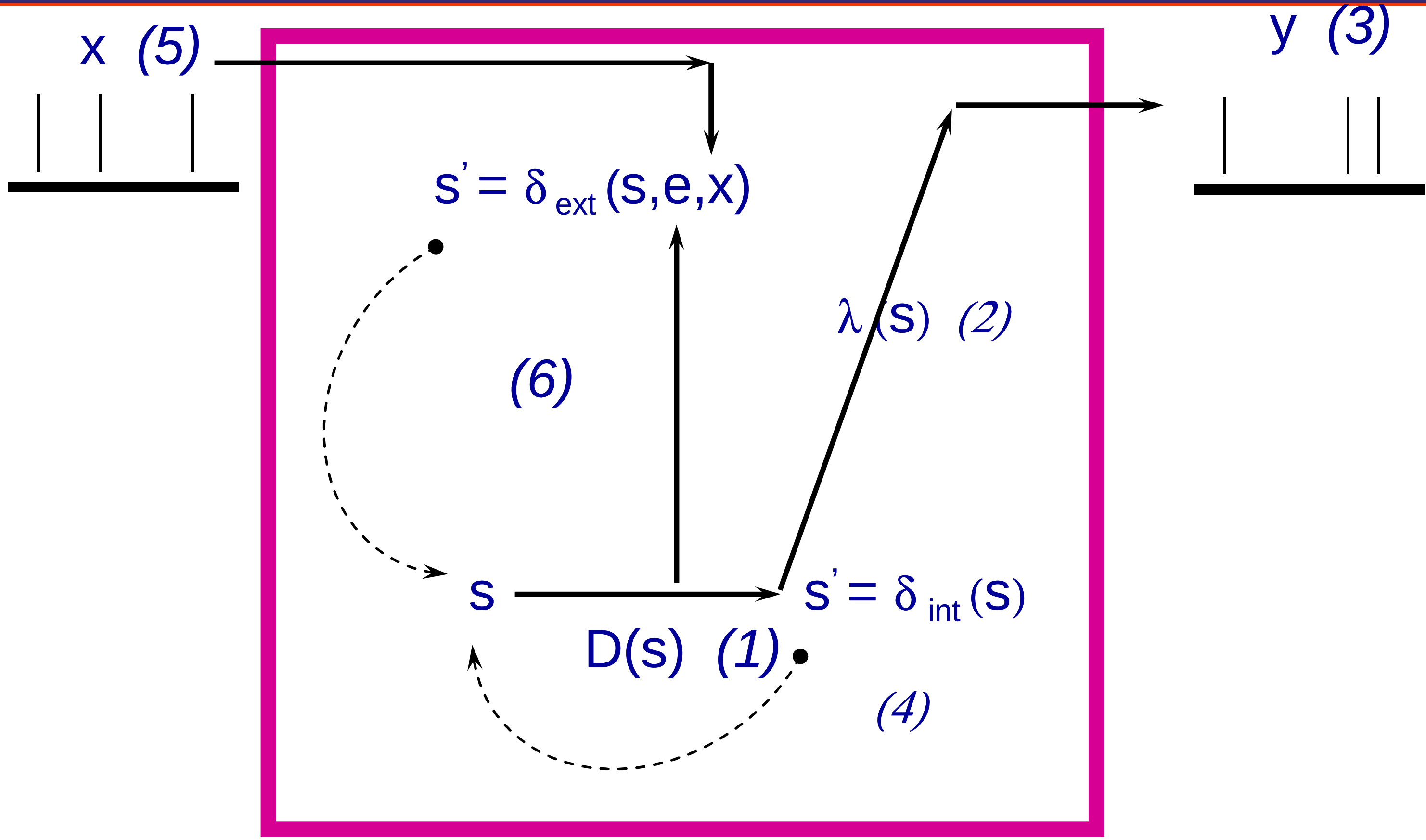
$$CM = < X, Y, D, \{M_i\}, EIC, EOC, IC, select >$$

DEVS atomic models semantics



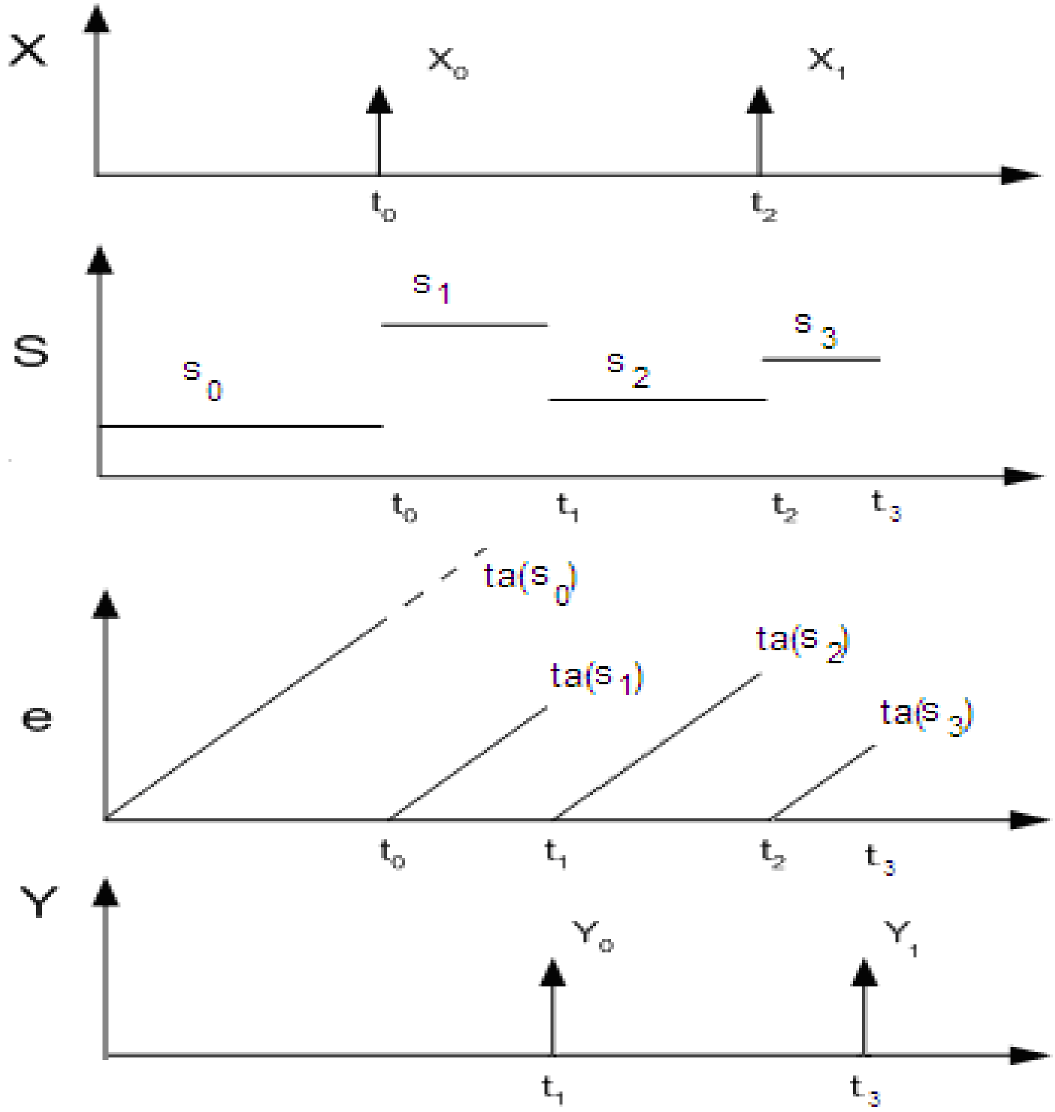
DEVS = < X, S, Y, δ_{int} , δ_{ext} , D, λ >

DEVS atomic models semantics



DEVS = < X, S, Y, δ_{int} , δ_{ext} , D, λ >

Dynamic behavior

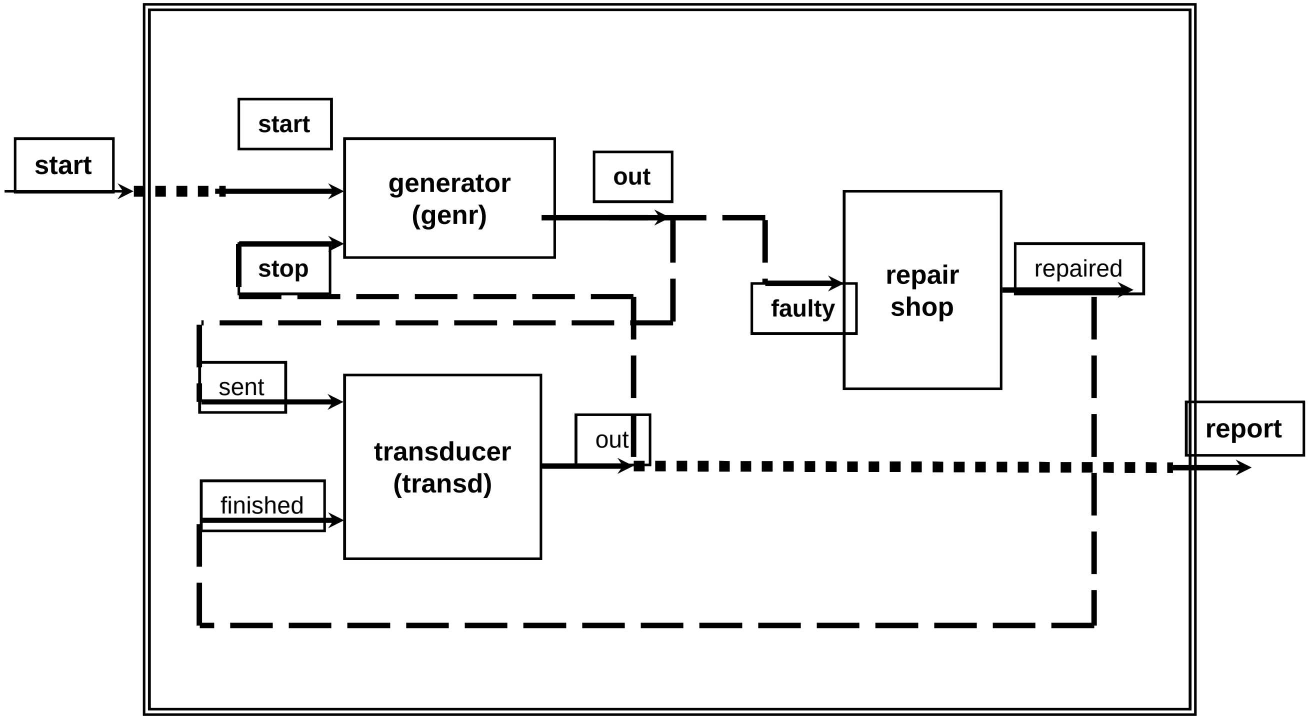


Coupled Models

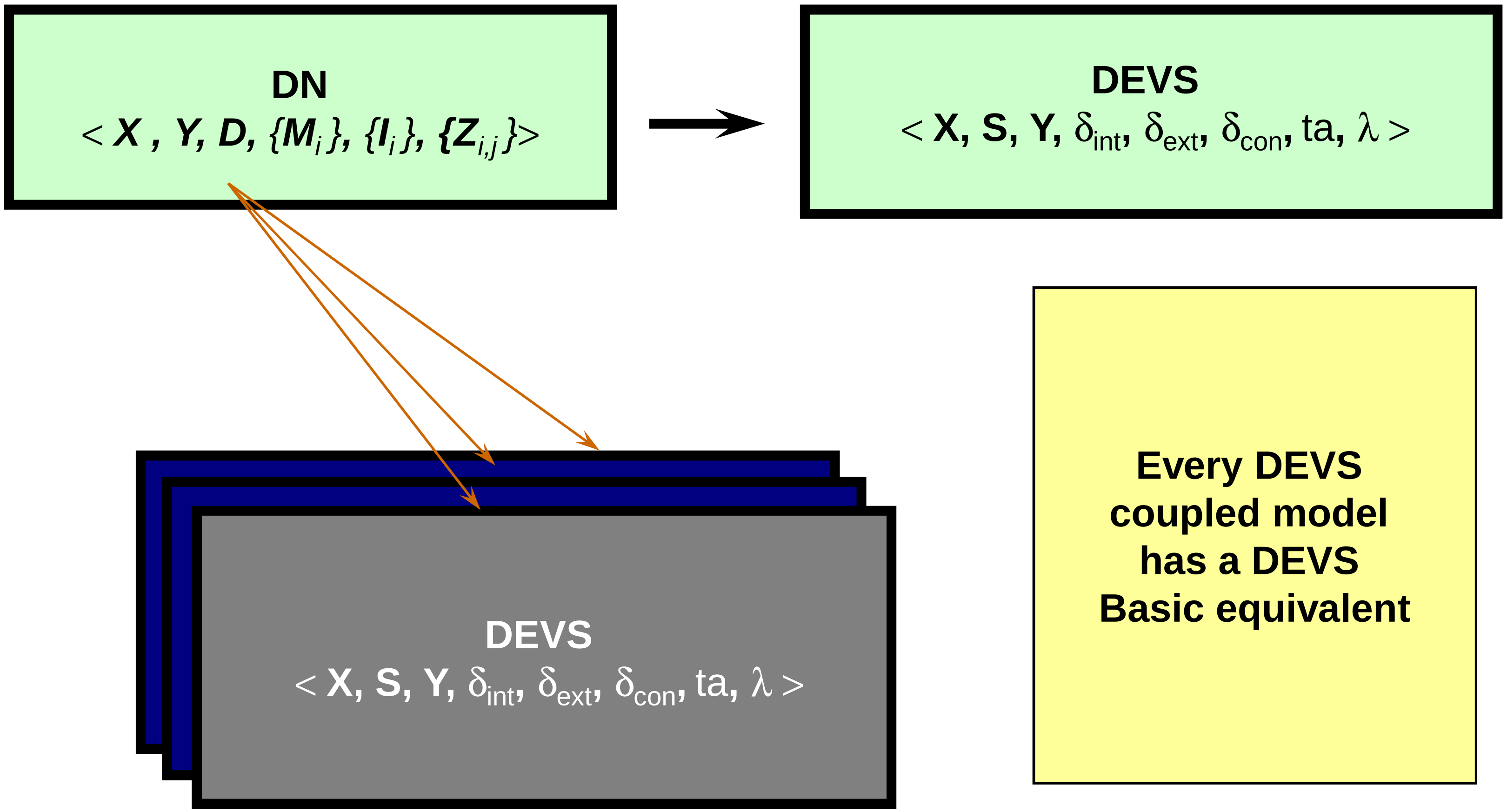
Components

couplings

- **Internal Couplings**
- **External Input Couplings**
- **External Output Couplings**



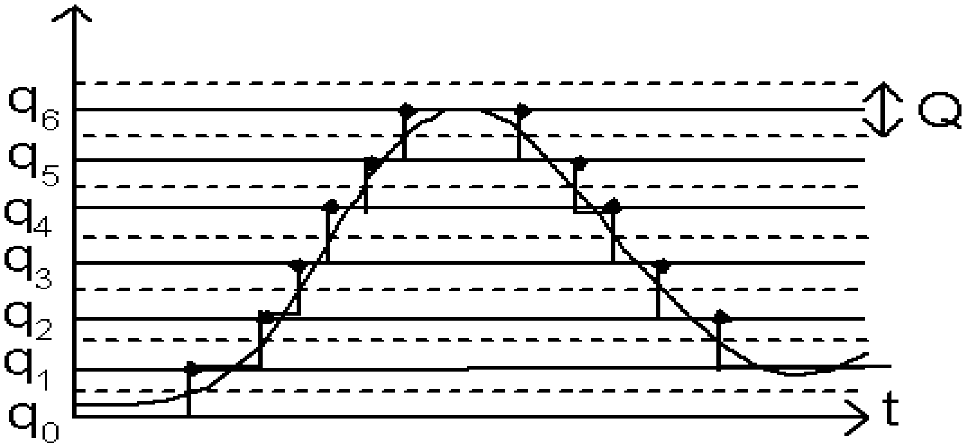
Closure Under Coupling



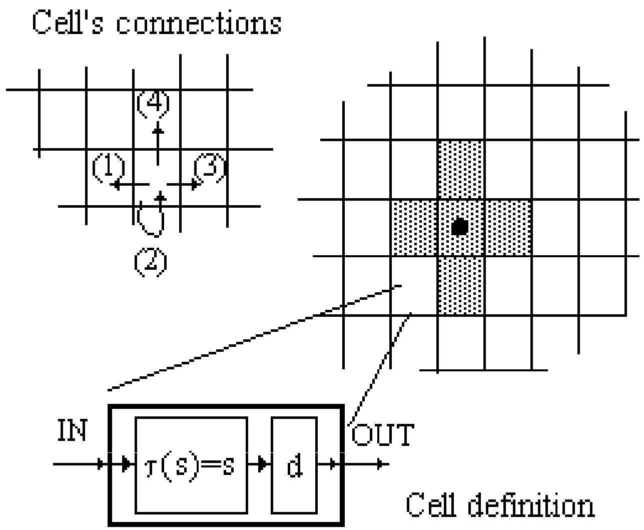
Every DEVS coupled model has a DEVS Basic equivalent

Quantized DEVS (QDEVS)

- Continuous signal represented by crossing of an equal spaced set of boundaries, separated by a *quantum* size
- Check for boundary crossing for every change in the model
- Outputs generated only when a crossing occurs
- Substantial reduction of the message updates frequency

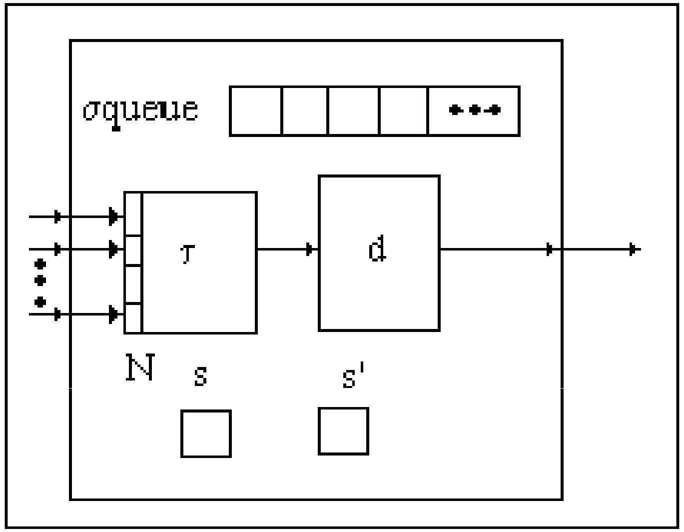


Cell-DEVS models

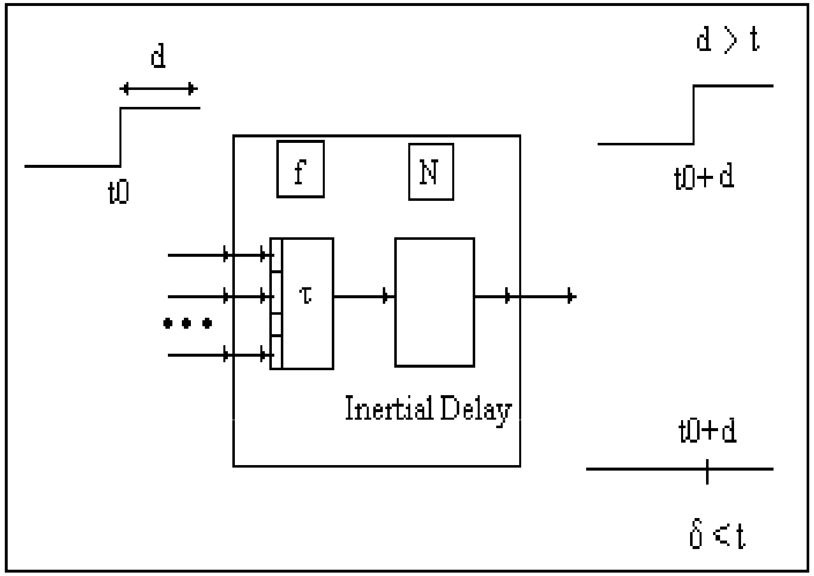


- Discrete-Events cell spaces
- Cells: atomic models. Automated coupling.
- Asynchronous execution using explicit delay functions
- Abstract simulation mechanism.

Cell-DEVS Atomic Models



Transport Delay



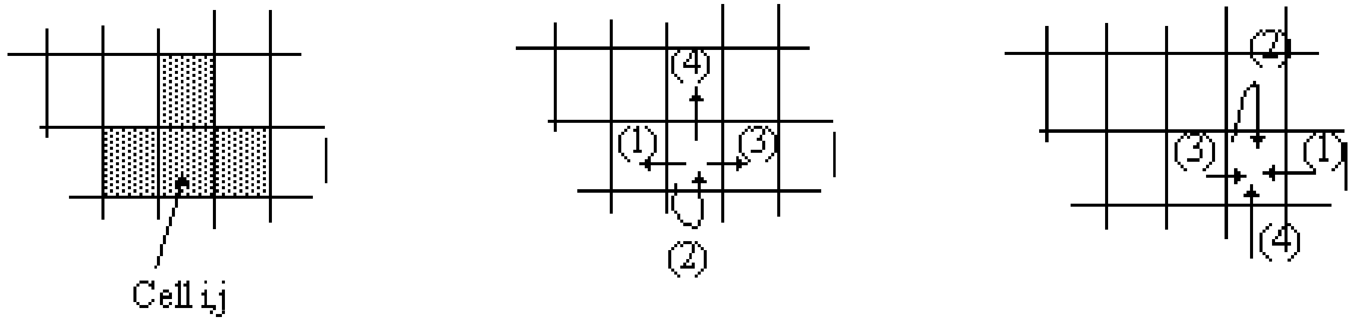
Inertial Delay

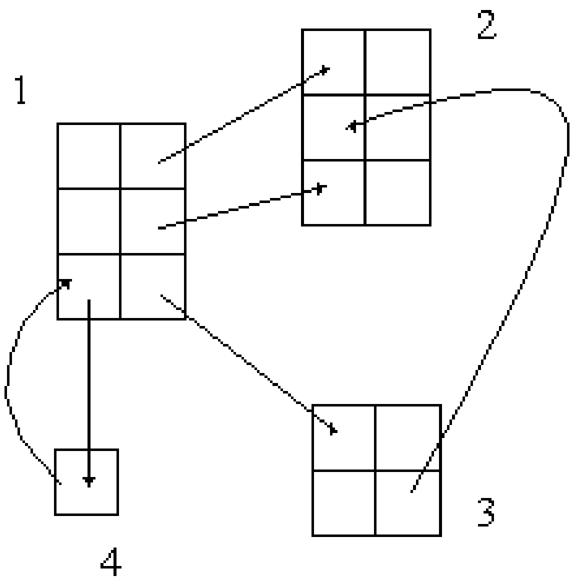
TDC= < X,Y, θ, N, **d**, τ, δ_{int}, δ_{ext}, λ, D>

- N inputs to a given cell
- Local computing function
- Inertial or Transport delays
- Outputs only if the cell state changes

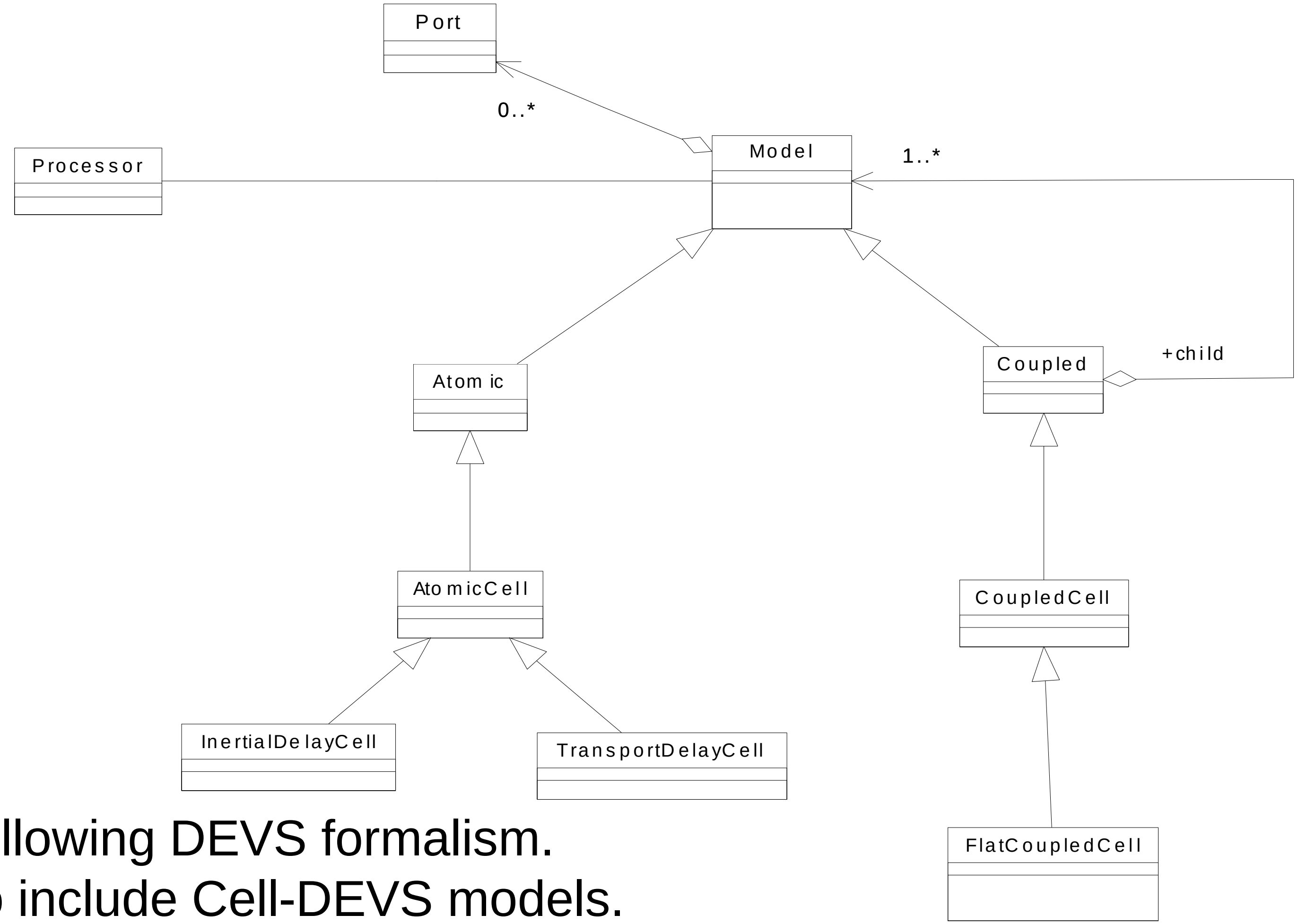
Coupled Cell-DEVS

$$GCC = < \mathbf{Xlist}, \mathbf{Ylist}, X, Y, \mathbf{n}, \mathbf{\{t_1,...,t_n\}}, \mathbf{N}, C, B, Z >$$



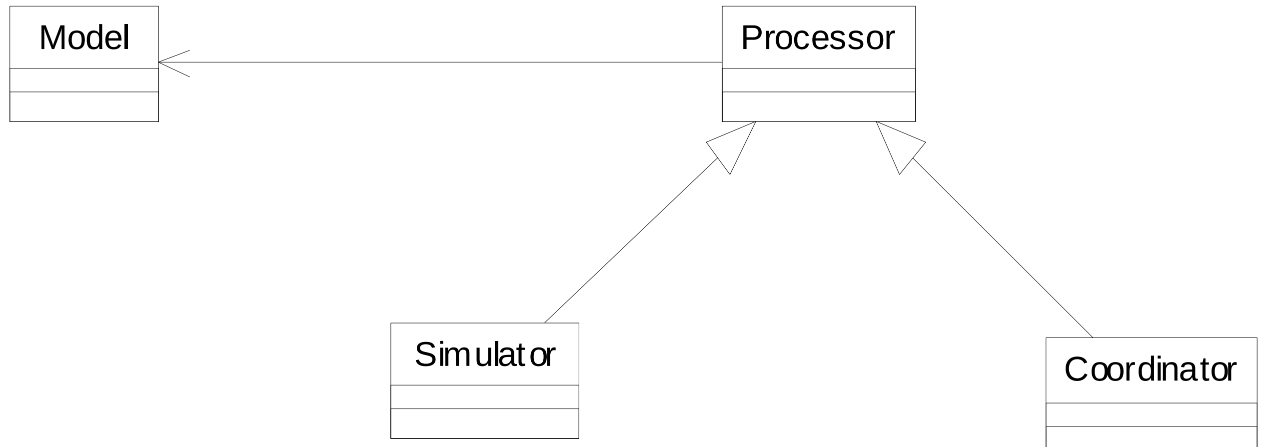


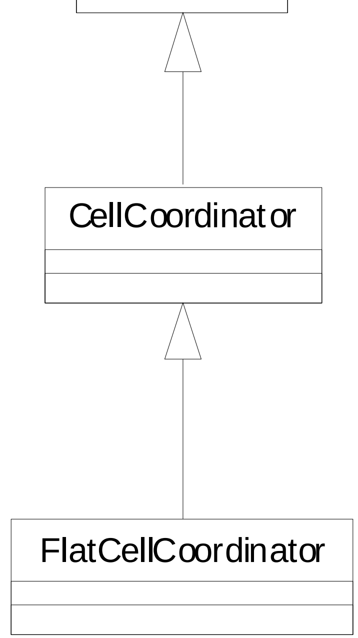
The CD++ toolkit



- Basic tool following DEVS formalism.
- Extension to include Cell-DEVS models.
- High level specification language for model definition.

CD++ simulator





Independent simulation mechanisms

("Abstract" simulator)

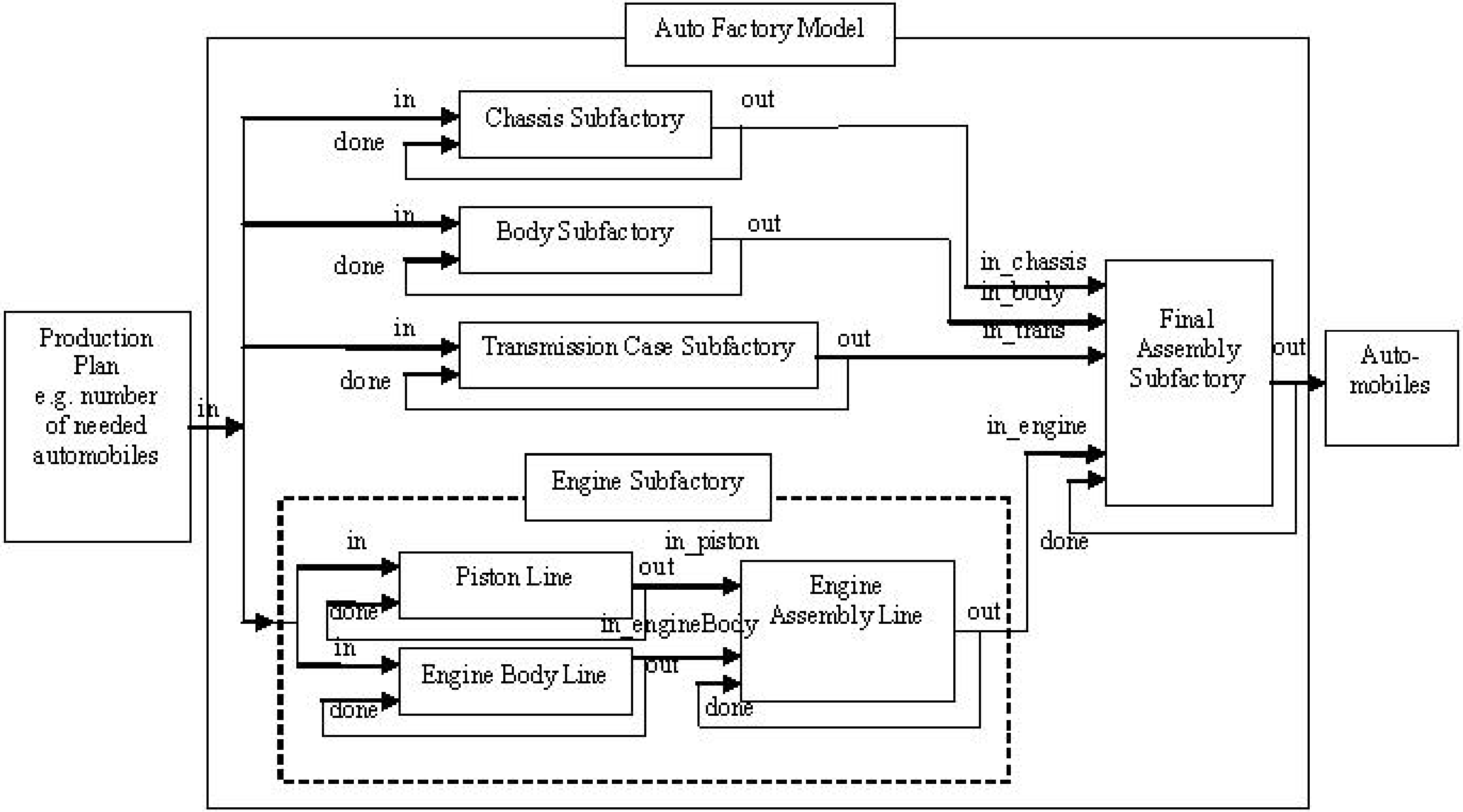
. Hierarchical

. Flat

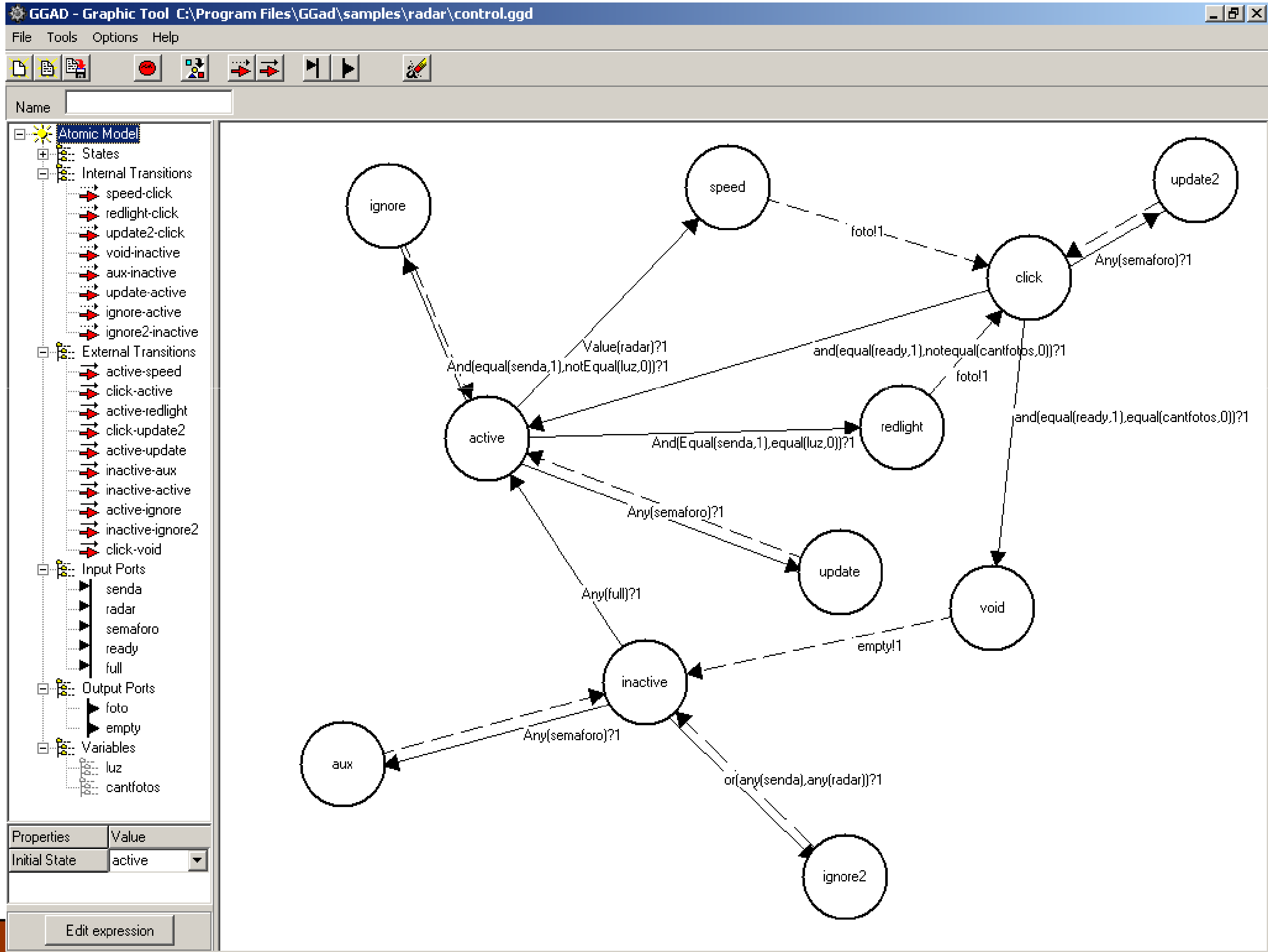
. Distributed/Parallel

. Real-Time

Auto-Factory DEVS model



DEVS Graphs Modeling environment



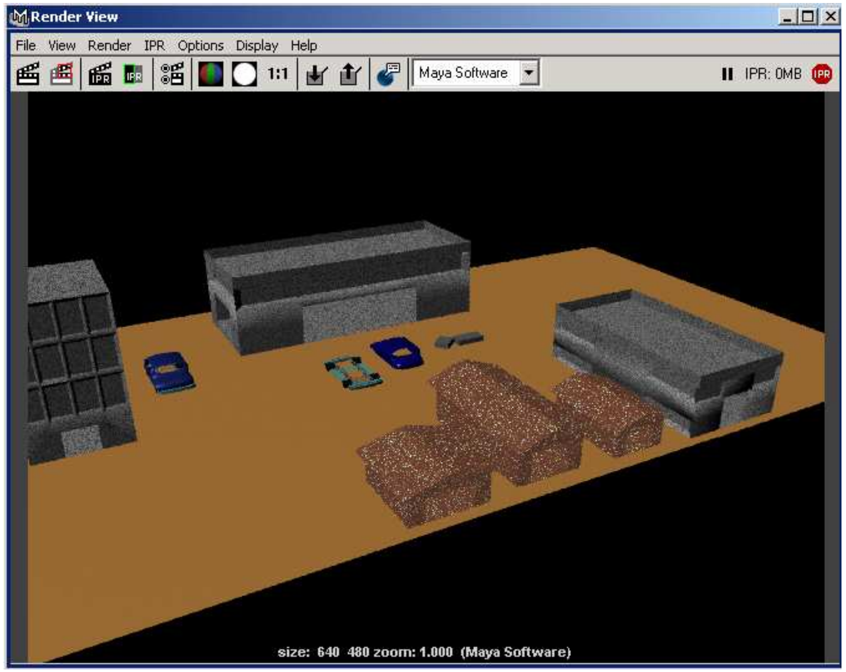
Engine Assembly Atomic

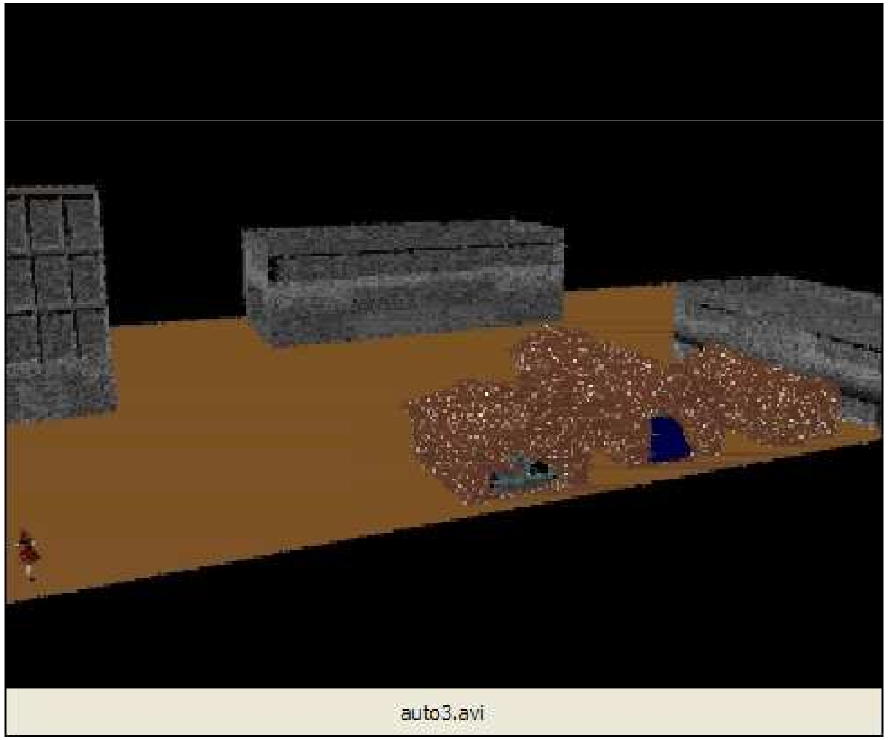
```
Model EngineAssem::EngineAssem(const string &name):Atomic(name),
    in_piston(addInputPort( "in_piston") ), in_engineBody(addInputPort(
    "in_engineBody") ), done(addInputPort("done") ), out( addOutputPort("out")),
    manufacturingTime( 0, 0, 10, 0 ) { } // Model constructor

Model &EngineAssem::externalFunction( const ExternalMessage &msg ) {
    if( msg.port() == in_piston ) {    // parts received one by one
          elements_piston.push_back( 1 ) ;
    if( elements_piston.size() == 1 && elements_engineBody.size()>=1)
          holdIn(active, manufacturingTime );
    for(int i=2;i<=msg.value;i++) //pushback if more than 1 received
            elements_piston.push_back( 1 ) ;
    }
    if( msg.port() == in_engineBody )   {         ...
}

Model &EngineAssem::internalFunction( const InternalMessage & ) {
          passivate();
}

Model &EngineAssem::outputFunction( const InternalMessage &msg ) {
    sendOutput( msg.time(), out, elements.front());
}
```

Auto Factory execution

```
X/00:000/top/in/2 to chassis
X/00:000/top/in/2 to body
X/00:000/top/in/2 to trans
X/00:000/top/in/2 to enginesubfact
D/00:000/chassis/02:000 to top
D/00:000/body/02:000 to top
D/00:000/trans/02:000 to top
X/00:000/enginesubfact/ in/2 to piston
X/00:000/enginesubfact/ in/2 to enginebody  ...
Y/02:000/chassis/out/1 to top
D/02:000/chassis/... to top
X/02:000/top/done/1 to chassis
X/02:000/top/in_chassis/1 to finalass ...
*/02:000/top to enginesubfact
*/02:000/enginesubfact to enginebody
Y/02:000/enginebody/out/1 to enginesubfact
D/02:000/enginebody/... to enginesubfact
X/02:000/enginesubfact/done/1 to enginebody
X/02:000/in_enginebody/1 to engineassem
D/02:000/enginebody/02:000 to enginesubfact
D/02:000/engineassem/02:000 to enginesubfact ...
```

Auto Factory





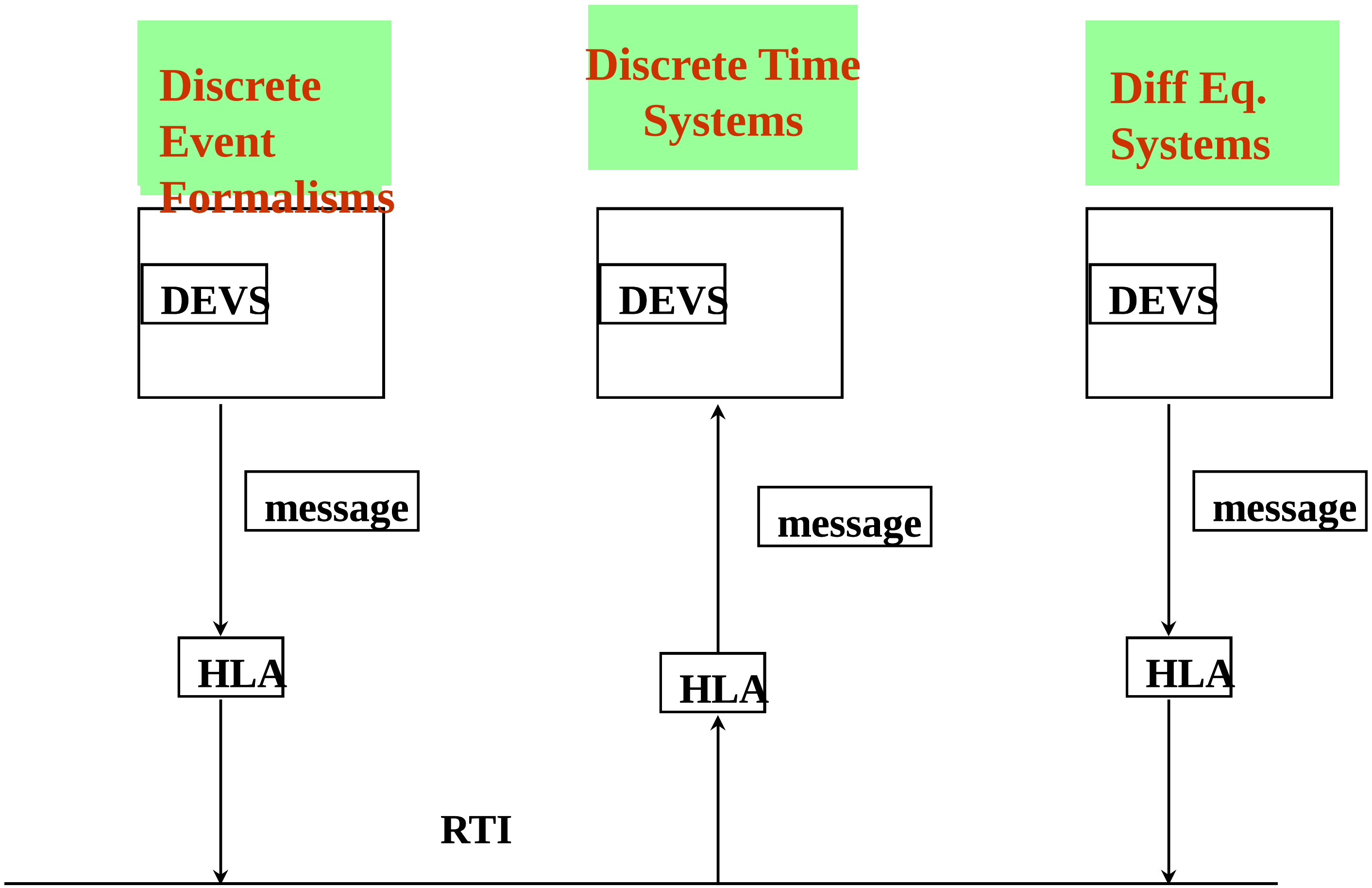
DEVS Success Stories

- Prototyping and testing environment for embedded system design (Schulz, S.; Rozenblit, J.W.; Buchenrieder, K.; Mrva, M.)
- Urban traffic models (Lee, J.K.; Lee, J-J.; Chi, S.D.; et al.)
- Watershed Modeling (Chiari, F. et al.)
- Decision support tool for an intermodal container terminal (Gambardella, L.M.; Rizzoli, A.E.; Zaffalon, M.)
- Forecast development of *Caulerpa taxifolia*, an invasive tropical alga (Hill, D.; Thibault, T.; Coquillard, P.)
- Intrusion Detection Systems (Cho, T.H.; Kim, H.J.)
- Depot Operations Modeling (B. Zeigler et al. U.S. Air Force)

DEVS Success Stories

- Supply chain applications (Kim, D.; Cao H.; Buckley S.J.)
- Solar electric system (Filippi, J-B.; Chiari, F.; Bisgambiglia, P.)
- M&S activities at JITC, AZ (B. Zeigler, J. Nutaro et al.)
- Representation of hardware models developed with heterogeneous languages (Kim, J-K.; Kim, Y.G.; Kim, T.G.)
- DEVS/HLA Research funded by DARPA received Honorable Mention in 1999 DMSO Awards

DEVS Bus Concept



UA/Lockheed distributed experimentation

JM:
- Detailed Surface Ship Models
- Sub/Surface Enemy Assets

Medusa:

Hi Fidelity Radar / Weapon Scheduling

Space Manager and Logger:

Pragmatic Event Cue Emission Propagation (with acoutics)

JM:
- Space Based Sensors
- Space Based Communication
- Land/Air Enemy Assets

Space Manager and Logger:

Pragmatic Event Cue Emission Propagation

DEVS/HLA
- quantization
- predictive filtering
- GIS/aggregation

LMGES -- NJ

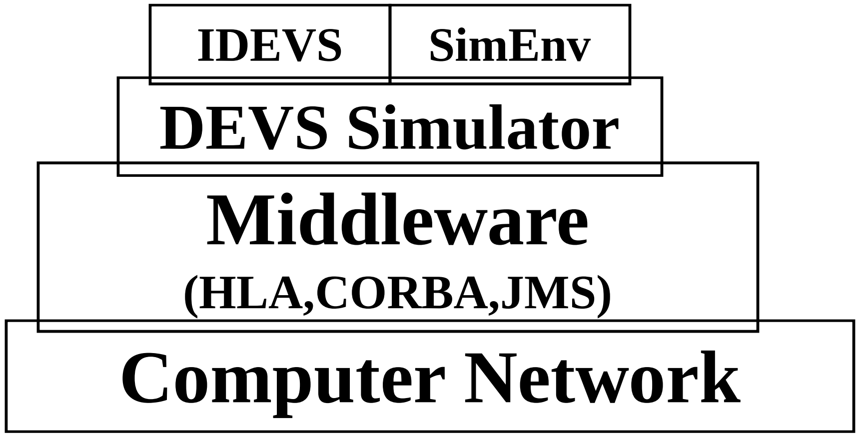
LMMS -- CA

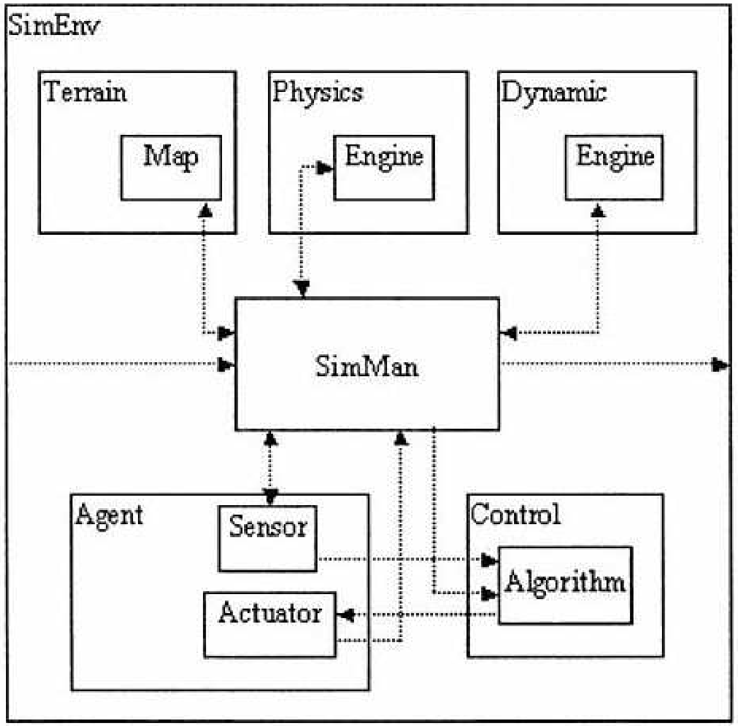
Component Model Reuse Matrix

Project / Model	Critical Mobile Target	Global Positioning System III	Arsenal Ship	Coast Guard Deep Water	Space Operations Vehicle	Common Aero Vehicle	Joint Composite Tracking Network	Integrated System Center	Space Based Laser	Space Based Discrimination	Missile Defense (Theater / National)
Radar Model	X		X	X	X	X	X				X
IR Sensor Model	X				X		X	X	X	X	X
Missile Model			X				X	X	X		X
Laser Model								X	X	X	X
Comm. Model	X			X		X	X	X			X
Command Control Model	X		X								X
Earth & Terrain Model	X	X	X		X						X
Weather Model	X										X
Waypoint & Heading Nav Model	X	X	X	X	X		X				X
Orbital Propagate Model	X	X			X			X	X	X	X
Ballistic Trajectory Model			X		X	X				X	X

U. of New Mexico Virtual Lab for Autonomous Agents

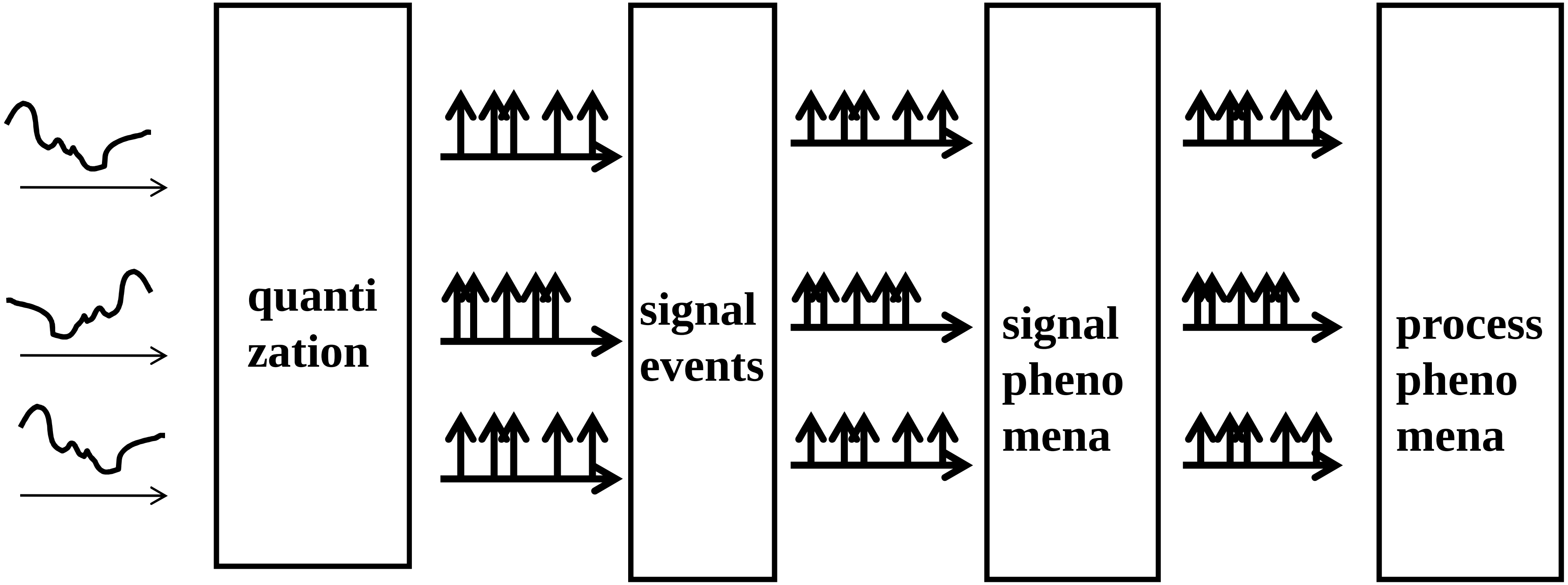
V-Lab: DEVS M&S environment for robotic agents with physics, terrain and dynamics (Mars Pathfinders for NASA).





Reported gains in development times thanks to the use of DEVS

DEVS framework for control of steel production



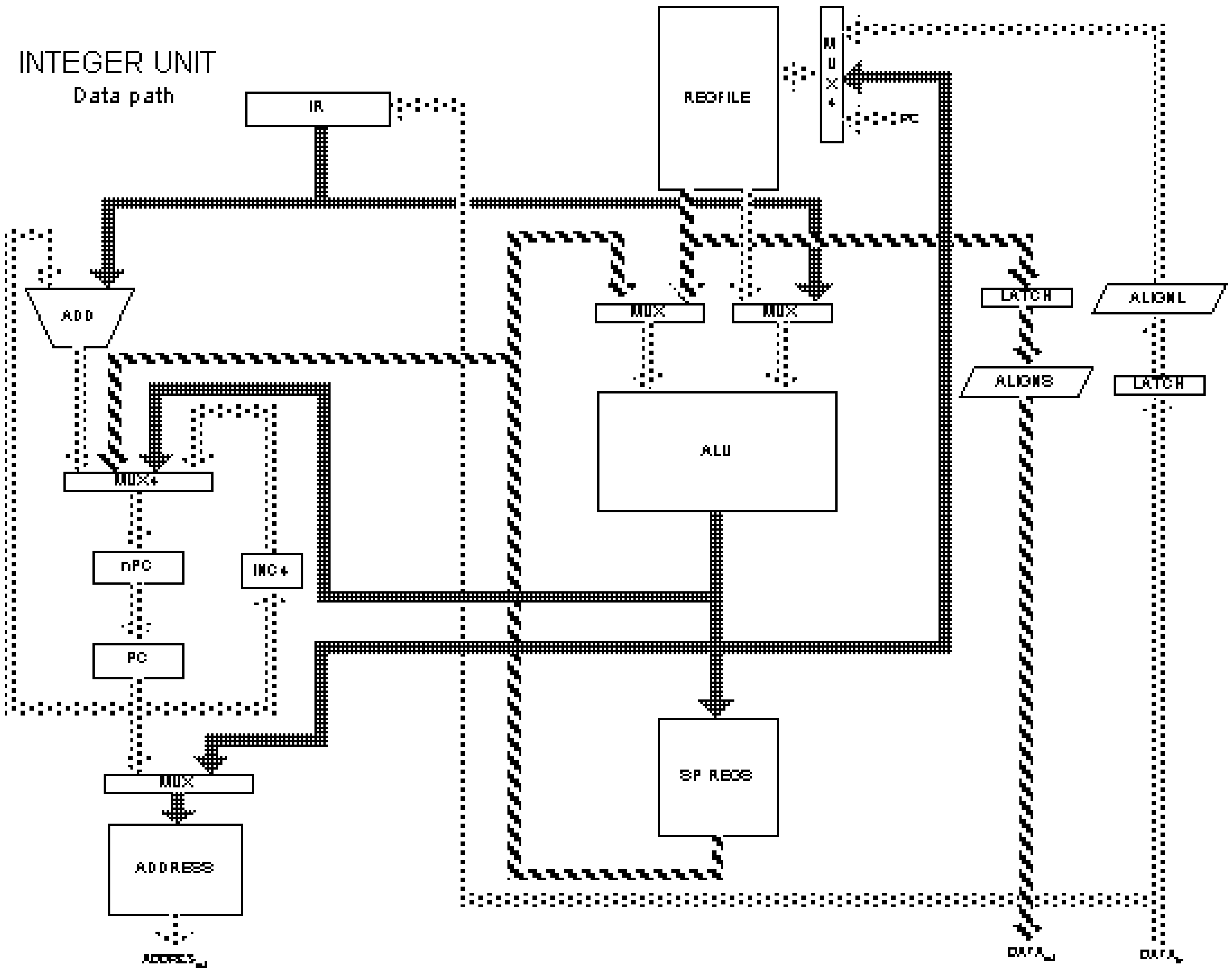
Large Scale:

- **Conceptual model contains 25,000 objects for 33 goals, 27 tasks,etc.**
- **Approximately 400,000 lines of code.**
- **14 man-years: 6 knowledge engineers and 12 experts**

One advantage of DEVS is *compactness*: high reduction in data volume

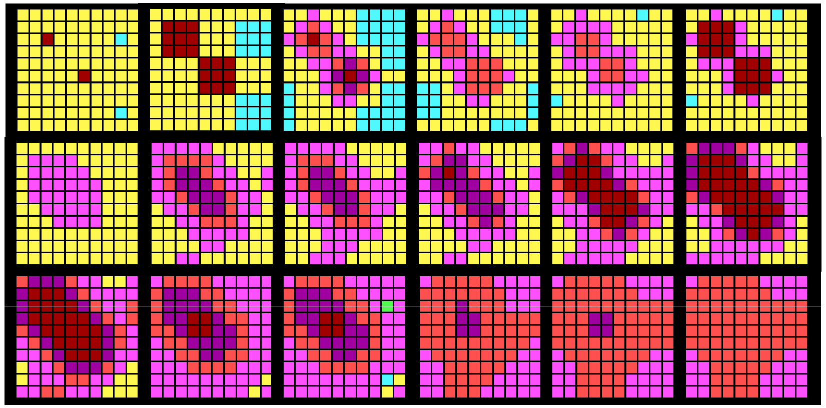
Effective analysis and control of the behavior of blast furnaces at high resolution

α-1 simulated computer

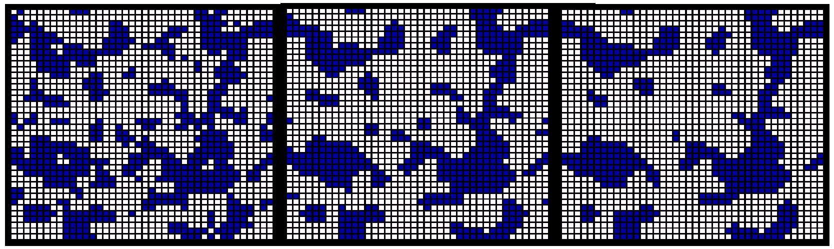


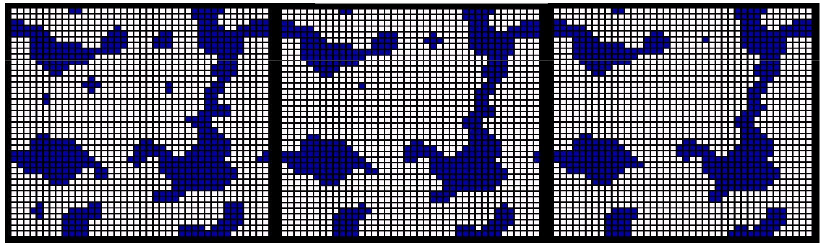
Physical Systems

Heat Spread

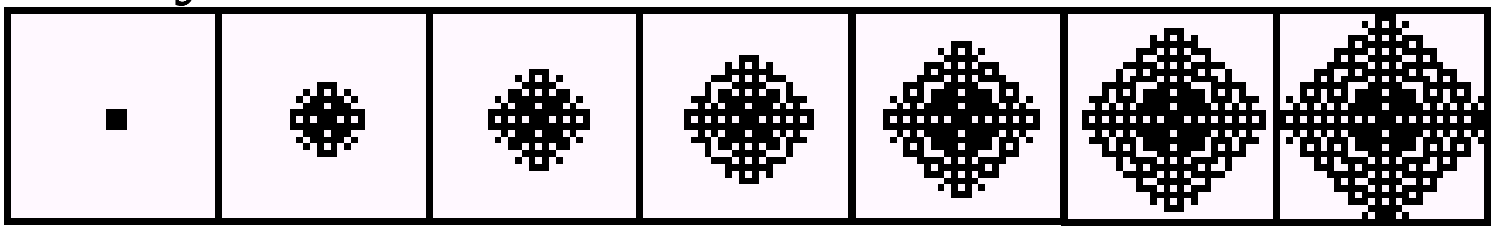


Surface Tension

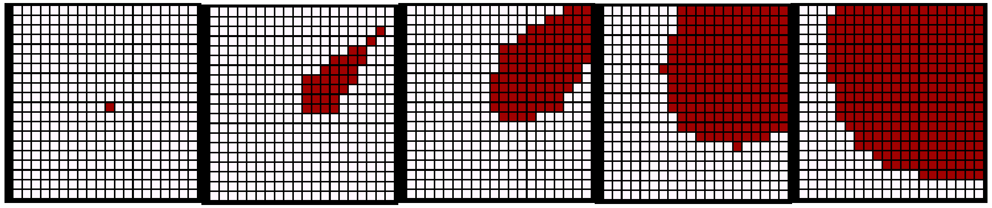


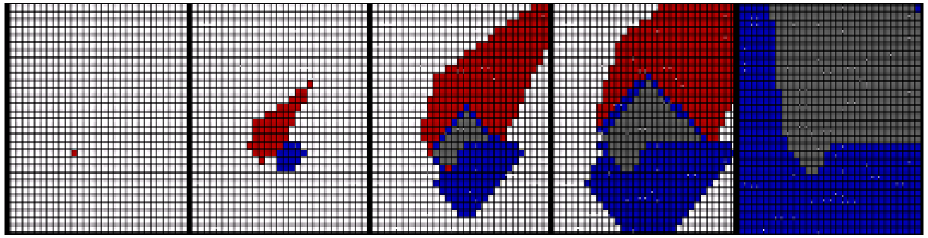


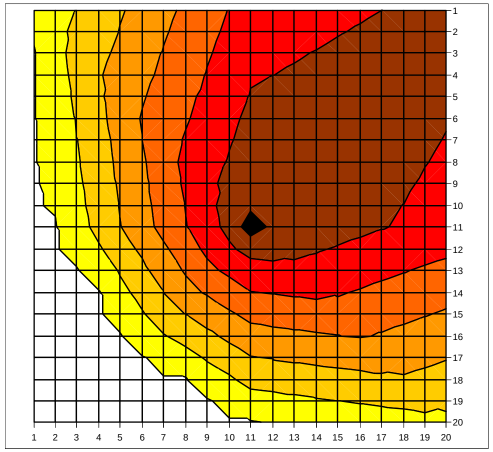
- Binary solidification

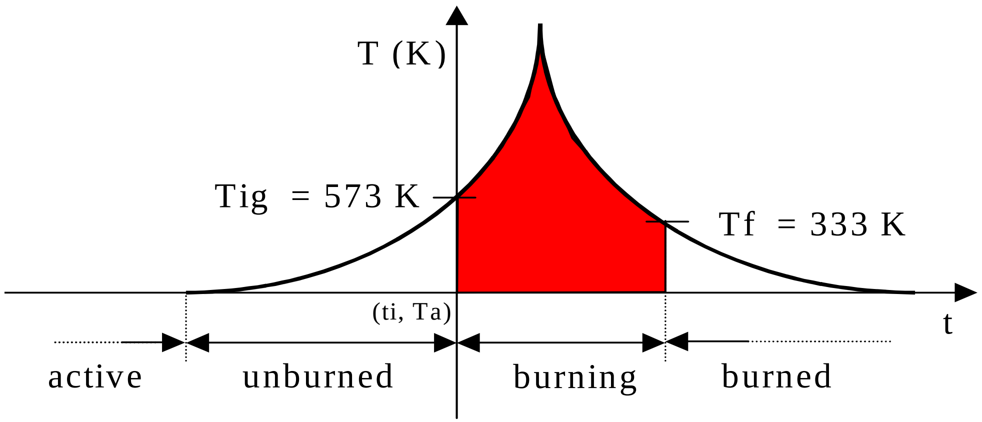


Fire Spread Modeling

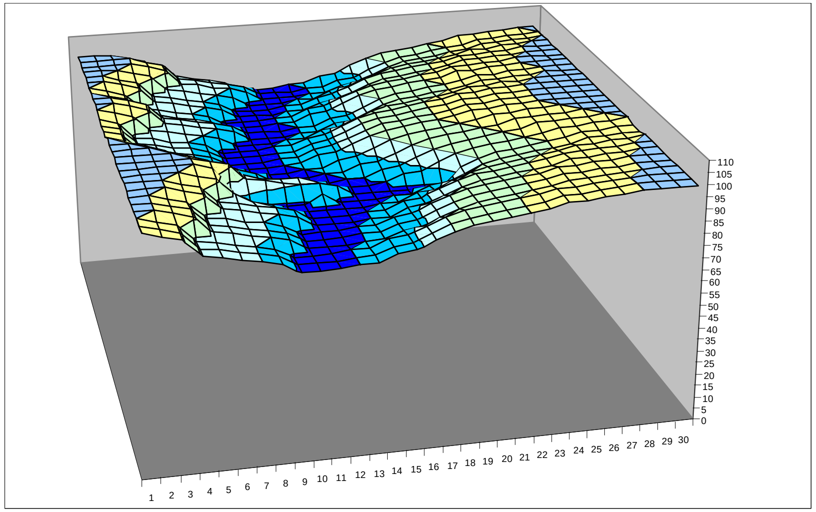


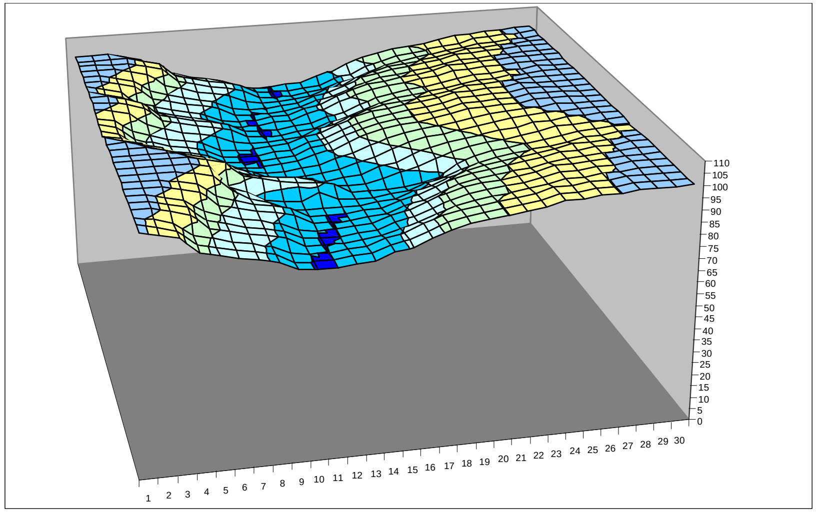


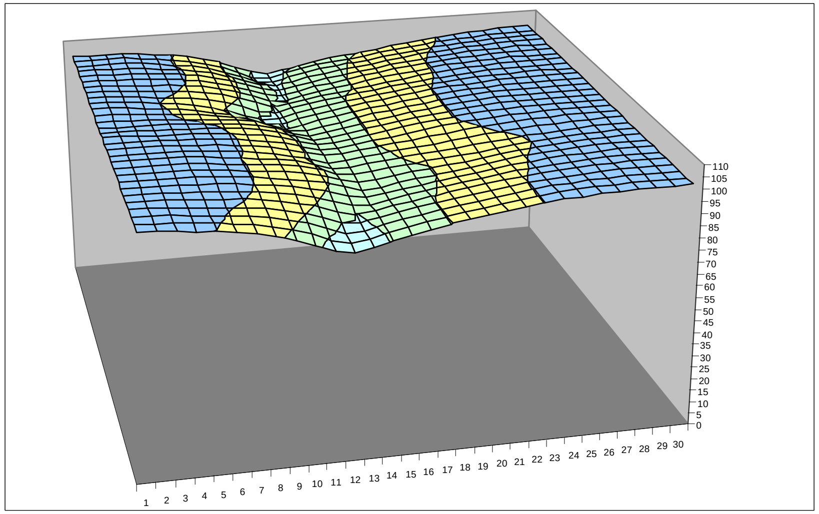


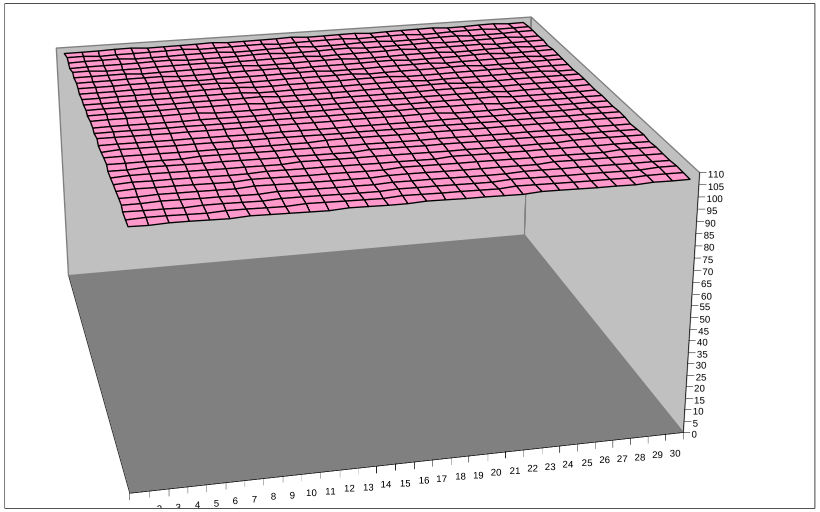


Watershed modeling



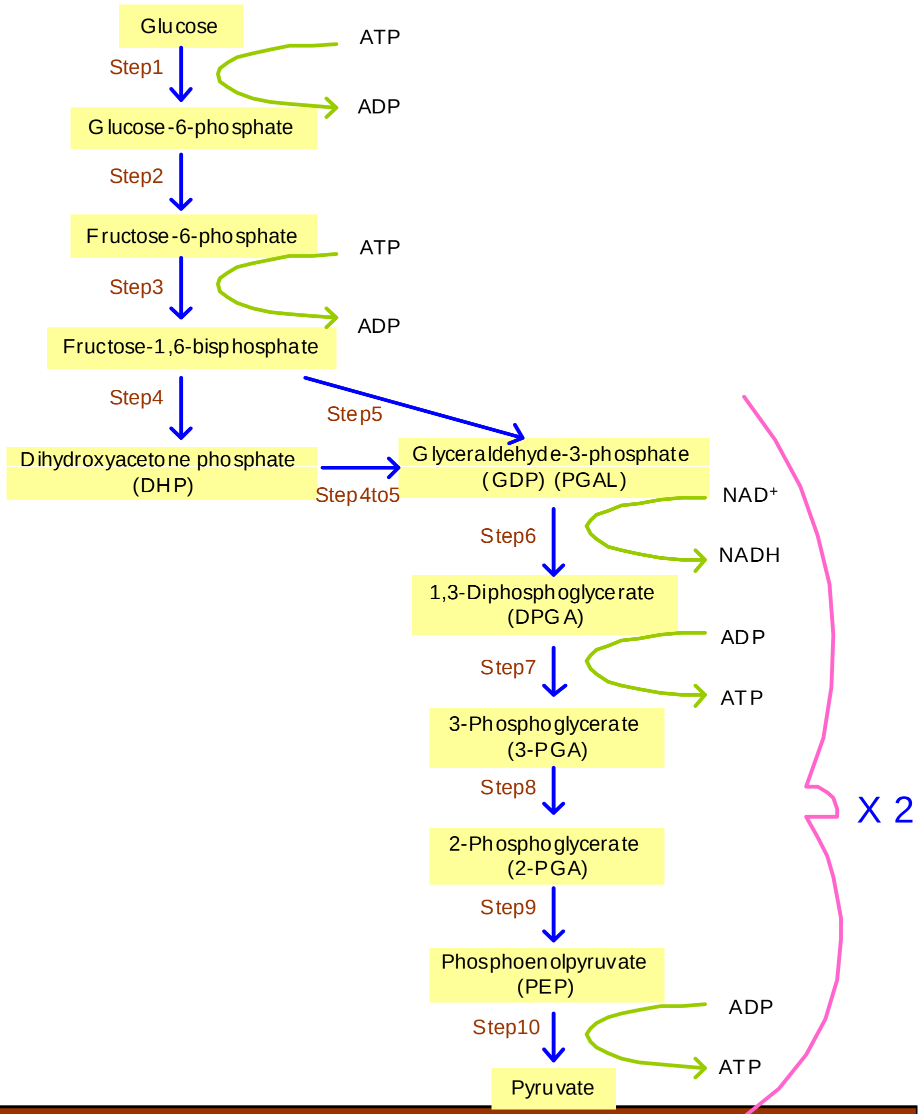




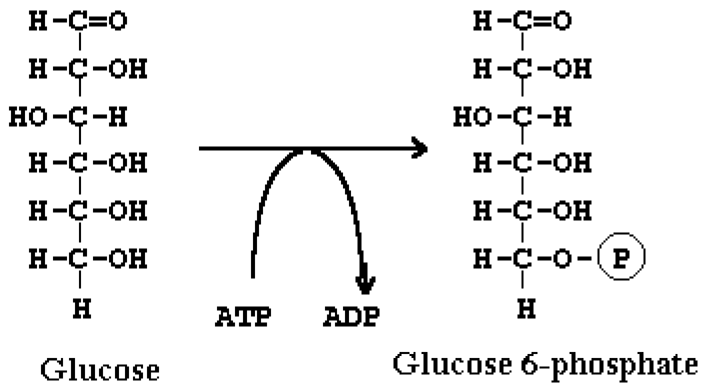


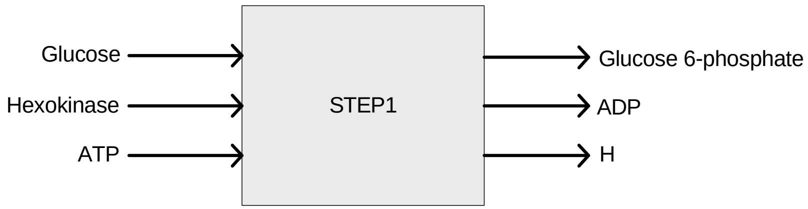
Metabolic pathways: Glycolisis

- Sequence of reactions used by cells to metabolize glucose
- Role: to produce energy
- Glycolysis generates about 15% of energy produced by aerobic respiration
- A sequence of ten reactions
- Converts one glucose molecule into two pyruvate molecules
- Produces NADH and ATP.
- Specific enzymes control each of the different reactions.



Step 1 Atomic Model





$Step1 = < S, X, Y, \delta_{int}, \delta_{ext}, ta, \lambda >$

S = {atpc, glucosec, ifhex, counter, phase, sigma}

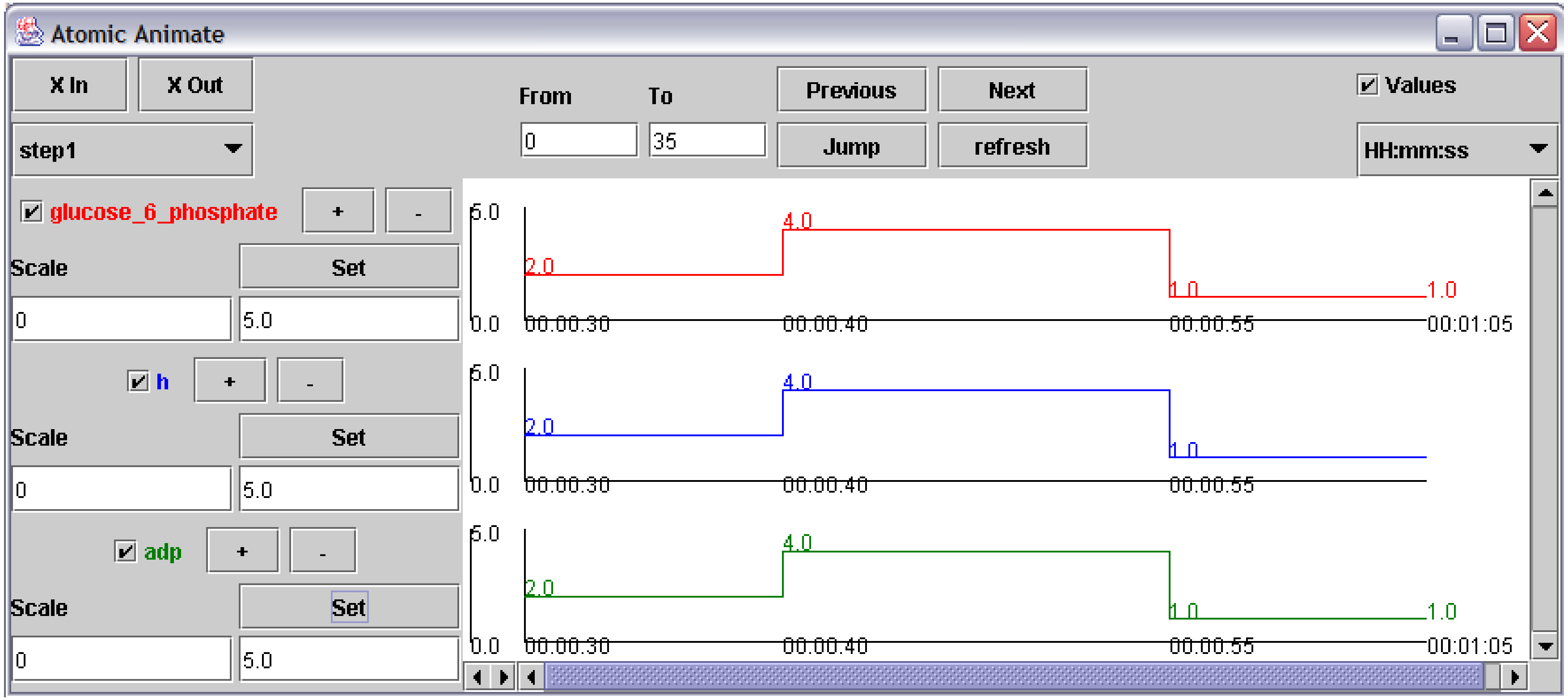
X = {glucose, ATPi, hexokinase}

Y = {glucose_6_phosphate, ADP, H}

δ_{int}, δ_{ext}, *ta* and λ using CD++ implementation.

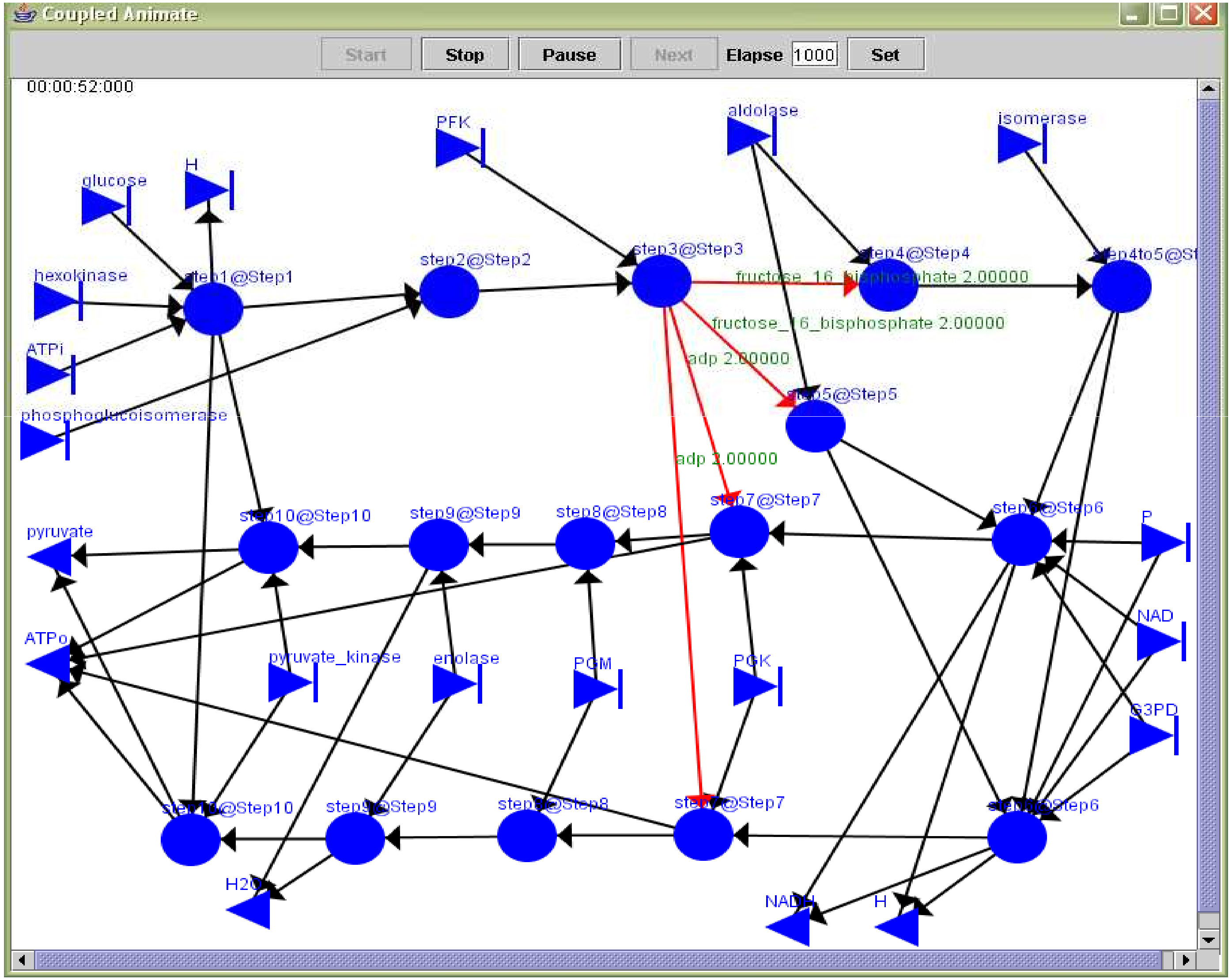
```
Model &Step1::externalFunction
        ( const ExternalMessage &msg ) {
  if( msg.port() == glucose ) {
      glucosec = glucosec + msg.value() ;
      if ( (atpc > 0 ) && (ifhex == true))
            holdIn( active, Prep_Gluc );
  }
  else if( msg.port() == ATPi ) {
      atpc = atpc + msg.value() ;
      if ( (glucosec>0) && (ifhex==true) )
            holdIn( active, Prep_ATPi );
  }
  else if ( msg.port() == hexokinase ) {
      ifhex = true ;
      if ( (glucosec > 0 ) && (atpc > 0) )
            holdIn( active, Prep_Hexo );
  }
}
```

Step 1 Test

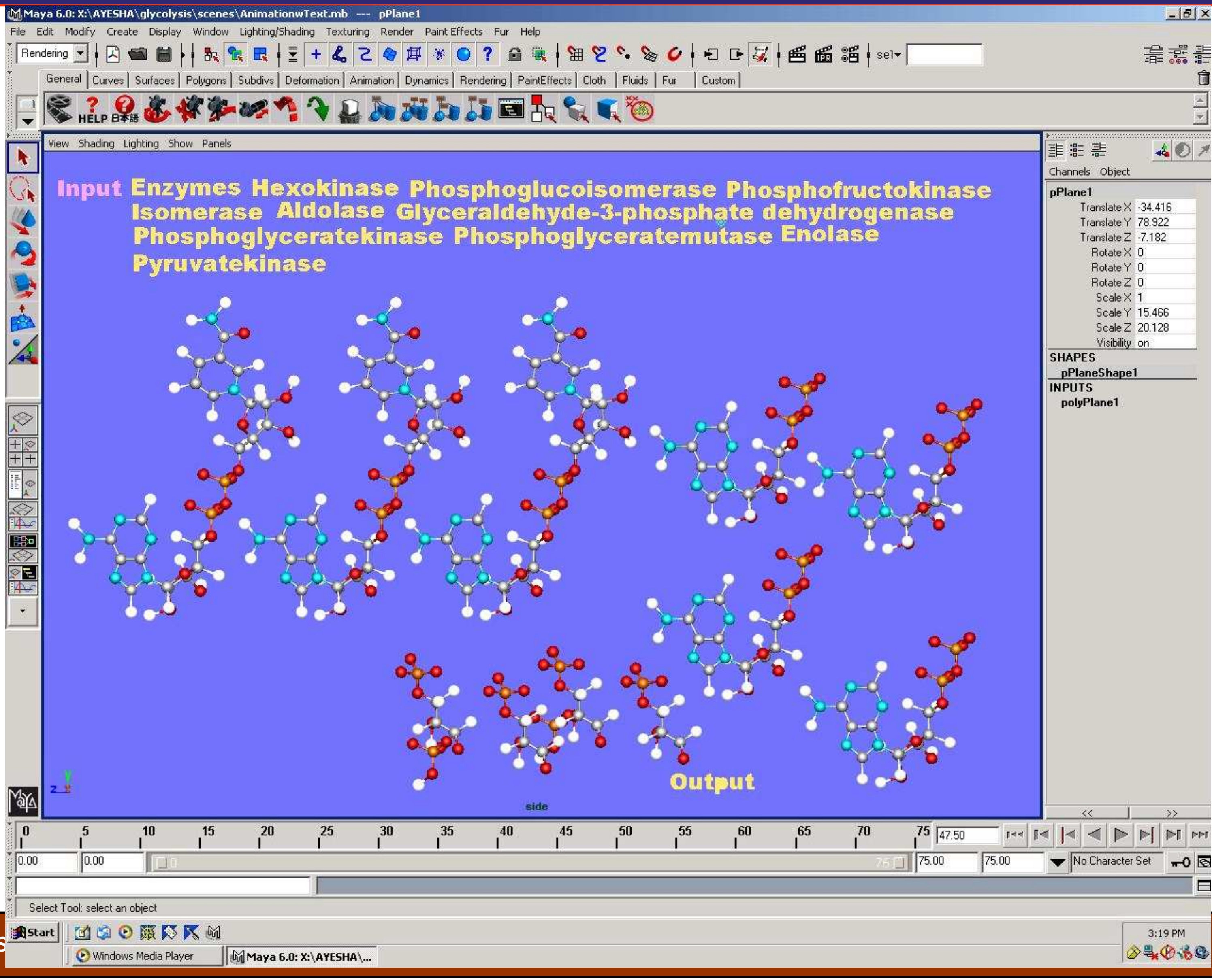


At time 40:00: after four *glucose* molecules entered the cell; four more outputs of each of the *ADP*, *glucose_6_phosphate*, and *H* molecules.

Coupled Animation of Glycolysis

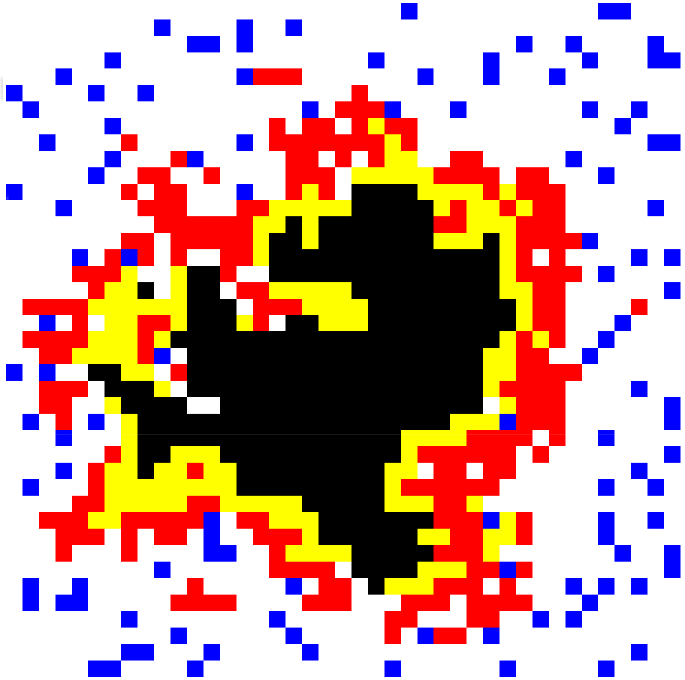


Glycolisis 3D visual results



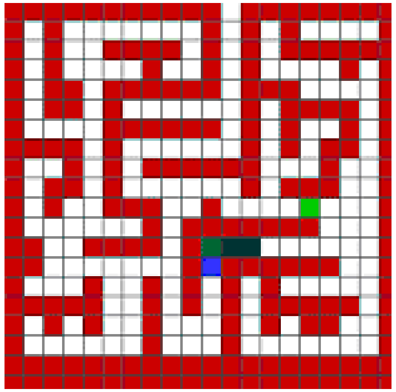
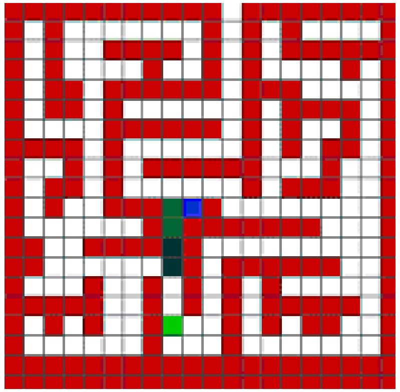
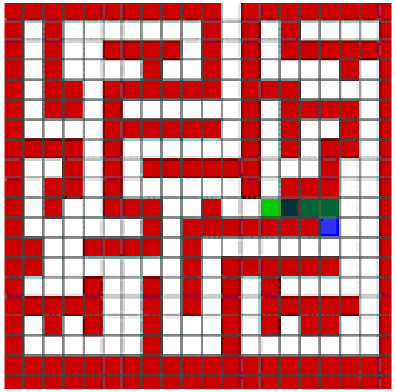
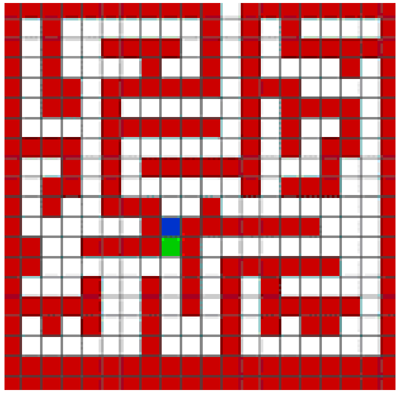
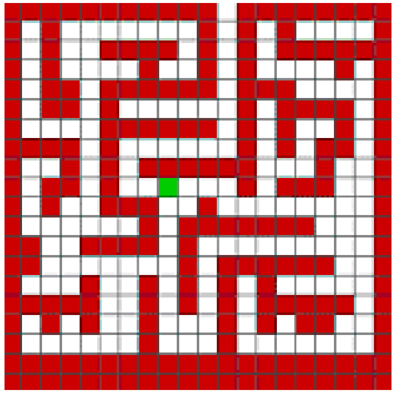
Tumor Victory

After 38 times steps, the immune cells have cleared away proliferative cells on the north side of the tumor.



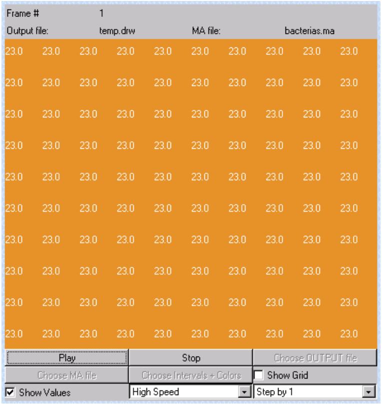
After 64 times steps, the tumor overwhelms the immune system.

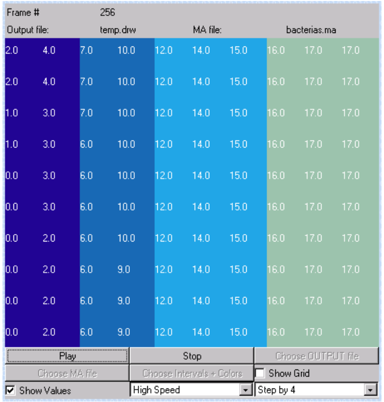
Pursuer/evader modeling

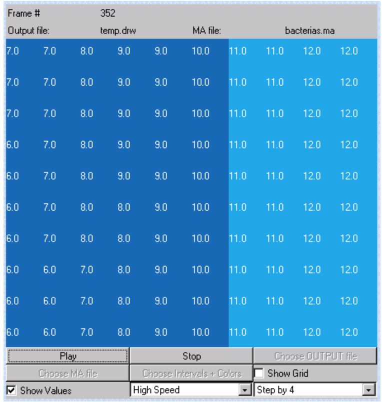






Vibrio Parahaemolyticus bacteria

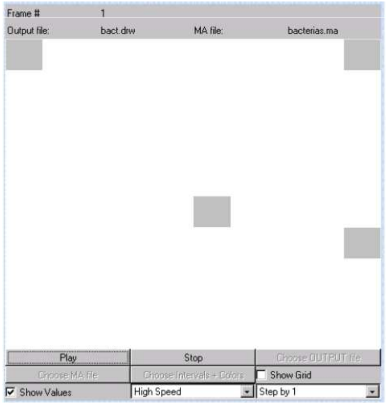
Temperature

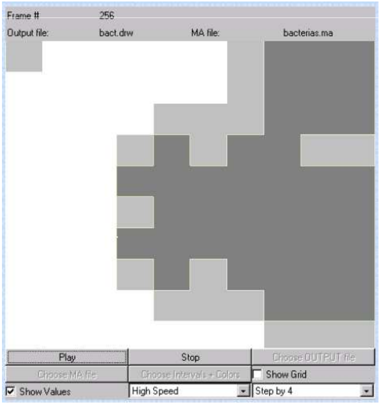


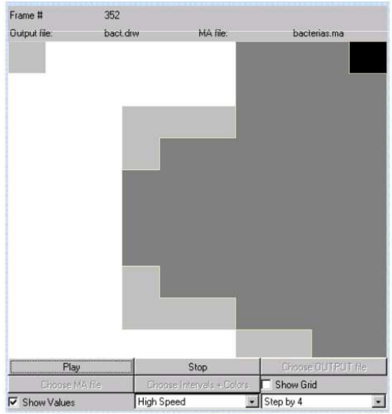




Bacteria concentration

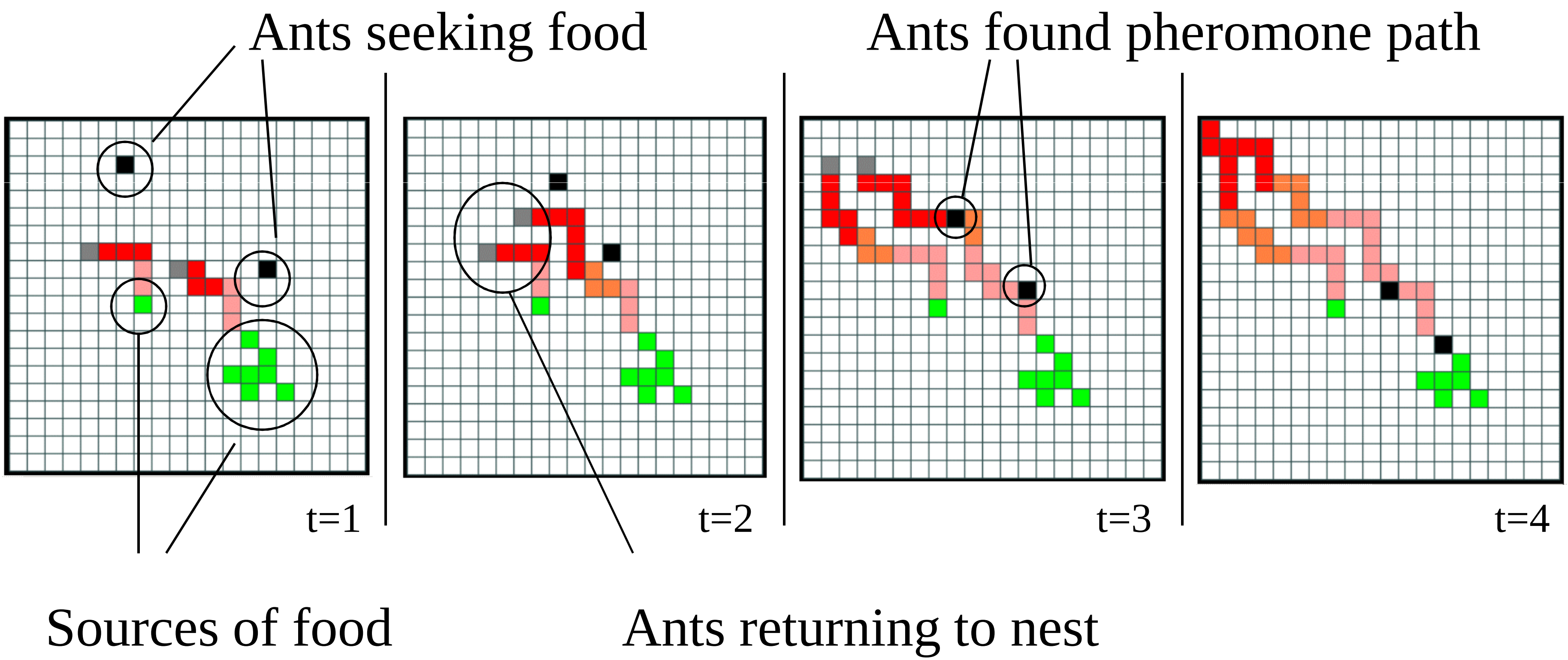




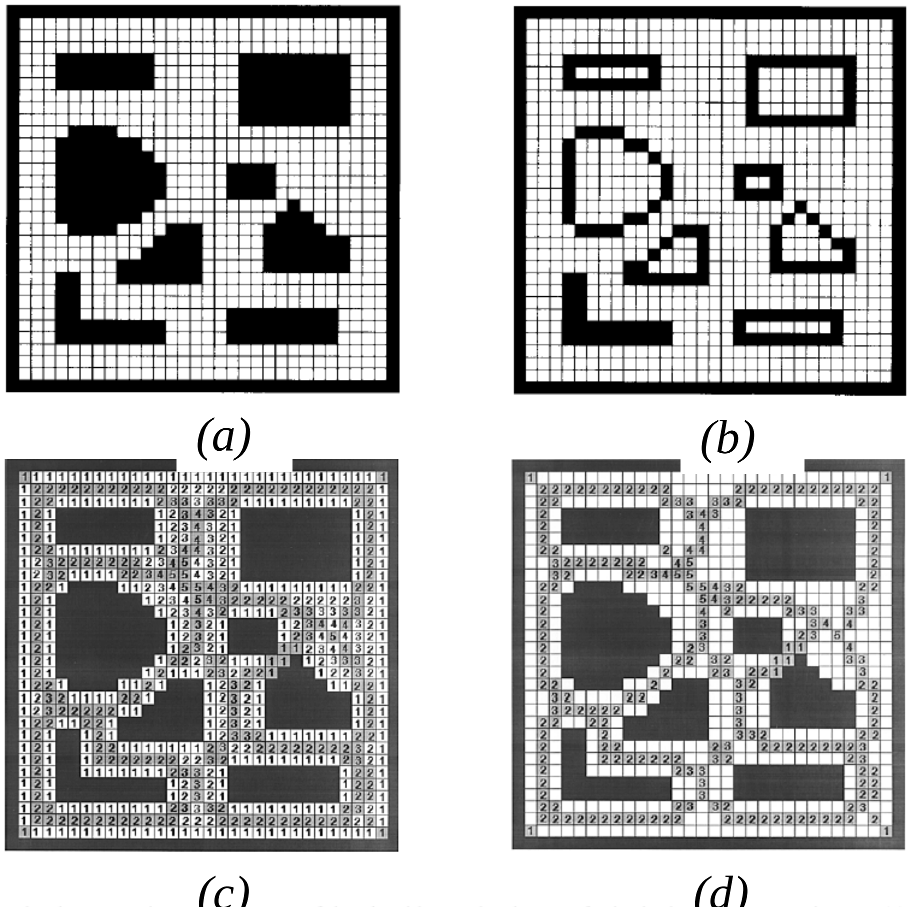


Initial | After 1.5 hr | After 4 hrs

Ants following pheromone paths

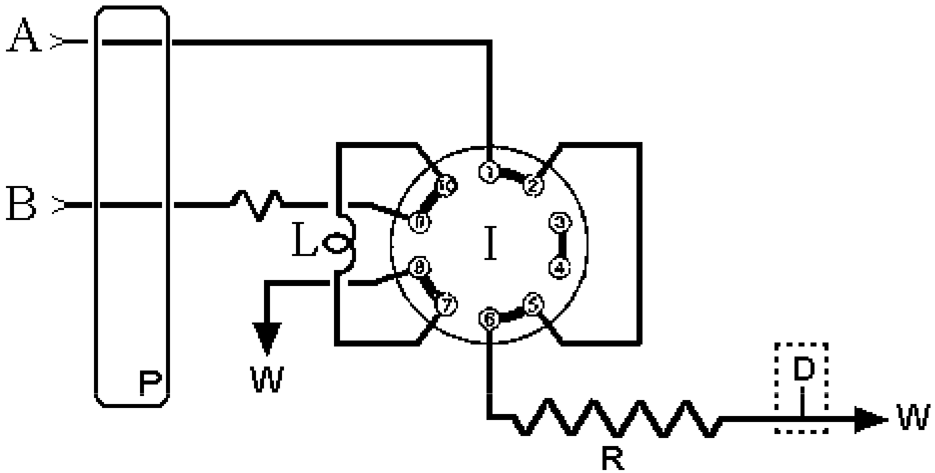


Path Planning Evolution



<u>Different phases of the algorithm:</u> *(a)* Configuration of obstacles, *(b)* Boundary detection, *(c)* Information for CA Expansion, *(d)* Optimal collision-free path

Flow Injection Analysis (FIA)

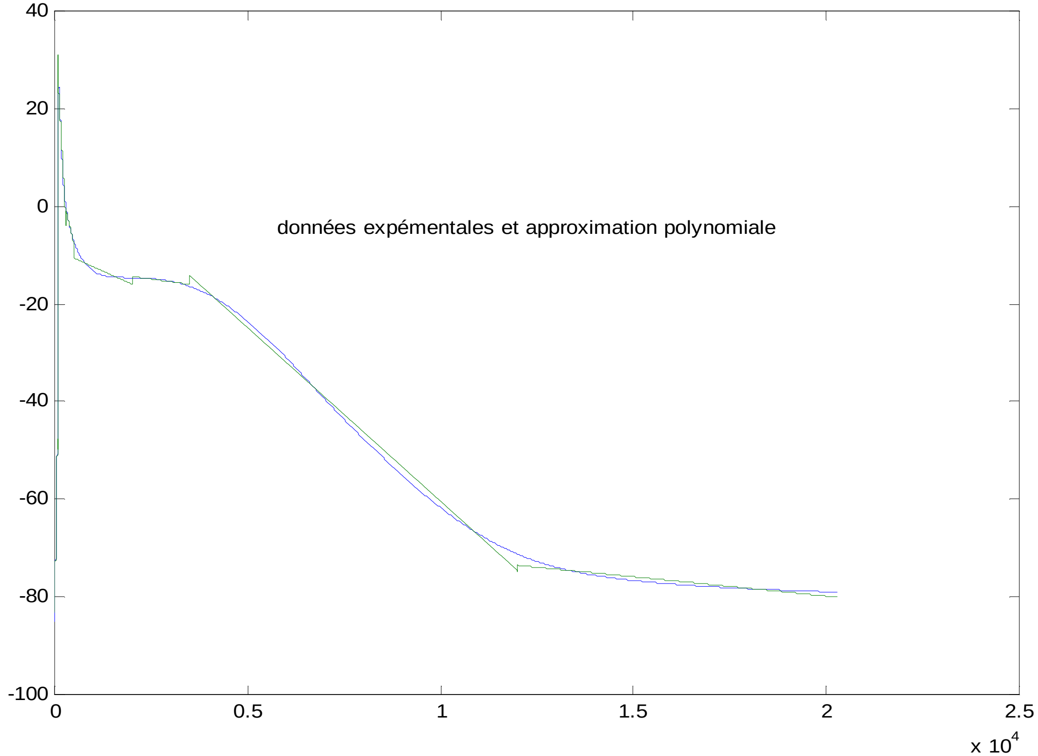


FIA manifold. P: pump; A,B: carrier and reagent lines; L: sample injection; I: injection valve; R: reactor coil; D: flow through detector; W: waste line.

- P pumps carrier solution A into valve I that connects to reactor R
- By turning valve I, sample B is injected into R
- Reactions in R between A and B are sensed by detector D

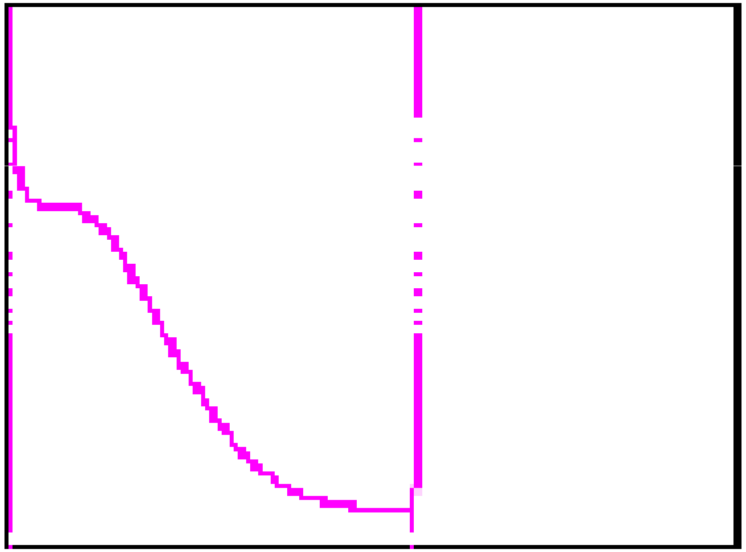
Heart tissue behavior

- Heart muscle excitable; responds to external stimuli by contracting muscular cells.
- Equations defined by Hodgkin and Huxley
- Every cell reproducing the original equations
- Discrete time
- Discrete event approximation
- G-DEVS, Q-DEVS

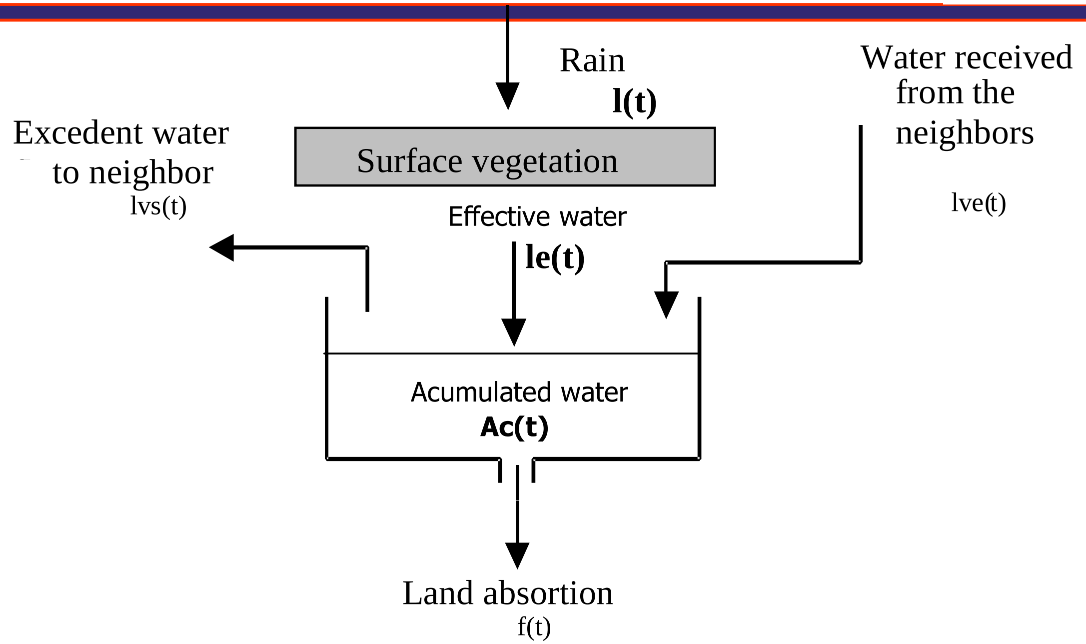


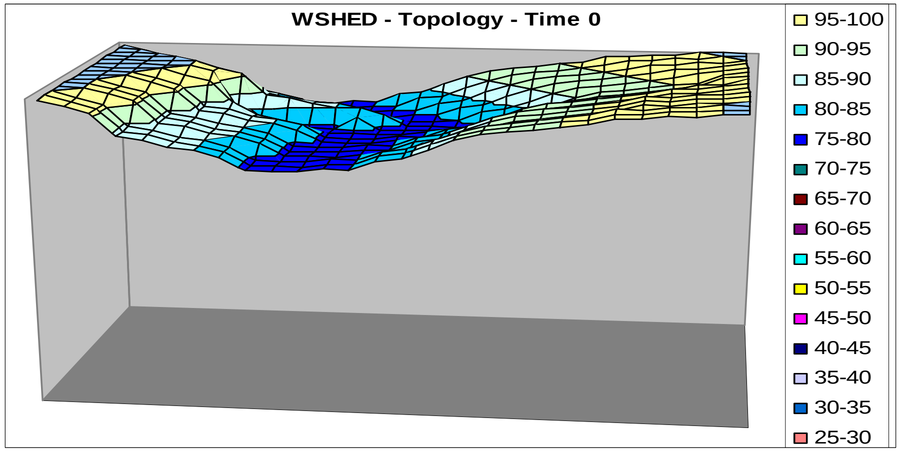
Test cases: a heart tissue model

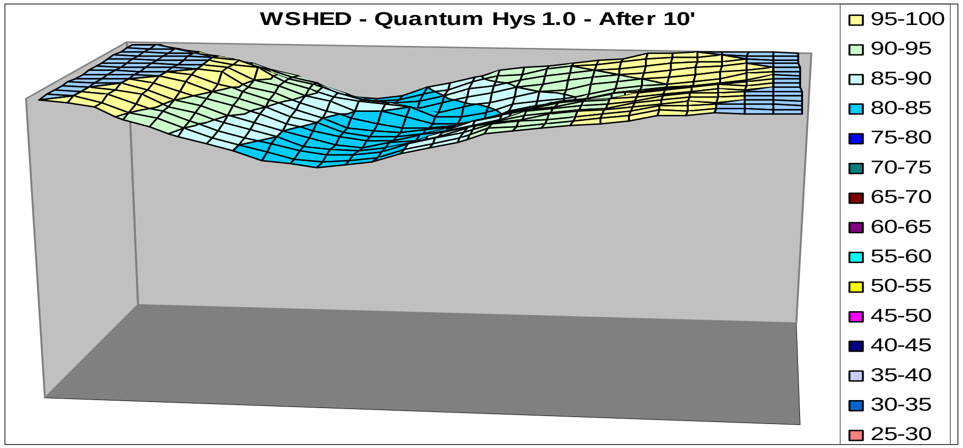
- Automated discretization of the continuous signal



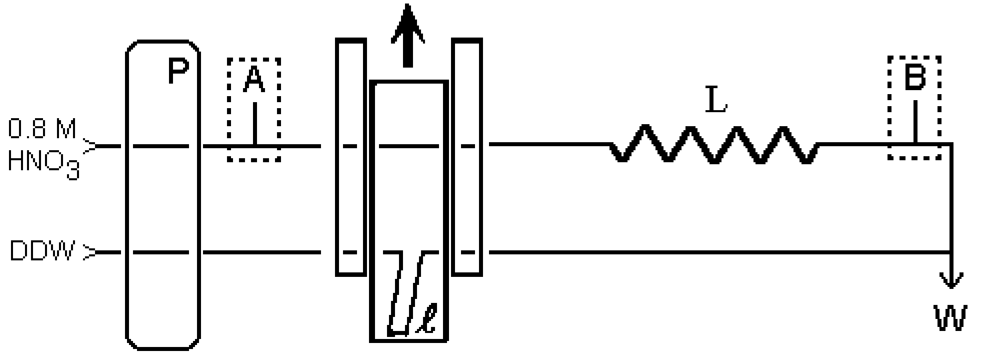
A Watershed model

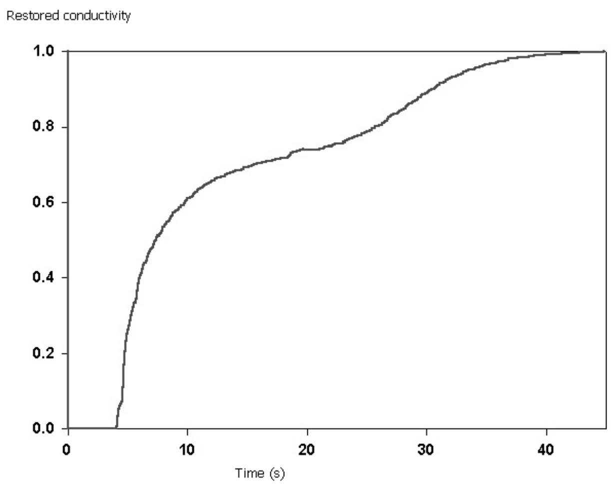


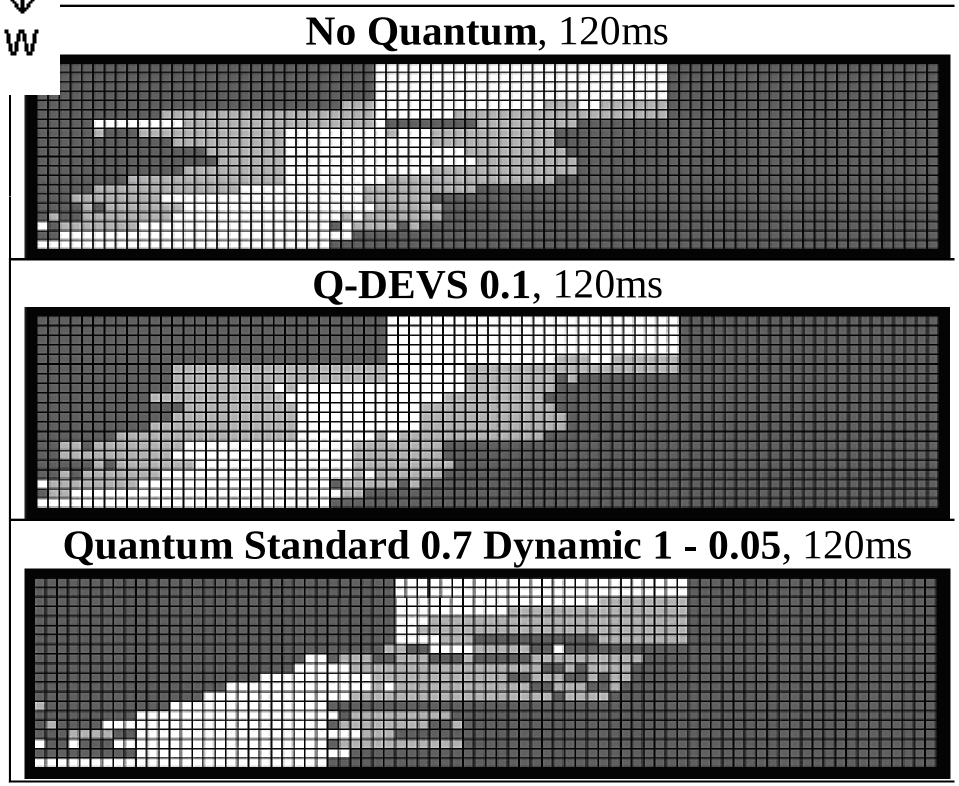




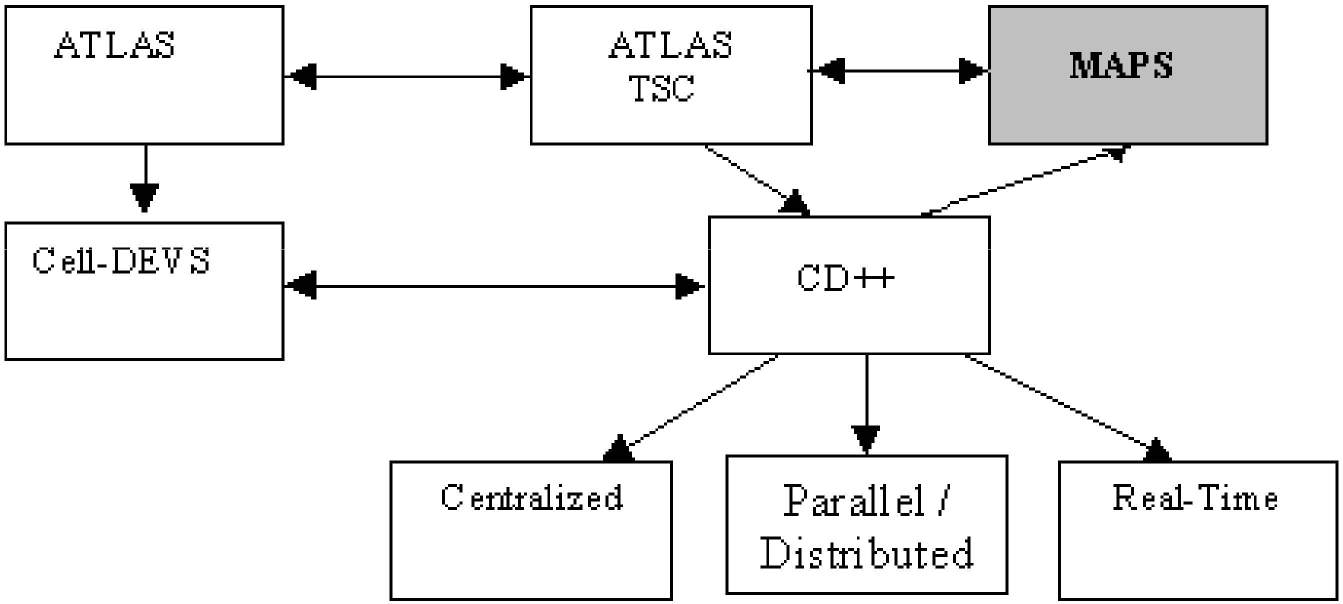
Flow Injection Analysis Model



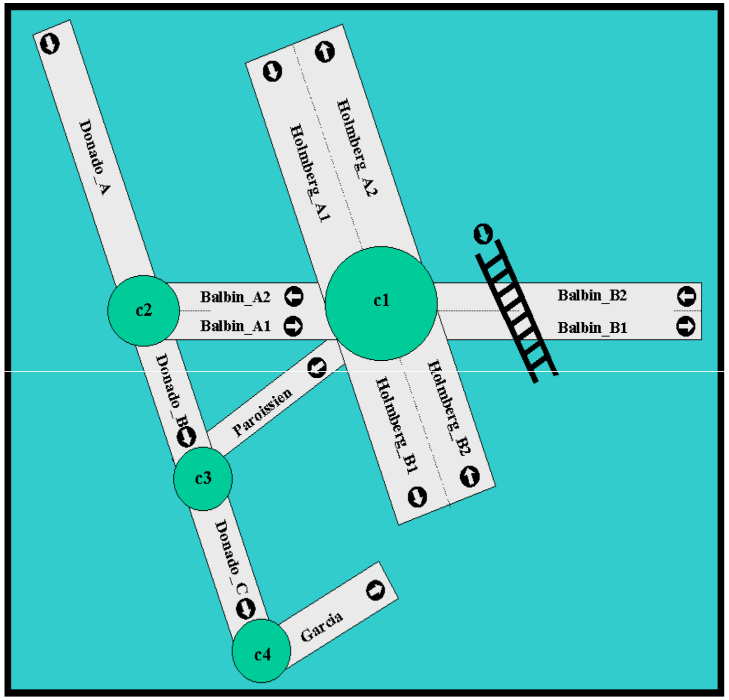




ATLAS SW Architecture

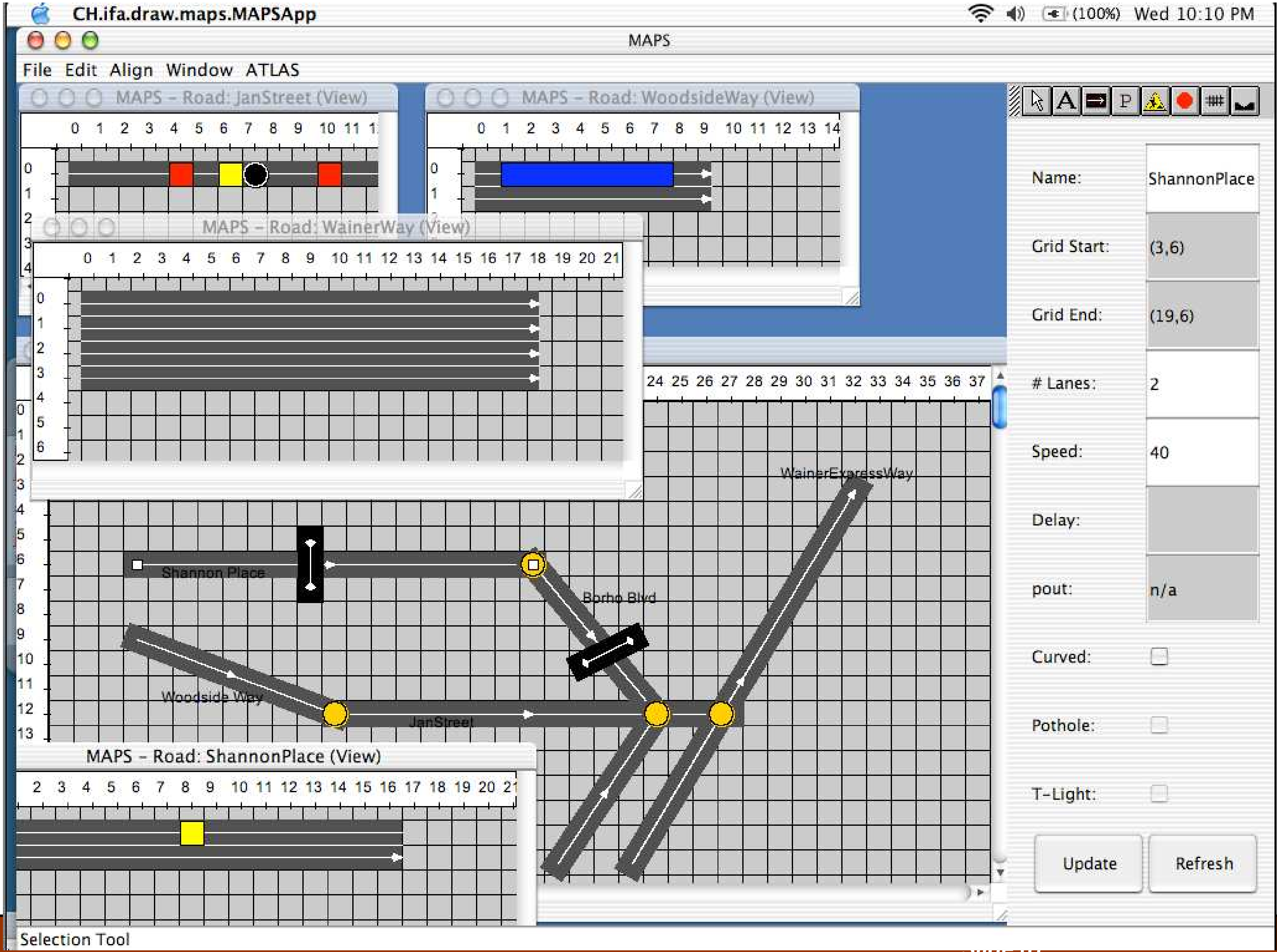


Modelling a city section

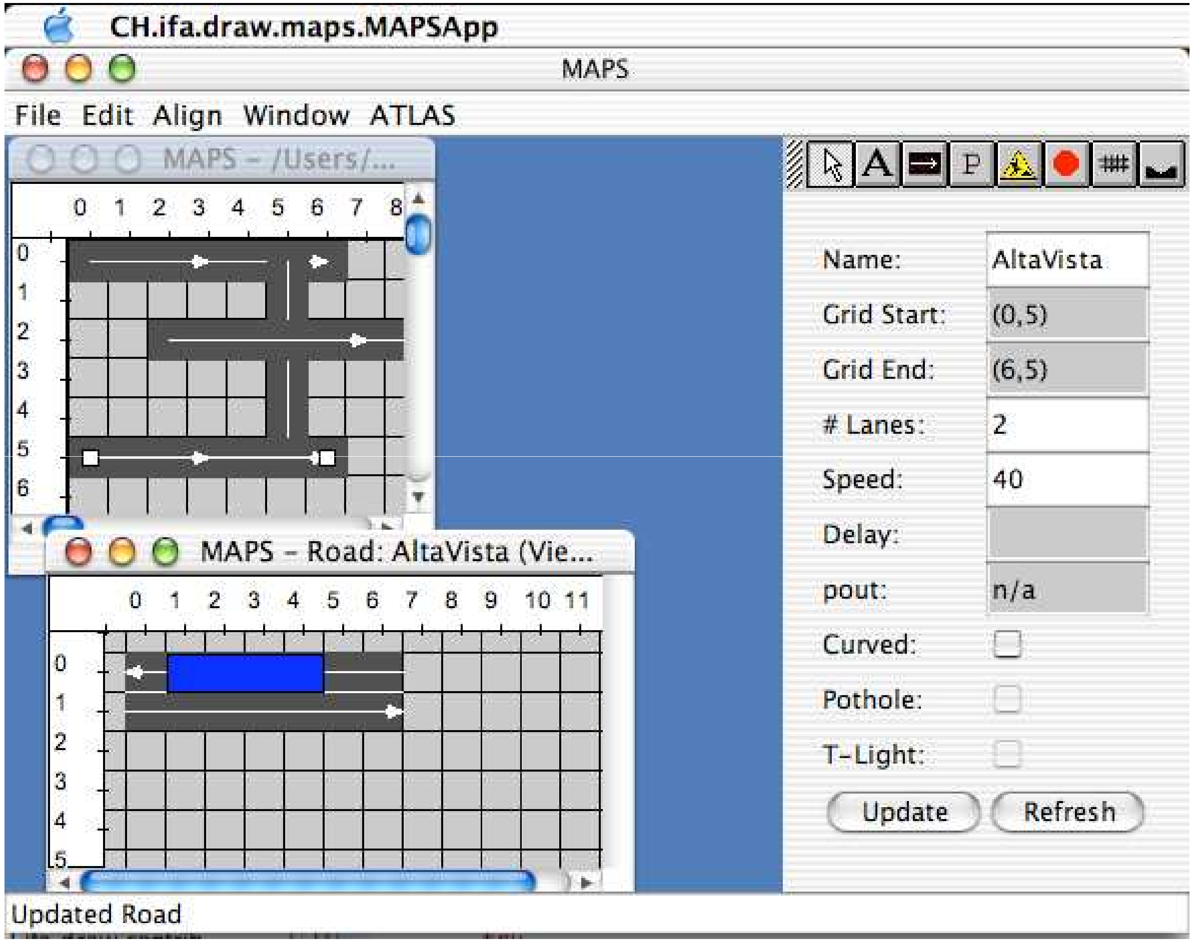


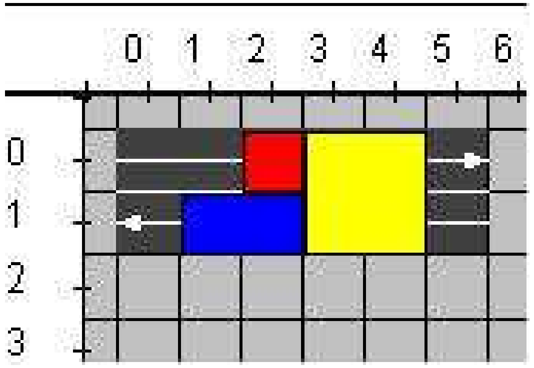
- **24-line specification**
- **1000 lines of CD++ specifications automatically generated**

Describing a city section



Defining a city section in MAPS





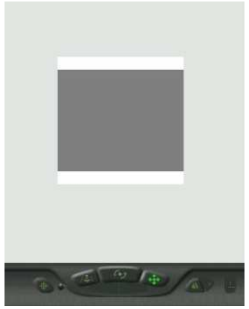
Exporting to TSC

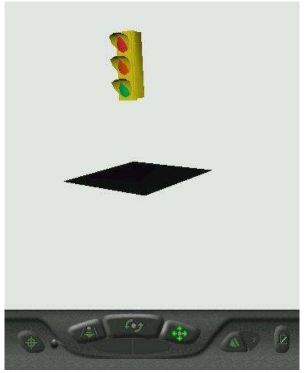
Carleton UNIVERSITY

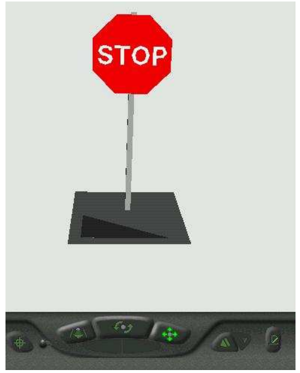
Generating ATLAS Code...

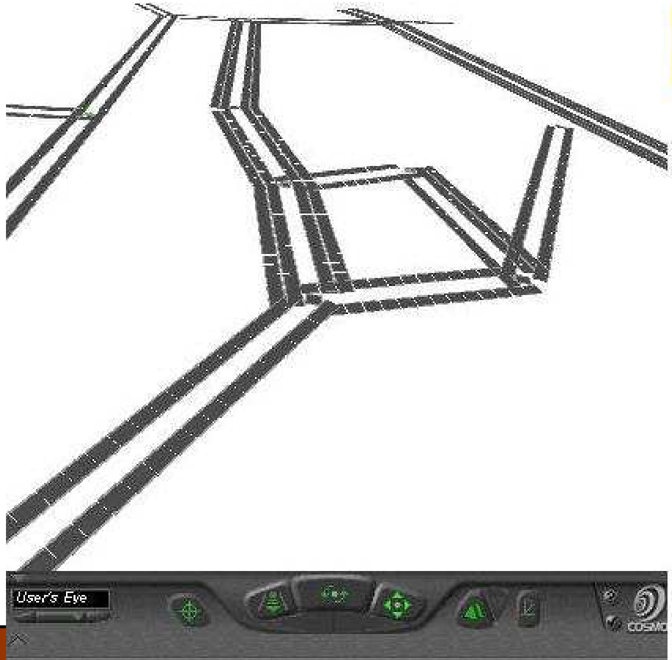
```
begin segments
ShannonPlaceGOS1=(3,6),(19,6),1,straight,go,40,0,parkNone
ShannonPlaceBACKS1=(3,6),(19,6),1,straight,back,40,0,parkNone
BorhoBlvdGOS1=(19,6),(24,12),1,curve,go,55,0,parkNone
BorhoBlvdGOS2=(24,12),(24,12),1,curve,go,55,0,parkNone
BorhoBlvdBACKS1=(19,6),(24,12),1,curve,back,55,0,parkNone
BorhoBlvdBACKS2=(24,12),(24,12),1,curve,back,55,0,parkNone
JanStreetGOS1=(11,12),(24,12),1,straight,go,55,0,parkNone
JanStreetGOS2=(24,12),(24,12),1,straight,go,55,0,parkNone
JanStreetGOS3=(24,12),(26,12),1,straight,go,55,0,parkNone
JanStreetGOS4=(26,12),(27,12),1,straight,go,55,0,parkNone
WainerWayGOS1=(23,18),(26,12),4,straight,go,55,0,parkNone
WainerWayGOS2=(26,12),(32,3),4,straight,go,55,0,parkNone
WoodsideWayGOS1=(3,9),(4,9),2,straight,go,70,0,parkNone
WoodsideWayGOS2=(4,9),(10,11),2,straight,go,70,20,parkLeft
WoodsideWayGOS3=(10,11),(11,12),2,straight,go,70,0,parkNone
PittnerPlazaGOS1=(20,18),(24,12),2,curve,go,90,0,parkNone
PittnerPlazaGOS2=(24,12),(24,12),2,curve,go,90,0,parkNone
end segments

begin crossings
ShannonPlace&BorhoBlvd = (19,6),40,withoutTL,withoutHole,0,0.5
BorhoBlvd&JanStreet = (24,12),55,withoutTL,withoutHole,0,0.5
JanStreet&WainerWay = (26,12),55,withoutTL,withoutHole,0,0.5
JanStreet&WoodsideWay = (11,12),55,withoutTL,withoutHole,0,0.5
end crossings

begin ctrlElements
in JanStreetGOS1 : stopsign,4,30
in JanStreetGOS1 : stopsign,10,20
end ctrlElements

begin jobsites
in ShannonPlaceBACKS1 : 0,8,1,666
in JanStreetGOS1 : 0,6,1,80
end jobsites

begin railnets
rn1 = (ShannonPlaceGOS1,7),(ShannonPlaceBACKS1,7),(BorhoBlvdG
end railnets
```

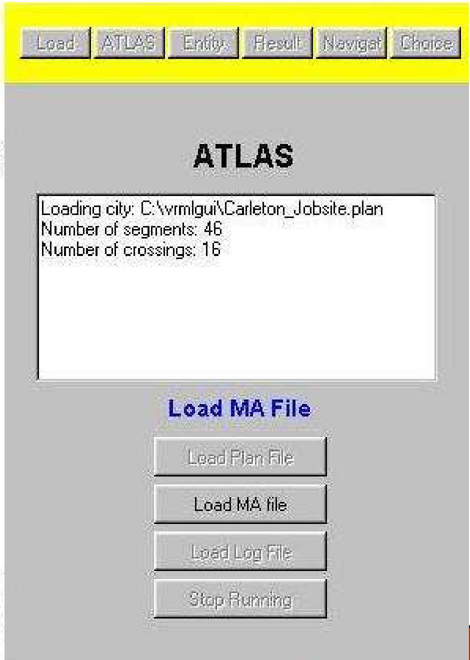
Visualizing outputs

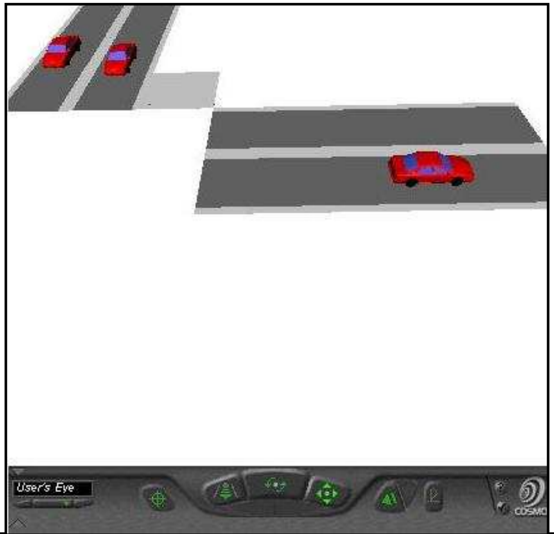






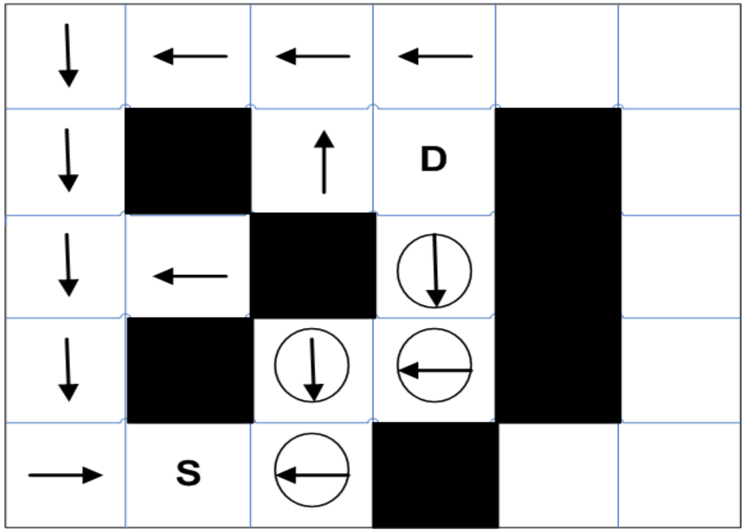




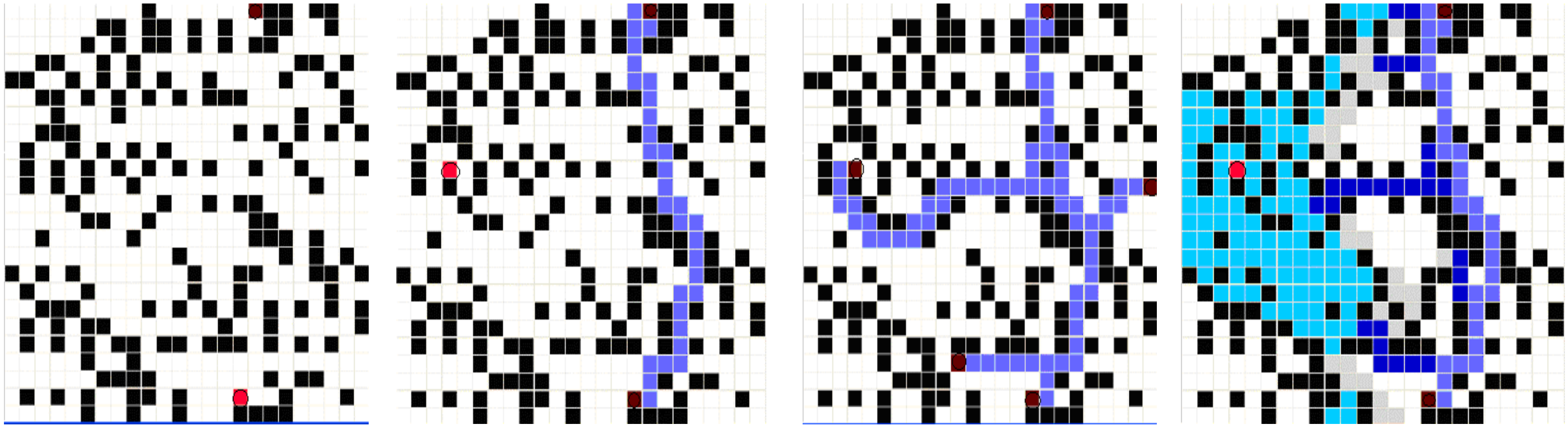


Modeling AODV routing

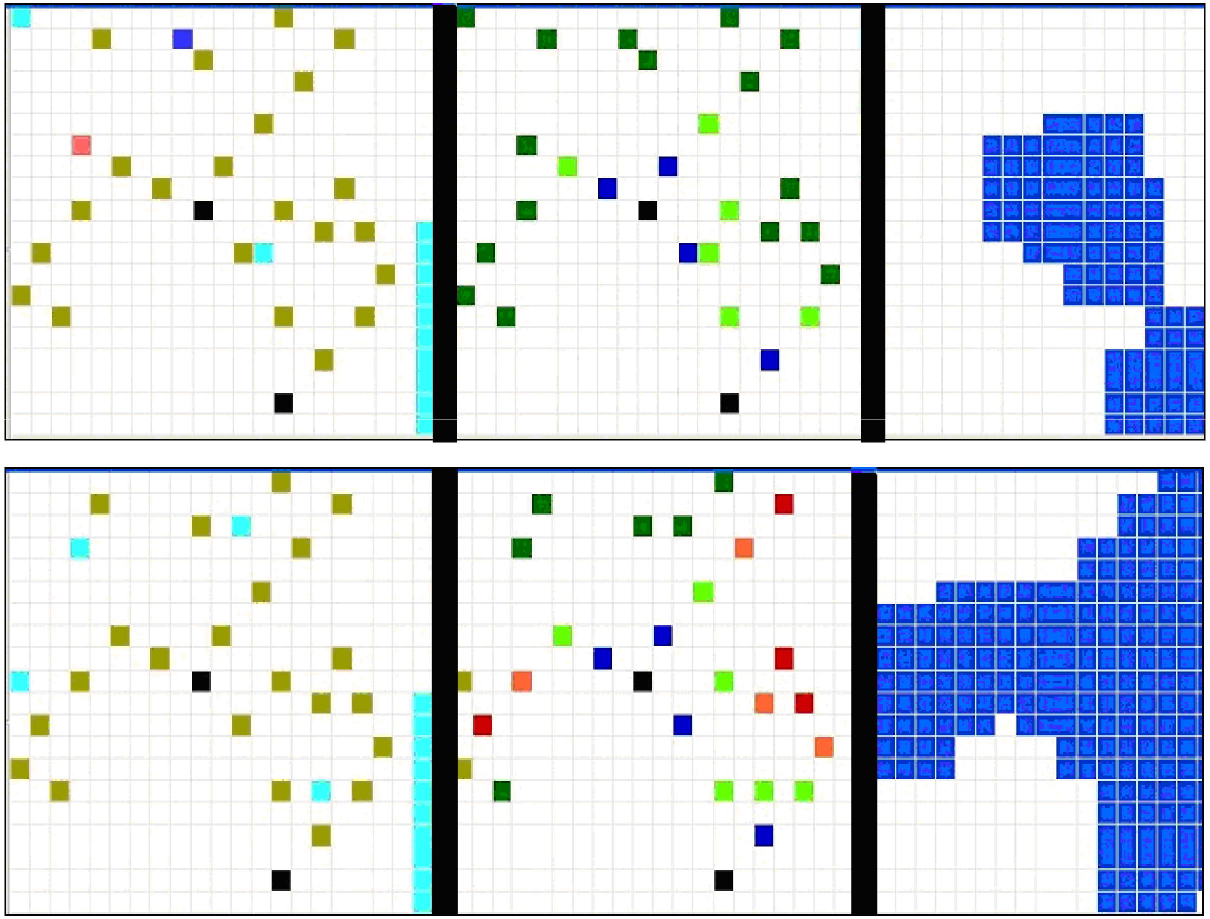
- Variant of the classical Lee’s Algorithm.
- S: node; D: a destination; black cells: dead.
- S broadcasts RREQ message to all its neighbors (*wave* nodes).
- Wave nodes re-broadcast, and set up a reverse path to the sender.
- The process continues until the message reaches the destination node D.
- Shortest path is selected



Simulation results

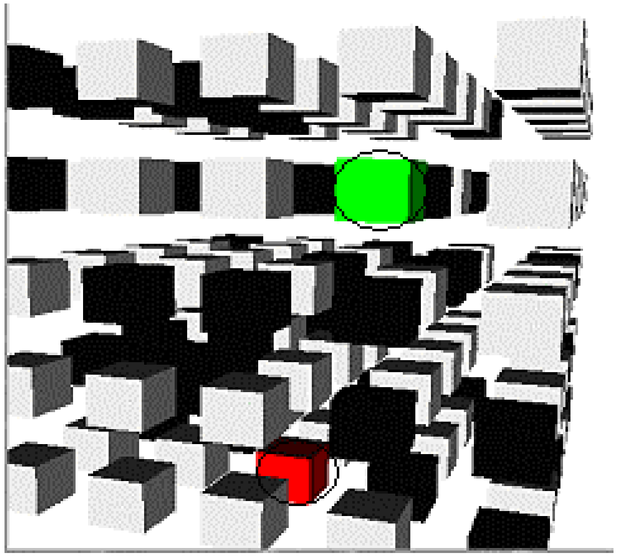


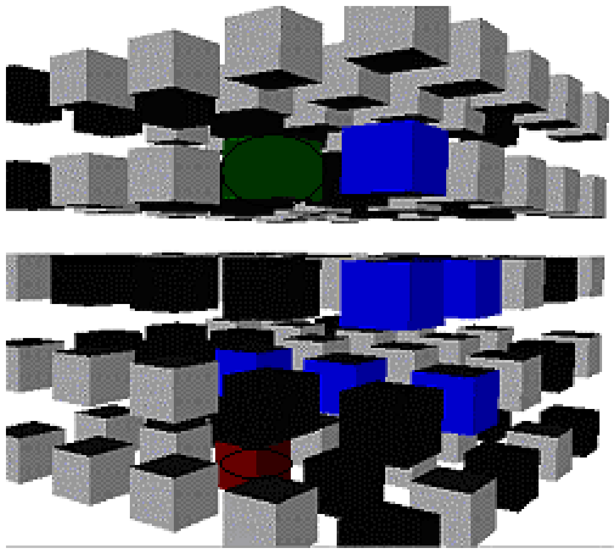
Execution results



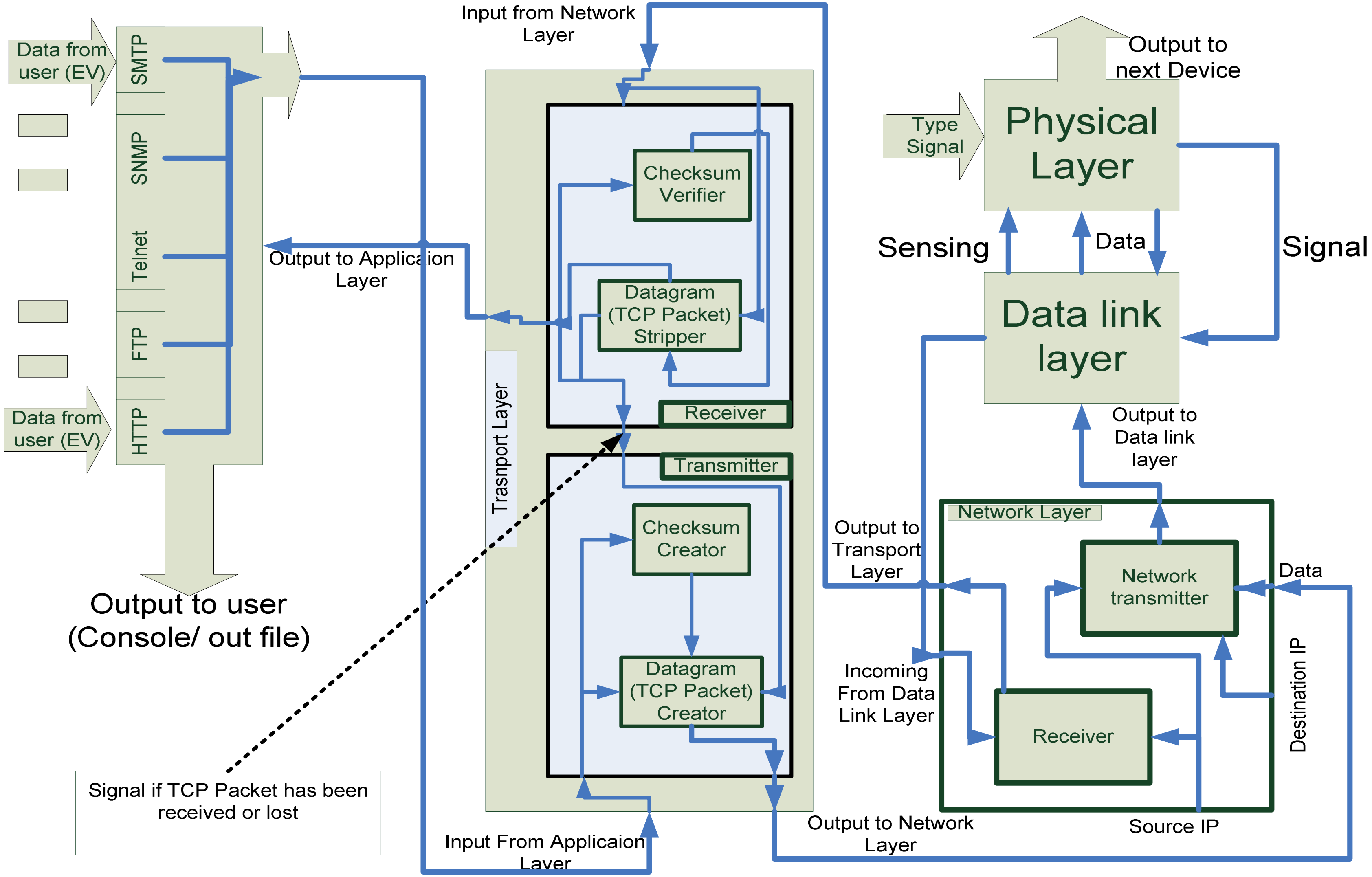
Internetworking Routing

- 3D Cell-DEVS model
- Plane 1: wireless network, Plane 2: wired.





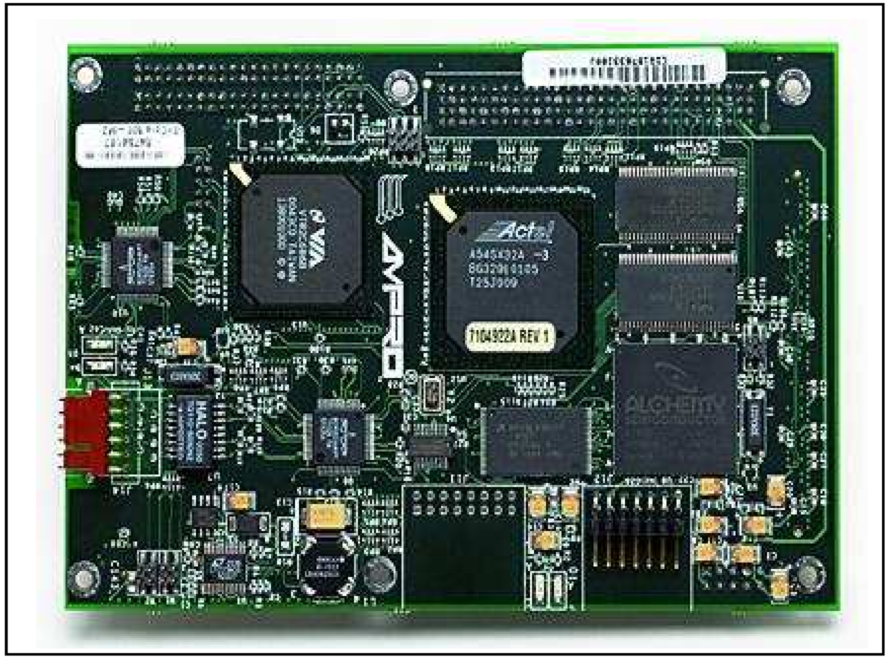
Host



Data from user (EV)
SMTP
SNMP
Telnet
FTP
HTTP
Data from user (EV)
Output to user (Console/ out file)
Input from Network Layer
Output to Application Layer
Checksum Verifier
Datagram (TCP Packet) Stripper
Receiver
Transmitter
Trasnport Layer
Checksum Creator
Datagram (TCP Packet) Creator
Signal if TCP Packet has been received or lost
Input From Applicaion Layer
Output to next Device
Type Signal
Physical Layer
Sensing
Data
Signal
Data link layer
Output to Data link layer
Network Layer
Output to Transport Layer
Network transmitter
Data
Incoming From Data Link Layer
Receiver
Destination IP
Output to Network Layer
Source IP

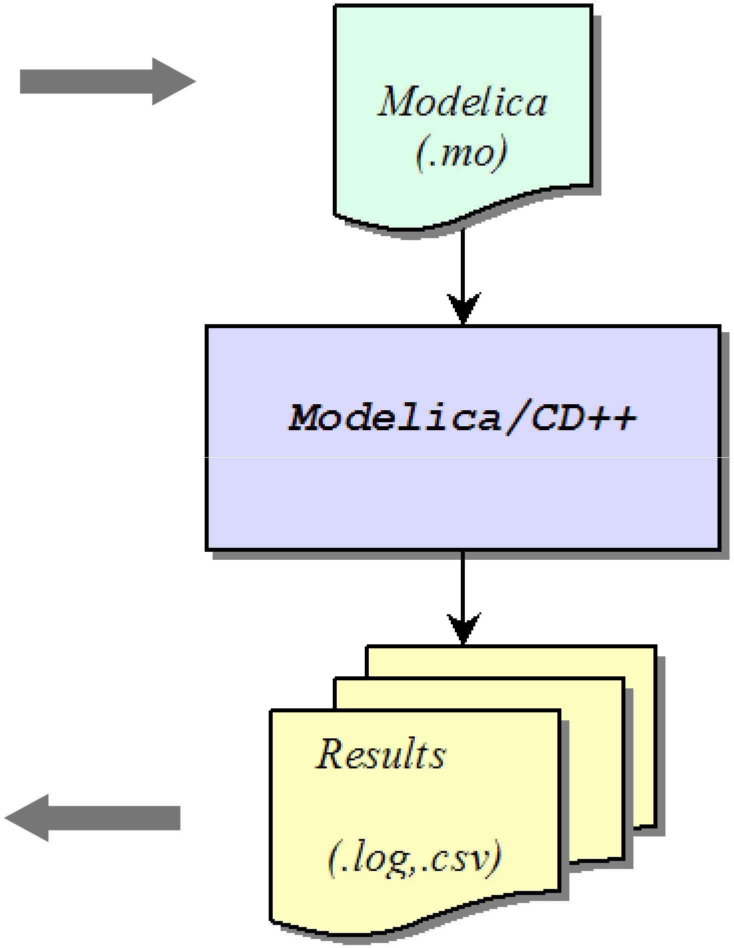
Network Prototyping

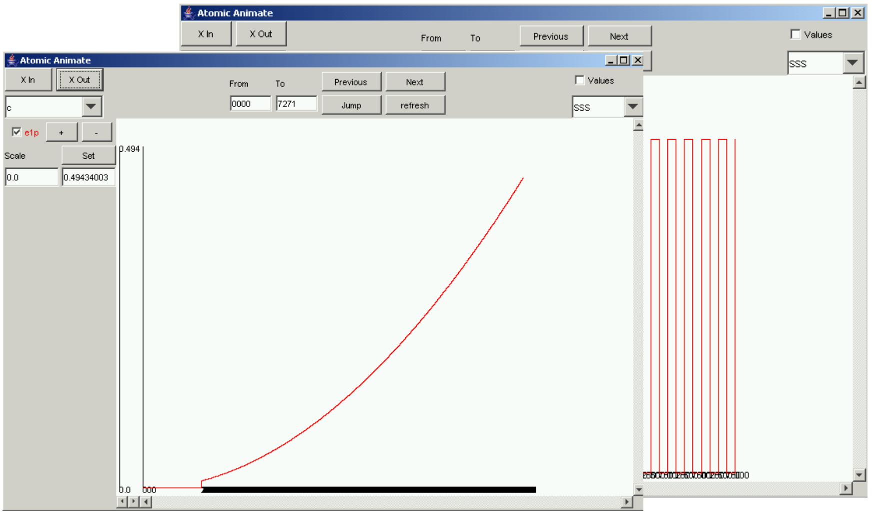
- Real time simulation on embedded linux microcontrollers
- Rapid design and testing potential network devices



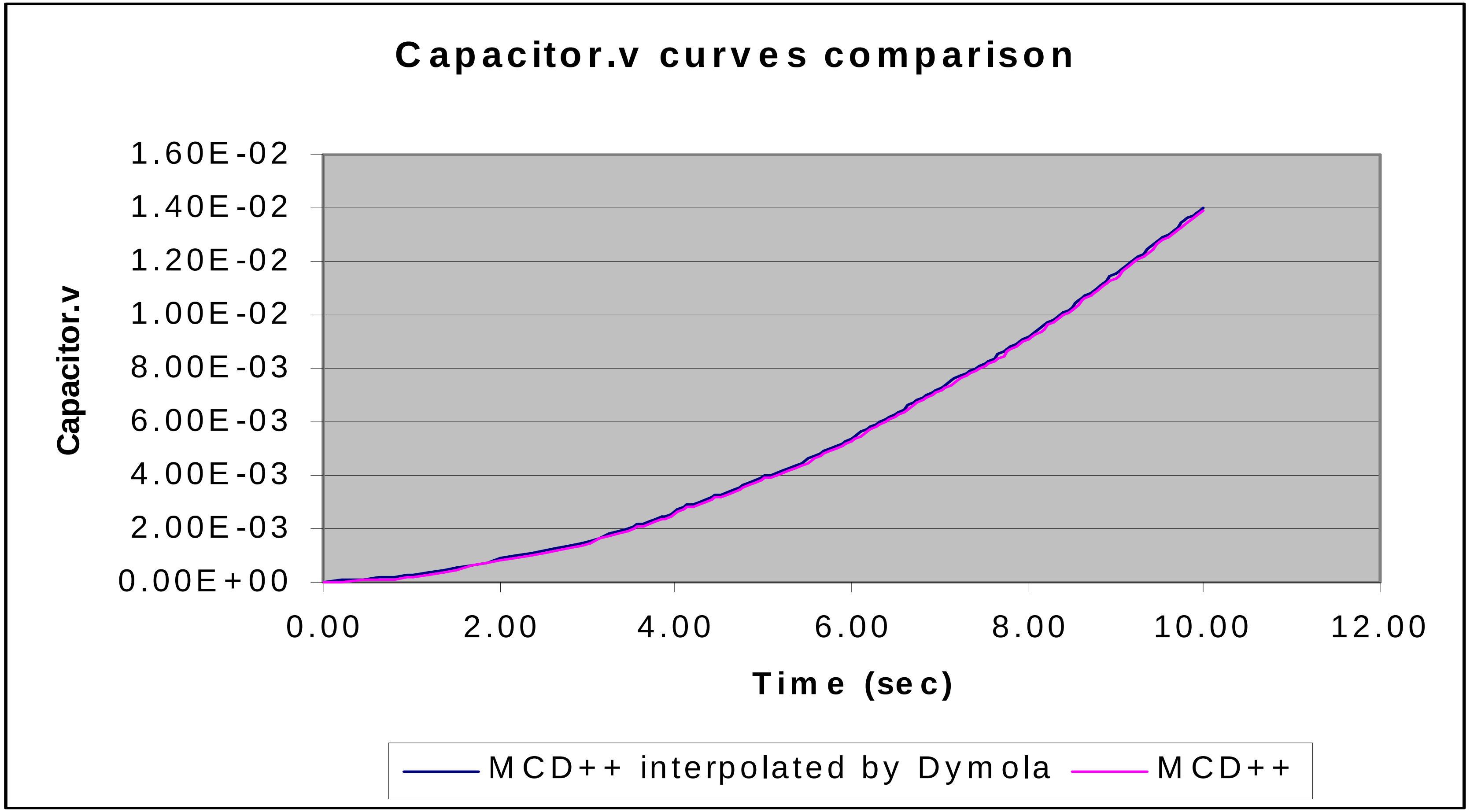
Modelica/CD++

```
model circuit
  Modelica.Electrical.Analog.Sources.PulseVoltage
                  V(V=10, width=50, period=2.5);
  Modelica.Electrical.Analog.Basic.Resistor R1(R=0.001);
  Modelica.Electrical.Analog.Basic.Inductor I1(L=500);
  Modelica.Electrical.Analog.Basic.Inductor I2(L=2000);
  Modelica.Electrical.Analog.Basic.Capacitor C(C=10);
  Modelica.Electrical.Analog.Basic.Resistor R2(R=1000);
  Modelica.Electrical.Analog.Basic.Ground Gnd;
equation
  connect(V.p, R1.p);
  connect(R1.n, I1.p);
  connect(R1.n, I2.p);
  connect(I2.n, C.p);
  connect(I2.n, R2.p);
  connect(C.n, I1.n);
  connect(R2.n, C.n);
  connect(I1.n, V.n);
  connect(V.n, Gnd.p);
end circuit;
```

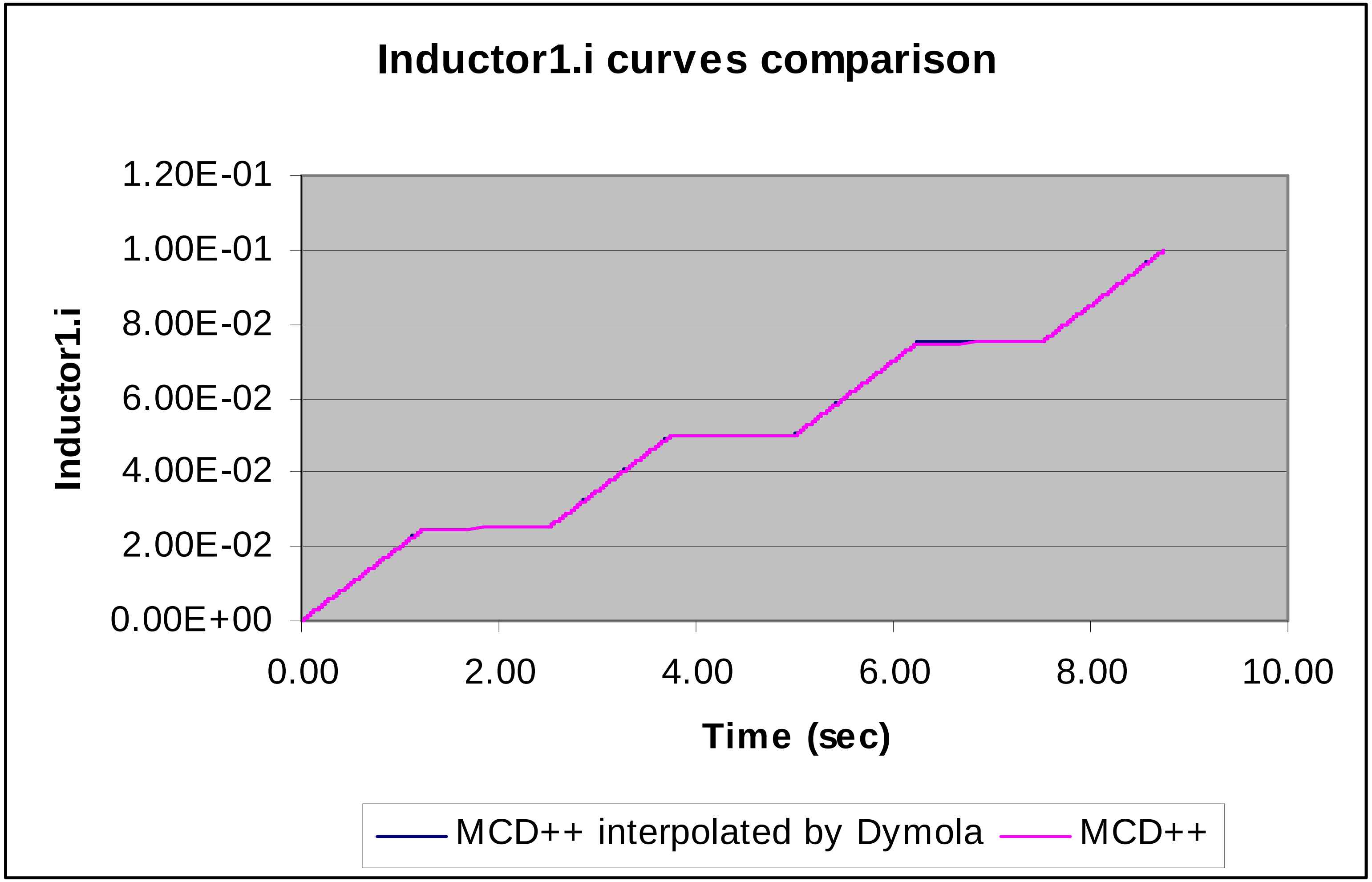




M/CD++ Execution Example



M/CD++ Execution Example



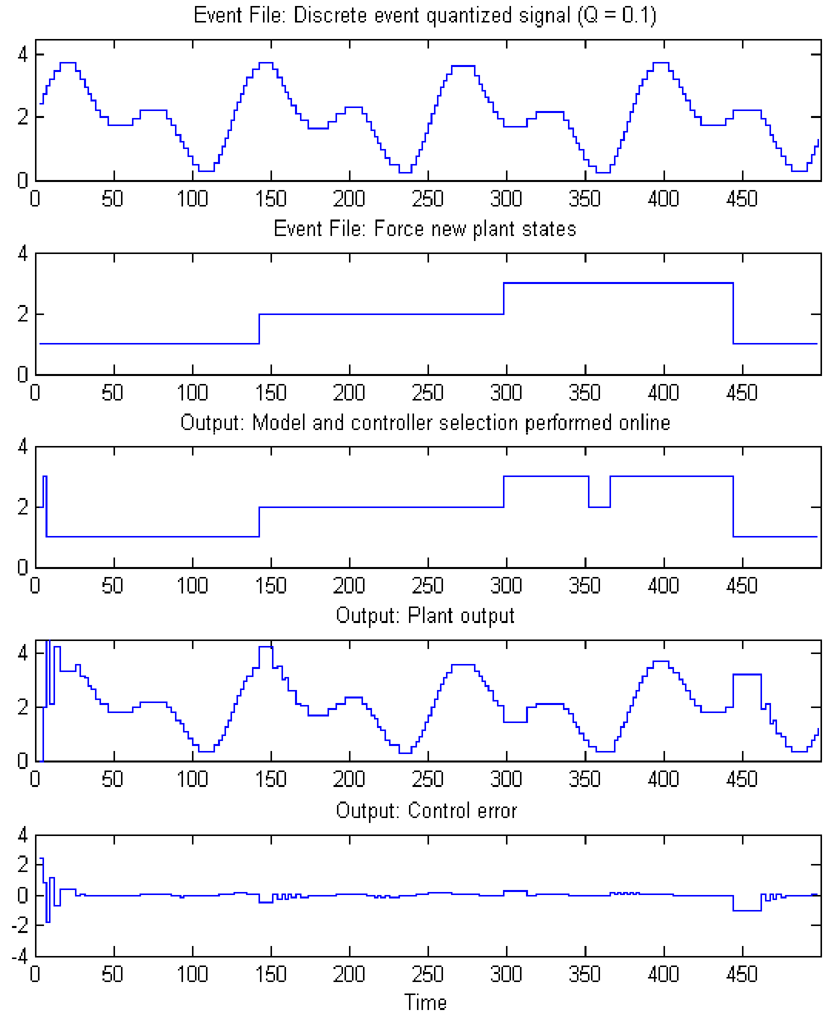
Sample Model Execution

- Multiple model controller allowed to operate as designed, and switch among plant identifying models

- Controller was able to find it and use its parameters

- Error existed only at the period coinciding w/each jump in plant parameters

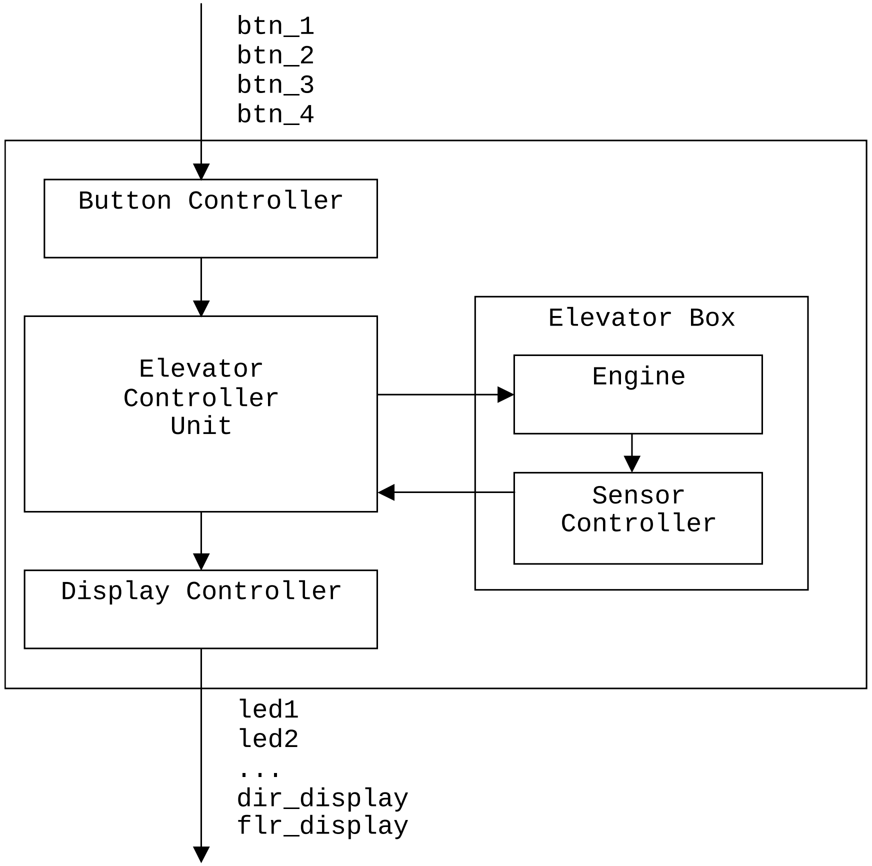
- Only at time 355 did a false model switch occur (due to two models having almost zero error)

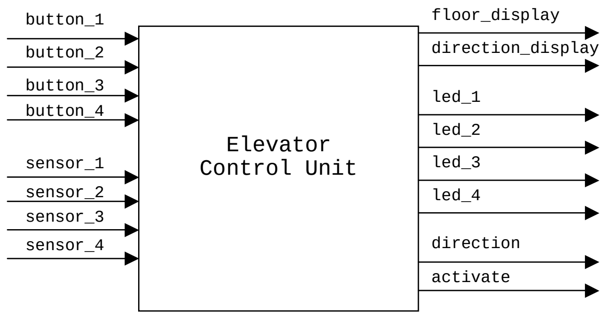


Incremental Prototyping

- We show how to develop incrementally a model based on simple components.
- The application executes in a simulated environment (i.e., all of the components remain executing in a virtual world).
- Simple model of an elevator with both hardware and simulated components.

An elevator control system





INPUTS

Time	Deadline	In-port	Out-Port	Value
00:11:500	00:11:700	btn_3	led3	1
00:14:600	00:14:800	sensor_2	flr_display	1
00:19:500	00:19:700	sensor_3	flr_display	1
00:25:100	00:25:300	btn_4	led4	1
00:30:000	00:30:200	sensor_4	flr_display	1
00:36:300	00:36:500	btn_3	led3	1
00:41:400	00:41:600	sensor_3	flr_display	1

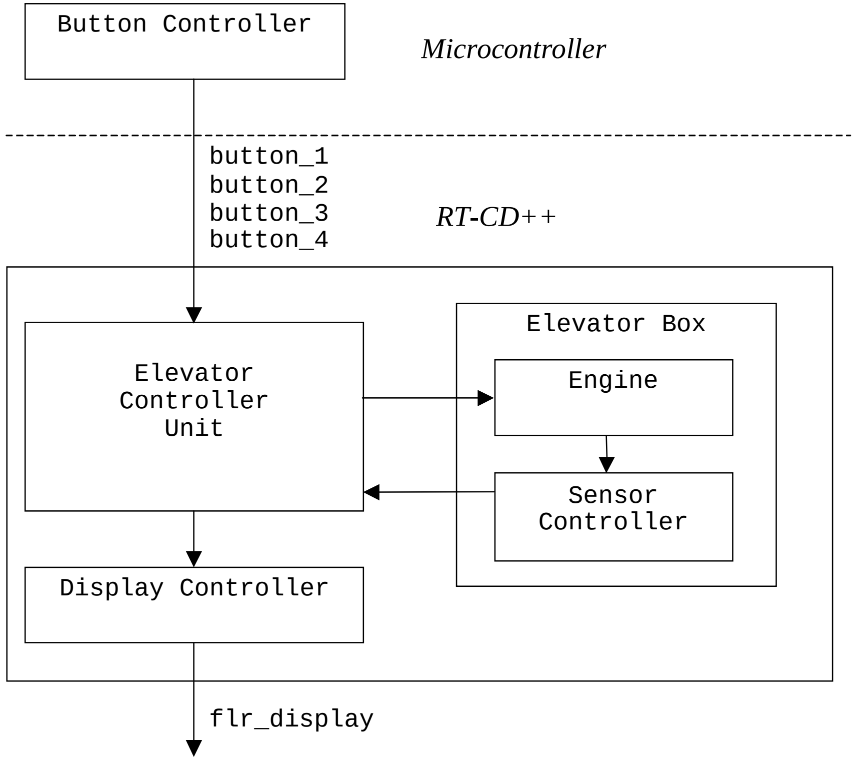
OUTPUTS

Time	Deadline	Out-port	Value
00:11:510	00:11:700	led3	1
00:11:510		dir_display	1
00:14:610	00:14:800	flr_display	2
00:19:510	00:19:700	led3	0
00:19:510		flr_display	3
00:19:510		dir_display	0
00:25:110	00:25:300	led4	1
00:25:110		dir_display	1

(…)

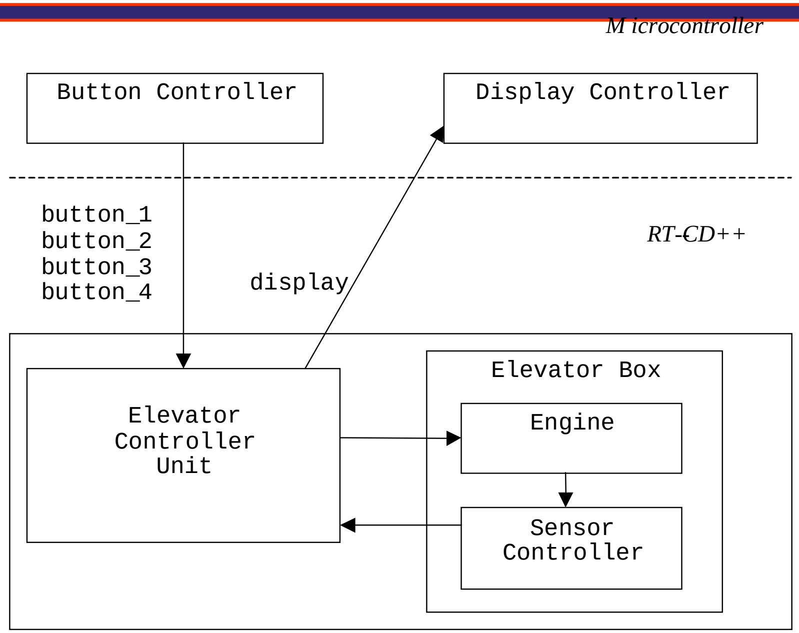
Replacing components

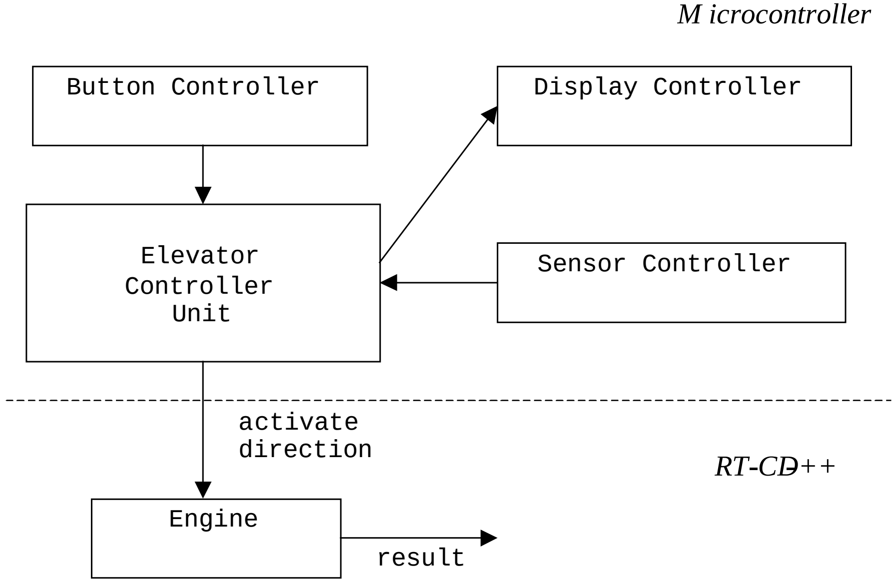
```
components:  elevBox   ec@ECU   dis@Display

in   : button_1  button_2  button_3  button_4
out  : flr_display
link : button_1 button_1@ec
link : button_2 button_2@ec
(…)
link : sensor_1@elevBox sensor_1@ec
link : sensor_2@elevBox sensor_2@ec
(…)
link : floor_disp@ec  flr_display@dis
link : floor_disp@ec  floor_disp
link : dir_disp@ec    dir_display@dis
link : led_1@ec       led_1@dis
(...)
[elevBox]
components:  sb@SensorController   eng@Engine
in   : activate  direction
out  : sensor_1 sensor_2 sensor_3 sensor_4
link : activate    activate@eng
link : direction   direction@eng
link : sensor_1@sb sensor_1
(...)
link : current_floor@eng sensor_triggered@sb
(...)
```



Time	Out-por	Value
00:08:170	flr_display	2
00:19:540	flr_display	1
00:30:130	flr_display	2
00:35:140	flr_display	3
00:40:150	flr_display	4
00:58:290	flr_display	3

Replacing components





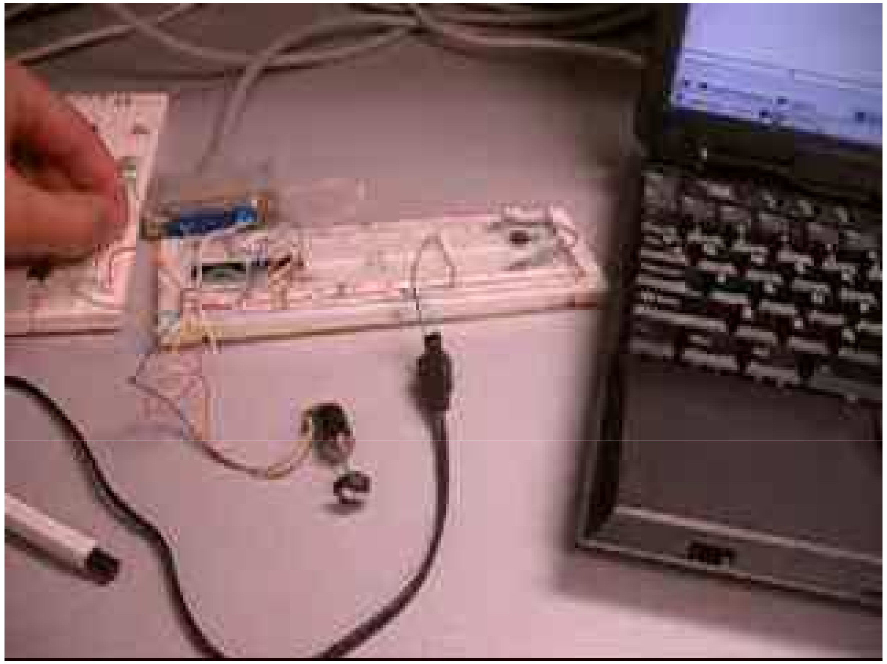
Time	Port	Value
00:06:120	direction	1
00:06:130	activate	1
00:15:930	activate	0
00:56:800	direction	2
00:56:810	activate	1
01:01:130	activate	0
01:22:710	direction	2

Time	Out-port	Value
00:06:130	result	1
00:15:930	result	0
00:56:810	result	2
01:01:130	result	0

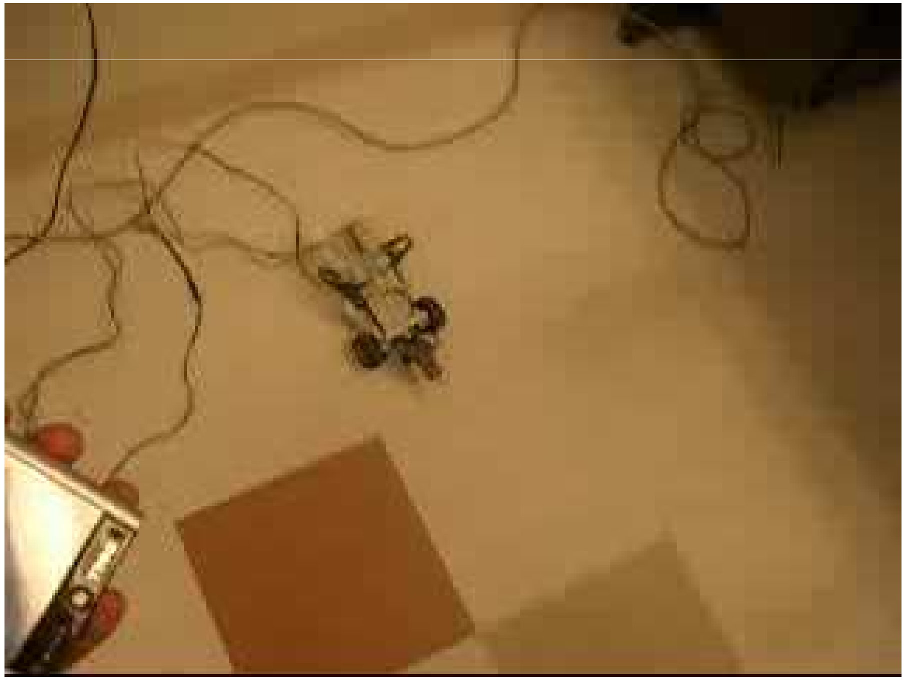
(…)

```
components:  eng@Engine
in   : activate_in direction_in
out  : result
link : activate_in    activate@eng
link : direction_in   direction@eng
```

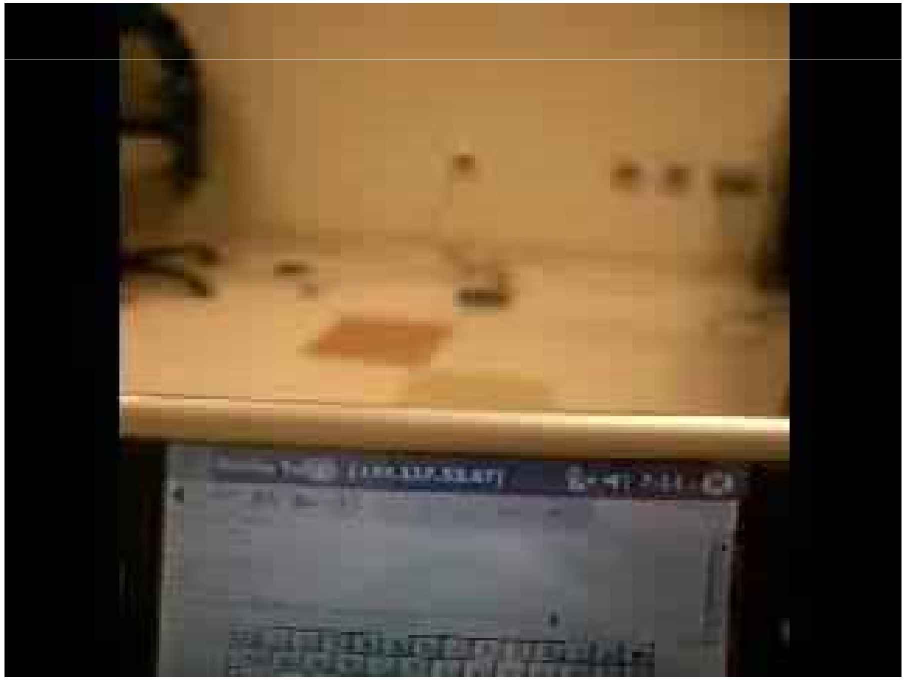
Building a Robot Controller

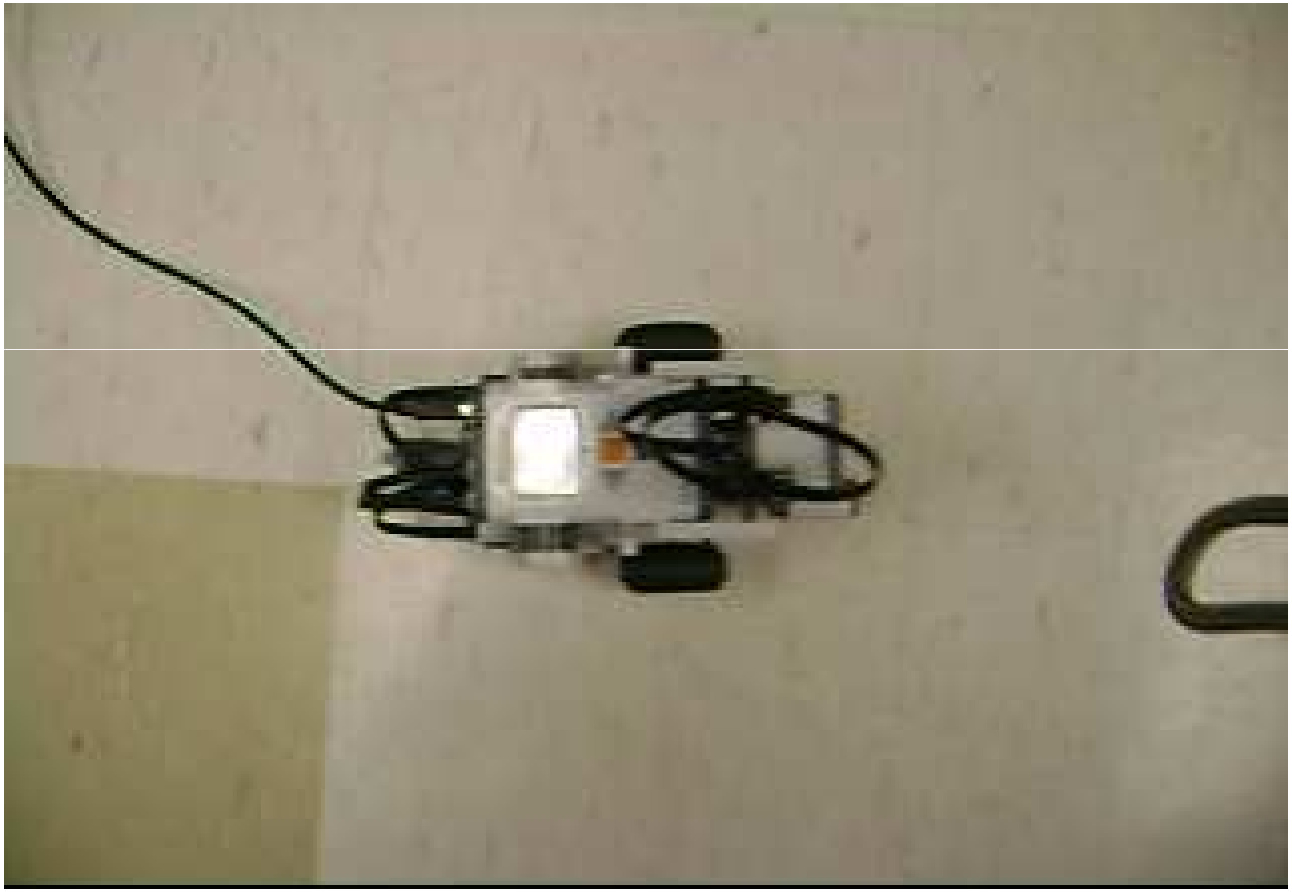


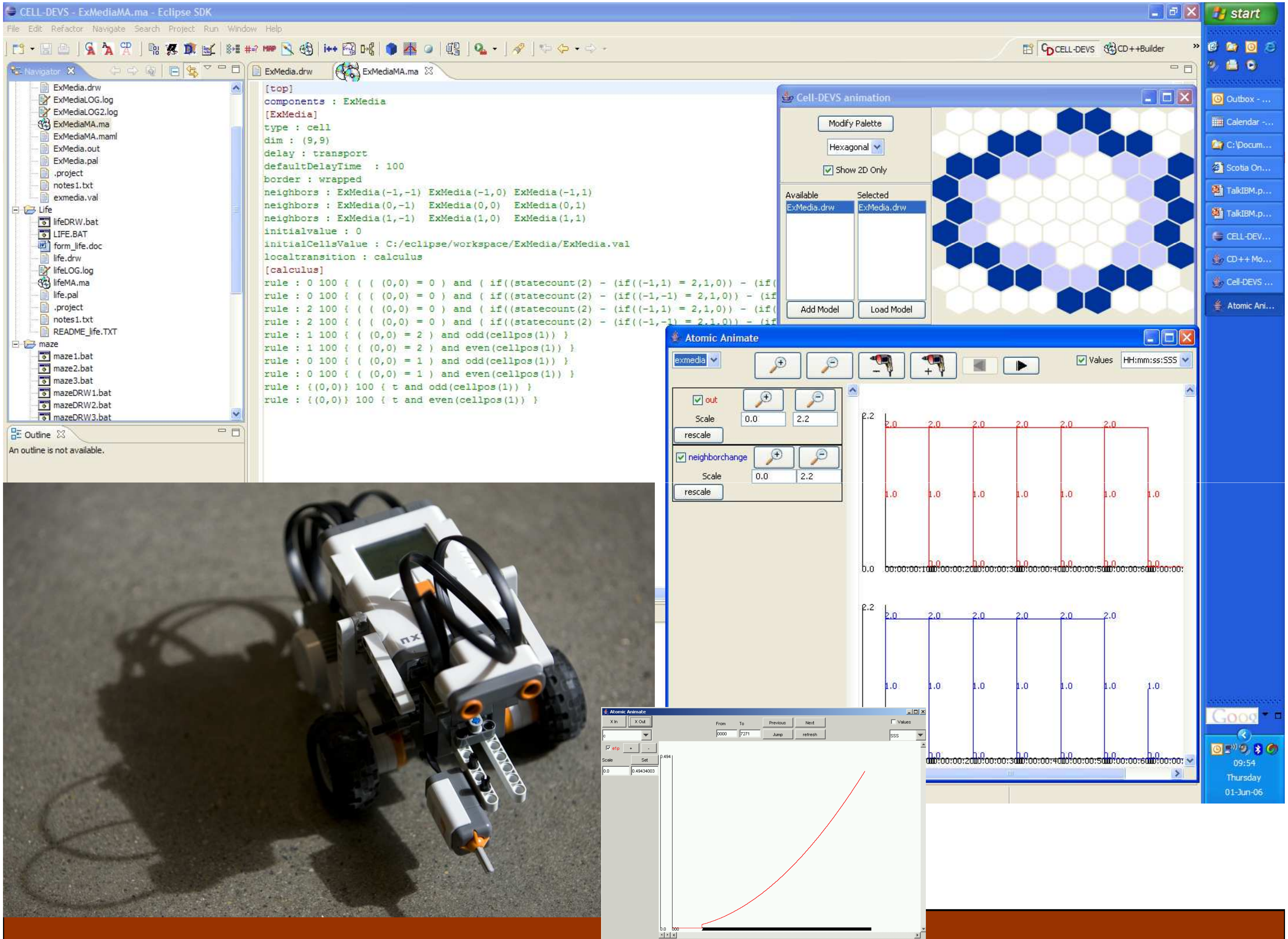
Integration Tests



Integration Tests

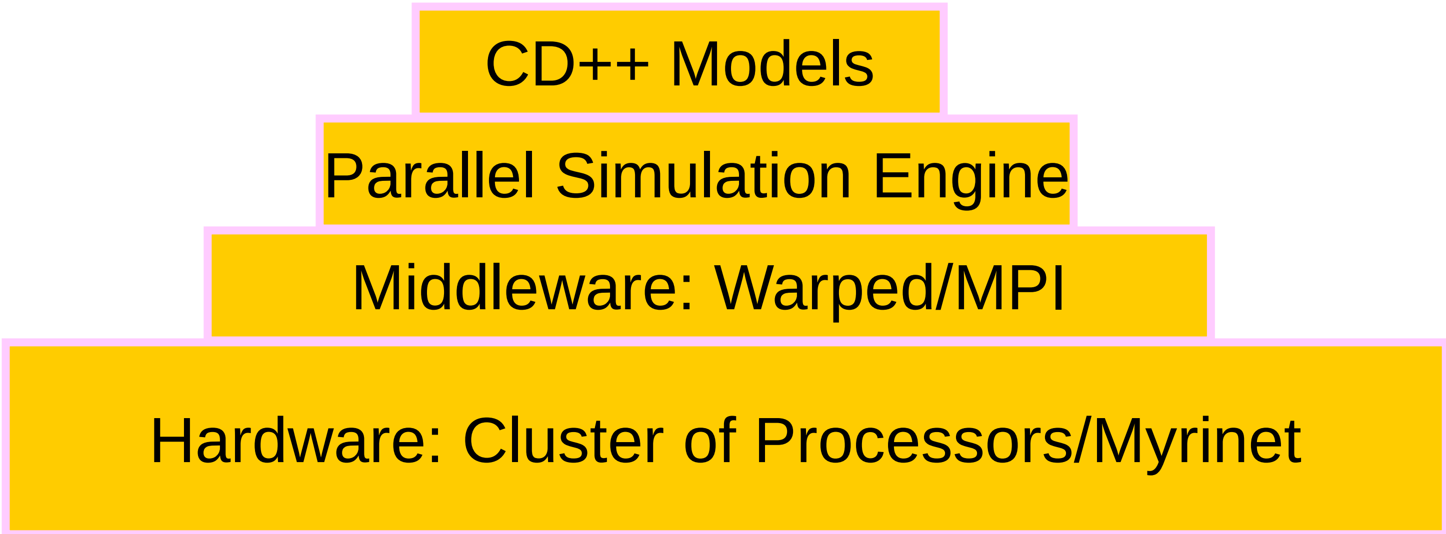





CELL-DEVS - ExMediaMA.ma - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Navigator
ExMedia.drw
ExMediaLOG.log
ExMediaLOG2.log
ExMediaMA.ma
ExMediaMA.maml
ExMedia.out
ExMedia.pal
.project
notes1.txt
exmedia.val
Life
lifeDRW.bat
LIFE.BAT
form_life.doc
life.drw
lifeLOG.log
lifeMA.ma
life.pal
.project
notes1.txt
README_life.TXT
maze
maze1.bat
maze2.bat
maze3.bat
mazeDRW1.bat
mazeDRW2.bat
mazeDRW3.bat
Outline
An outline is not available.
ExMedia.drw
ExMediaMA.ma
[top]
components : ExMedia
[ExMedia]
type : cell
dim : (9,9)
delay : transport
defaultDelayTime : 100
border : wrapped
neighbors : ExMedia(-1,-1) ExMedia(-1,0) ExMedia(-1,1)
neighbors : ExMedia(0,-1) ExMedia(0,0) ExMedia(0,1)
neighbors : ExMedia(1,-1) ExMedia(1,0) ExMedia(1,1)
initialvalue : 0
initialCellsValue : C:/eclipse/workspace/ExMedia/ExMedia.val
localtransition : calculus
[calculus]
rule : 0 100 { (((0,0) = 0) and (if((statecount(2) - (if((-1,1) = 2,1,0)) - (if(
rule : 0 100 { (((0,0) = 0) and (if((statecount(2) - (if((-1,-1) = 2,1,0)) - (if(
rule : 2 100 { (((0,0) = 0) and (if((statecount(2) - (if((-1,1) = 2,1,0)) - (if(
rule : 2 100 { (((0,0) = 0) and (if((statecount(2) - (if((-1,-1) = 2,1,0)) - (if(
rule : 1 100 { ((0,0) = 2) and odd(cellpos(1)) }
rule : 1 100 { ((0,0) = 2) and even(cellpos(1)) }
rule : 0 100 { ((0,0) = 1) and odd(cellpos(1)) }
rule : 0 100 { ((0,0) = 1) and even(cellpos(1)) }
rule : {(0,0)} 100 { t and odd(cellpos(1)) }
rule : {(0,0)} 100 { t and even(cellpos(1)) }
Cell-DEVS animation
Modify Palette
Hexagonal
Show 2D Only
Available
Selected
ExMedia.drw
ExMedia.drw
Add Model
Load Model
Atomic Animate
exmedia
Values
HH:mm:ss:SSS
out
Scale
0.0
2.2
rescale
neighborchange
Scale
0.0
2.2
rescale
CELL-DEVS
CD++Builder
start
Outbox - ...
Calendar - ...
C:\Docum...
Scotia On...
TalkIBM.p...
TalkIBM.p...
CELL-DEV...
CD++ Mo...
Cell-DEVS ...
Atomic Ani...
09:54
Thursday
01-Jun-06
nxt

Parallel CD++

- Execute DEVS models in parallel
- Layered architecture based on different middleware
  - Expansion to RTI: few lines of code



Carleton University

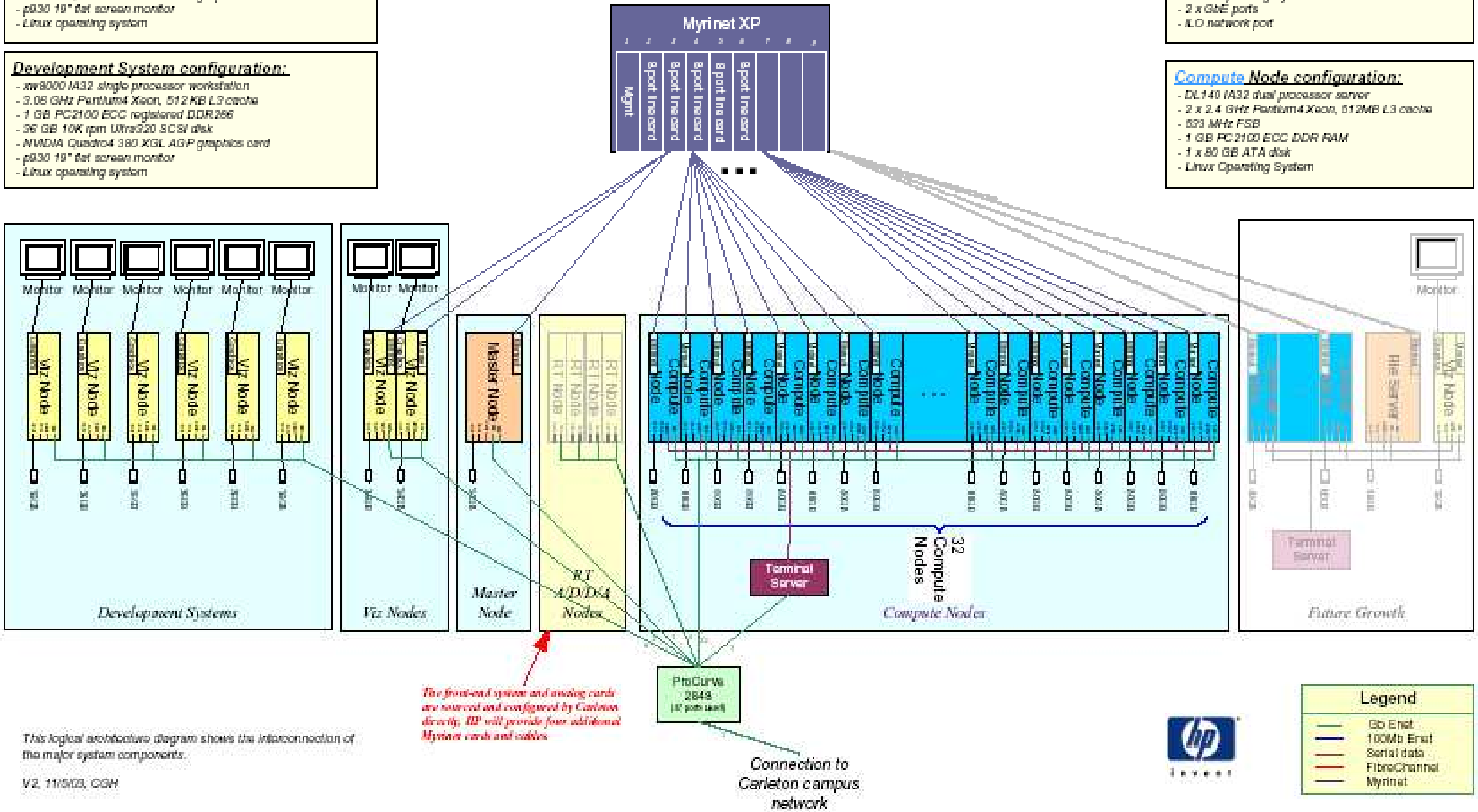
Advanced Laboratory for Real-time Simulation Cluster

Visualization Node configuration:
- xw8000 IA32 dual processor workstation
- 2 x 3.06 GHz Pentium4 Xeon, 512 KB L3 cache
- 1 GB PC2100 ECC registered DDR266
- 36 GB 10K rpm Ultra320 SCSI disk
- Myrinet XP PCI-X card
- NVIDIA Quadro4 980 XGL AGP graphics card
- p930 19" flat screen monitor
- Linux operating system

Development System configuration:
- xw8000 IA32 single processor workstation
- 3.06 GHz Pentium4 Xeon, 512 KB L3 cache
- 1 GB PC2100 ECC registered DDR266
- 36 GB 10K rpm Ultra320 SCSI disk
- NVIDIA Quadro4 380 XGL AGP graphics card
- p930 19" flat screen monitor
- Linux operating system

Master Node configuration:
- DL360-G3 IA32 2P server
- 2 x 3.06GHz Pentium4 Xeon, 512KB L3 cache
- 533 MHz FSB
- 1 GB PC2100 ECC DDR RAM
- 36 GB 10K rpm SCSI disk
- Linux Operating System
- 2 x GbE ports
- iLO network port

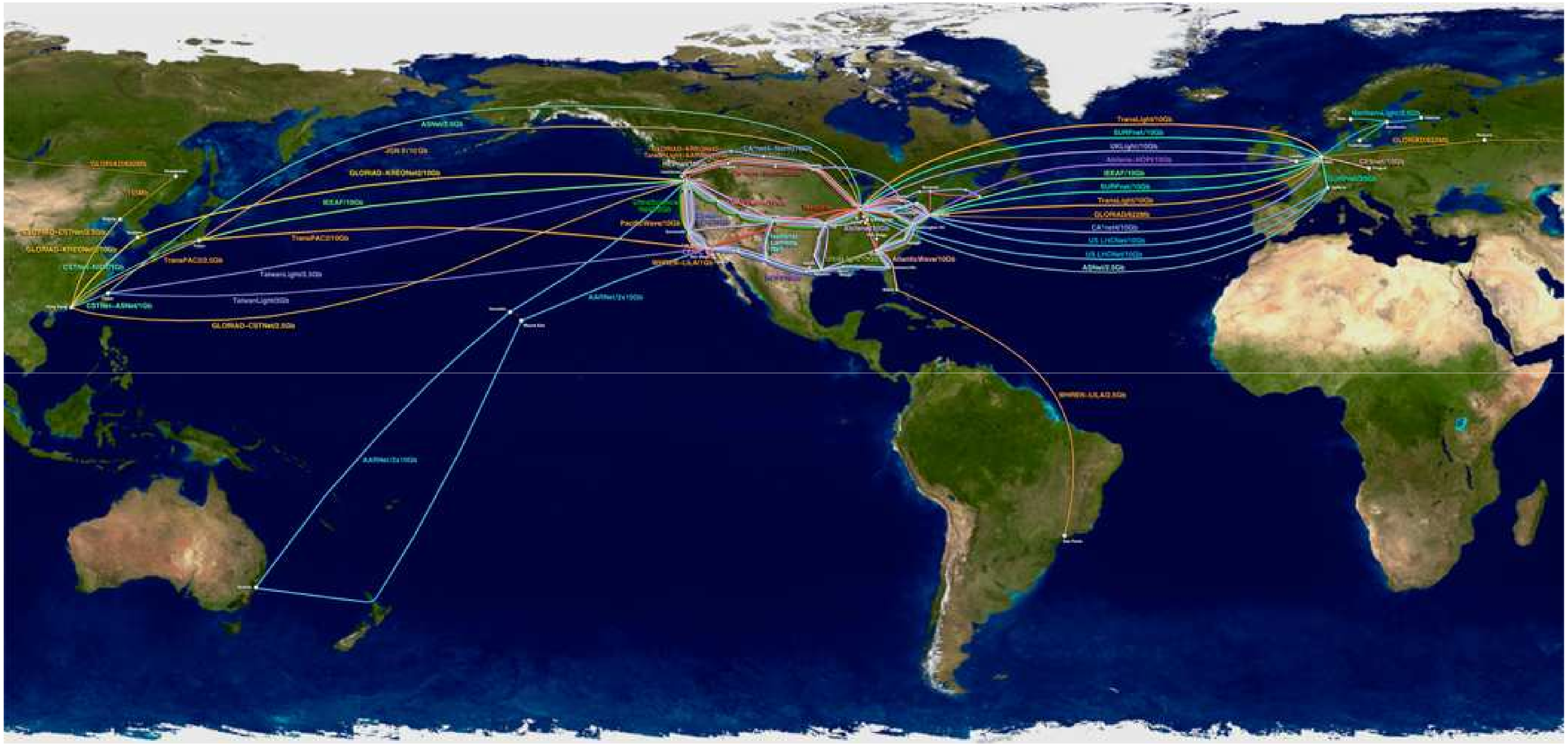
Compute Node configuration:
- DL140 IA32 dual processor server
- 2 x 2.4 GHz Pentium4 Xeon, 512MB L3 cache
- 533 MHz FSB
- 1 GB PC2100 ECC DDR RAM
- 1 x 80 GB ATA disk
- Linux Operating System



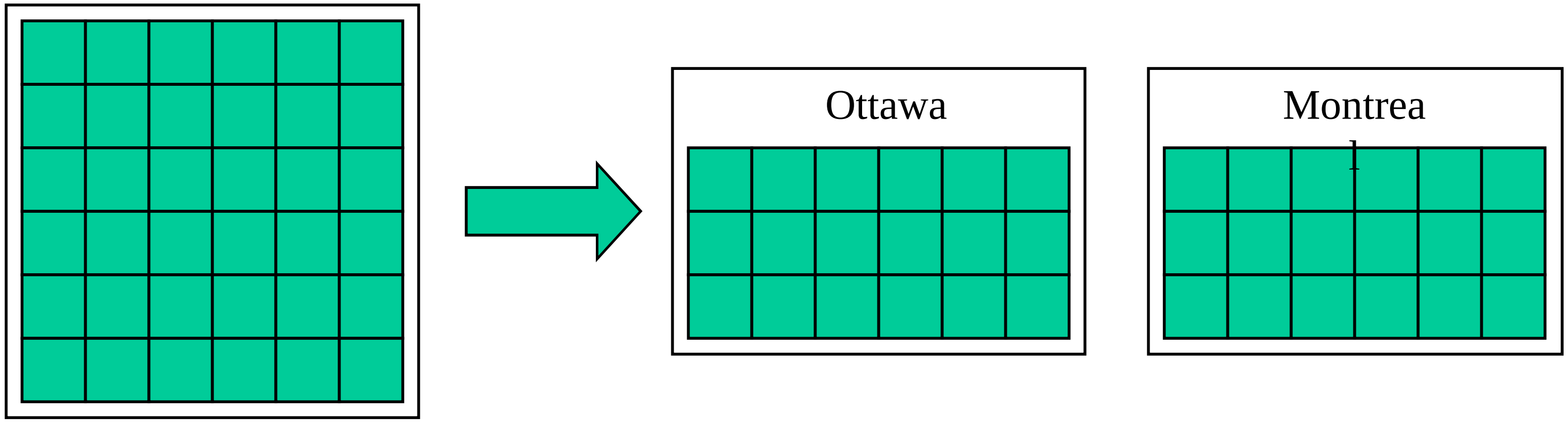
This logical architecture diagram shows the interconnection of the major system components.

V2, 11/5/03, CGH

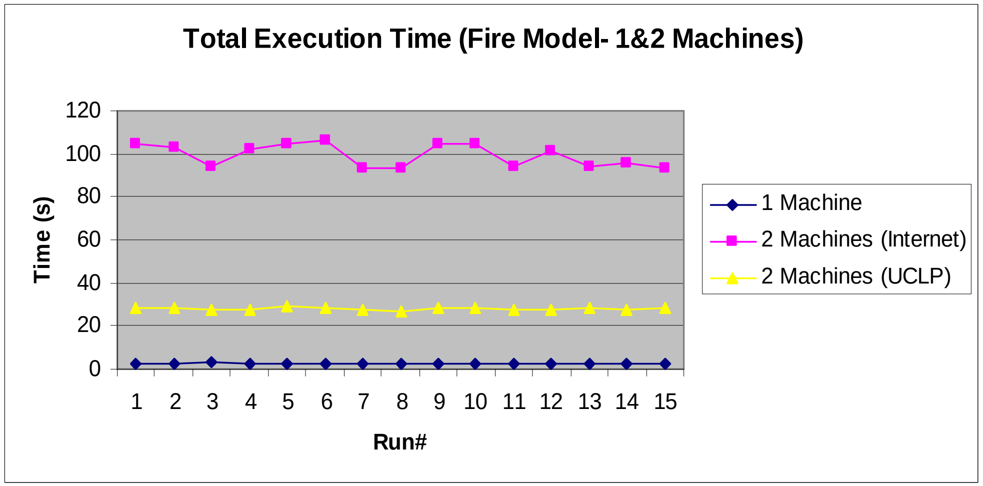
CANet4



Partitioned Fire Model test



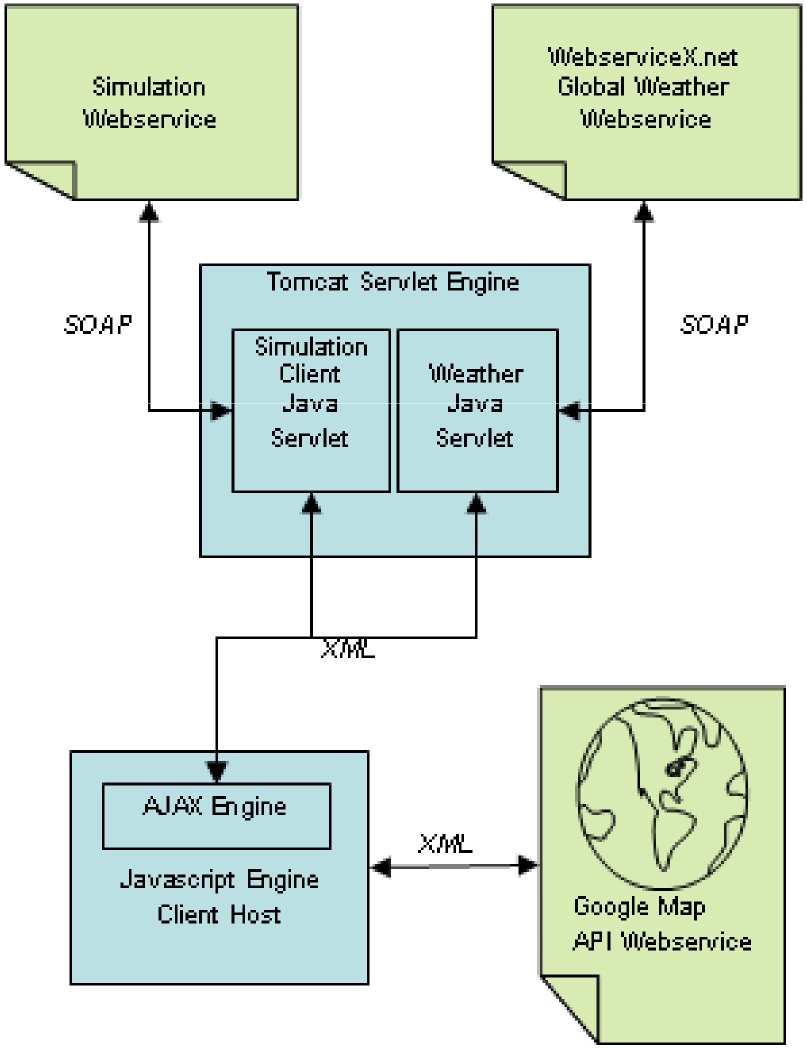
- Fire model (1-machine)



Fire Spreading Simulation Mashup

- Finalist at [IEEE Services Computing Contest]
- Service integrates prediction of forest fires,

  weather data and
  Google Maps



Carleton UNIVERSITY

1. **Simulation location**
2. Model
3. Computer Grid
4. Simulation

Log off

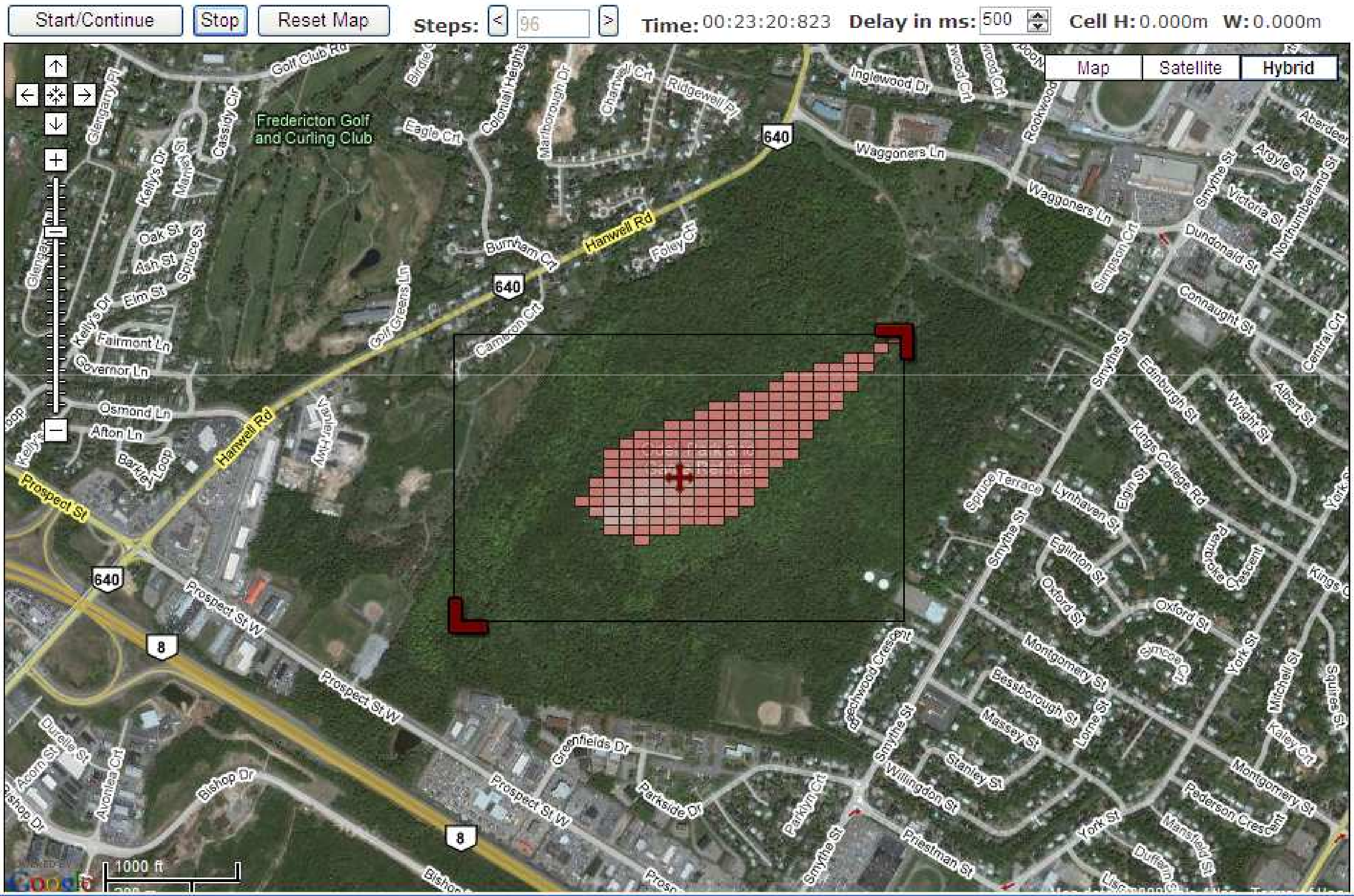
Select location and check weather

Use the fields and the map above to select the country and the town where to run the simulation. If the town does not appear or is not supported, please do a manual search with the google map.

Canada | Fredericton, N. B.

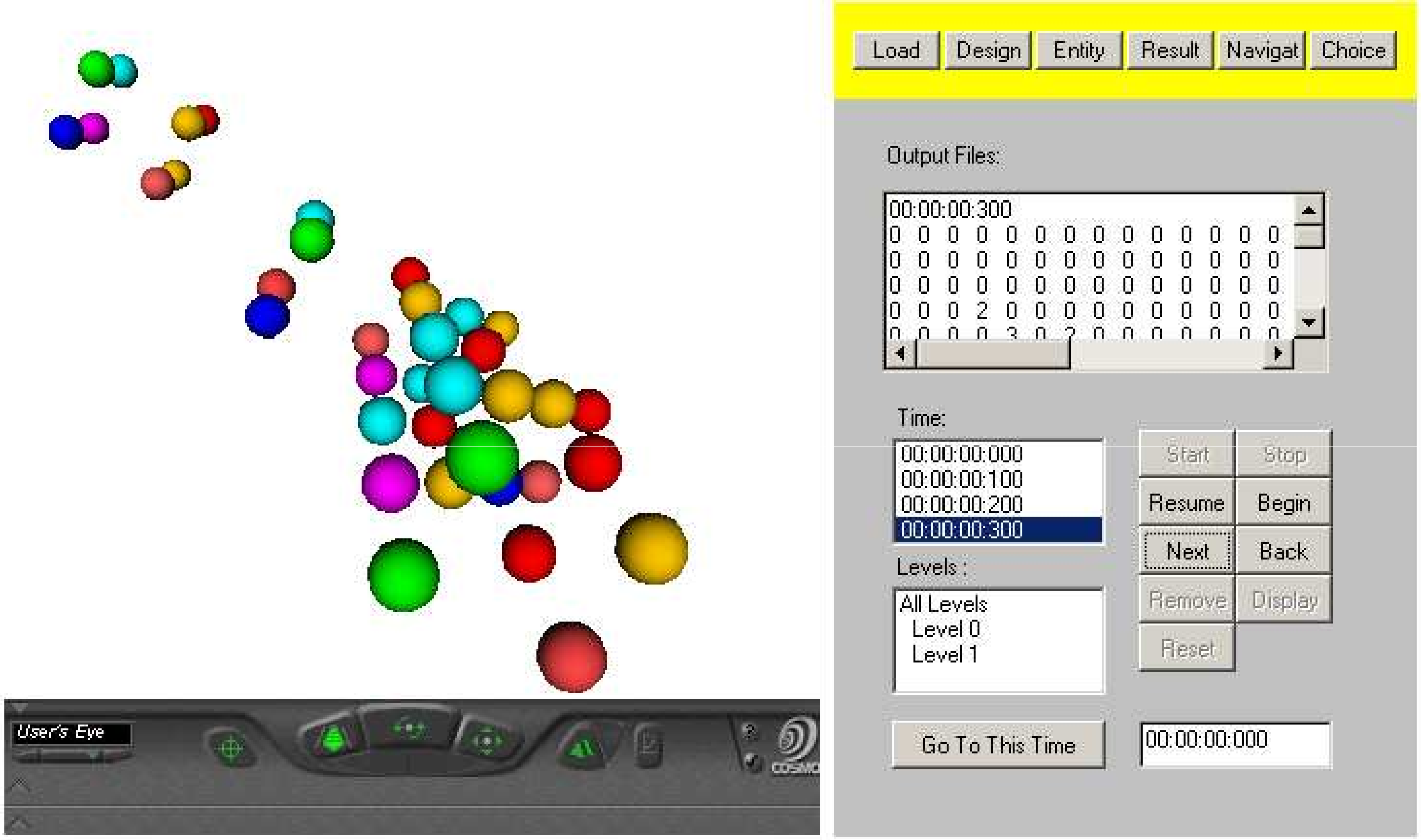
Next Step >

Mashup (Google Maps)



CD++ Visualization Engines

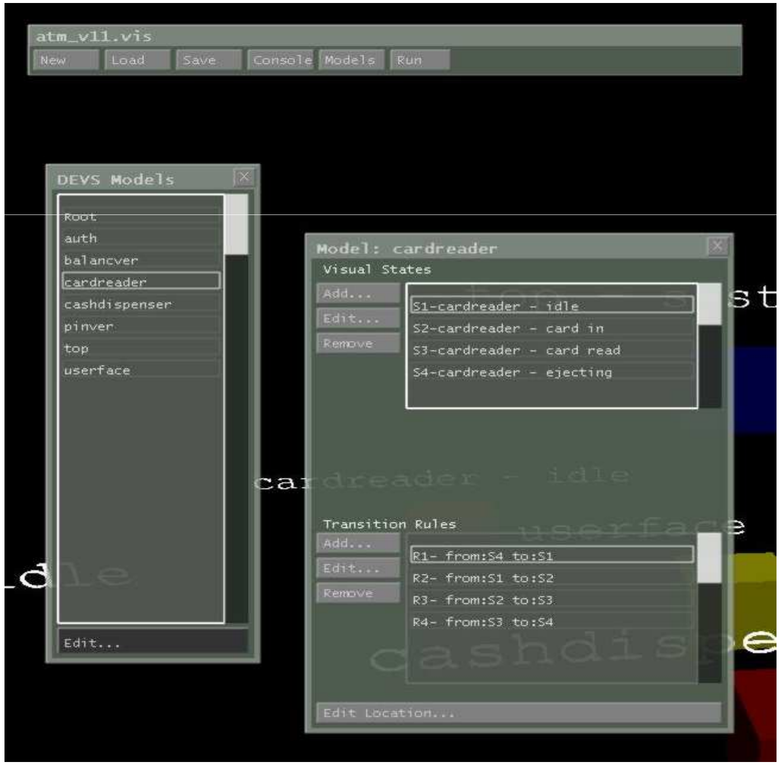
3D Visualization GUI

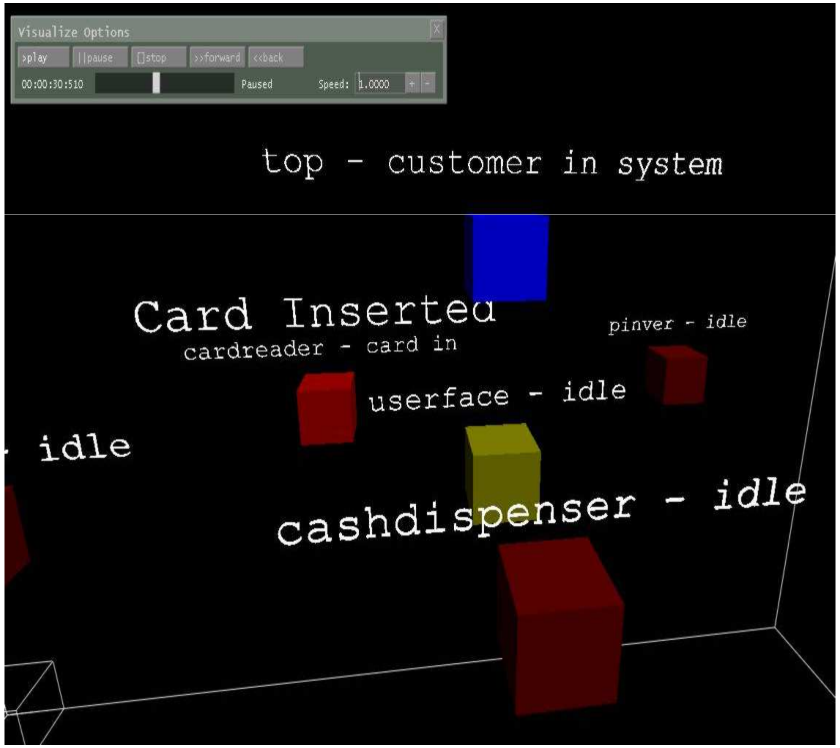


1. Change geometry, color and size of the nodes
2. Navigation
3. Edit individual node

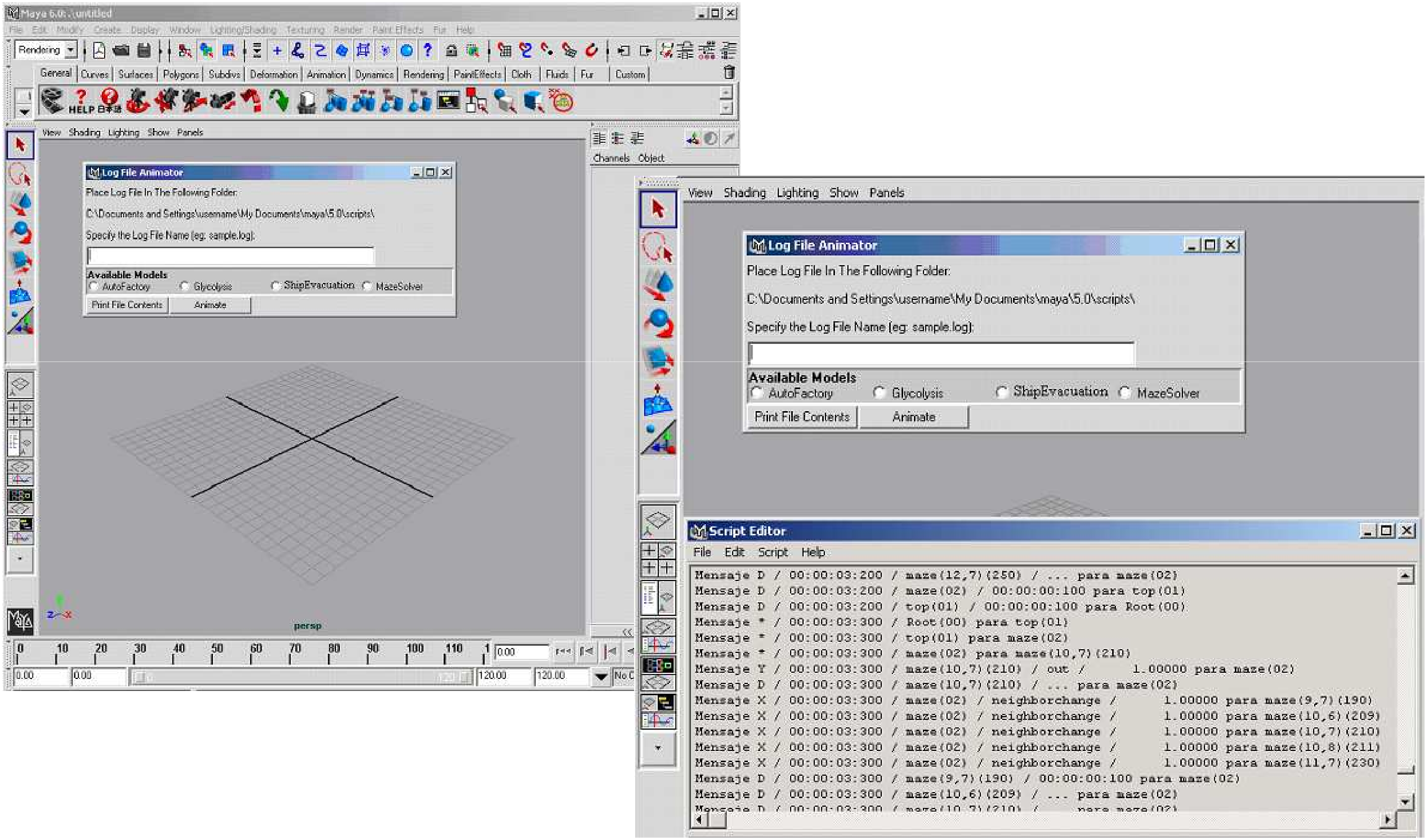
DEVSView

- Visual models extracted from CD++ simulation log file;
  visual state machines defined using the DEVSView user interface.





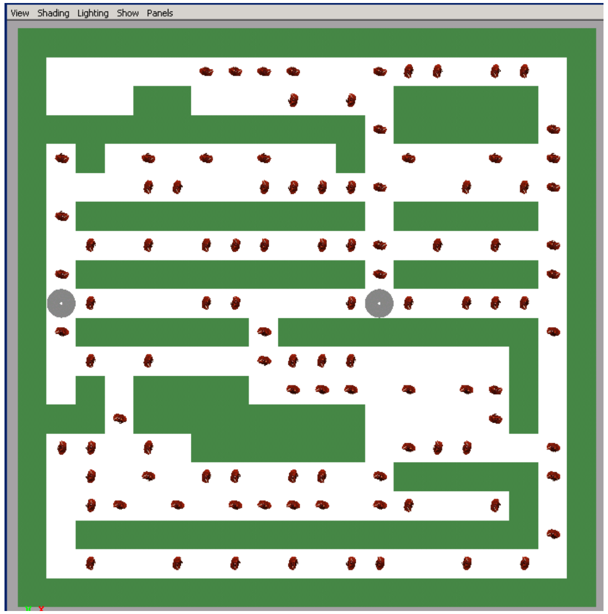
CD++/Maya

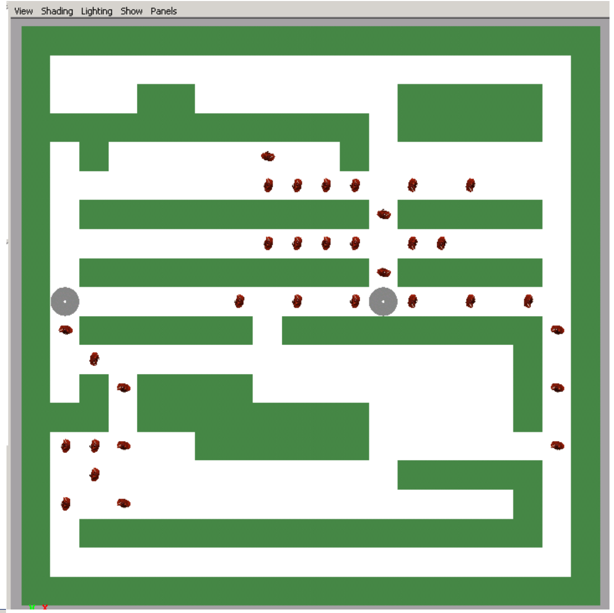

Log File Animator
Place Log File In The Following Folder:
C:\Documents and Settings\username\My Documents\maya\5.0\scripts\
Specify the Log File Name (eg: sample.log):
Available Models
AutoFactory
Glycolysis
ShipEvacuation
MazeSolver
Print File Contents
Animate
Script Editor
File Edit Script Help
Mensaje D / 00:00:03:200 / maze(12,7)(250) / ... para maze(02)
Mensaje D / 00:00:03:200 / maze(02) / 00:00:00:100 para top(01)
Mensaje D / 00:00:03:200 / top(01) / 00:00:00:100 para Root(00)
Mensaje * / 00:00:03:300 / Root(00) para top(01)
Mensaje * / 00:00:03:300 / top(01) para maze(02)
Mensaje * / 00:00:03:300 / maze(02) para maze(10,7)(210)
Mensaje Y / 00:00:03:300 / maze(10,7)(210) / out / 1.00000 para maze(02)
Mensaje D / 00:00:03:300 / maze(10,7)(210) / ... para maze(02)
Mensaje X / 00:00:03:300 / maze(02) / neighborchange / 1.00000 para maze(9,7)(190)
Mensaje X / 00:00:03:300 / maze(02) / neighborchange / 1.00000 para maze(10,6)(209)
Mensaje X / 00:00:03:300 / maze(02) / neighborchange / 1.00000 para maze(10,7)(210)
Mensaje X / 00:00:03:300 / maze(02) / neighborchange / 1.00000 para maze(10,8)(211)
Mensaje X / 00:00:03:300 / maze(02) / neighborchange / 1.00000 para maze(11,7)(230)
Mensaje D / 00:00:03:300 / maze(9,7)(190) / 00:00:00:100 para maze(02)
Mensaje D / 00:00:03:300 / maze(10,6)(209) / ... para maze(02)

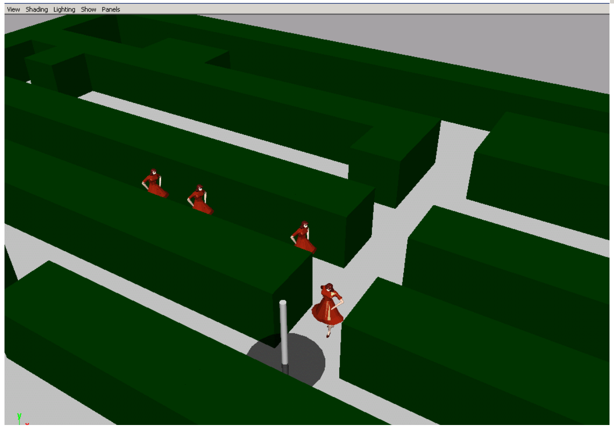
Simulated results

- Creation of a 3D version of the simulation
- Interpreted by the MEL scripts

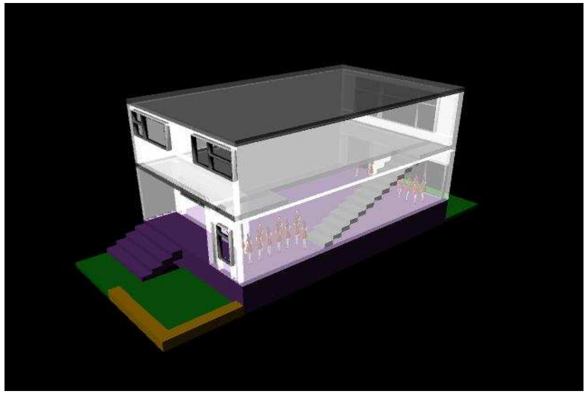
Evacuation Results

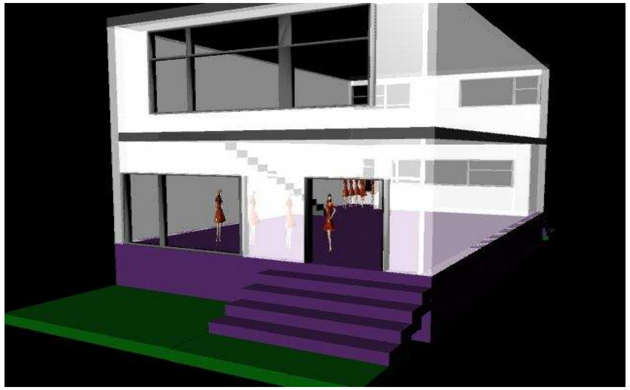




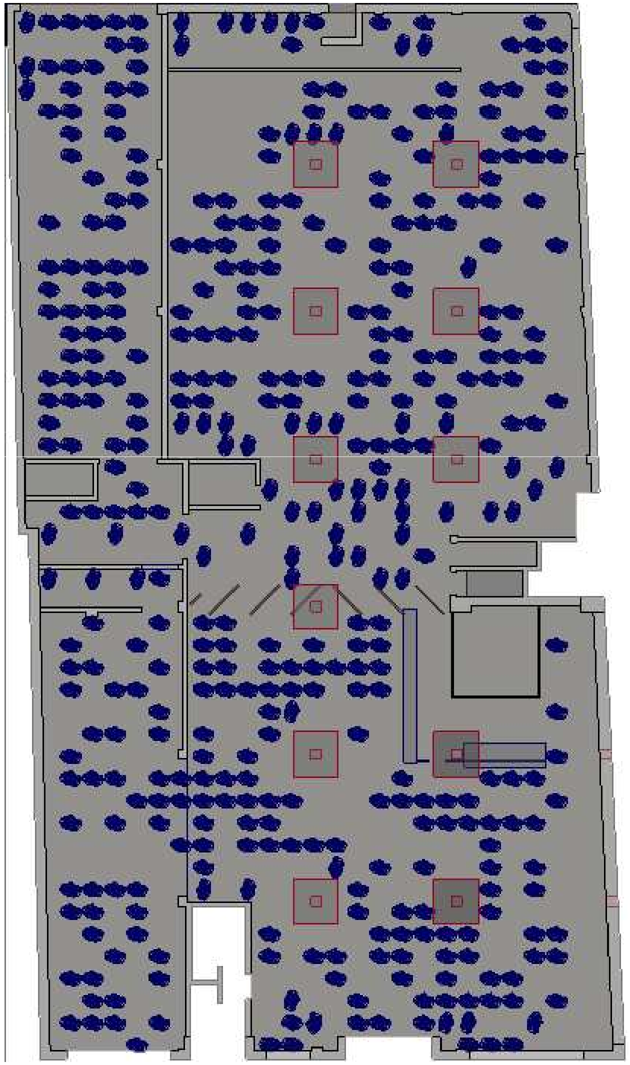


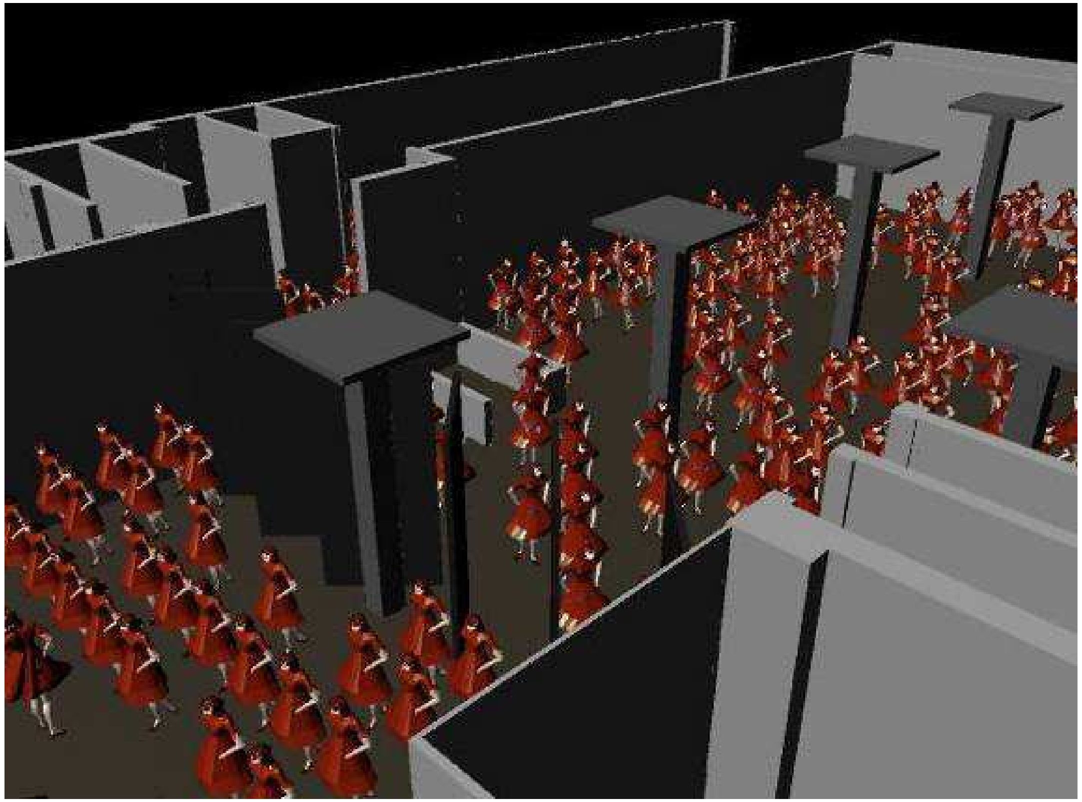
Evacuation Model

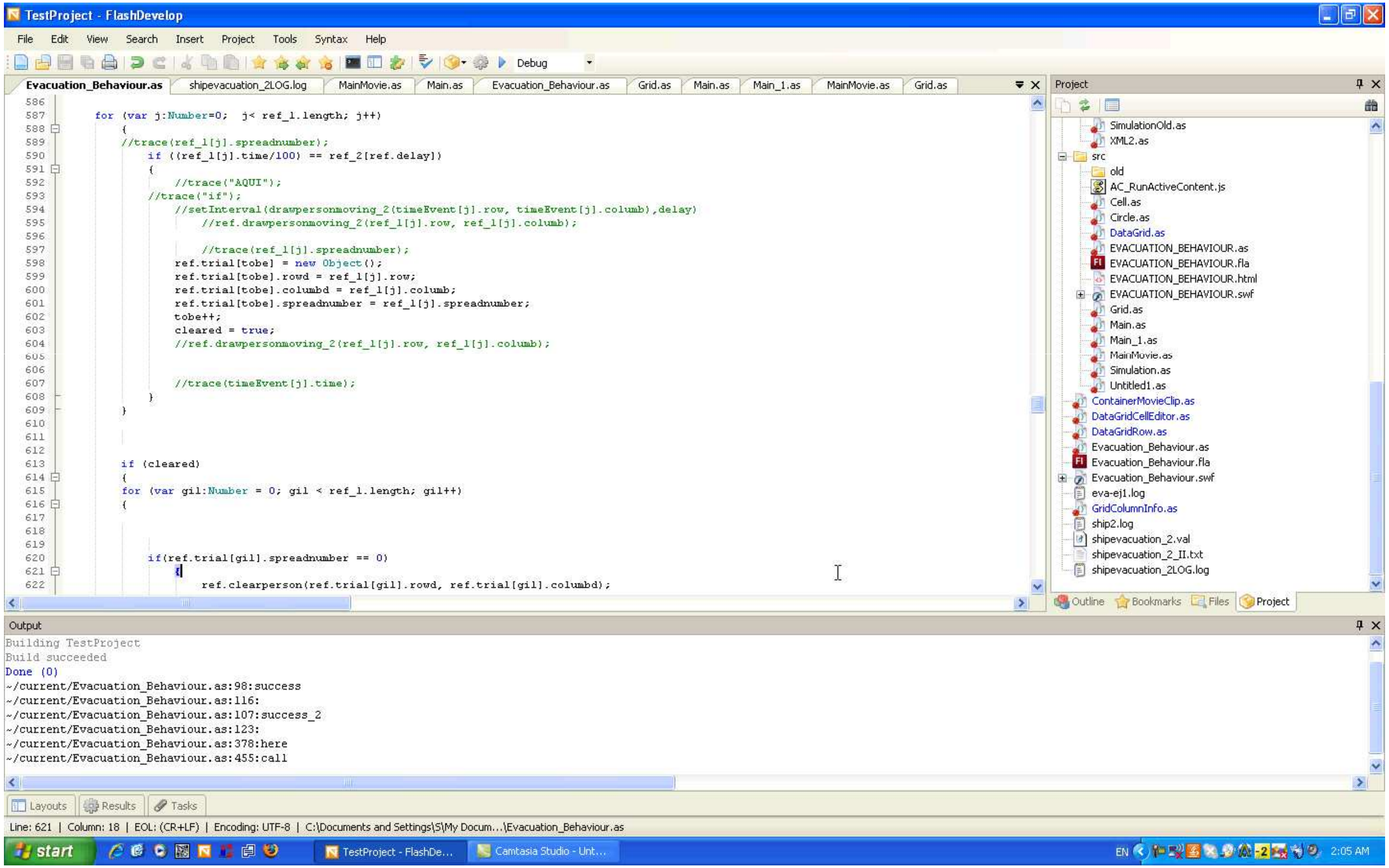




3D Visualization Scenario





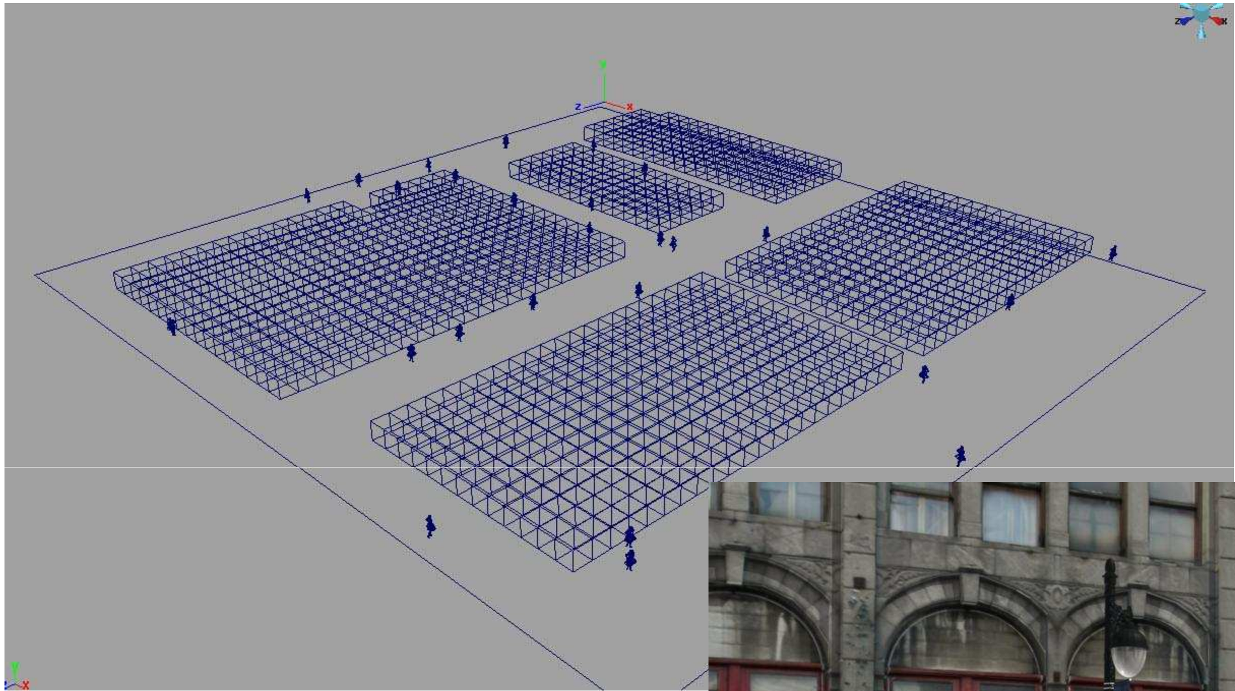

TestProject - FlashDevelop
File Edit View Search Insert Project Tools Syntax Help
Debug
Evacuation_Behaviour.as
shipevacuation_2LOG.log
MainMovie.as
Main.as
Evacuation_Behaviour.as
Grid.as
Main.as
Main_1.as
MainMovie.as
Grid.as
for (var j:Number=0; j< ref_l.length; j++)
{
//trace(ref_l[j].spreadnumber);
if ((ref_l[j].time/100) == ref_2[ref.delay])
{
//trace("AQUI");
//trace("if");
//setInterval(drawpersonmoving_2(timeEvent[j].row, timeEvent[j].columb),delay)
//ref.drawpersonmoving_2(ref_l[j].row, ref_l[j].columb);
//trace(ref_l[j].spreadnumber);
ref.trial[tobe] = new Object();
ref.trial[tobe].rowd = ref_l[j].row;
ref.trial[tobe].columbd = ref_l[j].columb;
ref.trial[tobe].spreadnumber = ref_l[j].spreadnumber;
tobe++;
cleared = true;
//ref.drawpersonmoving_2(ref_l[j].row, ref_l[j].columb);
//trace(timeEvent[j].time);
}
}
if (cleared)
{
for (var gil:Number = 0; gil < ref_l.length; gil++)
{
if(ref.trial[gil].spreadnumber == 0)
{
ref.clearperson(ref.trial[gil].rowd, ref.trial[gil].columbd);
Project
SimulationOld.as
XML2.as
src
old
AC_RunActiveContent.js
Cell.as
Circle.as
DataGrid.as
EVACUATION_BEHAVIOUR.as
EVACUATION_BEHAVIOUR.fla
EVACUATION_BEHAVIOUR.html
EVACUATION_BEHAVIOUR.swf
Grid.as
Main.as
Main_1.as
MainMovie.as
Simulation.as
Untitled1.as
ContainerMovieClip.as
DataGridCellEditor.as
DataGridRow.as
Evacuation_Behaviour.as
Evacuation_Behaviour.fla
Evacuation_Behaviour.swf
eva-ej1.log
GridColumnInfo.as
ship2.log
shipevacuation_2.val
shipevacuation_2_II.txt
shipevacuation_2LOG.log
Outline
Bookmarks
Files
Project
Output
Building TestProject
Build succeeded
Done (0)
~/current/Evacuation_Behaviour.as:98:success
~/current/Evacuation_Behaviour.as:116:
~/current/Evacuation_Behaviour.as:107:success_2
~/current/Evacuation_Behaviour.as:123:
~/current/Evacuation_Behaviour.as:378:here
~/current/Evacuation_Behaviour.as:455:call
Layouts
Results
Tasks
Line: 621 | Column: 18 | EOL: (CR+LF) | Encoding: UTF-8 | C:\Documents and Settings\S\My Docum...\Evacuation_Behaviour.as
start
TestProject - FlashDe...
Camtasia Studio - Unt...
EN
2:05 AM

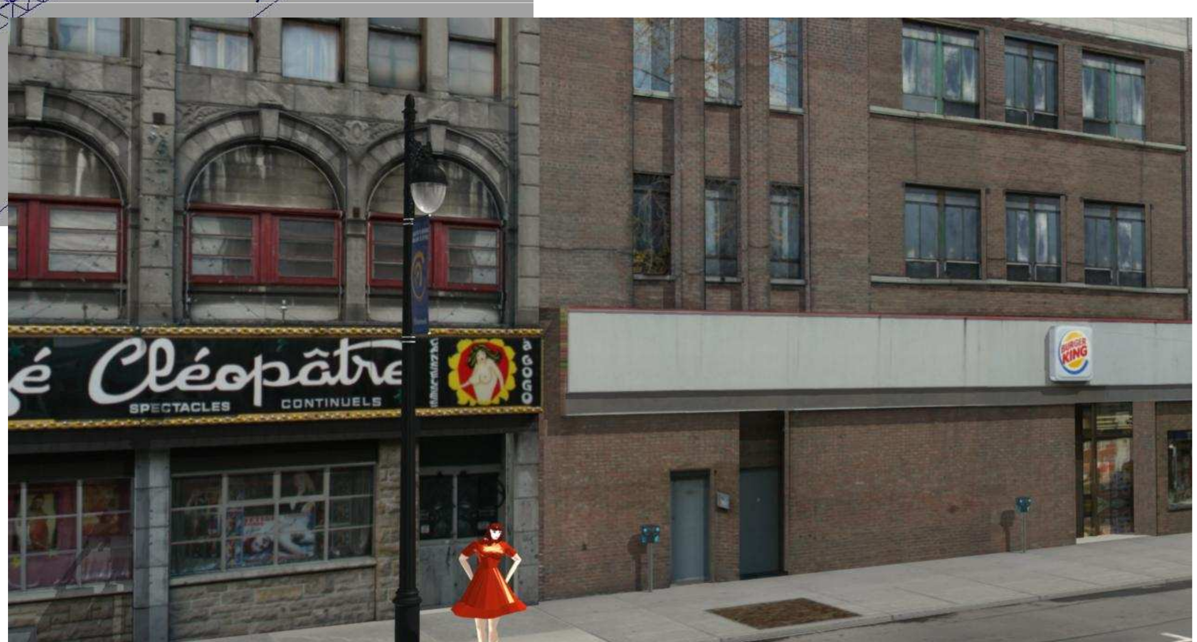
Advanced Visualization

Boulevard St. Laurent (MTL)

http://www.cims.carleton.ca/pose

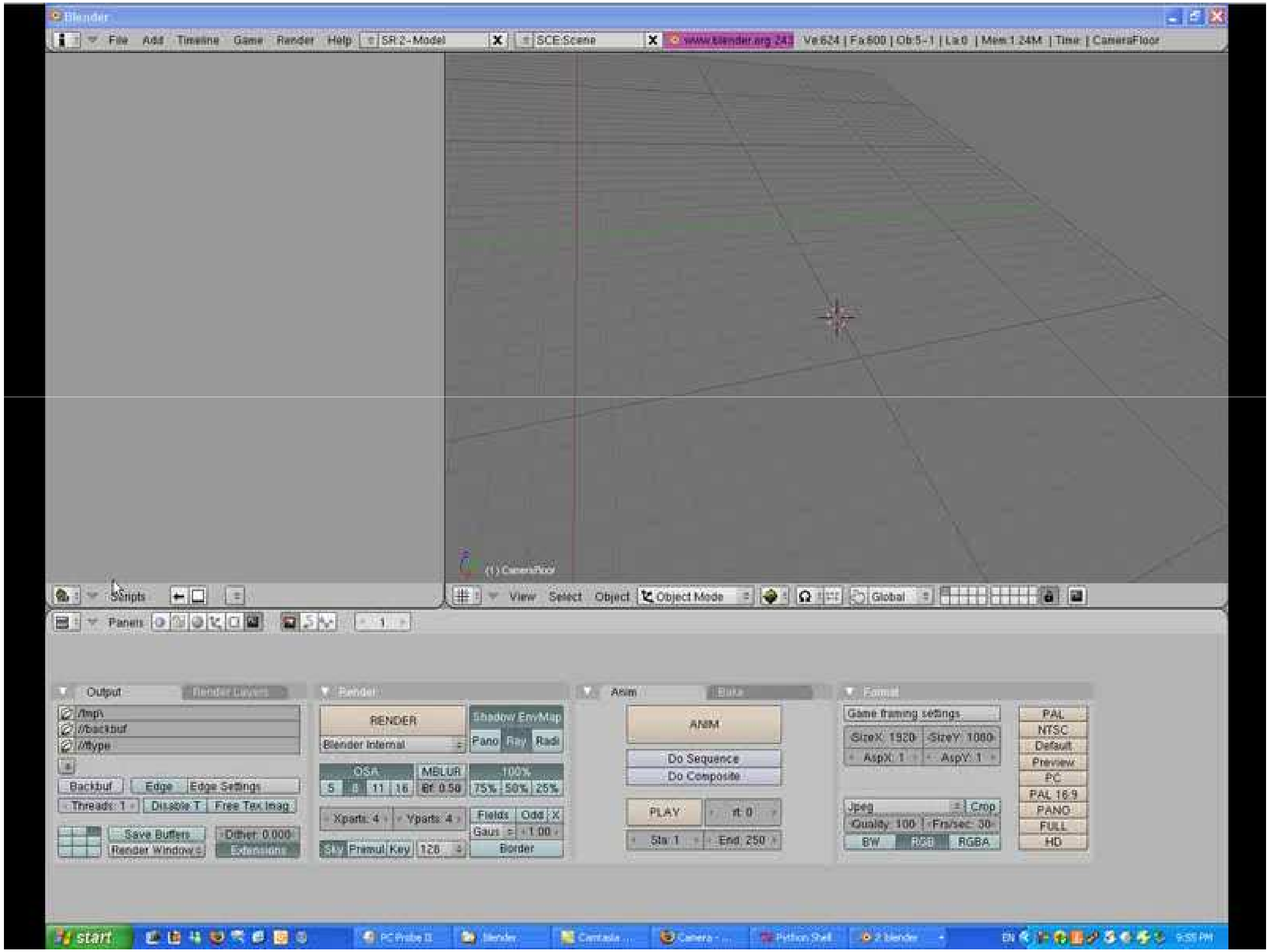
Evacuation in St. Laurent Blvd.

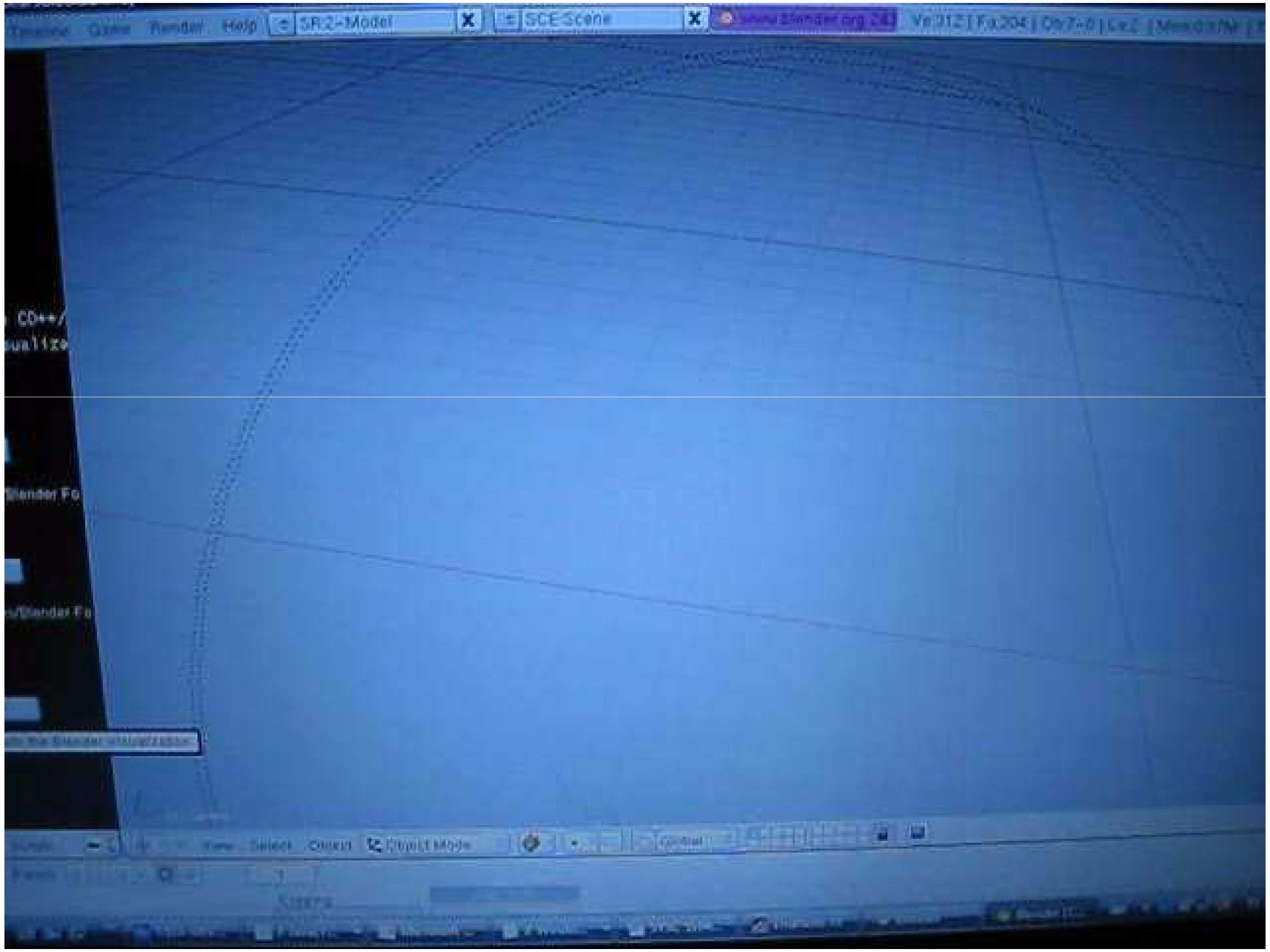




SAT Evacuation Animation

VIDEOS: youtube.com/arslab




SR:2-Model
SCE:Scene
Object Mode

Concluding remarks

- DEVS formalism: enhanced execution speed, improved model definition, model reuse.
- Hierarchical specifications: multiple levels of abstraction.
- Separation of models/simulators/EF: eases verification.
- Experimental frameworks: building validation tools
- Modeling using CD++: fast learning curve
- Parallel execution of models: enhanced speed
- The variety of models introduced show the possibilities in defining complex systems using Cell-DEVS.
- User-oriented approach. Development time improvement: test and maintenance.
- Incremental development

Further Information

http://cell-devs.sce.carleton.ca