Ministry of Education and Science of the Russian Federation
Peter the Great St. Petersburg State Polytechnic University
Institute of Computer Sciences and Technologies
Graduate School of Cyber-Physical Systems and Control

Technology Implementation Report

Multi-Agent Software Tools for Cognition-Based System Design
Discipline: Cognition-Based Multi-Agent Systems
15 May 2017

Student Group: 13541/8

Christopher W. Blake

Professor

Kapralov V.G.

St. Petersburg
2017

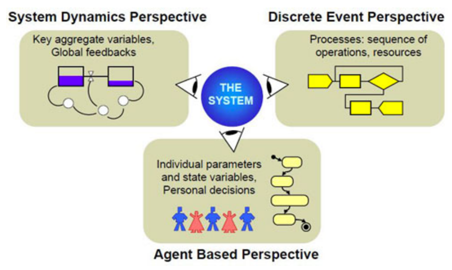
Contents

Overview 3
Comparison Points 3
Architecture 3
Agent Structure 4
Communication Protocols 4
Software Platforms 4
AgentSheets 6
AnyLogic 6
Java Agent Development Framework (JADE) 7
NetLogo 7
Behaviour Composer 7
Cougaar 8
Soar 8
StarLogo 9
iGEN 9
Implementation Example (AnyLogic) 10
Road Network Creation 10
Create Traffic Patterns 12
Add Traffic Controllers 15
Preview Simulation 16
Conclusion 16
References 17

Overview

There are many different tools for the design and development of multi-agent cognition-based systems. As of this writing, Wikipedia lists 13 pages for agent based programming, 20 pages for agent-based software, and 3 pages for agent-oriented programming languages. In addition, it lists 95 pages dedicated to different applications, protocols, and models of multi-agent systems. There are clearly many parallel and narrowly developed projects for particular domains such as biology, economics, logistics, or sociology. Unfortunately, a clear standard has not yet emerged, indicating that the field is not yet mature.

There is however a small-enough subset of more commonly utilized tools that can be reviewed. Of the hundreds of tools listed, less than 10 had their own meaningful content pages or developed websites. These were reviewed and their original websites were consulted to gain an understanding of their diversity, capability, and overall following. Of these software options all were Java based except one, and there are clearly three categories: Education, Standard Performance, and High Performance.


System Dynamics Perspective
Key aggregate variables, Global feedbacks
Discrete Event Perspective
Processes: sequence of operations, resources
THE SYSTEM
Individual parameters and state variables, Personal decisions
Agent Based Perspective
Figure 1: AnyLogic business simulation

Before discussing these software packages and toolkits, an overview of multi-agent systems will initially be provided. This is intended to act as a reminder of what such platforms and toolkits should provide. Each of the meaningful software tools is introduced and a summary is provided. Finally, an implementation example using AnyLogic is shown, which simulates cars driving through a city.

Comparison Points

A multi-agent system for cognition-based activities has to be flexible and consider various high-level objectives. Without meeting such objectives, it would be considered specific to a particular domain, which defeats the essential robust and open nature of a multi-agent system. Below is a possible list of considerations for such a system.

1. Different and future systems of intelligence.
2. Different communication protocols.
3. Different agent types.
4. Heterogeneity of open systems and other architectures.
5. Complete or partial decentralization, or monolithic structures.
6. Addition and subtraction of agents.
7. Addition and subtraction of system resources.

Architecture

The architecture environments can be discreet, virtual, or continuous and it can be described by the following properties.

1. Accessibility – possibility level of gathering information about the environment.
2. Determinism - if an action performed in the environment causes a definite effect.
3. Dynamics - how many entities influence the environment at a given moment.
4. Discreteness - if the number of possible actions in the environment is finite.
5. Episodicity - if agent actions in a time period influence other periods.
6. Dimensionality - if spatial characteristics are important factors of the environment, and if the agent considers space in its decision making.

Agent Structure

The definition of an agent is critical for multi-agent systems, since the architecture is built around them. As such there are typically three categories of agents: Software, Robotic, and Human. Additionally, they may be passive, which simply exist, or active agents which have simple or complex goals. Finally, there are the three fundamental components of being an agent, listed below.

1. Autonomy – to be able to perform completely, or nearly completely, alone. It may even be considered self-aware.
2. Local Perception – the agent is only capable of perceiving a portion of the environment relative to its duties and tasks.
3. Decentralization – There is no predetermined controlling agent. Although dynamically this sort of control may develop within a simulation.

Communication Protocols

Generally speaking, the majority of the projects and platforms use self-made proprietary protocols. However, a few more options appear to follow or at least support **Agent Communication Language (ACL)** and **Knowledge Query Manipulation Language (KQML)**.

ACL is the replacement for KQML and both were developed by an organization known as FIPA (Foundation for Intelligent Physical Agents). They are both based on speech act theory and define performatives also known as communicative acts. The actual content of a performative varies per project because of project goals, ontology, and overall design.

FIPA is a standards body for setting and developing standards for heterogeneous agent-based systems. It was founded in 1996 to create a full set of standards for implementing systems for agents to run within as well how to specify how agents communicate and interact. Unfortunately, it has never gained strong corporate commercial support and standards were transferred to an IEEE committee.

Software Platforms

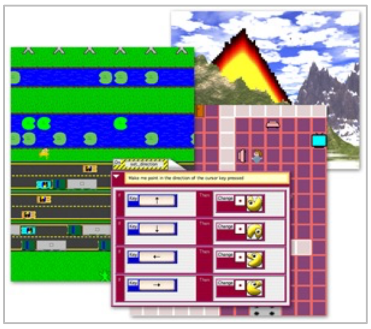
As previously mentioned, there are many different software platforms currently available, many of which are likely not being used outside of a small domain. Below is a list of platforms that were found. Most are not large enough to have their own Wikipedia page. However, the items in bold will be discussed further, as they have some larger degree of following or are significantly developed.

1. Agent Anytime Anywhere (AAA)
2. Agent Building and Learning Environment (ABLE)
3. AgentBuilder
4. AgentService
5. **AgentSheets**
6. Altreva Adaptive Modeler
7. **AnyLogic**
8. AOR Simulation
9. Ascape
10. BDI5Jade
11. **Behaviour Composer**
12. Boris
13. Brahms
14. Breve
15. Common-pool Resources and Multi-Agent Systems (CORMAS)
16. Construct
17. **Cougaar**
18. CybelePro

19. DALI
20. Descartes
21. DeX
22. DigiHive
23. Distributed Operator Model (D-OMAR)
24. ECHO
25. EVE
26. FAMOJA
27. FLAME
28. FLAME GPU
29. FLUXY
30. Framsticks
31. GAMA
32. GPU Agents
33. GROWlab
34. ICARO-T
35. **iGen**
36. Insight Maker
37. JABM
38. **Java Agent Development Framework (JADE)**
39. Jade's sim++
40. JAMEL
41. Janus
42. JAS-mine
43. Jason Interpreter of AgentSpeak
44. Java Auction Simulator API (JASA)
45. Java Enterprise Simulator (jES)
46. JCA-Sim
47. jEcho
48. JESS
49. JIAC
50. Laboratory for Simulation Development
51. MacStarLogo
52. MAGSY
53. MASON
54. Mesa
55. Micro and Multilevel Modelling Software (MIMOSE)
56. Multi Agent Development Kit (MaDKit)
57. Multi-Agent Modeling Language (MAML)
58. Multi-Agent Simulation Suite (MASS)
59. Multi-Agent Simulations for the Social Sciences (MAS-SOC)
60. Multimodeling Object-Oriented Simulation Environment (MOOSE)
61. **NetLogo**
62. Object Based Environment for Urban Simulation (OBEUS)
63. Omonia
64. OpenOME
65. OpenStarLogo
66. oRIS
67. Political Science Identity (PS-I)
68. Repast
69. Shell for Simulate Agent Systems (SeSAm)
70. SimAgent
71. SimBioSys
72. SimPack
73. SimPlusPlus
74. **Soar**
75. Spatial Modeling Environment (SME)
76. **StarLogo**
77. StarLogo TNG
78. StarLogoT
79. Strictly Declarative Modeling Language (SDML)
80. Sugarscape
81. Swarm
82. System Effectiveness Analysis Simulation (SEAS)
83. TerraME
84. Tryllian Agent Development Kit
85. VisualBots
86. VSEit
87. Xholon
88. ZEUS

AgentSheets

Agentsheets is a cyber-learning tool focused on education of programming through game design. It is mostly used in middle and high school levels and supported by the National Educational Technology Standards (NETS).



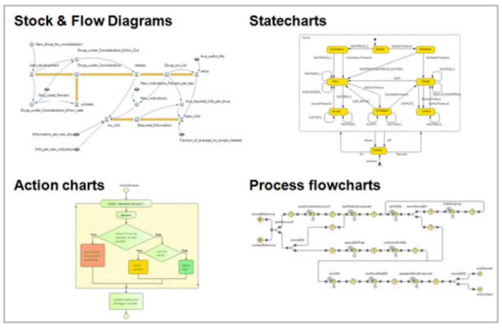
The programming language is primarily drag-and-drop at beginning levels to produce games equivalent to Frogger. The beginners environment is separated into a grid which can contain agents represented in various ways such as text, numbers, images, or animations. However, more advanced games with access to true java code and artificial intelligence can also be produced.

Outside of education, it is often used for modeling scientific phenomena involving tens of thousands of agents and scenarios such as mud slides, bridges collapsing, and ecosystems.

Primary Domain:	Education
First Release:	1991
Last Release	May, 2014
Main Platform:	Java
Licenses:	Proprietary

AnyLogic

AnyLogic is a commercial level simulation package without any particular domain. It even offers a free license for educational purposes and is built in a modular way using Java, to support extensibility. Its roots are from Saint Petersburg Technical University in the Distributed Computer Network research group.



It was named AnyLogic because it supports three different approaches to system design: System Dynamics, Discrete Event Simulation, and Agent-Based Modeling. Any of these approaches may be used or combined. However, its focus and architecture are primarily related to business concepts such as: Market and Competition, healthcare, Manufacturing, Supply Chain, Logistics, Retail, Business Processes, Social and Ecosystem Dynamics, Defense, Project/Asset Management, IT Infrastructure, Pedestrian Dynamics and Traffic, Aerospace, and Photovoltaics.

Primary Domain:	Business
First Release:	2000
Last Release	2016
Main Platform:	Java
Licenses:	Proprietary, Free

Java Agent Development Framework (JADE)

JADE is not a stand-alone software tool. It is instead a free framework toolkit developed for Java. Additionally, it follows the FIPA-ACL standard communication protocol. Hence it could be extended to communicate with other systems, if necessary.

Primary Domain:	None
First Release:	2000
Last Release	2015
Main Platform:	Java
Licenses:	Free

Its primary focus is to ease development of a standards-based simulation environment. It does this by providing a standardized environment of agent execution, agent design/communication, and a toolkit for management of intelligent agents.

There are two special agents required by all initialized environments:

1. Directory Facilitator Agent – manages agent availability.
2. Agent Management System – allows creation and destruction of agents and containers, and is the only agent allowed to stop the platform.

As JADE is a toolkit, it provides a superclass called Agent. All user created agents inherit from this class and hence additional or custom functionality can be created.

NetLogo

NetLogo is focused toward students and teachers, but is also used in research. Its focus is on simulating natural and social phenomena and is normally used for studying complex systems vs time. As instructions are given to agents, both micro- and macro-level patterns can be discovered.

It was developed upon the Logo programming language which is meant to be easily accessible to non-programming-experienced users such as children and domain experts. The basic building blocks are known as turtles, patches, links, and observers.

Primary Domain:	Psychology
First Release:	1999
Last Release	2016
Main Platform:	Java
Licenses:	GPL

The library toolkit offers many predefined controls such as switches, sliders, choosers, and inputs, and also supports creation of custom controls.

Behaviour Composer

Behaviour Composer is actually an extension and automation tool created for running NetLogo agent models. It was primarily developed for the Modelling4All project at University of Oxford. It enables simplified creation of defined micro-behaviours, which can be further customized if desired. These newly defined behaviours may also be shared on the web via URLS.

Primary Domain:	Psychology
First Release:	-
Last Release	-
Main Platform:	Java
Licenses:	BSD

From a technical side, the interface runs on Google's web toolkit and app engine, but everything runs within a java applet in a web browser on the user's computer.

Cougaar

Is a military-grade research project converted to commercial use. It is an open source toolkit/architecture which includes infrastructure and core services.

Primary Domain:	None
First Release:	1996
Last Release	2008
Main Platform:	Java
Licenses:	Open Source

It was initially developed by DARPA to use cognitive agents to solve military logistics problems. Even though its focus was logistics, it was developed in a generic manner to support other domains. As such, and because of the high levels of funding it received, it is considered to represent one of the most advanced reasoning and intelligent automation systems available. The Department of Defense converted the platform into a business solution, known as ActiveEdge, for building applications as robust as military applications.

The agents and environments are separate items and are possible to be developed independently. However, they must be combined during runtime. Some of the example services provided include Blackboard Publishing (subscription model), HTTP servlets, Knowledge Representation systems, and Agent Coordination Techniques.

Soar

Soar is an architecture focused on approximating rational behavior, with focus on functionality and performance by ensuring all primitive capabilities are available to realize human-level abilities. All decisions are made using short term sensory data and long term knowledge data. The goal is to create an intelligent agent that is as generic as possible.

Primary Domain:	None
First Release:	1983
Last Release	2014
Main Platform:	C/C++
Licenses:	BSD

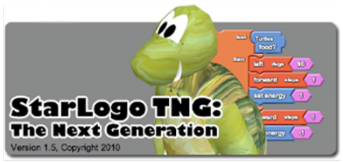
There is a single framework with built-in subcomponents for handling all major subtasks. Such subtasks include a single representation of knowledge (temporary and permanent), a single mechanism for generating goals (and subgoals), and a single learning mechanism. The learning mechanism has additionally been adjusted to support simultaneous different methods such as chunking, reinforcement, episodic, and semantic learning, all at runtime.

The platform offers primitive capabilities which can be combined to create human-level capabilities such as reactive decision making, situational awareness, deliberate reasoning and comprehension, planning and all forms of learning.

It incorporates a custom markup language known as Soar Markup Language (SML) which can be used for interfacing with other programing languages such as Java and Python.

StarLogo

StarLogo is an extension of the programming language Logo and is designed for educational purposes, in particular the study of decentralized systems. There are many flavors of StarLogo, which are listed below. Additionally, it is similar to NetLogo.



1. MacStarLogo – modified to run natively on Macintosh computers.
2. OpenStarLogo – a semi-open source version.
3. StarLogo TNG – inclusion of a 3D world, OpenGL graphics, and block structures.
4. StartLogo Nova – block structures and 3D visualization, but in a web browser.

Primary Domain:	Education
First Release:	-
Last Release	2011
Main Platform:	Java
Licenses:	BSD

iGEN

iGEN is a benchmarking tool used for comparing different multi-agent systems. There are seven different subcomponents each focused on a different benchmark element. Unfortunately, there is limited or no documentation easily available for its usage.

1. iGenCPU – CPU and fractal generation.
2. iGenRAM – Memory simulation of lottery
3. iGEnOLTP – OLTP database
4. iGENBATCH – BATCH database
5. iGENBIDW – Business Intelligence
6. iGENLDAPs – LDAP searching
7. iGENLDAPsm – LDAP SiteMinder emulation

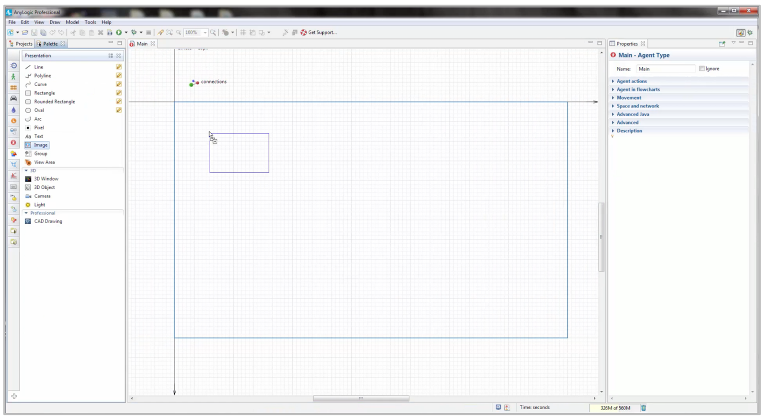
Implementation Example (AnyLogic)

AnyLogic is probably the most accessible and well-developed option from the reviewed software platforms. As such, it is the most likely candidate for commercial applications. To demonstrate AnyLogic, a model for simulating road traffic is described. This model provides insights about the following items:

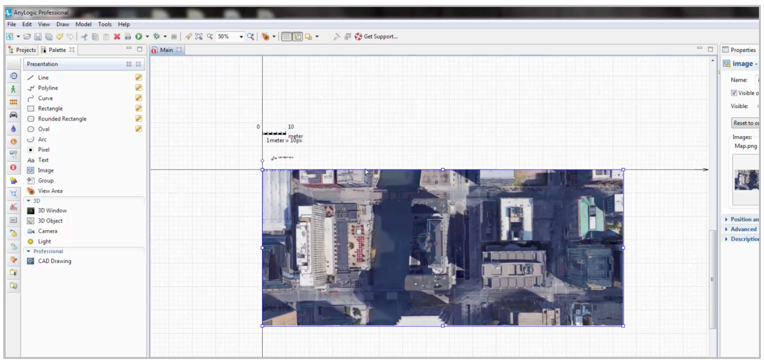
1. Road network creation based on satellite image.
2. Traffic flow logic.
3. Traffic lights and yield signs setup.

Road Network Creation

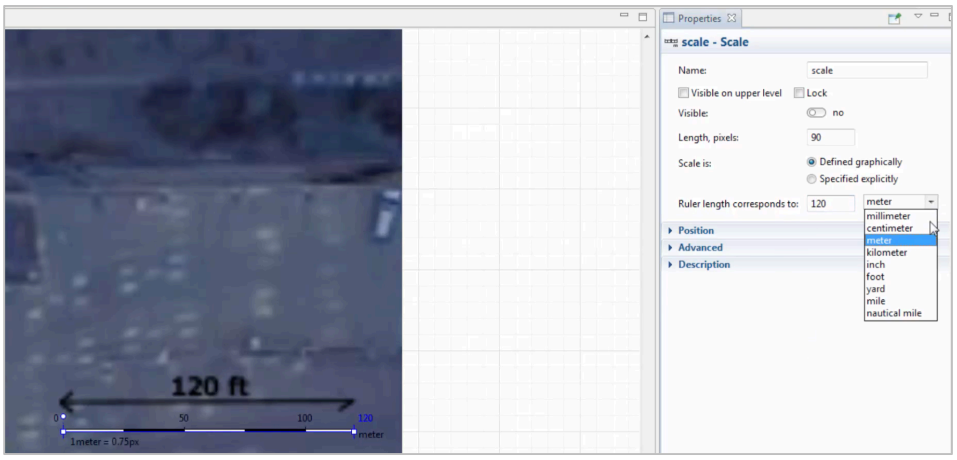
1. Create a new blank model.



2. Import a satellite image of the city.



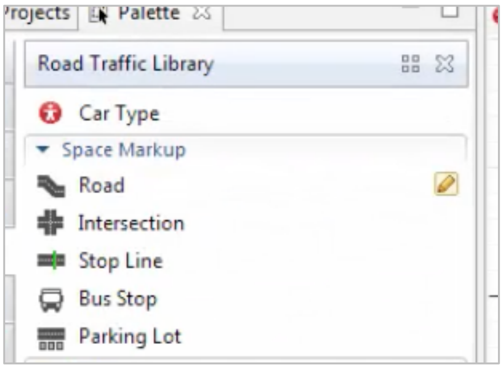
3. Adjust simulation scale to match image scale.

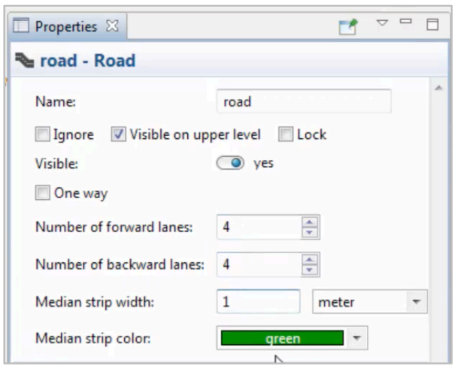


scale - Scale

Name: scale
Visible on upper level Lock
Visible: no
Length, pixels: 90
Scale is: [✓ Defined graphically] Specified explicitly
Ruler length corresponds to: 120 meter

4. Create roads using traffic library tools.




Properties

road - Road

Name: road

☐ Ignore ☑ Visible on upper level ☐ Lock

Visible: yes

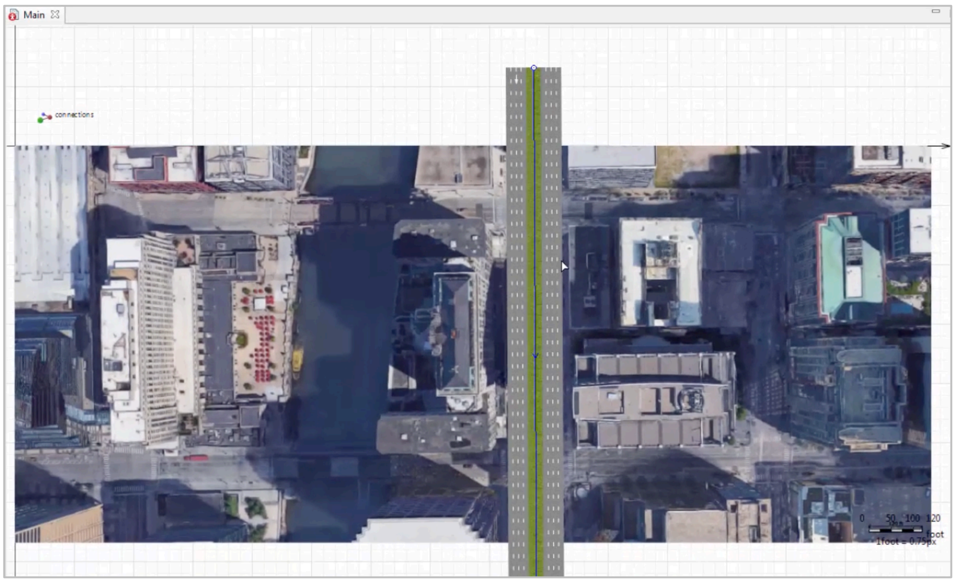
☐ One way

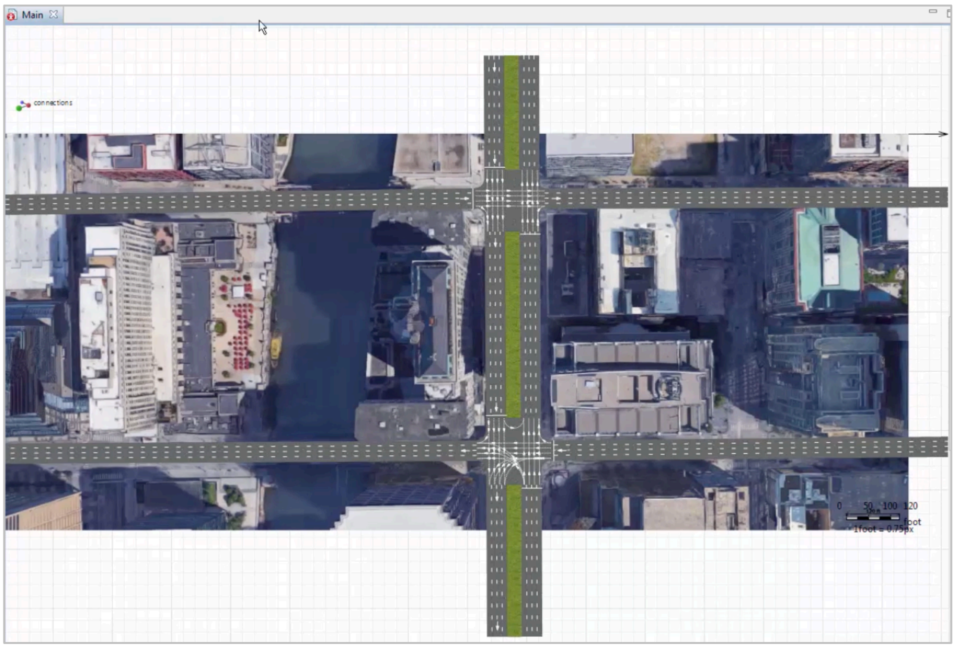
Number of forward lanes: 4

Number of backward lanes: 4

Median strip width: 1 meter

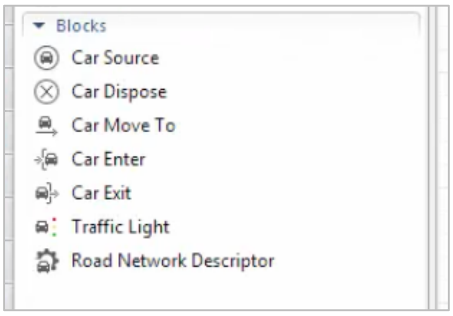
Median strip color: green



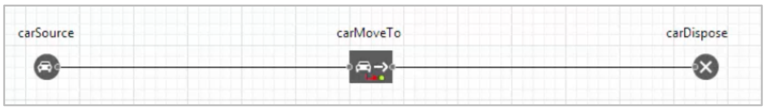


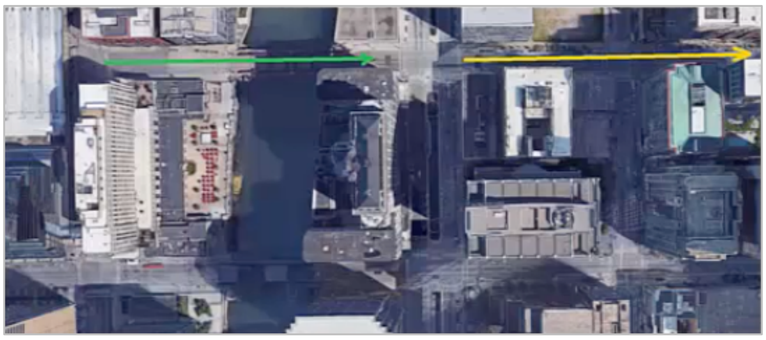
Create Traffic Patterns

1. Create car block flow charts for each traffic pattern, using traffic library.

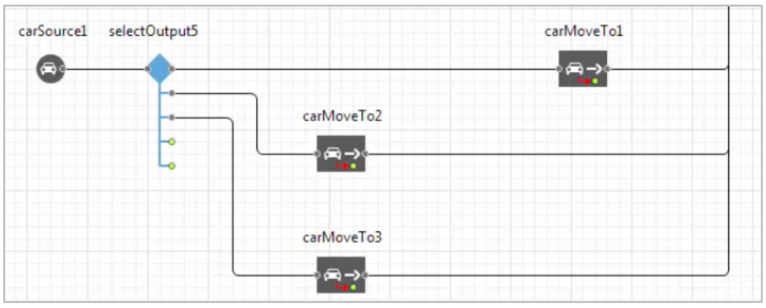


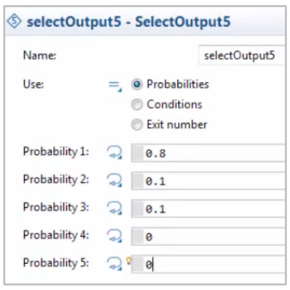
   a. Traffic Pattern: Top-left => Straight





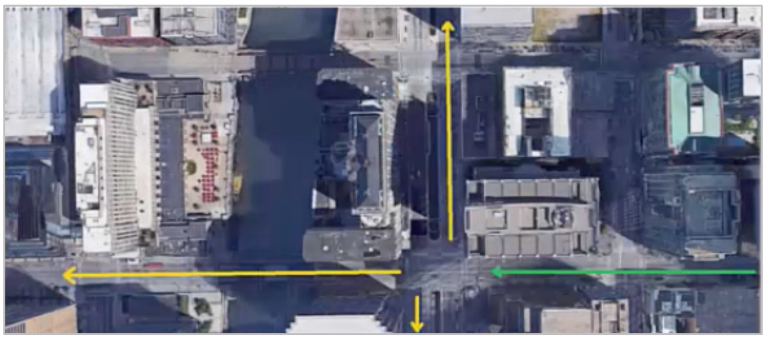
   b. Traffic Pattern: Bottom-right => Turn Right, Turn Left, Straight



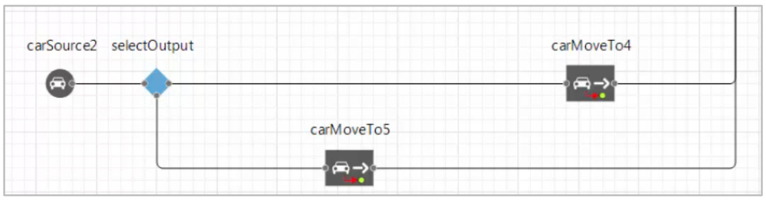


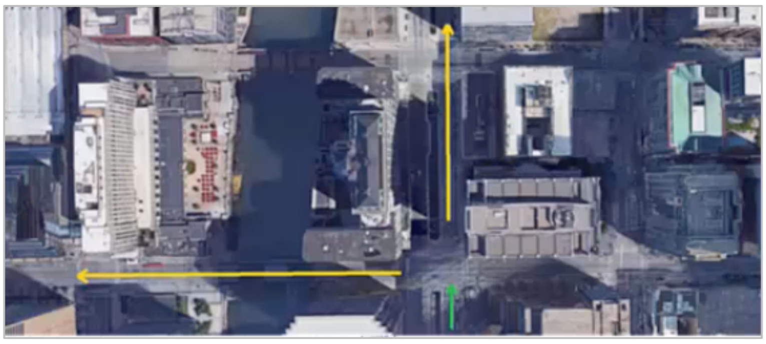
selectOutput5 - SelectOutput5

Name:	selectOutput5
Use:	[✓ Probabilities] Conditions Exit number
Probability 1:	0.8
Probability 2:	0.1
Probability 3:	0.1
Probability 4:	0
Probability 5:	0

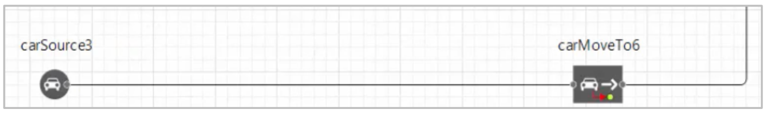


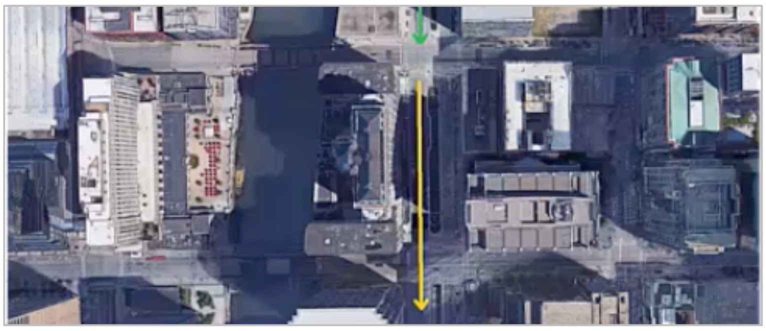
c. Traffic Pattern: Bottom => Turn Left, Straight



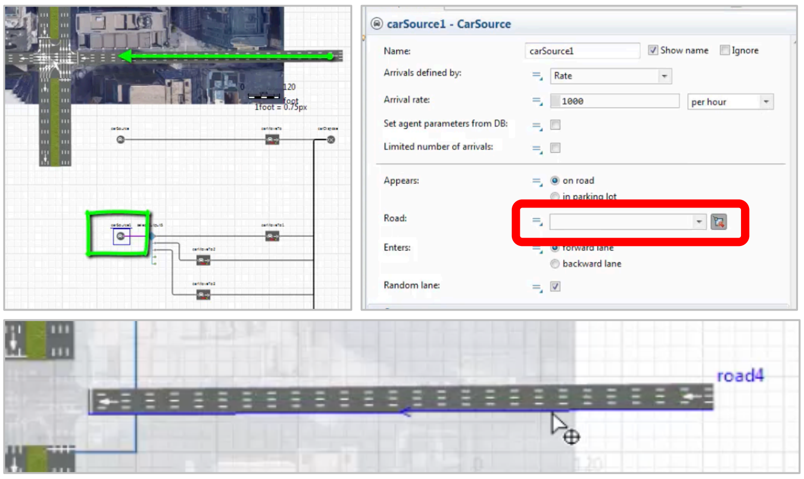


d. Top => Straight





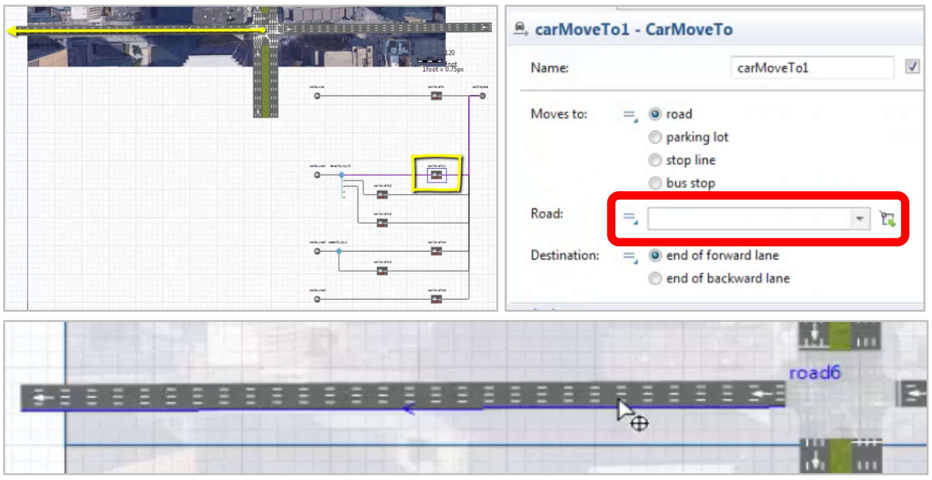
2. Associate car blocks with roads. (Repeat for each traffic pattern flow chart.)
   a. Car Source



carSource1 - CarSource

Name:	carSource1 [✓ Show name] Ignore
Arrivals defined by:	Rate
Arrival rate:	1000 per hour
Set agent parameters from DB:	☐
Limited number of arrivals:	☐
Appears:	[✓ on road] in parking lot
Road:	
Enters:	[✓ forward lane] backward lane
Random lane:	✓

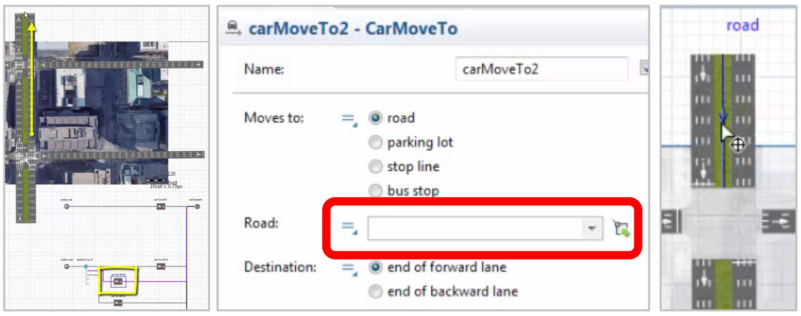
   b. Car Movement Option 1 (Straight)



carMoveTo1 - CarMoveTo

Name:	carMoveTo1
Moves to:	[✓ road] parking lot stop line bus stop
Road:	
Destination:	[✓ end of forward lane] end of backward lane

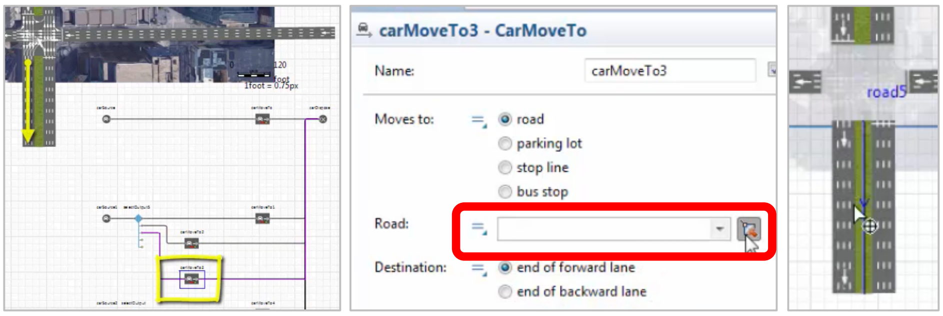
   c. Car Movement Option 2 (Turn Right)



carMoveTo2 - CarMoveTo

Name:	carMoveTo2
Moves to:	[✓ road] parking lot stop line bus stop
Road:	
Destination:	[✓ end of forward lane] end of backward lane

   d. Car Movement Option 3 (Turn Left)

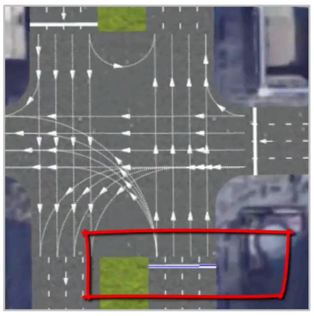
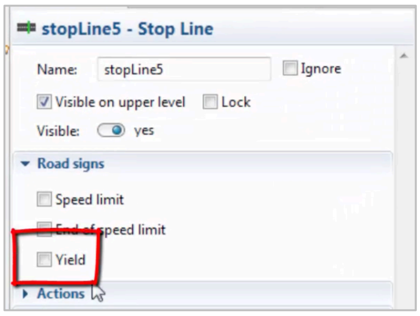


carMoveTo3 - CarMoveTo

Name:	carMoveTo3
Moves to:	[✓ road] parking lot stop line bus stop
Road:	
Destination:	[✓ end of forward lane] end of backward lane

Add Traffic Controllers

1. Yield Signs – Click a stop line and check the “Yield” box.

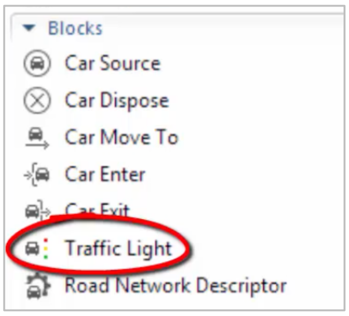



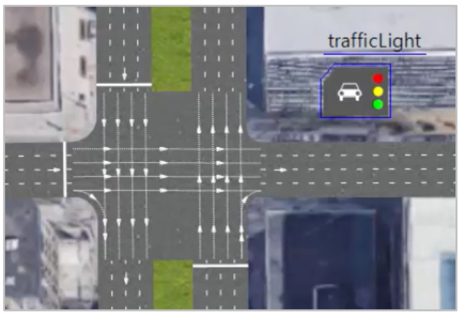
stopLine5 - Stop Line

Name: stopLine5 ☐ Ignore
☑ Visible on upper level ☐ Lock
Visible: yes

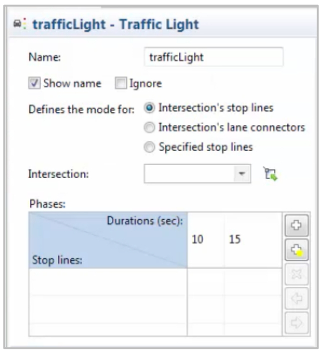
Road signs
☐ Speed limit
☐ End of speed limit
☐ Yield

2. Traffic Lights
    a. Add a stop light block near an intersection.





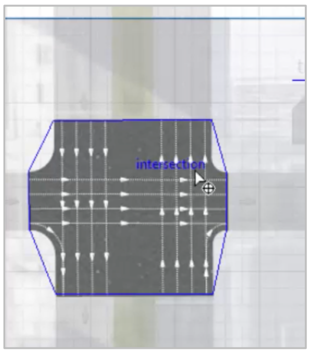
    b. Specify the intersection to control.



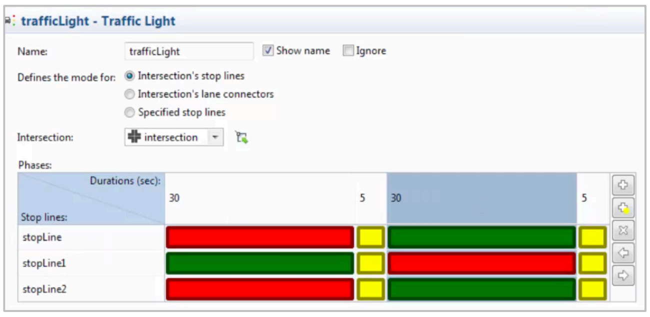
trafficLight - Traffic Light

Name: trafficLight
☑ Show name ☐ Ignore
Defines the mode for: (● Intersection's stop lines) ○ Intersection's lane connectors ○ Specified stop lines
Intersection:
Phases:

Durations (sec): / Stop lines:	10	15



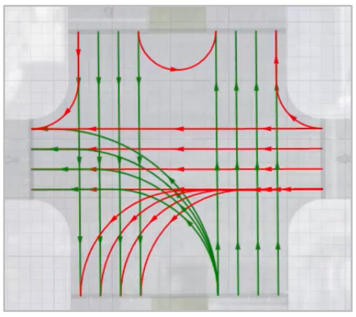
    c. Adjust stop light phases.



trafficLight - Traffic Light

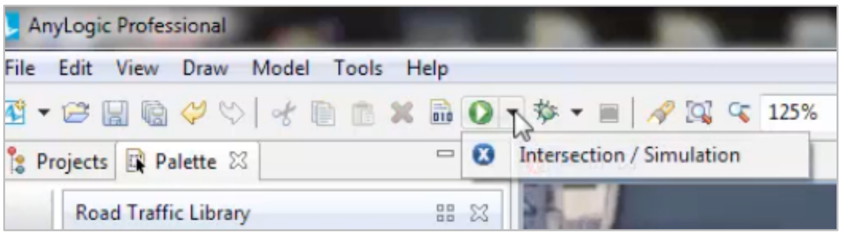
Name: trafficLight ☑ Show name ☐ Ignore
Defines the mode for: (● Intersection's stop lines) ○ Intersection's lane connectors ○ Specified stop lines
Intersection: intersection
Phases:

Durations (sec): / Stop lines:	30	5	30	5
stopLine	🟥	🟨	🟩	🟨
stopLine1	🟩	🟨	🟥	🟨
stopLine2	🟥	🟨	🟩	🟨

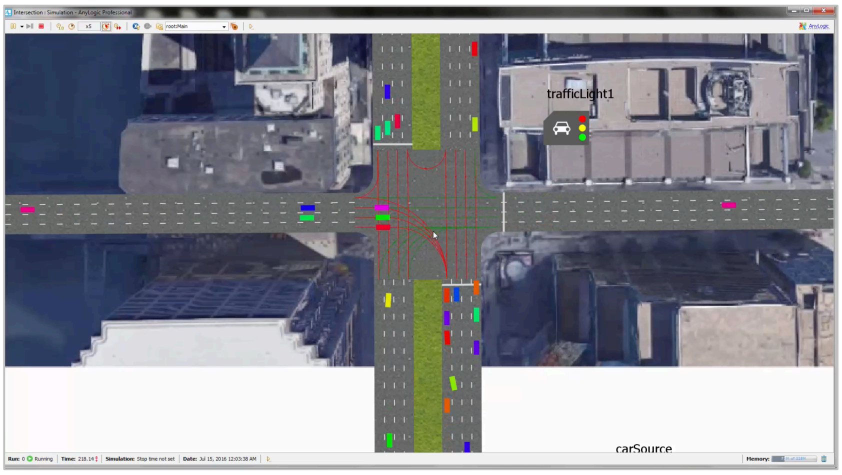


Preview Simulation

1. Start Simulation


AnyLogic Professional
File Edit View Draw Model Tools Help
125%
Projects
Palette
Intersection / Simulation
Road Traffic Library

2. Watch Simulation



Conclusion

Of the different software tools and packages reviewed, there appear to be 3 major different categories that would influence a decision. Additionally, all packages are Java based, except one which is C/C++ based. The majority of packages are focused on education. Two have standard, as expected, components and thus qualify as standard performance. The final two are very mature with many capabilities so they can be labeled as high performance. However, it should be noted that there is no common protocol amongst any of the software discussed, preventing natural cross communication capabilities.

1. Educational – AgentSheets, Behaviour Composer, NetLogo, StarLogo
2. Standard – JADE, Soar
3. High Performance – AnyLogic, Cougaar

Finally, an example implementation of AnyLogic was shown, since it is the most likely candidate for appearing in commercial applications. This example is able to be created in less than 30 minutes and simulates car traffic through a city. It includes three origin points for cars, two stop light systems, and one yield sign. After configuration, a live simulation is viewable, allowing easy previewing.

References

1. **Agent Architecture**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Agent_architecture
2. **Agent-Based Model**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Agent-based_model
3. **AgentSheets**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/AgentSheets
4. **AgentSheets**. Retrieved April 15, 2017, from http://www.agentsheets.com
5. **AnyLogic**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/AnyLogic
6. **AnyLogic**. Retrieved April 15, 2017, from http://www.anylogic.com
7. **Behaviour Composer**. Retrieved April 15, 2017, from http://m.modelling4all.org
8. **Behaviour Composoer**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/Behaviour_Composer
9. **Category: Multi-Agent Systems**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Category:Multi-agent_systems
10. **Cognitive Architecture**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Cognitive_architecture
11. **Comparison of Agent-Based Modeling Software**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software
12. **Cougaar**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/Cougaar
13. **Cougaar**. Retrieved April 15, 2017, from http://www.cougaarsoftware.com
14. **Discrete Event Simulation**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Discrete_event_simulation
15. **Distributed Artificial Intelligence**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Distributed_artificial_intelligence
16. **FIPA**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/FIPA
17. **iGen**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/IGen
18. **Intelligent Agent**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Intelligent_agent
19. **JADE**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/Java_Agent_Development_Framework
20. **JADE**. Retrieved April 15, 2017, from http://jade.tilab.com
21. **Multi-Agent System**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Multi-agent_system
22. **NetLogo**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/NetLogo
23. **NetLogo**. Retrieved April 15, 2017, from https://ccl.northwestern.edu/netlogo

24. **Soar**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/Soar_(cognitive_architecture)
25. **Soar**. Retrieved April 15, 2017, from http://soar.eecs.umich.edu
26. **Software Agent**. (n.d.) In Wikipedia. Retrieved April 14, 2017, from https://en.wikipedia.org/wiki/Software_agent
27. **StarLogo Nova**. Retrieved April 15, 2017, from http://www.slnova.org
28. **StarLogo TNG**. Retrieved April 15, 2017, from http://education.mit.edu/starlogo-tng
29. **StarLogo**. (n.d.) In Wikipedia. Retrieved April 15, 2017, from https://en.wikipedia.org/wiki/StarLogo