JaamSim
User Manual

Ausenco

The forefront of simulation

1 - Introduction

1.1 - Background

JaamSim (Java Animation Modelling and Simulation) is a discrete-event simulation environment developed by Ausenco as the foundation of all its simulation applications. JaamSim represents the lessons learned from the simulation projects carried out by Ausenco around the world for more than 35 years. Further background information on JaamSim can be found in the article "Discrete-Event Simulation in Java – a Practitioner's Experience", which is available on the Ausenco website: www.ausenco.com/simulation.

This User Manual documents the user interface and basic objects provided with JaamSim. These features are common to all simulation models created with this software. Documentation for application-specific objects such as those found in Ausenco's Transportation Logistics Simulator (TLS) is found in the manuals for the corresponding modules.

JaamSim is free open source software, licensed under GPLv3. The latest version of the software and manuals can be downloaded from the JaamSim website: www.jaamsim.com. The source code is published on GitHub: www.github.com/AusencoSimulation/JaamSim.

1.2 - Features

The key feature that makes JaamSim different from commercial simulation software is that it allows the user to develop new palettes of high-level objects for a given application in a standard programming language (Java). Objects created in this way automatically have 3D graphics, are available in the drag-and-drop interface, and have their inputs editable through the Input Editor. Users can focus on the logic for their objects without having to program a user interface, graphics, input/output processing or other model development tools.

Coding for new objects is done in Java using standard development tools such as Eclipse. There is no need for the specialised simulation languages, process flow diagrams, or scripting languages used by commercial off-the-shelf simulation software. Model logic can be coded directly in either an event- or process-oriented style using a few simple classes and methods provided by JaamSim.

JaamSim provides all the generic functionality needed for any simulation model:

- Controls for launching and manipulating simulation runs
- Drag-and-drop user interface
- 3D interactive graphics
- Input and output processing
- Model development tools and editors

It also provides palettes of basic objects that are used in typical simulation applications:

- Entities for discrete-event simulation
- Text objects for labelling and documentation
- Graphs for visualizing simulation outputs
- Probability distributions for random sampling
- Graphical objects for background maps and logos

The only task left to the simulation programmer is to create additional palettes of high-level objects needed for the specific application. For example, a highway simulation would require new objects for roadways, vehicles, traffic lights, etc. The new objects created in this way become extensions of JaamSim and automatically acquire the same features as the basic objects.

1.3 - Applications

Ausenco's Transportation Logistics Simulator (TLS) is an example of an application module that extends JaamSim for a specific purpose. TLS is designed to simulate mine-to-port and port-to-port supply chains, so it includes a number of additional object palettes to JaamSim:

- Mine objects (ore production, trucks, truck dumps, storage stockpiles and silos, etc.)
- Railway objects (trains, tracks, signals, stations, train loaders, train unloaders, etc.)
- Material handling objects (conveyors, stacker/reclaimers, shiploaders, surge silos, etc.)
- Marine shipping objects (ships, routes, ports, weather and ocean conditions, tides, etc.)

Manuals for these packages are available to licensed users in separate publications. Examples of models built using Ausenco's TLS can be found at http://www.youtube.com/javasimulation.

Chapters:

2 - Installing JaamSim

2.1 - System Requirements

JaamSim will run on most modern computers that support OpenGL graphics version 3.0 or later. This includes laptop computers with Intel Core i5 and i7 series processors with integrated graphics (second generation processors and later). The following system specifications are recommended for optimal performance:

- Intel Core i7 processor
- 16 GB of RAM
- NVIDIA Quadro 600 video card or better

For best results in models with high-end graphics, we recommend the NVIDIA GeForce 780 graphics card or better. Although it is possible to use a computer with an AMD graphics card, we find that NVIDIA cards provide better support for OpenGL graphics and are preferred for JaamSim.

2.2 - Installing Java

JaamSim requires a recent (Version 7 or later) installation of the Java Runtime Environment (JRE), available for download free-of-charge from http://www.oracle.com/. The 64-bit version of the JRE is preferred for 64-bit operating systems.

2.3 - Installing JaamSim

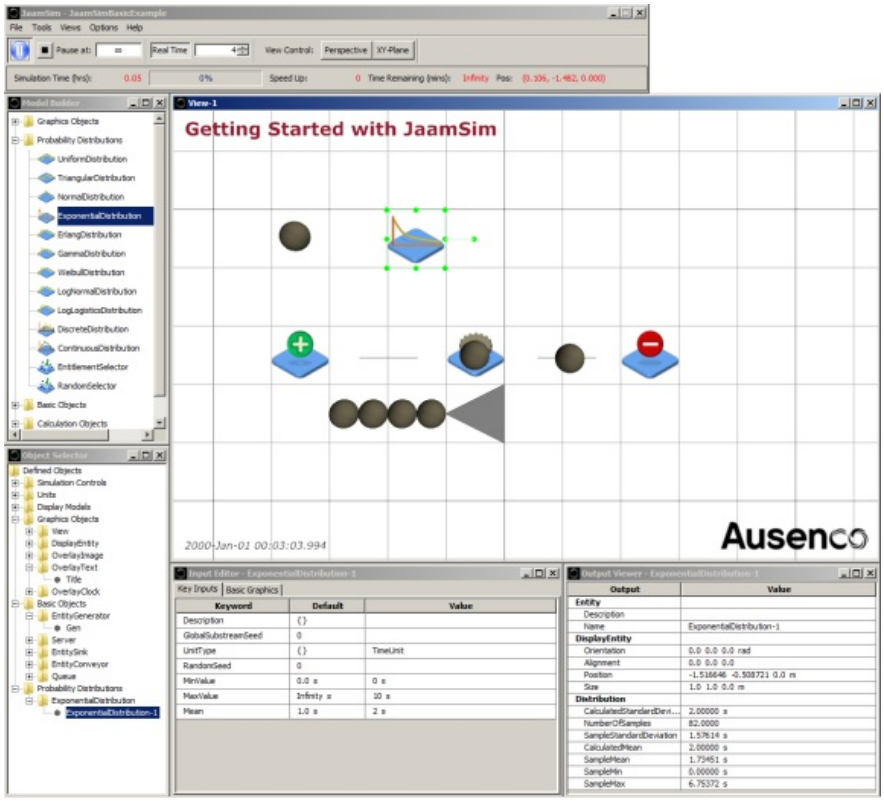
JaamSim consists of a single executable that can be copied directly to the user's computer. No special installation program is required. Copy the JaamSim executable file to a working directory, such as the directory containing the model input files. Launch JaamSim by double-clicking on the executable file.

TLS and other JaamSim-based applications are stand-alone executable files and are installed in the same way. The executable file (for example, TLS.exe for the Transportation Logistics Simulator) can be copied directly to the folder containing the model run files and launched directly from there or by a shortcut. It is not necessary to install JaamSim in addition to the application-specific executable.

3 - Getting Started with JaamSim

Welcome to JaamSim! In this example, users will be guided through building a basic model using the graphical user interface (GUI) that simulates entities being generated, processed, and consumed. The finished model should look similar to the model below.

Finished JaamSim Basic Example



This example will be divided into four steps:

- Step 1 - Creating Model Objects
- Step 2 - Putting the Objects Together
- Step 3 - Changing Model Graphics
- Step 4 - Adding a Probability Distribution

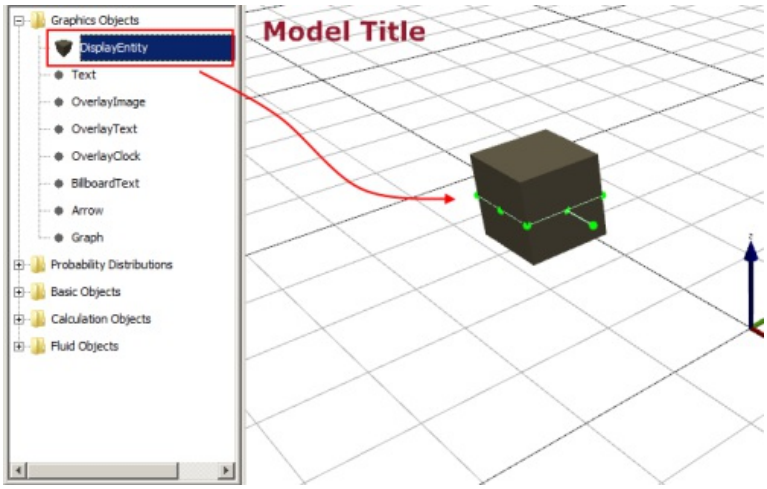
3.1 - Step 1 - Creating Model Objects

After launching JaamSim, the following windows will appear:

- **Control Panel**: provides a number of run control features
- **View Window**: displays a graphical representation of the model
- **Model Builder**: offers a selection of objects that can be added to the model
- **Object Selector**: lists the objects present in the model
- **Input Editor**: allows for editing of keywords for a selected object
- **Output Viewer**: displays outputs for a selected object

In the **Model Builder** expand the Graphics Objects palette and then drag-and-drop a DisplayEntity into the **View Window** (View-1). This creates a DisplayEntity object with a default name (DisplayEntity-1) and shape (Cube) and automatically selects it, denoted by green highlighting as seen below.

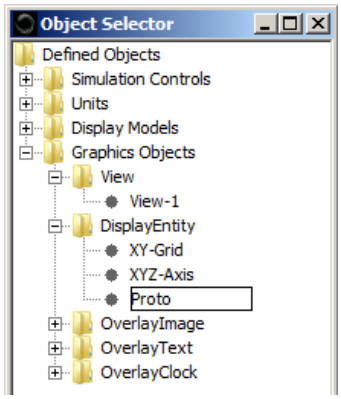
Creating the First Object



This DisplayEntity will serve as the prototype for entities that will be handled in the model. In the **Object Selector** select DisplayEntity-1 and press F2 or triple-click it to rename the object. Note that when renaming an object, or later when entering a keyword, the Enter button must be pressed when finished otherwise the change will not be completed:

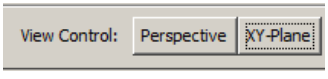
Object	Name
DisplayEntity-1	`Proto`

Renaming an Object



Since the model graphics will be 2-D aside from the Proto entity, set the View Control on the **Control Panel** to be XY-Plane so the view becomes bird's-eye:

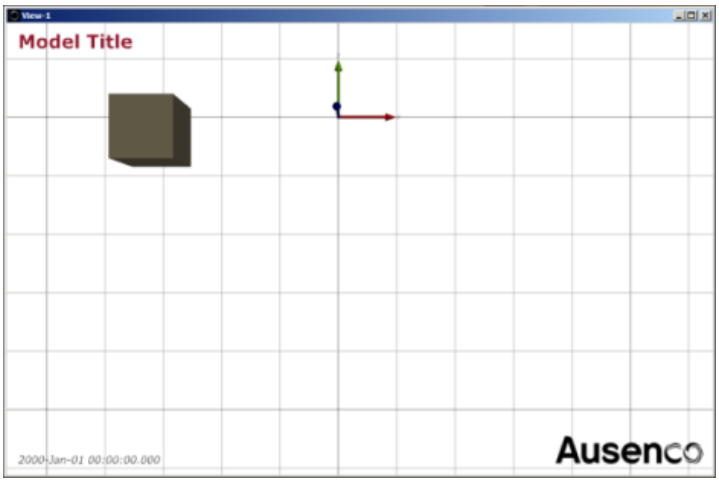
View Control: XY-Plane



Then adjust the viewing position as desired using the following actions:

Mouse/Keyboard Action	Effect on Viewing Position
Left Drag	Pan in the XY-plane
Scroll Wheel	Zoom in or out

Example of a Configured View



Create the rest of the objects for this model from the Basic Objects palette in the **Model Builder** in the following order from left to right:

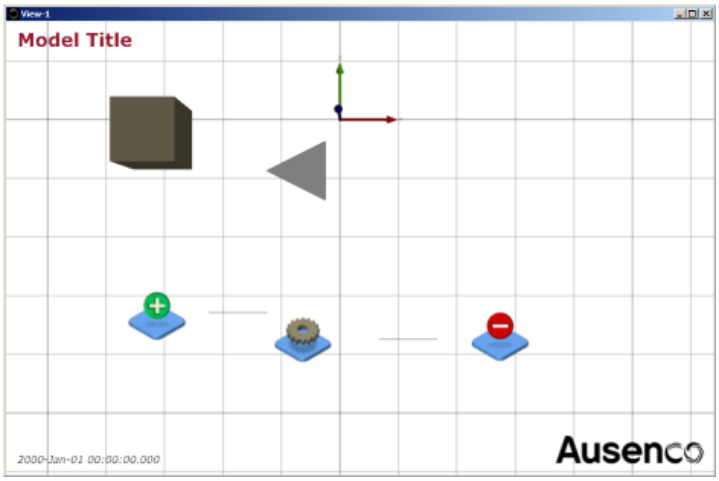
EntityGenerator ▶ EntityConveyor ▶ Server ▶ EntityConveyor ▶ EntitySink

and rename them using the same method as for the Proto entity:

Object	Name
EntityGenerator	`Gen`
EntityConveyor	`GenToServ`
Server	`Serv`
EntityConveyor	`ServToSink`
EntitySink	`Sink`
Queue	`ServQueue`

The re-positioning of these objects will be covered in a later step. The model so far should look similar to the Model Layout below, depending on how the viewing position was adjusted. Save the model from the File menu on the **Control Panel** with the name `JaamSimBasicExample.cfg` when prompted. We recommend saving regularly when building models.

Model Layout



3.2 - Step 2 - Putting the Objects Together

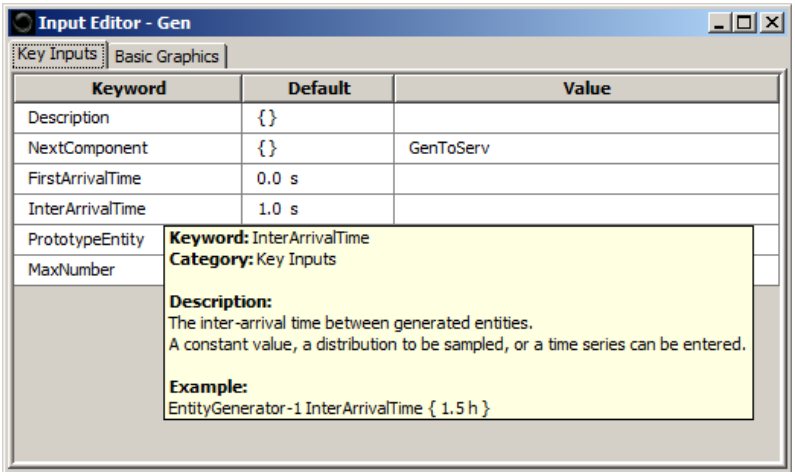
Now that the objects have been placed in the model, they need to be set to interact with one another to generate (Gen), transport (GenToServ, ServToSink), process (Serv), and consume (Sink) Proto entities. This is done by assigning values to the appropriate keywords for each object.

First connect the appropriate objects together by setting the following keywords:

Object	Keyword	Value
Gen	NextComponent	GenToServ
GenToServ	NextComponent	Serv
Serv	NextComponent	ServToSink
ServToSink	NextComponent	Sink

As a note, hovering the cursor over each individual keyword in the Keyword column will display a brief description of what each keyword does. The mouseover for InterArrivalTime is shown below.

Mouseover Keyword Tips



Set the time at which Proto entities will be added to the model by Gen, and the time they will spend at each stage:

Object	Keyword	Value
Gen	InterArrivalTime	2 s
GenToServ	TravelTime	1 s
Serv	ServiceTime	1 s
ServToSink	TravelTime	1.5 s

The Server object requires a Queue object for entities waiting to be processed. Set the following keyword to associate ServQueue with Serv:

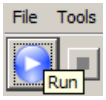
Object	Keyword	Value
Serv	WaitQueue	ServQueue

Lastly, configure Gen to produce instances of Proto, which is done by setting the following keyword:

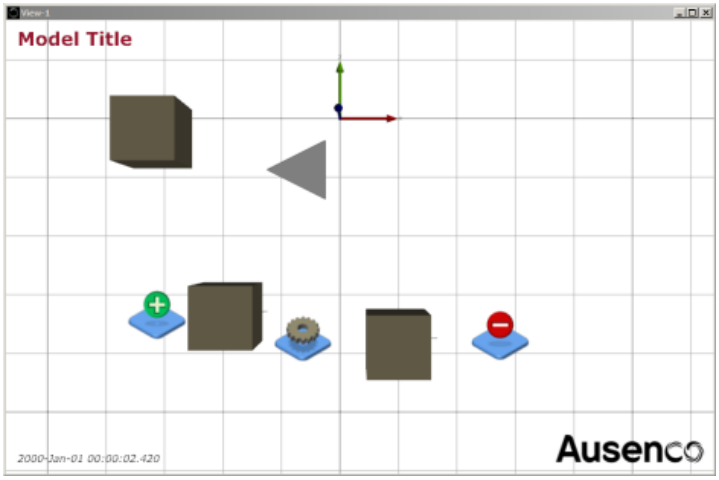
Object	Keyword	Value
Gen	PrototypeEntity	Proto

Save the model and press Play to run the simulation.

Playing the Simulation

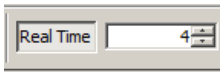


Simulation Screenshot



The Real Time Factor controls how fast the simulated time elapses in the model. The default Real Time Factor is 1, meaning each second watching the model corresponds to 1 simulated second, and can be changed by using the arrows on the Real Time Factor or by entering a desired value.

Real Time Factor



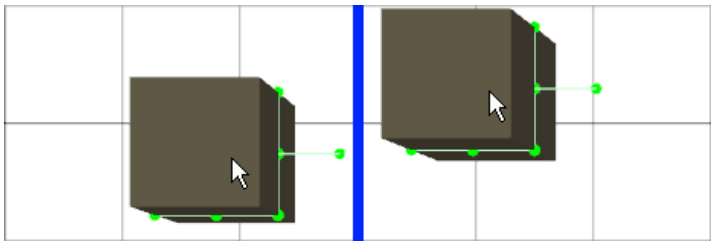
To continue working on the model, either Pause the model, which allows the simulation to later resume starting from the time Paused, or Stop the model, which resets the simulation.

3.3 - Step 3 - Changing Model Graphics

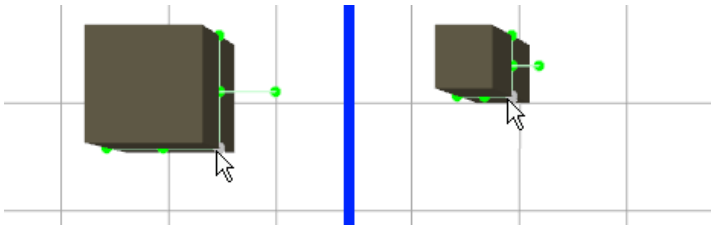
A few graphical adjustments will now be made to improve the appearance of the model. While the changes in this step will not impact the functionality of the model, graphics are invaluable when confirming proper operation in larger models.

Objects can be moved and resized by selecting them and holding down Control and left dragging. If a handle is selected during this process, the object will be resized, otherwise it will move following the cursor as illustrated in the figure below.

Moving an Object



Resizing an Object



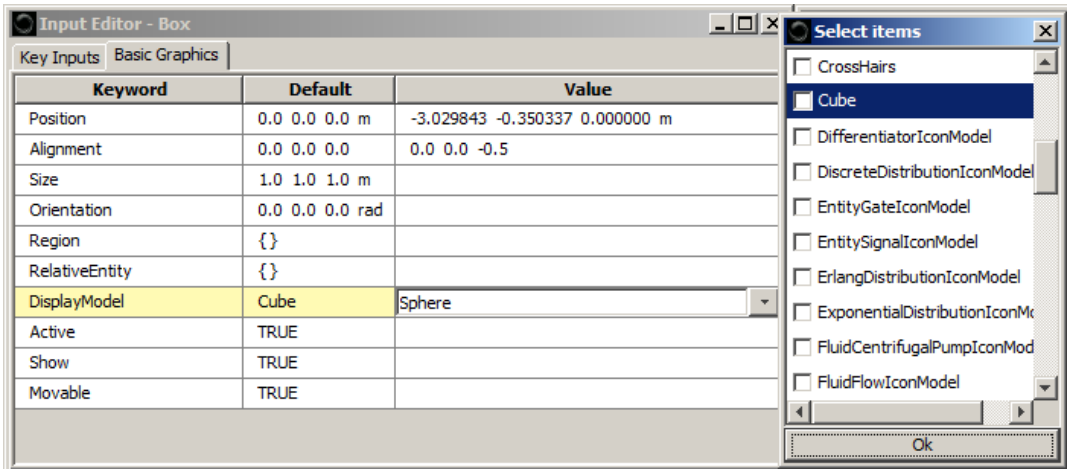
Use this method to place Proto somewhere off to the side and ServQueue near Serv, and then resize the two so they are not as large relative to the main model objects.

To demonstrate that the choice of entity processed is purely graphical, change the Proto entity to a Sphere by setting the DisplayModel keyword for Proto in the Basic Graphics tab in the **Input Editor**:

Object	Keyword	Value
Proto	DisplayModel	Sphere

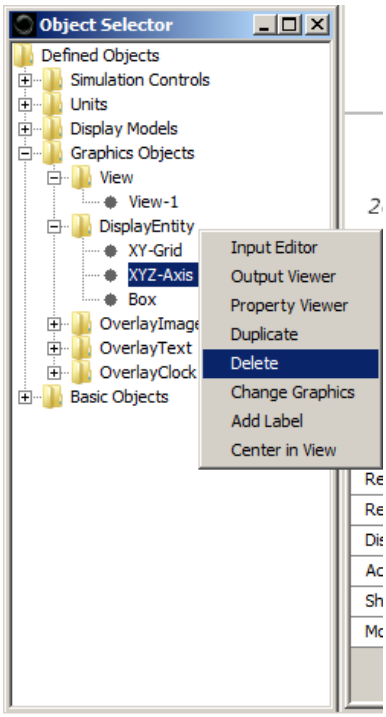
If using the drop-down menu, be sure to clear the check next to Cube as shown below.

Clearing the Previous DisplayModel Keyword



Since the XYZ-Axis marker isn't beneficial to this model, deleting it will make the model less cluttered. To do this, select the XYZ-Axis by going into **Object Selector** and expanding the Graphics Objects palette, then the DisplayEntity palette. Right-click the XYZ-Axis and choose Delete.

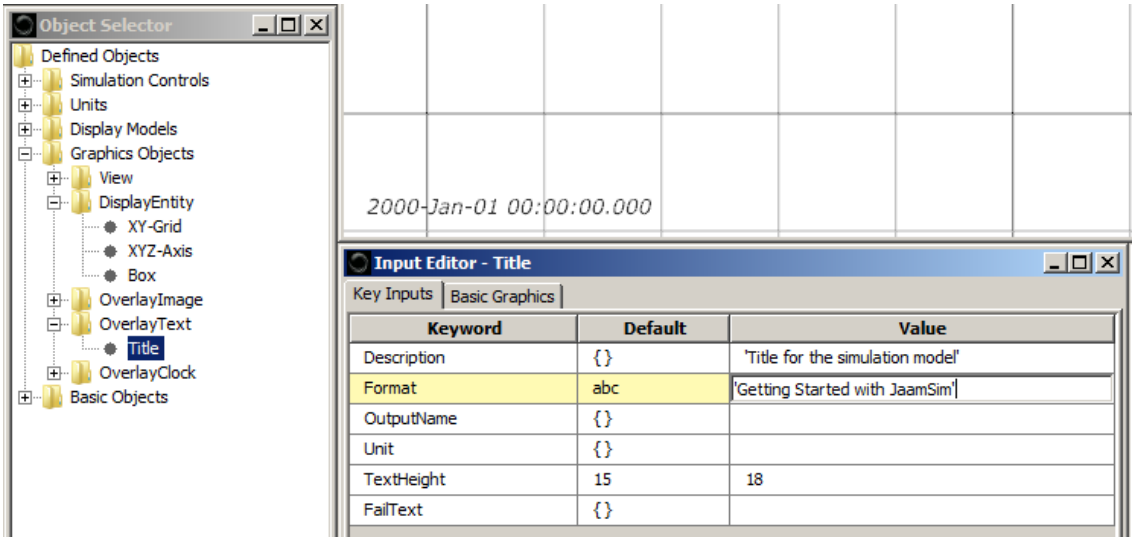
Deleting Objects



Lastly, change the 'Model Title' in the **View Window** by returning to **Object Selector** and expanding the OverlayText palette. Select the Title object and enter the following keyword under the Key Inputs tab in the **Input Editor** :

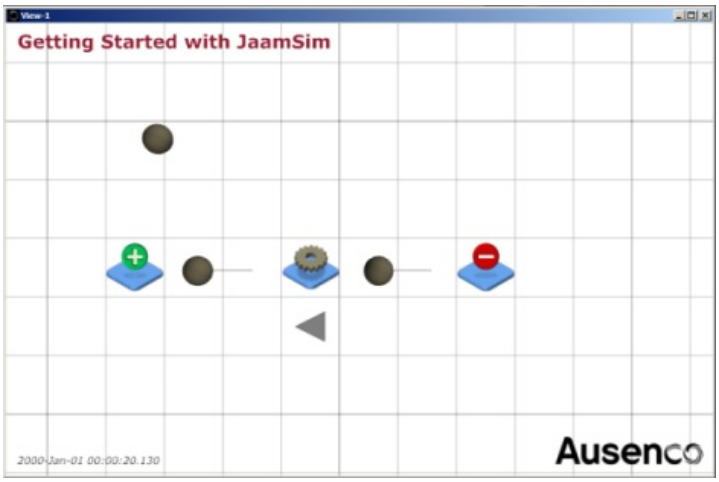
Object	Keyword	Value
Title	`Format`	`'Getting Started with JaamSim'`

Changing the Model Title



Press Play again. Gen should now produces Spheres instead of Cubes. Stop or Pause the simulation when ready to proceed.

Changing Proto DisplayModel



3.4 - Step 4 - Adding a Probability Distribution

The simulation created thus far is not particularly interesting because the Proto entities are being generated in a uniform sequence. To make the model more dynamic, we will add a probability distribution to control when each Proto entity is generated.

From the **Model Builder** expand the Probability Distributions palette and create an ExponentialDistribution object and rename it:

Object	Name
ExponentialDistribution-1	GenIATDist

Also set the following keywords to define GenIATDist:

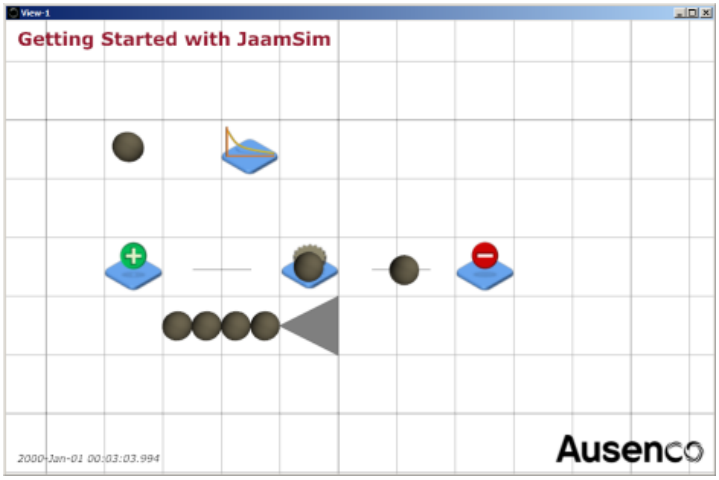
Object	Keyword	Value
GenIATDist	UnitType	TimeUnit
GenIATDist	MinValue	0 s
GenIATDist	MaxValue	10 s
GenIATDist	Mean	2 s

Now change the InterArrivalTime for Gen to be GenIATDist:

Object	Keyword	Value
Gen	InterArrivalTime	GenIATDist

Save the model again, and press Play. Observe that Proto entities are generated randomly and there are now times where Proto entities are waiting in ServQueue to be processed.

Simulation with a Probability Distribution



Congratulations, you have successfully constructed a JaamSim simulation model!

4 - Graphical User Interface

The graphical user interface (GUI) for JaamSim is also used by TLS and by any other simulation software based on JaamSim. The interface consists of the Control Panel, one or more view windows, the Model Builder, the Object Selector, the Input Editor, and the Output Viewer.

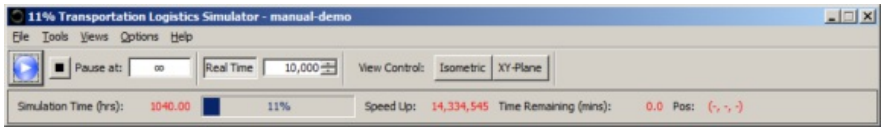
List of interface elements:

- Control Panel
- View Windows
- Model Builder
- Object Selector
- Input Editor
- Output Viewer

4.1 - Control Panel

The Control Panel provides a number of run controls and output displays to monitor and control the progress of a simulation run. The control panel for an example simulation run is shown in the figure below.

JaamSim Control Panel



The panel is divided into three rows, consisting of the menu bar, tool bar, and status bar, which are described in the following sections.

Menu Bar

File

Clicking on the File entry displays a menu with actions related to saving and loading model input configuration files:

Menu Entry	Description
New	Launches a new, blank model with no objects defined.
Open...	Loads a saved configuration from file.
Save	Saves the model under the present input configuration file name.
Save As…	Saves the current model as a new input configuration file.
Import…	Imports a 3D-model/image of a supported format as a ColladaModel/ImageModel and also creates a corresponding DisplayEntity.
Print Input Report	Prints the present inputs in a standard file format (.inp).
Exit	Closes all windows and exits JaamSim.

Tools

Clicking on the Tools entry displays a menu allowing the user to bring up each of the windows that make up the JaamSim graphical user interface:

Menu Entry	Description
Show Basic Tools	Shows the four main tools: Model Builder, Object Selector, Input Editor, and Output Viewer.
Close All Tools	Closes all of the tools that are open.
Model Builder	Drag and drop creation and placement of simulation objects.
Object Selector	Tree listing of all objects in the present simulation model.
Input Editor	View and edit keyword inputs for the selected simulation object.
Output Viewer	Key output values of the selected object.
Property Viewer	Detailed list of the internal properties of the selected object.
Log Viewer	Console for viewing input warnings and error messages.

The Property Viewer is typically used by programmers who are developing and debugging JaamSim applications, and so its usage is beyond the scope of this manual.

Views

A View is a window showing a graphical 3D representation of the model. The Views entry displays a menu containing a list of currently defined views of the system, as well as a single action:

Menu Entry	Description
Define new View	Creates a new View object and displays its window.

Options

Clicking on the Options entry in the menu bar generates a menu containing the following entries:

Menu Entry	Description
Show Position	If checked, the status bar displays the current position of the mouse cursor within the active view window.
Always On Top	If checked, the control panel will always remain on top of other windows.
Graphic Debug Info	If checked, information on video memory usage and rendering time will be shown as an overlay on the view windows.

Help

Clicking on the Help entry in the menu bar displays a single menu option that shows the software version number and copyright information.

Tool Bar

The Tool Bar contains controls for manipulating the simulation run and the 3D view for the active display window.

Run Controls

Tool Bar Item	Description
Run/Pause Simulation	Starts, pauses and resumes the simulation run.
Stop Simulation	Stops the simulation run and prepares it for re-starting at time zero.
Pause at:	Entering a value here (in hours of simulated time) pauses the simulation when it reaches that time.
Real Time	If pressed, the simulation speed is held to a constant multiple of wall-clock time.
Real Time Multiplier	The ratio of simulated time to wall-clock time that is used when the Real Time feature is active.

View Controls

Two buttons are provided to adjust the camera position.

Tool Bar Item	Description
Perspective	Moves the camera for the active view window to give the viewer a sense of depth in all dimensions.
XY-Plane	Moves the camera for the active view window to give a bird's eye view directly above the x-y plane of the simulation.

Note that both of these view options provide full 3D perspective graphics. Only the camera position is changed when a button is clicked.

Status Bar

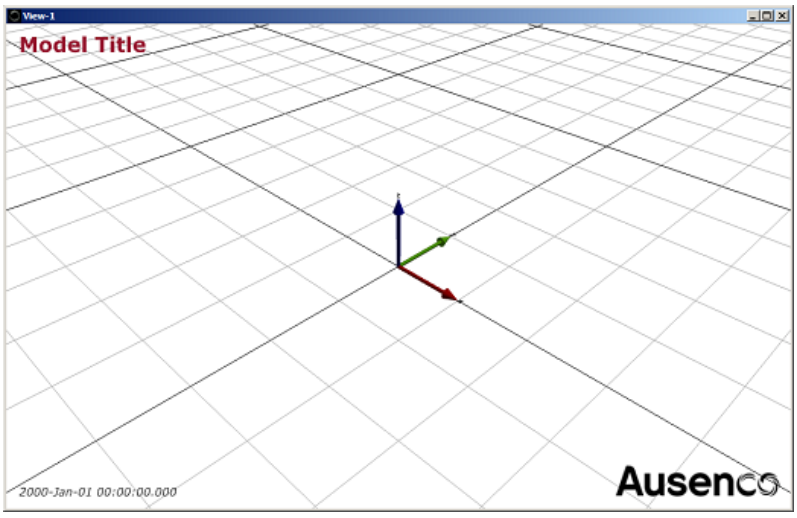
The Status Bar consists of indicators to illustrate the progress and status of the simulation run:

Status Bar Item	Description
Simulation Time	The number of simulated hours elapsed.
Progress Bar	Percentage of simulated time completed.
Speed Up	The ratio of simulated time to wall-clock time.
Time Remaining	Minutes of wall-clock time remaining until the simulation is completed.
Pos	Location of the mouse cursor in the active view window expressed in { x, y, z } coordinates.

4.2 - View Windows

View windows similar to the figure below display a graphical representation of a simulation. Multiple view windows can be defined depicting different parts of a model. Each view window is an instance of a View object and can be modified interactively.

View Window with Default Graphical Entities



Default Objects

When a new model is created, a default view window shown in the above figure is created with some default graphical objects that appear in this window: XY-Grid, XYZ-Axis, AusLogo, Title, and Clock. The XY-Grid and XYZ-Axis objects are intended as visual aids for placing new objects, are regular static graphics DisplayEntity objects. The `Movable` keyword is set to FALSE for both objects, so that they cannot be moved accidentally and do not respond to mouse clicks. The other three objects are the Ausenco wordmark, a place holder title for the model, and calendar and clock to show simulation time. These objects are Overlay Objects (OverlayImage, OverlayText, and OverlayClock respectively) that appear in a fixed position on the view window and are not part of the 3D scene.

The position and format of the default objects can be modified through the Input Editor or deleted using the Object Selector.

Moving the Camera

The basic camera movements are zoom, pan, and orbit. The user can zoom the camera by using the mouse wheel and pan the camera by clicking and dragging the mouse cursor around the view window. Dragging the mouse with the right button depressed causes the camera to orbit around the current point of interest. These movements are described in more detail below along with a number of other useful camera manipulations.

Mouse/Keyboard Action	Effect on the Camera
Left Click	Selects Point of Interest. The point on the surface of the object under the pointer becomes the point of interest. If no object is under the pointer, then the point on the xy-plane is used.
Scroll Wheel	Zooms the camera in or out. The camera is moved towards or away from the point of interest. One click moves the camera 10% closer to or farther away from the point of interest.
Left Drag	XY-plane pan. The camera is moved in the xy-plane from its present position so that the pointer stays fixed on the same object in the scene. If no object is under the pointer, then the point on the xy-plane is used. The point of interest is reset to the pointer position. Movement is restricted to no closer than 1 degree from the horizon to protect the user from an accidental trip to infinity.
Shift + Left Drag	Z-axis pan. The camera is moved along the z-axis from its present position so that the pointer stays fixed on the same object in the scene. If no object is under the pointer, then the point on the xy-plane is used. The point of interest is reset to the pointer position.
Right Drag	Orbit. The camera orbits left/right and up/down around the point of interest, following the mouse movement.
Shift + Right Drag	Look around. The camera looks left/right and up/down, following the mouse movement. The point of interest is unchanged.

Moving and Resizing Objects

Individual objects can be moved around using mouse controls that are analogous to the camera controls. To avoid moving an object accidentally, it is necessary to press the Control key during any movement. After selecting an object in the Object Selector or by clicking it in a view window, its position, size, and orientation can be manipulated interactively by pressing the Control key and dragging the entire object, a corner of the object, or its rotation handle using the mouse. By default, dragging an object moves it in the xy-plane. An object can be moved in the z-direction by holding down both the Control and Shift keys and dragging the object up and down. These actions are described in more detail in the following table.

Mouse/Keyboard Action	Effect on the Object
Left Click	Selects the object. The object under the pointer is selected for input/output viewing and for re-positioning, re-sizing, or rotating. The selected object is indicated by a green rectangle that encompassed the object. Green handles are also provided to allow the object to be re-sized or rotated.
Control + Left Drag	Moves the object parallel to the xy-plane (holding its z coordinate constant) following the pointer.
Shift + Control + Left Drag	Moves the object parallel to the z-axis (holding its x and y coordinates constant) following the pointer.
Control + Left Drag on a Handle	Resizes/rotates the object. The selected object is re-sized or rotated using the selected handle.

Moving and Re-shaping Linear Objects

In addition to the standard controls described above, linear objects, such as an Arrow, can be re-shaped by adding and removing joints and by moving individual joints. The actions for linear objects are described in the following table.

Mouse/Keyboard Action	Effect on the Object
Left Click on the Line	Selects the line. The line under the pointer is selected for input/output viewing and for re-positioning, re-sizing, or rotating. The selected line turns green indicating it has been selected and green handles appear at each joint in the line.
Control + Left Drag on the Line	Moves the entire line in the xy-plane. The selected line is moved in a plane parallel to the xy-plane, following the pointer.
Shift + Control + Left Drag on the Line	Moves the entire line along the z-axis. The selected line is moved along the z-axis, following the pointer.
Control + Left Drag on a Handle	Moves location of the joint in the xy-plane. The lines on either side of the joint adjust to following the joint.
Shift + Control + Left Drag on a Handle	Moves the location of the joint along the z-axis.
Alt + Control + Left Click on the Line	Adds a new joint and Handle to the line.
Shift + Alt + Control + Left Click on a Handle	Deletes the joint and re-connects the joints on either side of the deleted joint.

Context Menu

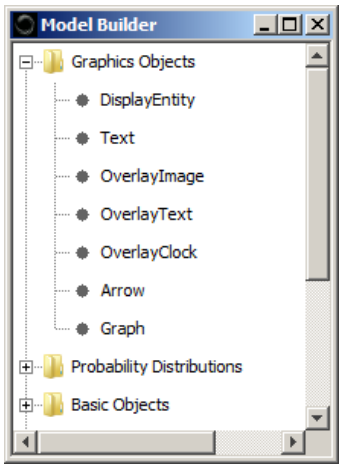
Right-clicking in the view window generates a menu listing all objects whose graphical representation lies under the mouse cursor. Selecting an object here (or in the case that there is only one object underneath the mouse cursor) generates a submenu with the following entries:

Menu Item	Description
Input Editor	Selects the object and opens its Input Editor window.
Output Viewer	Selects the object and opens its Output Viewer window.
Property Viewer	Selects the object and opens its Property Viewer window.
Duplicate	Creates a copy of the selected object in the current view window.
Delete	Deletes the selected object.
Change Graphics	Opens a dialog box to select a new DisplayModel graphical representation for the selected object.
Add Label	Creates a Text object with its name and RelativeEntity keywords set to the selected object.
Center in View	Selects the object and centers the current View on it.

4.3 - Model Builder

The Model Builder provides palettes of objects that can be dragged and dropped to construct a new model or to modify an existing one. The figure below shows the Model Builder with the Graphical Objects palette opened, ready for dragging and dropping.

Model Builder



Once an object has been created by dragging and dropping it from the Model Builder into a view window, its inputs are entered using the Input Editor.

When a model is saved, JaamSim generates an input configuration file that contains all of the objects that have been created along with their input values.

JaamSim includes a number of built-in palettes of objects:

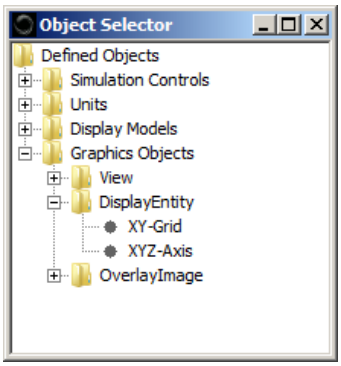
- Simulation Controls
- Units
- Graphics Objects
- Display Models
- Probability Distributions
- Basic Objects
- Calculation Objects
- Fluid Objects

The Simulation Controls, Units, and Display Models palettes contain objects that do not have a graphical representation and cannot be dragged and dropped by the user. They appear in the Object Selector, but not in the Model Builder. The remaining palettes contain the standard objects that are needed for many simulation models. A JaamSim application, such as Ausenco's TLS, will include additional palettes of objects created specifically for that application.

4.4 - Object Selector

The Object Selector is an index of all the objects that have been created for the present model, including ones that were created automatically by JaamSim. Objects are grouped according to their palette and type in a tree format that mirrors the structure of the Model Builder. A specific object can be selected either by clicking its node in the Object Selector or by clicking it in a view window. The figure below shows the Object selector with the Graphics Objects and DisplayEntity nodes expanded.

Object Selector

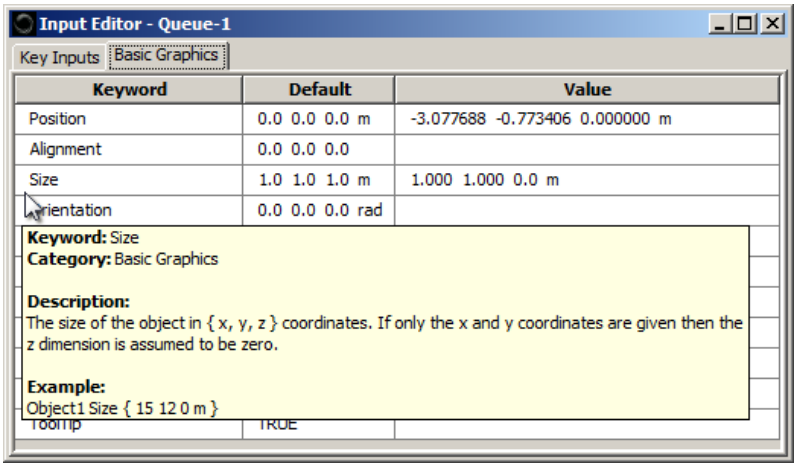


Objects that have been created using the Model Builder can be renamed or deleted using the Object Selector. Once an object has been selected, it can be renamed by clicking on its entry in the Object Selector or by pressing F2, similar to the convention in Windows. It can be deleted by pressing the Delete key, or by right-clicking the object in a view window or the Object Selector and selecting Delete.

4.5 - Input Editor

The Input Editor allows the user to modify inputs for objects already defined in the model or to assign inputs to newly-defined objects. When an object is selected, its parameters appear in the Input Editor window, grouped under a number of tabs. If a keyword has a default value assigned, it is shown in the Default column. Hovering the mouse cursor over a keyword will display a tooltip containing a brief description of the keyword and an example input. The figure below shows the Input Editor with the tooltip for Size.

Input Editor with Keyword Tooltip Mouseover



Entering and Editing Keyword Inputs

Keywords are modified by clicking the Value column and entering a new value, including units. Numbers must be entered without spaces or commas and Boolean keywords must take the value TRUE or FALSE (case sensitive). If an entry is made in the Value column, it will overwrite the default. If an input is not valid, an error message will be displayed showing the cause of the error.

Depending on the object, different keyword values will have different data types. These include:

- Numbers without units. Specified with or without a decimal point, e.g. 5, 5.0, and 5. are equivalent entries.
- Numbers with units. A pre-defined unit must be included, e.g. 1000 mm, 1.0 m, and 0.001 km are equivalent entries.
- Boolean values. Indicated as either TRUE or FALSE (case-sensitive).

- Colours. Specified by a colour keyword or by an RGB value, e.g. pink and 255 192 203 are equivalent entries.
- Strings. The text must be enclosed in single quotes(') if it contains any spaces, e.g. 'a b c'. Note that 'abc' and abc are equivalent entries.
- Objects. For example View1 indicates the view window, View1, created automatically by JaamSim.
- Object outputs. Specified by an object name and an output name, e.g. Queue1 QueueLength is the output named "QueueLength" for the queue object Queue1.

Curly braces are used to delineate distinct entries for keywords such as points, e.g. { 0.0 1.0 0.0 m } { 1.0 1.0 0.0 m }.

A drop-down menu is available for many types of inputs. For a Boolean input, the user is offered a choice of TRUE or FALSE. For an object input, the user is offered a list of all the objects of the appropriate type. Similar drop-down menus are provided for most other types of inputs.

Mathematical Expressions

Many keywords that expect a number can also accept other types of inputs that return a number, such as a Probability Distribution, a TimeSeries, or a mathematical expression. A probability distribution or time series can be specified simply by entering the object name. A mathematical expression is specified by entering the actual equation using the following rules:

- an expression must be enclosed in single quotes (') if it contains any spaces
- any object names must be enclosed by square brackets ([])
- spaces in an expression are optional
- outputs and attributes of an object are referenced using a dot notation: [ObjectName].OutputName
- the name "this" can be used as a substitute for the name of the object that is evaluating the expression
- all the normal rules for mathematical order of operation apply to expressions
- there is no limit to the length or complexity of an expression

The following are examples of valid expressions. They assume that the present object has an output A and another entity, Entity1, has an output B:

- '1 + 2*3'
- '1 + 2 * 3^2'
- '1 + 2*this.A'
- '1 + 2*[Entity1].B'

The mathematical functions min, max, and abs can also be used in expressions, e.g.

- '1 + 2*max(3,4)'
- '1 + 2 * max(1+this.A, 3*[Entity1].B)'

Logical operations can also be performed in expressions using the standard Java symbols:

- equal to: ==
- not equal to: !=
- less than: <
- less than or equal to: <=
- greater than: >
- greater than or equal to: >=
- logical AND operation: &&
- logical OR operation: ||
- logical NOT operation: !

The values 1 and 0 are interpreted as TRUE and FALSE, respectively. All other (non-zero) values are interpreted as TRUE. The following are examples of valid logical expressions:

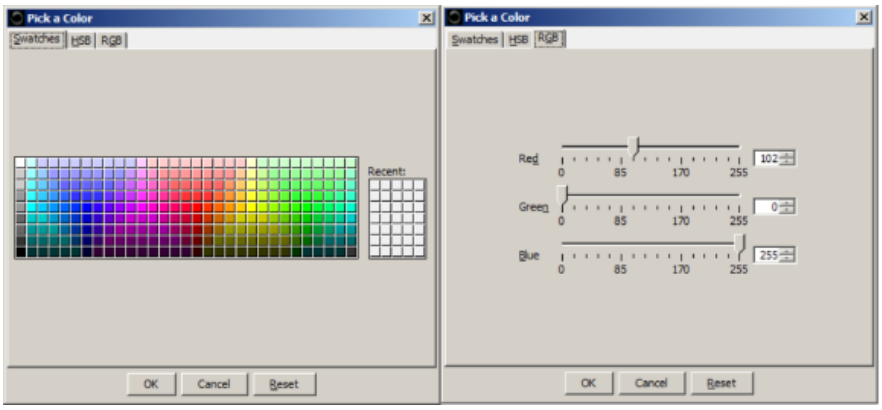
- 'this.A >= 1'
- '(this.A >= 1) && ([Entity1].B == 0)'

Colour Selection

For readability, it is often preferable to specify a colour input using a colour keyword instead of an RGB value, e.g. pink instead of 255 192 203. A table of standard colour keywords are provided in the last section of this manual.

Alternatively, the Input Editor provides a pop-up menu containing a graphical palette (shown in the figure below) to aid colour selection. The user can select a colour or adjust the RGB values via slider controls.

Colour Selection Pop-up Window



4.6 - Output Viewer

The Output Viewer displays the pre-programmed outputs that are available for the selected object. Output values are updated continuously as the simulation progresses. Numerical outputs are displayed in SI units unless a Preferred Unit is specified.

The output system provided by JaamSim is the basis for all model graphics and reports. An object can be animated by connecting a pre-defined Action in its DisplayModel to one of its Outputs.

The figure below shows the outputs provided for a Queue object named "Que".

Output Viewer

Output Viewer - Que

Output	Value
Entity	
Name	Que
DisplayEntity	
Position	0.493151 2.431863 0.0 m
Size	0.431764 0.365344 0.0 m
Orientation	0.0 0.0 0.0 rad
Alignment	0.0 0.0 0.0
Queue	
QueueLength	0
NumberAdded	5
QueueLengthMinimum	0
QueueLengthMaximum	1
NumberRemoved	5
QueueLengthAverage	0.0900710
QueueLengthStandardDeviation	0.286283
QueueLengthDistribution	{ .909929031696273, .09007096830372699 }

Attributes

In addition to the pre-programmed outputs, users can create custom outputs, called "attributes", for individual objects. Attributes can be added using the `AttributeDefinitionList` keyword that is provided in the Key Inputs tab for every entity. The format `{ AttributeName1 InitialValue1 Unit1 } ... { AttributeNameN InitialValueN UnitN }` is used for this keyword. For example, the input `{ A 1 km } { B 9 }` would create two new attributes: the first, named A, would have an initial value of 1 km, while the second, named B, would have an initial value of 9 and be dimensionless.

Attributes appear in the Output Viewer immediately after being defined through the `AttributeDefinitionList` keyword.

5 - Simulation Controls Palette

The Simulation object (named "Simulation") is used to store inputs used to define basic parameters of the model, such as run duration. Unlike other objects, the Simulation object is created automatically when a new model is started. It does need to be dragged and dropped by the user, and for this reason it does not appear in the Model Builder.

List of objects:

- Simulation

5.1 - Simulation

As with other objects, the simulation parameters are changed using the Input Editor.

Key Inputs:

Keyword	Description
RunDuration	Duration of the simulation run in which statistics will be recorded.
InitializationDuration	Duration of the initialization period from which the run statistics are discarded before the simulation is run for the time specified in RunDuration. Allows the model to reach steady-state before statistics are recorded. The total length of the simulation run is the sum of InitializationDuration and RunDuration.
SimulationTimeScale	The number of discrete time steps in one hour, e.g. a value of 3600 means that the smallest time step is one second.
PrintInputReport	If TRUE, an input report is generated once the simulation is loaded. The report can otherwise be generated from the File menu.
RealTimeFactor	A numerical multiplier that defines how quickly, with respect to wall-clock time, the simulation runs.
RealTime	If TRUE, then the simulation runtime is held constant at a multiple of real time. If FALSE, then the simulation runs as fast as possible (and may vary) throughout the run.
ExitAtStop	If TRUE, JaamSim will close all windows and exit when the simulation run is finished.
ShowModelBuilder	If TRUE, the Model Builder will be displayed when the input file is loaded.
ShowObjectSelector	If TRUE, the Object Selector will be displayed when the input file is loaded.
ShowInputEditor	If TRUE, the Input Editor will be displayed when the input file is loaded.
ShowOutputViewer	If TRUE, the Output Viewer will be displayed when the input file is loaded.
ShowPropertyViewer	If TRUE, the Property Viewer will be displayed when the input file is loaded.
ShowLogViewer	If TRUE, the Log Viewer will be displayed when the input file is loaded.

6 - Units Palette

JaamSim performs all internal calculations in SI units (meters, kilograms, seconds, etc.). However, many quantities employed by simulation users are more conveniently specified in other unit systems. To this end, JaamSim natively supports a number of commonly used units, shown in table below.

Unit Type	Default Unit	Supported Units
DimensionlessUnit		
TimeUnit	seconds (s)	min, h, d, w, y, ms, us, ns
DistanceUnit	meters (m)	m, km, nmi, mi, ft, in, mm
SpeedUnit	meters per second (m/s)	m/s, km/h, knots, mph
AccelerationUnit	meters per squared second (m/s^2)	ft/s^2
MassUnit	kilograms (kg)	tonnes, kt, Mt
MassFlowUnit	kilograms per second (kg/s)	(any mass unit)/(h, d, y)
VolumeUnit	cubic meters (m^3)	km^3, bbl, mbbl, mmbbl
VolumeFlowUnit	cubic meters per second (m^3/s)	(any volume unit)/(h, d, y)
AngleUnit	radians (rad)	degrees
AngularSpeedUnit	radians per second (rad/s)	rad/h, deg/s, deg/h
EnergyUnit	joules (J)	kWh
EnergyDensityUnit	joules per cubic meter (J/m^3)	kWh/m^3
SpecificEnergyUnit	joules per kilogram (J/kg)	kWh/t
PowerUnit	watts (W)	kW, MW
CostUnit	dollars ($)	
CostRateUnit	dollars per second ($/s)	$/h, $/d
LinearDensityUnit	kg/m	t/m, kt/m
LinearDensityVolumeUnit	m^3/m	
DensityUnit	kg/m^3	
PressureUnit	Pa	kPa, psi
ViscosityUnit	Pa-s	P, cP
AreaUnit	m^2	cm^2, mm^2, in^2
RateUnit	/s	/h, /d, /w, /y

Units are mandatory for most numerical inputs with the exception of pure numbers or ratios, or in a few cases for older inputs that have not yet been updated with this requirement. Inputs that are pure numbers are indicated by the DimensionlessUnit type.

All Unit objects have the same keywords.

Key Inputs:

Keyword	Description
ConversionFactorToSI	Two numbers that specify the numerator and denominator, respectively, of the multiplicative factor to convert from the new unit to SI base units.
PreferredUnit	The unit to be used in the Output Viewer for this unit type. If no unit is entered, the Output Viewer defaults to the SI unit. The value entered for this keyword is shared between all the units for a given unit type.

6.1 - Defining a New Unit

In addition to the pre-existing unit types defined in JaamSim, the user can specify new units using a Unit object. A new unit object can be created by right clicking on an existing unit and selecting duplicate. After the unit is renamed appropriately, its `ConversionFactorToSI` input can be set to the desired value. For example, to create a new TimeUnit for "fortnight" (two weeks), the `ConversionFactorToSI` keyword should be set to { 1209600 1 }. Once this has been done, keyword values in units of time will subsequently support the Fortnight unit when values are entered.

6.2 - Setting a Preferred Unit

Model outputs are normally shown in the appropriate SI unit in the Output Viewer. In many cases, it is useful to show outputs in another unit. For example, it might be better to view a time output in terms of hours instead of seconds (the SI unit for time). This change can be accomplished by setting the `PreferredUnit` keyword for a TimeUnit to { h }. Any TimeUnit can be used for this purpose. The same result would be obtained by using s (seconds), h (hours), d (days), or any other TimeUnit. Changing the `PreferredUnit` keyword for one TimeUnit changes this keyword's value for all TimeUnits.

7 - Graphics Objects Palette

Graphical Objects are used to create 3D objects, pictures, text, graphs, arrows, and other visual components needed to visualise and monitor a simulation.

List of objects:

- DisplayEntity
- Text
- Overlay Objects
- BillboardText
- Arrow
- Graph
- View

7.1 - DisplayEntity

The DisplayEntity is the basic 3D graphical object in JaamSim. All JaamSim objects that have graphics are sub-classes of DisplayEntity. A DisplayEntity dragged and dropped from the Model Builder can be used to display a static object such as a 3D shape or a 2D picture.

The graphical appearance of a DisplayEntity and its subclasses is determined by its DisplayModel. The two objects work together to generate an object's display. In general, the DisplayEntity determines what is displayed, while its DisplayModel determines how it is displayed.

A number of different subclasses of DisplayModel are available to match the various subclasses of DisplayEntity. The `DisplayModel` keyword determines the DisplayModel used by a DisplayEntity. A 3D object will require a ColladaModel while a 2D picture will require a ImageModel. The inputs for DisplayEntity determine the position, size, and orientation for the graphical object, while the inputs for the DisplayModel determine the shape and appearance of the object.

One DisplayModel can be shared between multiple DisplayEntities. This is an essential feature for the case of complex 3D content built from thousands or millions of triangles. In this case, a ColladaModel (as subclass of DisplayModel) stores 3D information that can be shared between multiple DisplayEntities. The 3D content is loaded and stored only once, even though it is displayed many times in various locations. Even in the case of animated 3D content, only one ColladaModel is needed to display a different animation state in each location.

The default appearance of a DisplayEntity is a grey cube. Its appearance can be changed by right-clicking the DisplayEntity in either a view window or the Object Selector and selecting "Change Graphics". The user can then select between the available DisplayModels or can create a new DisplayModel by importing a file in one of the supported 3D graphics formats. Alternatively, the user can create a new DisplayModel through the Object Selector:

- Create a new DisplayModel by right clicking on an existing ColladaModel or ImageModel in the Object Selector and selecting "Duplicate".
- Rename the new object by triple clicking on the name in the Object Selector, entering the new name, and pressing Return key.
- Use the Input Editor to assign a 3D file to the `ColladaFile` keyword (for a ColladaModel) or a picture file to the `ImageFile` keyword (for a ImageModel).
- Use the Input Editor to select the new DisplayModel for the DisplayEntity's `DisplayModel` keyword.

All objects intended for visualization in a model display window in JaamSim have a set of Basic Graphics keywords used to define their appearance. These are found in the Basic Graphics tab of the Input Editor when the object is selected.

Basic Graphics Inputs:

Keyword	Description
Position	The point in the region at which the alignment point of the object is positioned.
Alignment	The point within the object about which its Position keyword is defined, expressed with respect to a unit box centered about { 0, 0, 0 }.
Size	The size of the object in { x, y, z } coordinates. When two coordinates are given it is assumed that z = 0. When the size is changed, the coordinates of the center are held fixed and the eight corners are moved.
Orientation	The three Euler angles defining the rotation of the object.
RelativeEntity	An object with respect to which the Position keyword is referenced. If that object is moved, any object connected by RelativeEntity will also move.
DisplayModel	The graphical representation of the object. Accepts a list of multiple Display Model objects for compatibility with level of detail and optional rendering.
Active	If TRUE, then the object is active and used in the simulation run.
Show	If TRUE, then the object is shown in the simulation view windows.
Movable	If TRUE, then the object can be positioned by dragging with the mouse. Non-movable objects do not respond to mouse interactions in the View windows.

7.2 - Text

The Text object is used to define static or dynamic text to be displayed in a view window. Text objects are used for labelling various parts of the model and for monitoring the status of the model.

The text displayed by the Text object is determined primarily by its `Format` keyword. Dynamic text can be introduced by including a Java format code such as %s in the `Format` input and specifying the variable to be displayed with the `OutputName` keyword. Use the Output Viewer to identify the outputs that are available for each object. If the output is a number with Units, the value can be converted from SI to a specified unit through the `Unit` keyword.

The appearance and style of the text is determined by the TextModel specified by the `DisplayModel` keyword. The default style ("DefaultTextModel") is black Verdana text. A new text style can be created through the following steps:

- Create a new TextModel by right clicking on the DefaultTextModel object in the Object Selector and selecting "Duplicate".
- Rename the new TextModel object by doubling clicking on the name in the Object Selector, entering the new name, and pressing Return key.
- Use the Input Editor to specify the characteristics for the new TextModel, such as font, italics, bold, colour, etc.
- Use the Input Editor to select the new TextModel for the Text object's `DisplayModel` keyword.

A Text object for a specific graphical entity can be generated automatically by right-clicking the object in a view window and selecting "Add Label". The `Format` keyword will be automatically populated with the name of the target entity and will have its `RelativeEntity` keyword set so that the label follows the target entity when it is moved.

The Text object has the following keywords in addition to those from DisplayEntity.

Key Inputs:

Keyword	Description
Format	The fixed and variable text to be displayed. If spaces are included, enclose the text in single quotes ('). If variable text is to be displayed using the OutputName keyword, include the appropriate Java format in the text, e.g. %s will display a text output and %.6f will display a number with six decimals of accuracy.
OutputName	The output value chain that returns the variable text to be displayed. If more than one output value is given, all outputs but the last should point to an entity output to query for the next output. For example, { Tank-1 Product Name }, returns the name of the product in Tank-1.
Unit	The unit in which to express the output value. Required only for outputs that return a number.
TextHeight	The size of the font as displayed in the view window.

7.3 - Overlay Objects

Overlay objects are special versions of other objects that are used for graphical display in a simulation model. Unlike other display objects, the position of overlay objects is referenced to the corner of a view window, and so the object does not move when the view is panned or zoomed. These objects are useful for labelling view windows or displaying the model name and company logos. Examples of Overlay Objects are the default Title, Clock, and Ausenco logo that are provided automatically when a new model is opened.

There are three types of overlay objects, each corresponding to a different graphical object type. The relationship between each overlay object, its parent object type, and its usage is summarized below.

Overlay Object	Parent Object	Usage
OverlayImage	DisplayEntity	Static image (Logos, other graphics)
OverlayText	Text	Static of dynamic text (Model name, states, rates)
OverlayClock	Text	Current time in the simulation model.

Each type of overlay object takes the keywords associated with its parent object. For instance, the `Format` and `TextHeight` keywords must be supplied for an OverlayText object. Additionally, instead of the Basic Graphics keywords common to other display objects, overlay objects have the following keywords:

Key Inputs:

Keyword	Description
`ScreenPosition`	A list of two numbers specifying the spacing (in pixels) between the left and top corner of the View window and the object.
`AlignRight`	If TRUE, the horizontal alignment is referenced to the right side of the view window instead of the left.
`AlignBottom`	If TRUE, the vertical alignment is referenced to the bottom side of the view window instead of the top.

7.4 - BillboardText

A BillboardText object is similar to a normal Text object, except that the text is always oriented towards the view window and its height is given in pixels instead of metres. BillboardText retains its coordinates in 3-space and in this way differs from the OverlayText object. Both static and dynamic text can be dispayed by a BillboardText object using the same keywords as Text.

The keywords for BillboardText are identical to those for Text and have the same meaning, except for the `TextHeight` keyword, which now gives the height of the text in pixels instead of metres. At the present time, this input must include the units of metres (m) even though it is interpreted as pixels. This deficiency will corrected in a future version of the software.

7.5 - Arrow

An Arrow object consists of a line (possibly multiple line segments) and an arrowhead. It can be used to point out objects in a view window. In addition to the Basic Graphics keywords, the table below describes keywords used for this object in particular.

Key Inputs:

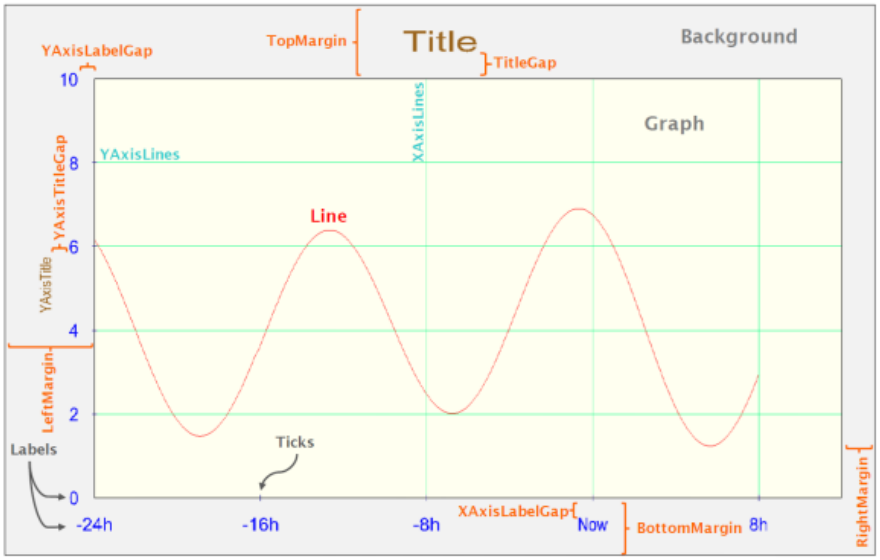
Keyword	Description
Points	A list of points (in { x, y, z } coordinates) defining the line segments that make up the arrow. When two coordinates are given it is assumed that z = 0.
ArrowSize	A set of { x, y, z } numbers that define the size of the arrowhead in those directions at the end of the connector.
Width	The width of the arrow line segments in pixels.
Colour	The colour of the arrow, defined using a colour keyword or RGB values.

Arrows (and other graphical objects that use the Points keyword) ignore the Position, Alignment and Orientation keywords, since their placement is defined completely by their Points coordinates.

7.6 - Graph

The Graph object is a real-time visual representation of one or more output values, displaying its current value in the context of its recent history. The figure below shows an example of a graph, with keywords related to formatting its appearance labelled directly on the graph. The formatting is applied by specifying a GraphModel object to the DisplayModel keyword.

Sample Graph with Formatting Keywords Defined Graphically



Key Inputs:

Keyword	Description
Title	Text for the graph title, enclosed in single quotes (') if it contains spaces.
NumberOfPoints	The number of data points to be displayed on the graph. This determines the resolution of the graph.
DataSource	One or more sources of data to be graphed against the primary y-axis.
LineColours	A list of colors for the data series to be displayed. For multiple colors, each color must be enclosed in braces as each color can be itself defined as a list of RGB values.
LineWidths	A list of widths (in pixels) for the data series to be displayed.
SecondaryDataSource	One or more sources of data to be graphed against the secondary y-axis.
SecondaryLineColours	A list of colors for the data series to be displayed. For multiple colors, each color must be enclosed in braces as each color can be itself defined as a list of RGB values.
SecondaryLineWidths	A list of widths (in pixels) for the data series to be displayed.

X-Axis Keywords:

Keyword	Description
XAxisTitle	Title text for the x-axis.
XAxisUnit	The unit to be used for the x-axis.
XAxisStart	The minimum value for the x-axis.
XAxisEnd	The maximum value for the x-axis.
XAxisInterval	The interval between x-axis labels.
XAxisLabelFormat	The Java format to be used for the tick mark values on the x-axis.
XLines	A list of time values between StartTime and EndTime where vertical gridlines are inserted.
XLinesColor	Color of the vertical gridlines (or a list corresponding to the color of each gridline defined in XLines), defined using a color name or RGB values.

Y-Axis Keywords:

Keyword	Description
YAxisTitle	Title text for the primary y-axis.
YAxisUnit	The unit to be used for the primary y-axis.
YAxisStart	The minimum value for the y-axis (in units of the target method).
YAxisEnd	The maximum value for the y-axis.
YAxisInterval	The interval between y-axis labels.
YAxisLabelFormat	The Java format to be used for the tick mark values on the y-axis.
YLines	A list of values at which to insert horizontal gridlines.
YLinesColor	Color of the horizontal gridlines, defined using a color name or RGB values.

Secondary Y-Axis Keywords:

Keyword	Description
SecondaryYAxisTitle	Title text for the secondary y-axis.
SecondaryYAxisUnit	The unit to be used for the secondary y-axis.
SecondaryYAxisStart	The minimum value for the secondary y-axis.
SecondaryYAxisEnd	The maximum value for the secondary y-axis.
SecondaryYAxisInterval	The interval between secondary y-axis labels.
SecondaryYAxisLabelFormat	The Java format to be used for the tick mark values on the secondary y-axis.

7.7 - View

When the Views menu is selected from the menu bar, a list of View objects is displayed. Each View object consists of an imaginary camera positioned in the world and a window in which to display the image. Clicking one of these objects will display the View in a new window (or bring the view window into focus if already displayed).

The graphical positions of objects are not used by any of the simulation model calculations. All graphics are for display purposes only.

Key Inputs:

Keyword	Description
ViewCenter	The position in space that the View window is looking at.
ViewPosition	The position in space that the View window is looking from.
WindowSize	The size of the view window, in pixels.
WindowPosition	The position of the view window, in pixels, with respect to the upper-left hand corner of the computer screen.
TitleBarText	Text to place in the title bar of the view window.
ShowWindow	If TRUE, the view window is displayed on-screen.

8 - Display Models Palette

The graphical appearance of a DisplayEntity and its subclasses is determined by its DisplayModel. The two objects work together to generate an object's display. In general, the DisplayEntity determines what is displayed, while its DisplayModel determines how it is displayed. A number of different subclasses of DisplayModel are available to match the various subclasses of DisplayEntity.

For example, the Text object (a subclass of DisplayEntity) and the TextModel (a subclass of DisplayModel) work together to display text. The text to be displayed is determined by the Text object, while the style of the displayed text (font, bold, italics, color, etc.) is determined by the TextModel object. In this case, the TextModel plays the same role as a text style in word processing software. The analyst can ensure that the same text style is used for multiple Text objects by sharing the same TextModel between all these objects.

Although DisplayModels determine the appearance of a DisplayEntity, the DisplayModel has no graphics itself and therefore cannot be dragged and dropped in the normal manner. To create a new DisplayModel, simply duplicate an existing DisplayModel. JaamSim starts with a pre-defined example of each type of Display Model that can be used for this purpose. Duplicate an example Display Model by right-clicking its entry in the Object Selector and selecting "Duplicate", then edit its attributes and name as necessary.

A selection of DisplayModels can be found in the "Display Models" palette in the Object Selector. Each of these subclasses of DisplayModel in intended for a particular subclass of DisplayEntity, as noted below.

List of objects:

- ColladaModel
- ImageModel
- TextModel
- ScreenPointsModel
- ArrowModel
- GraphModel
- DisplayModelCompat

All of these various Display Model objects have the same Basic Graphics keywords, described below, that control optional rendering and scaling at different drawing ranges.

Key Inputs:

Keyword	Description
`VisibleViews`	A list of Views for which this Display Model is shown. If empty, the model appears on all views.
`DrawRange`	A list of two values for the minimum and maximum distance from the camera this object is visible.
`ModelScale`	A list of three multiplicative factors by which to scale the model in the x-, y- and z-dimensions respectively.

8.1 - ColladaModel

ColladaModel objects are used to display custom 3D graphics in a simulation model. The COLLADA file format (.DAE) is an interchange file format used for 3D graphics. Any 3D object such as a DisplayEntity and most of its sub-classes can accept a ColladaModel as its DisplayModel.

A number of other 3D formats can be used in addition to Collada. At the present time, JaamSim supports DAE, OBJ, JSM, and JSB formats as well as zipped versions of these files (ZIP). The JSM and JSB formats are specific to JaamSim:

- The JSM format was introduced to allow 3D objects to be animated. These files can be created using a plug-in for Blender, an open-source 3D graphics program. Animation is still in the early stages of development at the present time. Please contact us for more information if you wish to use this feature.
- The JSB format is a binary format that allows complex 3D objects to be loaded much faster than is possible with the DAE and OBJ formats. A JSB file can be exported by right clicking on a ColladaModel in the Object Selector and selecting "Export 3D Binary File (*.jsb)". A present, the ColladaModel must have been created from a DAE file. Support for OBJ and JSM files will be added in a future version of the software. Note that the

exported JSB will look for the same textures as the DAE file and that these should be located in the same relative position to the JSB file as they were to the DAE file.

Managing 3D assets can be complicated because of the large file sizes and the need to provide separate files for the textures. We recommend that the user place each 3D asset and its texture files in its own ZIP file. This approach greatly reduces the number and size of the files being handled, and ensures that the files can be moved between computers without breaking the file paths. Note that it is often necessary to edit the DAE file with a text editor such as Notepad++ to convert any absolute file paths for texture file to relative ones.

The table below describes the additional inputs for ColladaModel objects:

Key Inputs:

Keyword	Description
ColladaFile	A file path to the DAE, OBJ, JSM, and JSB file to be used for this display model. A ZIP file containing one of these files and its related texture files can also be used, and is the recommended option for this input. The file path must be enclosed in single quotes (') if it contains spaces.
Actions	A list of animation Actions and the object Outputs to which they are to be connected, in the format { { Action1 Output1 } { Action2 Output2 } }. In this example, the actions Action1 and Action2 must be available for the graphical asset specified by the ColladaFile input. The outputs Output1 and Output2 must be valid outputs for the entity to which this ColladaModel is assigned. Animation is only available for 3D objects in the JSM format.

The following outputs are provided for ColladaModels:

Outputs:

Output Name	Description
Actions	A list of the actions that are provided in the JSM file.
Vertices	The number of unique vertices that are specified in the 3D model.
Triangles	The number of unique triangles that are specified in the 3D model.
VertexShareRatio	The ratio (number of triangles)/[(number of vertices)/3]. Equal to 1.0 if every vertex is used in only one triangle.

8.2 - ImageModel

ImageModel objects are used to display custom 2D graphics in a simulation model, such as a picture or a map. Both DisplayEntity and OverlayImage can accept an ImageModel object as an input. Image file types currently supported by JaamSim include BMP, GIF, JPG, PCX and PNG files, or a ZIP file containing any one of these file types.

The table below describes the additional inputs for ImageModel objects:

Key Inputs:

Keyword	Description
ImageFile	A file path to the image file to be used for this display model. Must be enclosed in single quotes (') if the path contains spaces.
Transparent	If TRUE, transparency is enabled for supported image types (GIF and PNG).
CompressedTexture	If TRUE, image compression is applied in order to alleviate memory issues with large images.

8.3 - TextModel

TextModel objects specify the general appearance of Text objects and of Overlay Objects that display text (OverlayText and OverlayClock), as well as BillboardText. A TextModel object can therefore be used as a style class, with all instances of these text objects that have the same style sharing the same TextModel.

The table below describes the additional inputs for TextModel objects.

Key Inputs:

Keyword	Description
FontName	The font face to be used for text with this TextModel (selectable via a drop-down menu)
FontColour	The color of the text, defined by a color keyword or RGB values.
FontStyle	Styles (BOLD, ITALIC) to be applied to the font (selectable via a pop-up menu).
DropShadow	If TRUE, a drop shadow is displayed below the text.
DropShadowColour	The colour of the drop shadow, defined by a color keyword or RGB values.
DropShadowOffset	The spacing between the text and its drop shadow, specified in { x, y, z } coordinates.

8.4 - ScreenPointsModel

The ScreenPointsModel is used to determine the graphics for entities that appear as a single- or multi-segment line, such as EntityConveyor and EntityDelay. It cannot be used by other objects such as DisplayEntity that are intended for 3D or 2D graphics.

ScreenPointsModel has no inputs other than the standard ones for DisplayModels.

8.5 - ArrowModel

The ArrowModel is used to determine the graphics for the Arrow entity.

Key Inputs:

Keyword	Description
ArrowSize	The x, y, z dimensions of the arrowhead.

8.6 - GraphModel

The GraphModel is used to determine the presentation style and proportions for various types of Graph objects. To make the graph scalable to any size, all length measurements are specified as fraction of the graph's total height.

Key Inputs:

Keyword	Description
TitleTextHeight	Text height for the graph title. Expressed as a fraction of the graph's total height.
XAxisTitleTextHeight	Text height for the x-axis title. Expressed as a fraction of the graph's total height.
YAxisTitleTextHeight	Text height for the y-axis title. Expressed as a fraction of the graph's total height.
LabelTextHeight	The text height for both x- and y-axis labels. Expressed as a fraction of the graph's total height.
TitleGap	The gap between the graph title and the top of the graph. Expressed as a fraction of the graph's total height.
XAxisTitleGap	The gap between the x-axis labels and the y-axis title. Expressed as a fraction of the graph's total height.
XAxisLabelGap	The gap between the x-axis and its labels. Expressed as a fraction of the graph's total height.
YAxisTitleGap	The gap between the y-axis labels and the y-axis title. Expressed as a fraction of the graph's total height.
YAxisLabelGap	The gap between the y-axis and its labels. Expressed as a fraction of the graph's total height.
TopMargin	Size of the gaps between the respective edges of the outer pane and the graph. Expressed as a fraction of the graph's total height.
BottomMargin	
LeftMargin	
RightMargin	
TitleTextModel	The TextModel used to determine the font, color, and style for the graph title. The dropshadow settings are ignored. Black Verdana text is used if this input is left blank.
AxisTitleTextModel	The TextModel used to determine the font, color, and style for the x- and y-axis titles. The dropshadow settings are ignored. Black Verdana text is used if this input is left blank.
LabelTextModel	The TextModel used to determine the font, color, and style for the labels next to the x- and y-axis tick marks. The dropshadow settings are ignored. Black Verdana text is used if this input is left blank.
GraphColor	The color of the graph background.
BackgroundColor	The color of the outer pane background.
BorderColor	The color of the graph border.

8.7 - DisplayModelCompat

JaamSim includes native DisplayModel objects that serve as default visual representations for objects, including some graphics for entities commonly modelled in Ausenco's Transportation Logistics Simulator module (shipping, rail, and terminal equipment).

The appearances of these objects are controlled by the additional DisplayModel keywords, listed in the table below:

Key Inputs:

Keyword	Description
Shape	The graphical appearance of the object, chosen from a selection of pre-defined shapes.
FillColour	The colour of the filled part of the display model, defined by a colour keyword or RGB values.
OutlineColour	The colour of the outline of the display model, defined by a colour keyword or RGB values.
Filled	If TRUE, the model will appear with a solid colour fill. Otherwise, the model will appear hollow.
Dashed	If TRUE, the display model outline will appear as a dashed line.
Bold	If TRUE, the display model outline will appear thicker than normal.

9 - Probability Distributions Palette

The probability distributions palette provides a standard selection of theoretical distributions as well as user-defined distributions.

9.1 - Theoretical Probability Distributions

The following set of standard probability distributions are available. The distributions were coded using algorithms adapted from "Simulation Modeling & Analysis", 4th Edition, by Averill M. Law. Random numbers for these distributions are generated by the Multiple Recursive Generator developed by L'Ecuyer ("Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators", Operations Res., 47: 159-164 (1999a)).

Distribution Name	Description
UniformDistribution	Generates samples with a constant probability between a minimum and maximum value.
TriangularDistribution	Generates samples from a triangular distribution between a minimum and maximum value. The distribution peaks at its mode.
NormalDistribution	Generates samples from a normal distribution.
ExponentialDistribution	Generates samples from a negative exponential probability distribution.
ErlangDistribution	Generates samples from an Erlang probability distribution.
GammaDistribution	Generates samples from a Gamma probability distribution.
WeibullDistribution	Generates samples from a Weibull probability distribution.
LogNormalDistribution	Generates samples from a Log-Normal probability distribution.
LogLogisticsDistribution	Generates samples from a Log-Logistics probability distribution.

9.2 - User-Defined Probability Distributions

Users can define a distribution point by point, if required. A user-defined distribution can be either a continuously varying value or one of a discrete set of values.

Distribution Name	Description
ContinuousDistribution	Generates samples over a continuous range of values.
DiscreteDistribution	Generates samples from a discrete set of values.

9.3 - Keywords for Probability Distributions

Each of the distributions uses the following standard keywords in addition to the ones that are specific to an individual distribution.

Key Inputs:

Keyword	Description
UnitType	The unit type for the value returned by the distribtion, e.g. { TimeUnit }. To keep the units consistent for other inputs, this input must be set first.
RandomSeed	The random seed to be used by the random number generator.
GlobalSubstreamSeed	The substream number to be used by the random number generator. The substream number entered for one probability distribution is applied globally to **ALL** random number streams in the model.
MinValue	The minimum value that can be returned by the distribution. A value less than the minimum is rejected and re-sampled from the distribution.
MaxValue	The maximum value that can be returned by the distribution. A value greater than the maximum is rejected and re-sampled from the distribution.

A number of outputs are standard to all distributions.

Outputs:

Output Name	Description
CalculatedMean	The mean value for the distribution calculated directly from the inputs. Ignores the values entered for the MinValue and MaxValue keywords.
CalculatedStandardDeviation	The standard deviation for the distribution calculated directly from the inputs. Ignores the values entered for the MinValue and MaxValue keywords.
NumberOfSamples	The total number of samples returned by the distribution.
SampleMin	The minimum of the samples returned by the distribution.
SampleMax	The maximum of the samples returned by the distribution.
SampleMean	The average of the samples returned by the distribution.
SampleStandardDeviation	The standard deviation of the samples returned by the distribution.

9.4 - Distributions that Return an Object

Two distributions are available that return an object instead of a number.

Distribution Name	Description
RandomSelector	Randomly selects an object from a list of objects and probabilities.
EntitlementSelector	Selects an object from a list of objects and probabilities based on an entitlement algorithm. The object returned is the one that minimizes the difference between the actual number returned for that object and the expected number to be returned based on the probabilities.

10 - Basic Objects Palette

The Basic Objects palette contains a number of fundamental objects that can be used to create simple models by drag and drop. Many of these objects are similar to the ones provided by commercial off-the-shelf simulation packages such as Arena, Simio, etc.

A number of objects such as Queue, TimeSeries, and TimeSeriesThreshold are also used extensively by TLS models. The use of these objects in TLS are described in the user manuals for that software.

List of objects:

- ReportGenerator
- EntityGenerator
- Server
- EntitySink
- EntityConveyor
- EntityDelay
- Assign
- Queue
- Seize
- Release
- Resource
- Branch
- Assemble
- EntityGate
- EntitySignal
- TimeSeries
- SignalThreshold
- ExpressionThreshold
- TimeSeriesThreshold
- Statistics

10.1 - ReportGenerator

The ReportGenerator object controls the initialization period and duration of the simulation run and prints the output report. Without a ReportGenerator object, a simulation run will continue indefinitely. No more than one ReportGenerator should be defined for a simulation model.

A simulation run is executed in two steps. The first step is the initialization period which should be long enough for the model to reach typical inventory levels for the various storage objects and typical lengths for the various queues. The statistics collected by each object are cleared at the end of the initialization period. The second step is the run duration over which the final set of statistics is collected. An output report is created at the completion of the run and the simulation is either paused or, if in batch mode, terminated.

The `InitializationDuration` and `RunDuration` keywords determine the initialization period and the duration of the simulation run respectively. Note that total length of the simulation run is equal to the sum of these values.

The output report is a text file whose file name is the same as the configuration file name with the extension changed to .rep. The output report is created in the same directory as the configuration file, unless another directory is specified using the `ReportDirectory` keyword. The output file consists of a listing of the output values that appear in the Output Viewer. Only the outputs for statistical values or attributes values are printed. To avoid unnecessary clutter, outputs that simply show the present state of the system, such as the present length of a queue or the present position of an object, are ignored.

Each output is printed on a separate line using the following formats:

- Output with units: <object name> Output[<output name>, <unit>] <value>

  e.g.: Queue1 Output[AverageQueueTime, s] 24.758558850846765

- Output without units: <object name> Output[<output name>] <value>

  e.g.: Queue1 Output[NumberAdded] 143

This format was chosen to make it easier to parse by a spreadsheet macro or a scripting language. Typically, selected outputs for a series of simulation runs are collected in a spreadsheet to form a summary table.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
ReportDirectory	A string containing the file path to the directory in which the output report is to be printed. If left blank, the report is placed in the same directory as the configuration file.
InitializationDuration	The length of the initialization period for the simulation run.
RunDuration	The length of the simulation run.

Outputs:

None.

10.2 - EntityGenerator

The EntityGenerator object creates a series of entities that are passed to the next object in a process. The PrototypeEntity keyword identifies the entity to be copied. This entity can be any type of object, no matter how complex. Copies retain both the graphics of the prototype as well as the values of all its inputs.

The rate at which entities are generated is determined by the InterArrivalTime and FirstArrivalTime keywords. These inputs have units of time and can be a constant value, a TimeSeries, a Probability Distribution, or a mathematical expression.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
OperatingThresholdList	A list of threshold objects that must all be open for operation to proceed. Operation is stopped if any one of the thresholds is closed.
FirstArrivalTime	The time at which the first entity is to be generated.
InterArrivalTime	The elapsed time between one generated entity and the next.
PrototypeEntity	The entity to be copied by the generator.
MaxNumber	The maximum number of entities to be generated.

Outputs:

Output Name	Description
obj	Not applicable to the EntityGenerator.
NumberAdded	Not applicable to the EntityGenerator.
NumberProcessed	The total number of entities that have been generated.
ProcessingRate	The rate at which entities are generated (entities per second).

10.3 - Server

The Server object processes an incoming entity and then passes it to the next object. Entities that are waiting to be processed are held by a Queue object identified by the keyword `WaitQueue`. All entities received by the Server first pass through this Queue object, even if the Server is idle.

The rate at which entities are processed is determined by the `ServiceTime` keyword. This input has units of time and can be a constant value, a TimeSeries, a Probability Distribution, or a mathematical expression.

Default Graphic:

Key Inputs:

Keyword	Description
`Description`	Optional free-form text that describes the object.
`AttributeDefinitionList`	User-defined outputs to be added to this entity.
`NextComponent`	The next object to which the generated entity is passed.
`ServiceTime`	The time required to process the incoming entity.
`WaitQueue`	A Queue object in which to hold entities that are waiting to be processed.

Outputs:

Output Name	Description
`obj`	Entity last received by the Server.
`NumberAdded`	Number of entities received by the Server.
`NumberProcessed`	The total number of entities that have been processed by the Server.
`ProcessingRate`	The rate at which entities have been serviced (NumberProcessed/Total Time).

10.4 - EntitySink

The EntitySink object destroys each incoming entity.

Default Graphic:

Key Inputs:

Keyword	Description
`Description`	Optional free-form text that describes the object.
`AttributeDefinitionList`	User-defined outputs to be added to this entity.

Outputs:

Output Name	Description
obj	Entity last received by the EntitySink.
NumberAdded	Number of entities received by the EntitySink.
NumberProcessed	The total number of entities that have been destroyed by the EntitySink.
ProcessingRate	The rate at which entities have been destroyed (NumberProcessed/Total Time).

10.5 - EntityConveyor

The EntityConveyor object moves an incoming entity along a path at a fixed speed, and then passes it to the next object. The travel time for the EntityConveyor is determined by the TravelTime keyword. This time must be a constant value.

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
TravelTime	The time required convey an entity from the start to the end.
Points	The coordinates for the start, intermediate joints, and end of the EntityConveyor.
Width	The width of the line representing the conveyor in pixels.
Color	The color of the line representing the conveyor.

Outputs:

Output Name	Description
obj	Entity last received by the EntityConveyor.
NumberAdded	Number of entities received by the EntityConveyor.
NumberProcessed	The total number of entities that have been processed by the EntityConveyor.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).

10.6 - EntityDelay

The EntityDelay object delays an incoming entity by a variable duration before passes it to the next object. The delay is represented as motion along a line that is similar in appearance to the EntityConveyor object. It differs, however, in that the entities moving along the line can pass one another due to their having different delay times.

The duration of each entities delay is determined by the Duration keyword. This input has units of time and can be a constant value, a TimeSeries, a Probability Distribution, or a mathematical expression.

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which an entity will be passed once it has completed its delay.
Duration	The time required to process the incoming entity.
Points	The coordinates for the start, intermediate joints, and end of the EntityDelay.
Width	The width of the line representing the conveyor in pixels.
Color	The color of the line representing the conveyor.

Outputs:

Output Name	Description
obj	Entity last received by the EntityDelay.
NumberAdded	Number of entities received by the EntityDelay.
NumberProcessed	The total number of entities that have been processed by the EntityDelay.
ProcessingRate	The rate at which entities have been serviced (NumberProcessed/Total Time).

10.7 - Assign

The Assign object performs one or more attribute assignments when it receives an incoming entity. The Assign object is the only place where an attribute's value can be modified.

Attribute assignments can be performed for the Assign object itself, the received entity, or for any other entity in the model. The received entity is passed to the next object without delay, once the assignments have been performed.

The attribute assignment to be performed are specified by the AttributeAssignmentList keyword. Examples of the inputs to this keyword are:

- { 'this.A = this.A + 1' }
- { 'this.obj.B = this.obj.B + 1' }
- { '[Entity1].C = [Entity1].C + 1' }

In this example, the entries A, B, and C are the attributes of the Assign object, the entity received, and Entity1, respectively.

The right-hand side of each assignment equation is a mathematical expression to be evaluated. The left-hand side identifies the attribute whose value is to be modified. The attribute is named using the same conventions as an expression. Note that only attributes can be assigned; it is not possible to assign a new value to an output. Outputs can only appear in the expression on the right-hand side of assignment equation.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the received entity is passed.
AttributeAssignmentList	The assignment equations to be performed on receipt of an entity.

Outputs:

Output Name	Description
obj	Entity received by the Assign object.
NumberAdded	Number of entities received by the Assign object.
NumberProcessed	The total number of entities that have been processed by the Assign object.
ProcessingRate	The rate at which entities have been received (NumberProcessed/Total Time).

10.8 - Queue

A Queue object defines a location for simulation entities to wait for processing by other simulation entities. For instance, a Server requires a Queue in which incoming entities can wait for processing by the Server.

Default Graphic:

Key Inputs:

Keyword	Description
Spacing	The amount of graphical space between objects in the queue.
MaxPerLine	Maximum number of objects in each row of the queue.
PrintReport	If TRUE, a queue report file (QUE file) is output summarizing the contents of the queue every time an object is added or removed.

Outputs:

Output Name	Description
QueueLength	The present number of objects in the queue.
NumberAdded	The total number of objects that have been added to the queue.
NumberRemoved	The total number of objects that have been removed from the queue.
QueueLengthAverage	The average number of objects in the queue.
QueueLengthStandardDeviation	The standard deviation of the number of objects in the queue.
QueueLengthMinimum	The fewest number of objects in the queue.
QueueLengthMaximum	The largest number of objects in the queue.
QueueLengthDistribution	A list of decimal values summing to 1.0. The first entry is the fraction of time that the queue is empty. The second entry is the fraction of time that the queue contains one object, and so on. The number of entries is equal to the maximum queue length plus one.

10.9 - Seize

The Seize object allocates one or more units of a specified Resource on receiving an incoming entity. If the number of available Resource units is insufficient, the received entity is directed to a Queue object identified by the keyword WaitQueue. Entities waiting for a Resource at more than one Seize object are processed on a first-in-first-out basis.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
Resource	The Resource to be seized.
NumberOfUnits	The number of units of the Resource to be seized.
WaitQueue	A Queue object in which to hold entities that are waiting for sufficient Resource units to be available.

Outputs:

Output Name	Description
obj	Entity last received by the Seize object.
NumberAdded	Number of entities received by the Seize object.
NumberProcessed	The total number of entities that have been processed by the Seize object.
ProcessingRate	The rate at which entities have been processed by the Seize object (NumberProcessed/Total Time).

10.10 - Release

The Release object de-allocates one or more units of a specified Resource on receiving an incoming entity. The Seize block with the entity that has waited the longest for the Resource is notified to process this entity.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
Resource	The Resource to be released.
NumberOfUnits	The number of units of the Resource to be released.

Outputs:

Output Name	Description
obj	Entity last received by the Release object.
NumberAdded	Number of entities received by the Release object.
NumberProcessed	The total number of entities that have been processed by the Release object.
ProcessingRate	The rate at which entities have been processed by the Release object (NumberProcessed/Total Time).

10.11 - Resource

The Resource object provides a pool of units that can be seized and released by the Seize and Release objects.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
Capacity	The number of units that can be seized.

Outputs:

Output Name	Description
UnitsSeized	The total number of units that have been seized to date.
UnitsReleased	The total number of units that have been released to date.
UnitsInUse	The number of units that are in use at present.
UnitsInUseAverage	The average number of units that are in use.
UnitsInUseStandardDeviation	The standard deviation of the number of units that are in use.
UnitsInUseMinimum	The fewest number of units that have been in use at any time.
UnitsInUseMaximum	The largest number of units that have been in use at any time.
UnitsInUseDistribution	A list of decimal values summing to 1.0. The first entry is the fraction of time that no units were in use. The second entry is the fraction of time that one unit was in use, and so on. The number of entries is equal to the maximum number of units that have been in use at any time plus one.

10.12 - Branch

The Branch object directs an incoming entity to a destination that is chosen from a list of alternatives. The value for the Choice keyword determines the destination that is chosen: 1 = first branch, 2 = second branch, etc. The input to Choice can be a constant value, a random distribution, a time series, or a mathematical expression. The use of mathematical expression allows the choice to be made based on the attributes of the incoming entity.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponentList	A list of possible objects to which incoming entities can be passed.
Choice	A number that determines which downstream branch will be chosen for the incoming entity: 1 = first branch, 2 = second branch, etc.

Outputs:

Output Name	Description
obj	Entity last received by the Branch object.
NumberAdded	Number of entities received by the Branch object.
NumberProcessed	The total number of entities that have been processed by the Branch object.
ProcessingRate	The rate at which entities have been processed by the Branch object (NumberProcessed/Total Time).

10.13 - Assemble

The Assemble object combines a number of sub-components into an assembled part. Sub-components waiting for processing are collected into a series of Queue objects, identified by the WaitQueueList keyword, one for each type of sub-component.

Incoming sub-components are directed to a queue using the Choice keyword. Normally, this keyword would refer to an attribute of the incoming entity. For example, if the incoming entity has an attribute named Type, whose value was 1 for the first type sub-component, 2 for the second type of sub-component, etc., then the entry for the Choice keyword should be the expression: this.obj.Type.

The assembly process begins when there is at least one sub-component entity in each of the queues. Once this occurs, the first sub-component in each queue is removed and destroyed, and a new assembled part is created by copying the object specified by the PrototypeEntity keyword. The time required to complete the assembly process is given by the ServiceTime keyword.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
OperatingThresholdList	A list of Threshold objects which must be open for the assembly operation to proceed.
ServiceTime	The time required to complete the assembly process.
WaitQueueList	A set of Queue objects, one for each type of sub-component to be assembled.
Choice	The index of the queue in which an incoming sub-component is to be placed: 1 = first queue, 2 = second queue, etc.
PrototypeEntity	The entity whose copies will represent the assembled part.

Outputs:

Output Name	Description
obj	Entity last received by the Assemble object.
NumberAdded	Number of entities received by the Assemble object.
NumberProcessed	The total number of assembled parts that have been created by the Assemble object.
ProcessingRate	The rate at which assembled parts have been created (NumberProcessed/Total Time).

10.14 - EntityGate

The EntityGate object allows incoming entities to be blocked temporarily. When open, the EntityGate allows incoming entities to pass through without delay. When closed, incoming entities are directed to a Queue where they are held until such time as the EntityGate opens. When the EntityGate opens, the queued entities are released one at time, separated by a delay determined by the `ReleaseDelay` keyword. This delay does not apply to entities that arrive when the EntityGate is open. However, if queued entries are still being released, an incoming entity is placed at the end of the queue even though the EntityGate is open.

The EntityGate's state, either open or closed, is determined by the input to the `OperatingThresholdList` keyword, which specifies a list of Threshold objects such as SignalThreshold, TimeSeriesThreshold, or ExpressionThreshold. All of the specified Threshold objects must be open for the EntityGate to be open.

Default Graphic:

Key Inputs:

Keyword	Description
`Description`	Optional free-form text that describes the object.
`AttributeDefinitionList`	User-defined outputs to be added to this entity.
`NextComponent`	The next object to which the incoming entities are passed.
`OperatingThresholdList`	A list of Threshold objects that determine whether the EntityGate is open or closed.
`WaitQueue`	A Queue object in which to hold incoming entities that are blocked when EntityGate is closed.
`ReleaseDelay`	The time required to remove each entity from the Queue.
`InitialState`	The EntityGate's state at the start of the simulation run: TRUE = open, FALSE = closed.

Outputs:

Output Name	Description
`obj`	Entity last received by the EntityGate.
`NumberAdded`	Number of entities received by the EntityGate.
`NumberProcessed`	The total number of entities that have been processed by the EntityGate.
`ProcessingRate`	The rate at which entities have been received by the EntityGate (NumberProcessed/Total Time).

10.15 - EntitySignal

The EntitySignal object sets the state of a SignalThreshold object when it receives an incoming entity. The SignalThreshold and its new state are specified by the `TargetSignalThreshold` and `NewState` keywords, respectively.

Default Graphic:

Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which the generated entity is passed.
TargetSignalThreshold	The SignalThreshold object whose state will be changed by this EntitySignal.
NewState	The state to be set for the SignalThreshold: TRUE = open, FALSE = closed.

Outputs:

Output Name	Description
obj	Entity last received by the EntitySignal .
NumberAdded	Number of entities received by the EntitySignal .
NumberProcessed	The total number of entities that have been processed by the EntitySignal .
ProcessingRate	The rate at which entities have been received by the EntitySignal object (NumberProcessed/Total Time).

10.16 - TimeSeries

The TimeSeries object simulates a numerical value that varies with time. Many JaamSim keywords are structured to accept a constant value, a Probability Distribution, or a Time Series, which makes this object a powerful and flexible component of any simulation model.

The data for the object consists of a series of time stamps and values as specified by the `Value` keyword. The time stamps can be irregularly spaced and it is possible to repeat the time series over and over again during the simulation using the `CycleTime` keyword.

The value returned by a TimeSeries object has a specific unit type specified by the `UnitType` keyword. To maintain consistency, the `UnitType` must be specified prior to any other unit. Once specified, it cannot be changed.

Default Graphic:

Key Inputs:

Keyword	Description
UnitType	One of the pre-defined unit types within JaamSim. The list of valid UnitTypes can be found in the Object Selector under Units. Once set, this input cannot be changed by the user.
Value	A series of timestamps, along with the associated value for the TimeSeries at the given time. The UnitType and must be specified before the Value keyword to ensure that the inputs pass validation.
CycleTime	When the time series will repeat from the start. By default, the TimeSeries object will not cycle.

The format for timestamps follows the ISO 8601 standard, and can take one of the following three forms:

- YYYY-MM-DD (e.g. 2013-08-25 for August 25, 2013)
- YYYY-MM-DDTHH:mm:SS (e.g. 2013-08-25T04:25:00 for 4:25 AM on August 25, 2013)
- 'YYYY-MM-DD HH:mm:SS' (e.g. '2013-08-25 04:25:00' for 4:25 AM on August 25, 2013)

Additionally, timestamps will also accept the full RFC8601 format following the forms:

- YYYY-MM-DDTHH:mm:SS.dddddd (e.g. 2013-08-25T04:25:00.1 for 0.1 seconds past 4:25 AM on August 25, 2013)

- HHHHHHHH:mm:SS.dddddd (e.g. 50000:00:00.000001 for 1 microsecond past 50,000 hours)

The first timestamp must occur on January 1, 00:00. The year in TimeSeries `Value` lists does not need to match the Simulation `StartDate` keyword value; the model assumes that both the timestamps and the simulation begin at the same time. When the simulation reaches the next timestamp of the TimeSeries object, the associated value will be updated in the model. If the simulation time is greater than the `CycleTime`, the TimeSeries will repeat from the beginning of the `Value` list.

Example: Repeated TimeSeries values

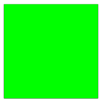
```
Define TimeSeries { TS }
TS  UnitType { SpeedUnit }
TS  Value { { 2014-01-01T00:00:00 0.00 km/h } { 2014-01-03T00:00:00 0.76 km/h } { 2014-01-11T00:00:00 0.24 km/h } }
TS  CycleTime { 14 d }
```

In this example, if the simulation started on 2014-01-01, it would take on the new values on January 3 (0.76 km/h) and then January 11 (0.24 km/h). On 2014-01-15, the first value of the list would be read again and the returned value from the TimeSeries would be 0. The TimeSeries would take on new values on January 17 and 25, and this 14-day cycle would repeat until the end of the simulation run.

10.17 - SignalThreshold

The SignalThreshold object varies its state between open and closed when instructed to do so by one or more EntitySignal objects. It has no internal logic of its own for changing state.

Default Graphic:



Key Inputs:

Keyword	Description
`Description`	Optional free-form text that describes the object.
`AttributeDefinitionList`	User-defined outputs to be added to this entity.

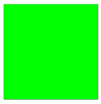
Graphics:

Keyword	Description
`OpenColour`	The color to display when the SignalThreshold is open.
`ClosedColour`	The color to display when the SignalThreshold is closed.
`ShowWhenOpen`	If TRUE, the SignalThreshold is displayed when it it open. If FALSE, it is hidden when open.
`ShowWhenClosed`	If TRUE, the SignalThreshold is displayed when it it closed. If FALSE, it is hidden when closed.

10.18 - ExpressionThreshold

The ExpressionThreshold object varies its state between open and closed depending on the value returned by the mathematical expression specified by the `OpenCondition` keyword. The expression is re-evaluated with every time advance.

Default Graphic:



Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
OpenCondition	A mathematical expression that returns a boolean value: TRUE or 1 = open, FALSE or 0 = closed.

Graphics:

Keyword	Description
OpenColour	The color to display when the ExpressionThreshold is open.
ClosedColour	The color to display when the ExpressionThreshold is closed.
ShowWhenOpen	If TRUE, the ExpressionThreshold is displayed when it it open. If FALSE, it is hidden when open.
ShowWhenClosed	If TRUE, the ExpressionThreshold is displayed when it it closed. If FALSE, it is hidden when closed.

10.19 - TimeSeriesThreshold

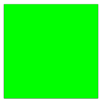
The TimeSeriesThreshold object varies its state between open and closed depending on the state of a TimeSeries object specified by its TimeSeries keyword.

The trigger levels for the TimeSeriesThreshold object are determined by its MinOpenLimit and MaxOpenLimit keywords. For the TimeSeriesThreshold to be open, the present value for the TimeSeries must be within the range specified by these two keywords.

The LookAhead and Offset keywords provide additional flexibility:

- If a non-zero value is specified for the LookAhead keyword, then the value for TimeSeries must be within the range specified by the MinOpenLimit and MaxOpenLimit keywords over the entire LookAhead duration, starting from the present time.
- If a non-zero value is specified for the Offset keyword, then the above rule is modified so that the LookAhead duration begins at the present time plus the offset.

Default Graphic:



Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
AttributeDefinitionList	User-defined outputs to be added to this entity.
UnitType	The unit type for the threshold (e.g. DistanceUnit, TimeUnit, MassUnit).
TimeSeries	The name of time series object for which the threshold applies.
MaxOpenLimit	The limit over which the threshold is closed.
MinOpenLimit	The limit under which the threshold is closed.
LookAhead	The amount of time that the threshold must remain within MinOpenLimit and MaxOpenLimit to be considered open.
Offset	The amount of time that the TimeSeriesThreshold adds to the present time for each TimeSeries lookup.

Graphics:

Keyword	Description
OpenColour	The color to display when the TimeSeriesThreshold is open.
ClosedColour	The color to display when the TimeSeriesThreshold is closed.
ShowWhenOpen	If TRUE, the TimeSeriesThreshold is displayed when it it open. If FALSE, it is hidden when open.
ShowWhenClosed	If TRUE, the TimeSeriesThreshold is displayed when it it closed. If FALSE, it is hidden when closed.

10.20 - Statistics

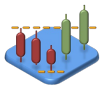
The Statistics object collects statistical information on the entities it receives.

The quantity that is tracked is specified using the SampleValue keyword, which can accept any valid mathematical expression. Some typical examples for this input are:

- 'this.obj.A' - the sample value is the attribute A carried by the received entity
- 'this.SimTime - this.obj.t' - the sample value is the simulation time that has elapsed since the attribute t for the received entity was set to the simulation time at some earlier point in the model

The TestEntity keyword should be set to the prototype for the entities received by the Statistics object. This entity is needed to test whether the attributes used in the SampleValue expression actually exist.

Default Graphic:



Key Inputs:

Keyword	Description
Description	Optional free-form text that describes the object.
TestEntity	The prototype entity used to generate the entities received by the Statistics object.
NextComponent	The next object to which the generated entity is passed.
SampleValue	A mathematical expression that returns the sample value whose statistics are to be collected.

Outputs:

Output Name	Description
obj	Entity last received by the Statistics object.
NumberAdded	Number of entities received by the Statistics object.
NumberProcessed	The total number of assembled parts that have been created by the Statistics object.
ProcessingRate	The rate at which assembled parts have been created (NumberProcessed/Total Time).
SampleMinimum	The smallest value that was observed.
SampleMaximum	The largest value that was observed.
SampleAverage	The average of the values that were observed.
SampleStandardDeviation	The standard deviation of the values that were observed.
StandardDeviationOfTheMean	The estimated standard deviation of the sample average.

11 - Advanced Topics

JaamSim has been designed such that simple simulation models can be realized using the Graphical User Interface alone. However, some advanced functions will require direct editing of the model input configuration file (`.cfg` file). The configuration file is in plain text and can be created using a text editor. We recommend the use of Notepad++, an open-source editor available for download at http://notepad-plus-plus.org.

This section discusses the structure of the input configuration file as well as some examples that require editing the configuration file outside of the GUI.

List of topics and objects:

- Editing the Configuration File
- Launching in Batch Mode
- ScriptEntity
- VideoRecorder

11.1 - Editing the Configuration File

Basic Structure

A JaamSim input configuration file consists of a series of lines, akin to a scripting language. Each line consists of a combination of object names, keywords, and values contained within braces. One or more spaces are used to separate these elements. Braces are also used to denote sets of arguments within the outer braces required for arguments in general. Blank lines and extra whitespace are ignored by JaamSim.

Comments beginning with a double quote can be used to document the input files. There is no need to use a double quote mark to end the comment, since JaamSim automatically recognizes the end of a comment with the end of the line. If a comment extends for several lines, each line must start with a double quote.

Object Definitions

In JaamSim, an object is initialized by a Define statement. The statement contains `Define` followed by the object type, and the object name enclosed by braces. Multiple objects can be defined at the same time, provided that they are of the same type. For instance, the following two lines define respectively a single Arrow object and three Arrow objects.

```
Define Arrow  { SingleArrow }
Define Arrow  { Arrow1 Arrow2 Arrow3 }
```

JaamSim ignores extraneous whitespace, allowing the object definition to span multiple lines provided that all objects are surrounded by braces. Object instances can only be referenced after they have been defined.

Object Inputs

Once an object is defined, its keyword values can be set using a command of the following form:

```
<object name> <keyword> { <value1> <value2> … }
```

where `value1`, `value2`, … is the list of values for the keyword separated by one or more spaces. For instance, the following line sets the colour of the `Arrow1` object to be black:

```
Arrow1 Colour { 'black' }
```

Multiple parameters for an object can be set in one line containing the object name followed by keyword and value pairs.

```
Arrow1 Colour { 'black' } Width { 2 }
```

Inner curly braces are used for keywords that accept multiple input values.

```
Arrow1 Points { { 0 0 0 m } { 1 1 1 m } }
```

Include Statements

The user can store input data in multiple files and then refer to these files in an input configuration file using Include statements. These statements refer to other input configuration files by filename and path, surrounded by single quotes:

```
Include  '..\Base File\InputFile.cfg'
```

Include statements are particularly useful in studies where the analyst wishes to vary only a few inputs across many simulation runs. Using Include statements, the analyst can create a base case configuration file and then use Include statements to create simple incremental configuration files for the additional runs:

```
Include       '..\Base File\Basecase.cfg'
Arrow1 Width { 2.0 }
```

This example includes the contents of `Basecase.cfg` and, modifies the already-defined object `Arrow1` keyword `Width` value to `2.0`. Note that the changes from the base case configuration must appear after the Include statement. These simple configuration files are useful because it is easy to tell exactly how the configuration differs from the base case configuration.

Group "Objects"

Group objects bundle multiple objects together to simplify inputs. Instead of referring to a long list of objects, a single group is used instead. The group may be used to set the value for the keyword for all members instead of setting the value for each member of the group. Certain keywords also accept Group objects as values.

Considerable caution must be exercised when using Group objects because they do not behave in the same way as other JaamSim objects. Although they are defined in the same way as other objects, they are used only as an input convenience. When JaamSim reads the configuration file, it replaces each group by its list of objects. Groups are ignored once the configuration file has been read.

Key Inputs:

Keyword	Description
`List`	A list of names of the objects included in the list, enclosed by braces.

Example:

The following example demonstrates the use of groups:

```
Define     Arrow       { Arrow1 Arrow2 Arrow3 }
Define     Group       { ArrowList }
ArrowList  List        { Arrow1 Arrow2 Arrow3 }
ArrowList  Color       { 'black' }
```

In this example, a group of three Arrow objects is created and used to assign the color 'black' to each Arrow.

By using the `List` keyword, a fourth Arrow can be added to the Group:

```
Define     Arrow       { Arrow4 }
ArrowList  List        { ArrowList Arrow4 }
ArrowList  Color       { 'black' }
```

RecordEdits Statement

The RecordEdits statement is used to preserve the organisation and formatting of a configuration file that has been prepared manually by the user.

It is usually best to construct a complex model manually using a text editor. These inputs are carefully formatted and organised, and include comments to document to model design. However, once this material has been created, the easiest way to position the objects and to add graphics such as titles, labels, etc. is through the graphical user interface. If the model was then saved, all the formatting and comments would normally be lost.

JaamSim avoids this predicament with the RecordEdits statement. On saving, JaamSim copies all inputs before the RecordEdits statement line-by-line to the saved file, and then saves all the changes to the model using computer-written inputs. The following example illustrates this structure:

```
" Manually prepared inputs:
" - Everything before the RecordEdits statement is unchanged when JaamSim saves a file.

RecordEdits

" Computer written inputs:
" - Everything that appears after the RecordEdits statement is written by the computer.
```

The "Save" functionality in JaamSim is disabled when an input file is loaded without a RecordEdits statement. In this case, the "Save As" operation adds a RecordEdits statement to the end of the original configuration file and then writes out the new inputs.

11.2 - Launching in Batch Mode

JaamSim models can be executed in batch mode with minimized graphics. Batch mode is useful when only the output reports are required or when a series of runs are to be executed without user intervention.

The model can also be launched, configured and started automatically from the command line or a batch file using the command:

```
JaamSim.exe run1.cfg -tags
```

Here, `run1.cfg` is the name of the input file to be loaded and `-tags` are the optional tags for the run. Multiple tags must be separated by a space. The following tags are supported:

Tag	Description
`-b or -batch`	Start the simulation immediately after the input file has been read, and exit when the run has completed. This tag is useful for batch file execution.
`-m or -minimize`	Minimize the graphical user interface, making the simulation run faster when visualizations are not required (for instance, in overnight simulation runs).

A series of simulation runs can be executed one after the other by using a batch file that contains a series of these commands. For example, a batch file containing the following two lines would execute two runs: `run1.cfg` and `run2.cfg`:

```
JaamSim.exe run1.cfg -b
JaamSim.exe run2.cfg -b
```

Note that the batch file and input configuration files must be in the same directory for this example to work.

11.3 - ScriptEntity

The ScriptEntity object takes only one keyword, which is the path to a script (`.scr`) file.

Key Inputs:

Keyword	Description
`Script`	Name of the file containing the scripting instructions to be loaded.

The script file contains sets of model configuration inputs preceded by the `Time` keyword (belonging to the ScriptEntity object). Originally developed for video capture, a script can be used to change window views, to create automatic zooming and panning, and to turn on and off video capture during a run. Furthermore, keywords defined in the script file can be used to modify simulation and object parameters initially defined in the input configuration file.

Key Inputs for .scr File:

Keyword	Description
`Time`	The simulated time at which the subsequent inputs are executed.

Example

The following inputs can be entered into a `.scr` file and then referenced by a ScriptEntity object in order to slow down the model at a given point in the simulation run:

```
ScriptEntity1 Time            { 24.0 h }
Simulation    RealTime        { TRUE }
Simulation    RealTimeFactor  { 1200 }

ScriptEntity1 Time            { 30.0 h }
Simulation    RealTime        { FALSE }
```

11.4 - VideoRecorder

JaamSim includes functionality to capture still images or videos of simulation models via the VideoRecorder object. The length and playback speed of the video is governed by three keywords:

- `CaptureStartTime` is the simulated time at which video capture begins;
- `CaptureFrames` is the number of frames to capture;
- `CaptureInterval` is the simulated time elapsed between frames.

The video recorder output frame rate is fixed at 30 frames per second.

The video recorder naturally composites the selected views (from CaptureViews) onto a surface with dimensions specified in CaptureArea, with windows borders not rendered. The example inputs below illustrate the capture of a 20-second long video (in 1080p resolution) spanning 600 hours of simulated time:

```
Define  VideoRecorder   {  VideoRecorder1  }

VideoRecorder1       VideoCapture            { TRUE }
VideoRecorder1       CaptureInterval         { 1 h }
VideoRecorder1       CaptureStartTime        { 0 h }
VideoRecorder1       CaptureFrames           { 600 }
VideoRecorder1       CaptureArea             { 1920 1080 }
VideoRecorder1       SaveVideo               { TRUE }
VideoRecorder1       CaptureViews            { Overview }
VideoRecorder1       VideoBackgroundColor    { skyblue }
```

Key Inputs:

Keyword	Description
`CaptureStartTime`	The simulated time at which video capture begins.
`VideoBackgroundColor`	The background colour for parts of the video frame not covered by a view window.
`CaptureFrames`	The number of frames to capture.
`CaptureInterval`	The amount of simulated time elapsed between captured frames.
`SaveImages`	If TRUE, individual screen captures are made for each frame and output to image files.
`SaveVideo`	If TRUE, video capture is enabled and an AVI file is output by the video recorder.
`CaptureArea`	The size of the rendered image or video in pixels.
`CaptureViews`	A list of views to include in the video or image.
`VideoName`	The name of the video output file.

12 - Colour Name Index with RGB Values

Colour Name	Colour	R	G	B		Colour Name	Colour	R	G	B
lavenderblush		255	240	245		chocolate		210	105	30
pink		255	192	203		rawsienna		199	97	20
lightpink		255	182	193		sienna		160	82	45
palevioletred		219	112	147		brown		138	54	15
hotpink		255	105	180		lightsalmon		255	160	122
deeppink		255	20	147		darksalmon		233	150	122
violetred		208	32	144		salmon		250	128	114
mediumvioletred		199	21	133		lightcoral		240	128	128
raspberry		135	38	87		coral		255	114	86
thistle		216	191	216		tomato		255	99	71
plum		221	160	221		orangered		255	69	0
orchid		218	112	214		red		255	0	0
violet		238	130	238		crimson		220	20	60
magenta		255	0	255		firebrick		178	34	34
purple		128	0	128		indianred		176	23	31
mediumorchid		186	85	211		burntumber		138	51	36
darkorchid		153	50	204		maroon		128	0	0
darkviolet		148	0	211		sepia		94	38	18
blueviolet		138	43	226		white		255	255	255
indigo		75	0	130		gray99		252	252	252
mediumpurple		147	112	219		gray98		250	250	250
lightslateblue		132	112	255		gray97		247	247	247
mediumslateblue		123	104	238		gray96		245	245	245
slateblue		106	90	205		gray95		242	242	242
darkslateblue		72	61	139		gray94		240	240	240
ghostwhite		248	248	255		gray93		237	237	237
lavender		230	230	250		gray92		235	235	235
blue		0	0	255		gray91		232	232	232
darkblue		0	0	139		gray90		229	229	229
navy		0	0	128		gray89		227	227	227
midnightblue		25	25	112		gray88		224	224	224
cobalt		61	89	171		gray87		222	222	222
royalblue		65	105	225		gray86		219	219	219
cornflowerblue		100	149	237		gray85		217	217	217
lightsteelblue		176	196	222		gray84		214	214	214
lightslategray		119	136	153		gray83		212	212	212
slategray		112	128	144		gray82		209	209	209
dodgerblue		30	144	255		gray81		207	207	207

Colour Name	Colour	R	G	B		Colour Name	Colour	R	G	B
aliceblue		240	248	255		gray80		204	204	204
powderblue		176	224	230		gray79		201	201	201
lightblue		173	216	230		gray78		199	199	199
lightskyblue		135	206	250		gray77		196	196	196
skyblue		135	206	235		gray76		194	194	194
deepskyblue		0	191	255		gray75		191	191	191
peacock		51	161	201		gray74		189	189	189
steelblue		70	130	180		gray73		186	186	186
darkturquoise		0	206	209		gray72		184	184	184
cadetblue		95	158	160		gray71		181	181	181
azure		240	255	255		gray70		179	179	179
lightcyan		224	255	255		gray69		176	176	176
paleturquoise		187	255	255		gray68		173	173	173
cyan		0	255	255		gray67		171	171	171
turquoise		64	224	208		gray66		168	168	168
mediumturquoise		72	209	204		gray65		166	166	166
lightseagreen		32	178	170		gray64		163	163	163
manganeseblue		3	168	158		gray63		161	161	161
teal		0	128	128		gray62		158	158	158
darkslategray		47	79	79		gray61		156	156	156
turquoiseblue		0	199	140		gray60		153	153	153
aquamarine		127	255	212		gray59		150	150	150
mintcream		245	255	250		gray58		148	148	148
mint		189	252	201		gray57		145	145	145
seagreen		84	255	159		gray56		143	143	143
mediumspringgreen		0	250	154		gray55		140	140	140
springgreen		0	255	127		gray54		138	138	138
emeraldgreen		0	201	87		gray53		135	135	135
mediumseagreen		60	179	113		gray52		133	133	133
cobaltgreen		61	145	64		gray51		130	130	130
darkseagreen		143	188	143		gray50		127	127	127
honeydew		240	255	240		gray49		125	125	125
palegreen		152	251	152		gray48		122	122	122
lawngreen		124	252	0		gray47		120	120	120
greenyellow		173	255	47		gray46		117	117	117
limegreen		50	205	50		gray45		115	115	115
forestgreen		34	139	34		gray44		112	112	112
sapgreen		48	128	20		gray43		110	110	110
green		0	128	0		gray42		107	107	107
darkgreen		0	100	0		gray41		105	105	105

Colour Name	Colour	R	G	B		Colour Name	Colour	R	G	B
darkolivegreen		85	107	47		gray40		102	102	102
olivedrab		107	142	35		gray39		99	99	99
olive		128	128	0		gray38		97	97	97
ivory		255	255	240		gray37		94	94	94
lightyellow		255	255	224		gray36		92	92	92
lightgoldenrodyellow		250	250	210		gray35		89	89	89
cornsilk		255	248	220		gray34		87	87	87
lemonchiffon		255	250	205		gray33		84	84	84
beige		245	245	220		gray32		82	82	82
yellow		255	255	0		gray31		79	79	79
khaki		240	230	140		gray30		77	77	77
lightgoldenrod		255	236	139		gray29		74	74	74
palegoldenrod		238	232	170		gray28		71	71	71
darkkhaki		189	183	107		gray27		69	69	69
banana		227	207	87		gray26		66	66	66
gold		255	215	0		gray25		64	64	64
goldenrod		218	165	32		gray24		61	61	61
darkgoldenrod		184	134	11		gray23		59	59	59
brick		156	102	31		gray22		56	56	56
floralwhite		255	250	240		gray21		54	54	54
seashell		255	245	238		gray20		51	51	51
oldlace		253	245	230		gray19		48	48	48
linen		250	240	230		gray18		46	46	46
antiquewhite		250	235	215		gray17		43	43	43
papayawhip		255	239	213		gray16		41	41	41
blanchedalmond		255	235	205		gray15		38	38	38
eggshell		252	230	201		gray14		36	36	36
bisque		255	228	196		gray13		33	33	33
moccasin		255	228	181		gray12		31	31	31
navajowhite		255	222	173		gray11		28	28	28
wheat		245	222	179		gray10		26	26	26
peachpuff		255	218	185		gray9		23	23	23
tan		210	180	140		gray8		20	20	20
burlywood		222	184	135		gray7		18	18	18
melon		227	168	105		gray6		15	15	15
sandybrown		244	164	96		gray5		13	13	13
cadmiumyellow		255	153	18		gray4		10	10	10
carrot		237	145	33		gray3		8	8	8
orange		255	128	0		gray2		5	5	5
flesh		255	125	64		gray1		3	3	3
cadmiumorange		255	97	3		black		0	0	0