JaamSim

User Manual

Software Version: 2016-07
April 7, 2016

Table of Contents

1 Introduction

JaamSim (Java Animation Modelling and Simulation) is a discrete-event simulation software package first developed in 2002 as the foundation for simulation applications. JaamSim includes a drag-and-drop graphical user interface, 3D animation, and a full set of built-in objects for model building. It is object oriented, extremely fast, and scalable to the largest of applications. Windows, Linux, and OSX are all supported.

JaamSim represents the lessons learned from simulation projects we have performed around the world for more than 35 years. Further background information on JaamSim can be found in the JaamSim blog: www.jaamsim.com/blog.

JaamSim is free open source software, licensed under Apache 2.0. 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/jaamsim/jaamsim. Presentations and tutorials for JaamSim can be found by following the Videos link on the JaamSim website.

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.

JaamSim provides all the key functions needed for any simulation model:

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

It also provides a full suite of basic objects for model building:

- Objects for process flow-type models (servers, queues, etc.);
- Text objects for labelling and documentation;
- Graphs for visualizing simulation outputs;
- Probability distributions for random sampling; and
- Graphical objects for background maps and logos.

Advanced users can create additional palettes of high-level objects for their applications. A separate Programming Manual can be downloaded from the JaamSim website.

2 Installing JaamSim

2.1 System Requirements

JaamSim runs under Windows, Linux, and OSX on most modern computers.

Support is required for OpenGL graphics version 3.0 or later (OpenGL 2.1 is also supported with some degraded functionality). These specifications are provided by typical laptop computers with Intel Core i3, i5, and i7 series processors with integrated graphics (second generation "Sandy Bridge" processors and later).

For models with high-end graphics, a NVIDIA GeForce graphics card is recommended. It is not necessary to use the more expensive NVIDIA Quadro workstation type graphics cards. Although it is possible to use a computer with an AMD graphics card, NVIDIA cards generally 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 www.oracle.com. The 64-bit version of the JRE is preferred for 64-bit operating systems.

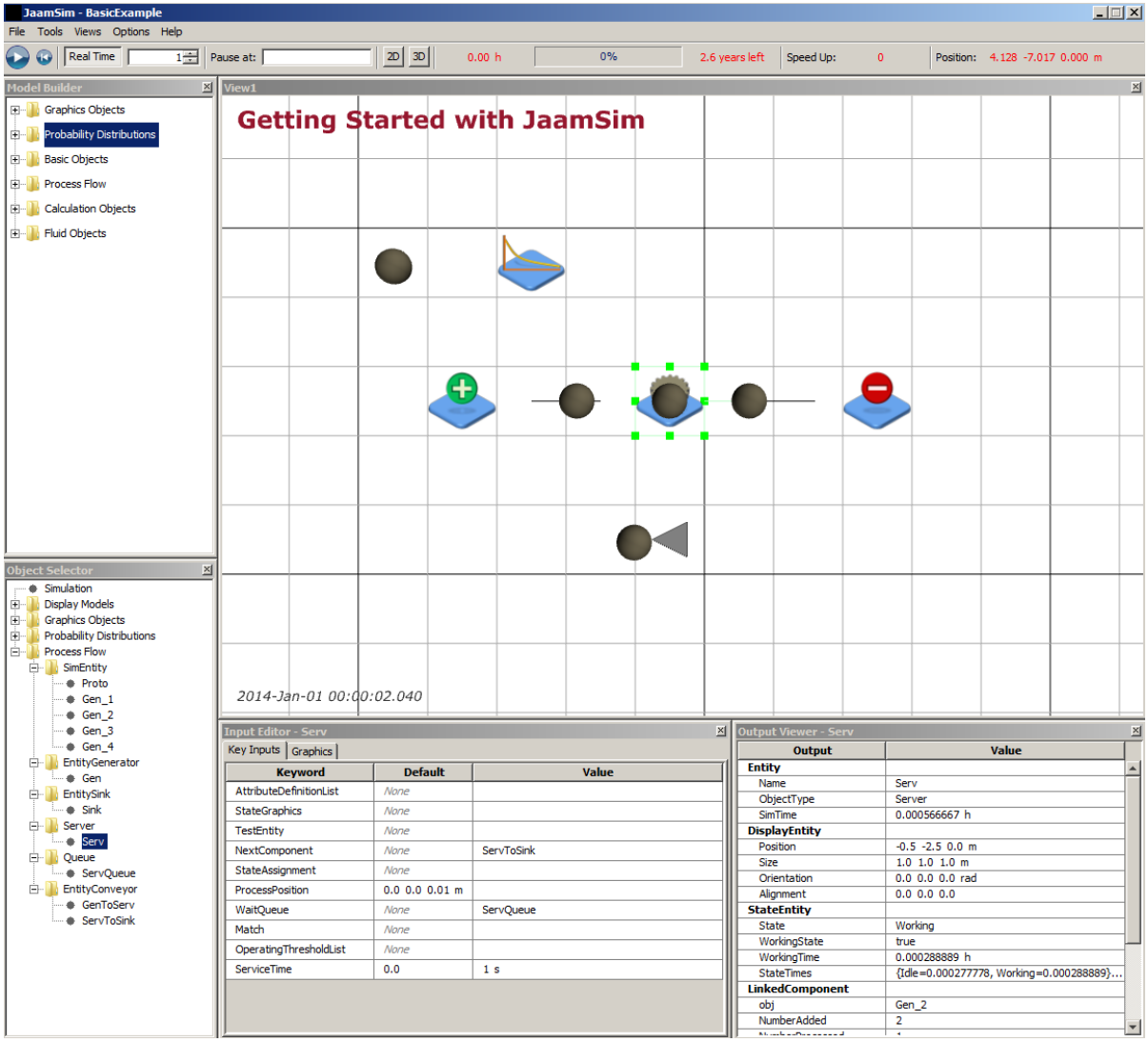
2.3 Installing and Launching JaamSim

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.

3 JaamSim Basic Example

In this example, you will be guided through building a basic model using the graphical user interface (GUI). The example model simulates a typical single-server queuing system, with entities being generated, processed, and consumed. The finished model will look similar to Figure 3-1.

Figure 3-1 Screenshot of the 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

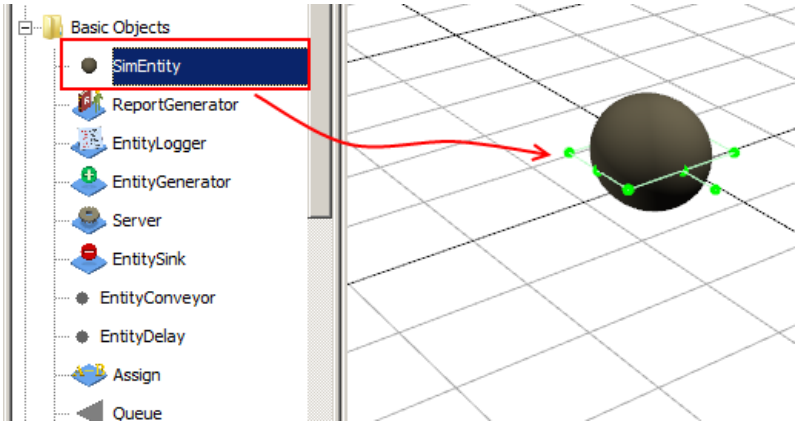
3.1 Step 1: Creating Model Objects

After launching JaamSim, the following windows will appear:

- **Control Panel** – provides a number of run control feature;
- **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; and
- **Output Viewer** – displays model outputs for a selected object.

In the **Model Builder**, expand the Process Flow palette and then drag-and-drop a SimEntity into the **View Window** (View1). This creates a SimEntity object with a default name (SimEntity1) and shape (Sphere) and automatically selects it, denoted by green highlight as shown in Figure 3-2.

Figure 3-2 Highlighted SimEntity in View Window



This object will serve as the prototype for entities that will be handled in the model. In the **Object Selector**, select SimEntity1 and press **F2** or triple-click it to rename the object.

Table 3-1 Objects to Create For Step 1 (Part 1)

Model Builder Palette	Object Type	Name
Process Flow	SimEntity	Proto

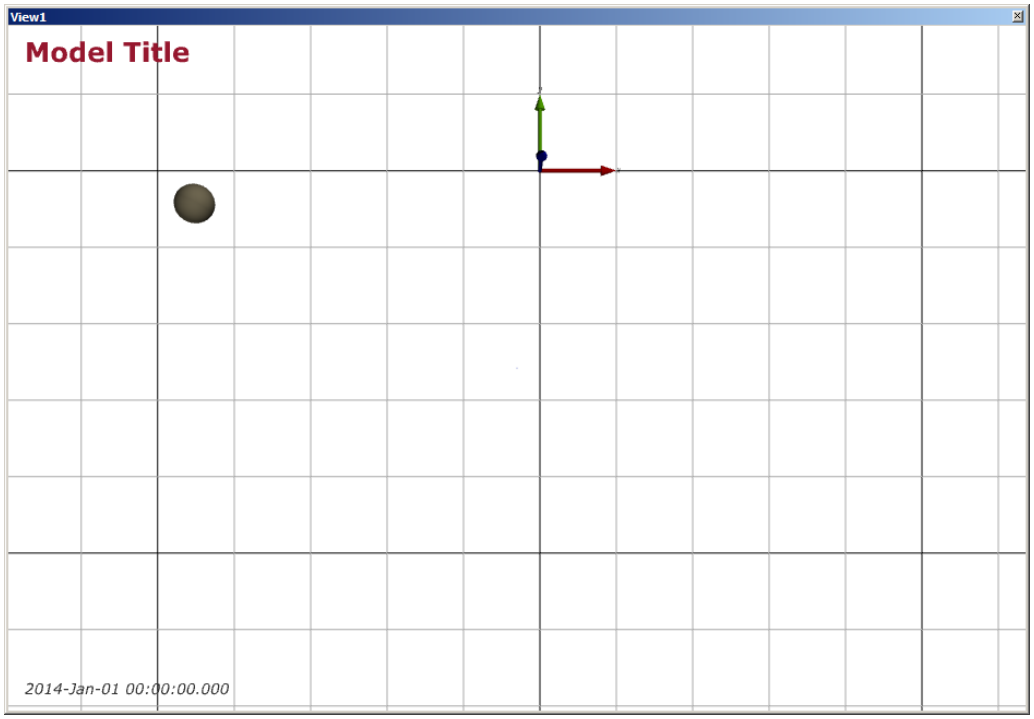
Since the model graphics will be 2D aside from the Proto entity, click the **2D** button on the **Control Panel**, so that the view becomes bird's eye. Adjust the viewing position using the actions listed in Table 3-2:

Table 3-2 View Adjustments

Keyword	Description
Left Click + Drag	Pan in the XY-plane
Scroll Wheel	Zoom in or out

After adjusting the view position and zoom, the View window should appear similar to Figure 3-3.

Figure 3-3 Configured View Window



Before creating any additional objects, it is helpful to turn on the Snap-to-Grid setting by clicking on the Options entry in the Control Panel's menu bar and selecting the 'Snap to Grid' menu item.

Now create and rename the objects listed in Table 3-3:

Table 3-3 Objects to Create for Step 1 (Part 2)

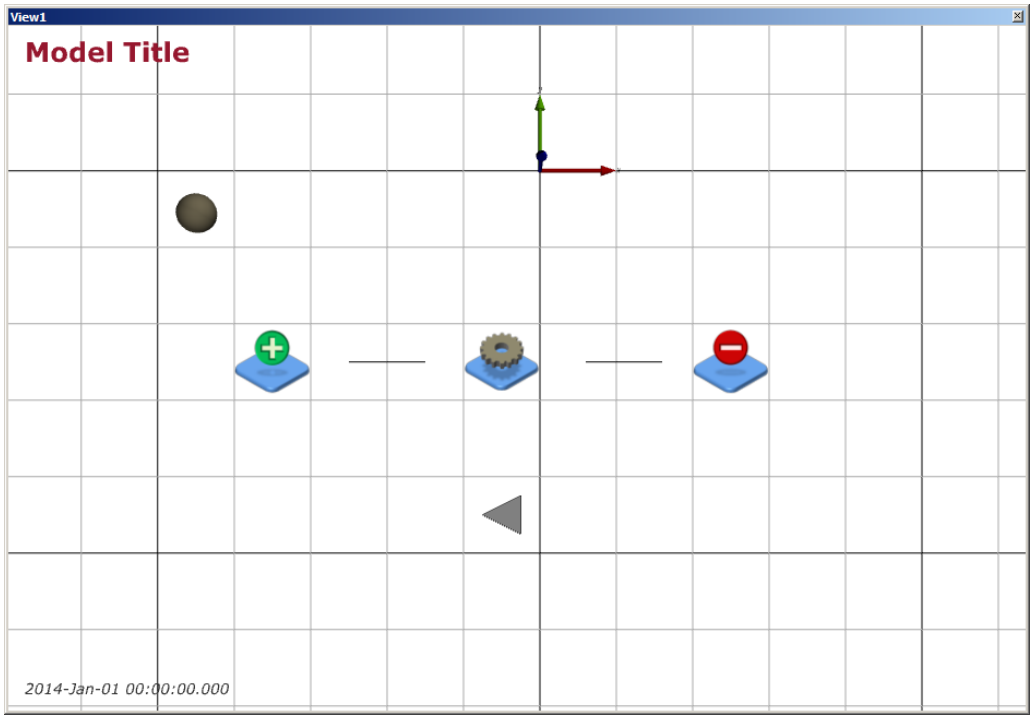
Model Builder Palette	Object Type	Name
Process Flow	EntityGenerator	Gen
Process Flow	EntityConveyor	GenToServ
Process Flow	Server	Serv
Process Flow	EntityConveyor	ServToSink
Process Flow	EntitySink	Sink
Process Flow	Queue	ServQueue

An object can be moved by selecting it and using **CTRL + Left Click + Drag.** Position the first five objects from left to right in the following order:

Gen - GenToServ - Serv - ServToSink - Sink

Place the final object, ServQueue, below the Serv object. After positioning the objects, the model should look similar to Figure 3-4.

Figure 3-4 Screenshot of Step 1



3.2 Step 2: Putting the Objects Together

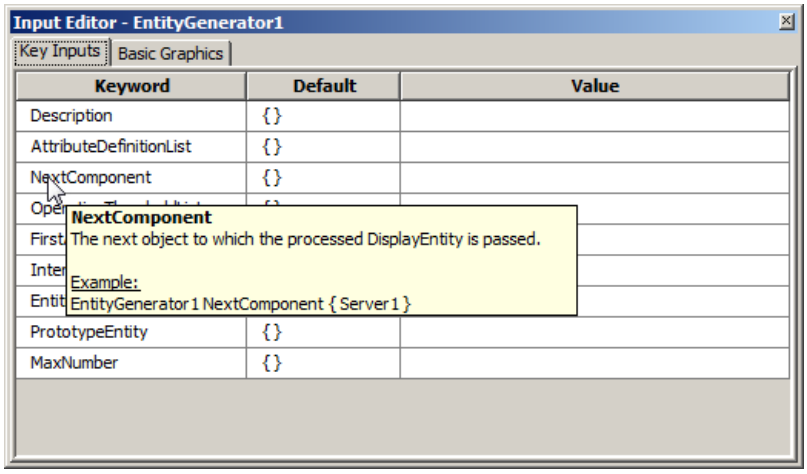
In this step, the objects placed in Step 1 must be set to interact with one another. Connect the objects together by setting the keyword values listed in Table 3-4.

Table 3-4 Object Connections

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

For reference, hovering the cursor over keywords in the Input Editor will display a brief description of the keyword. For example, the mouse-over for the NextComponent keyword is shown in Figure 3-5.

Figure 3-5 Mouse-Over Tool Tips



Set the time at which Proto entities will be added to the model by Gen, and the time that entities will spent at each stage of the model:

Table 3-5 Transit and Handling Durations

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

Server objects require Queue objects to hold entities waiting to be processed. Set the WaitQueue keyword on Serv to send waiting entities to ServQueue:

Table 3-6 Serv Inputs

Tab	Keyword	Value
Key Inputs	WaitQueue	ServQueue

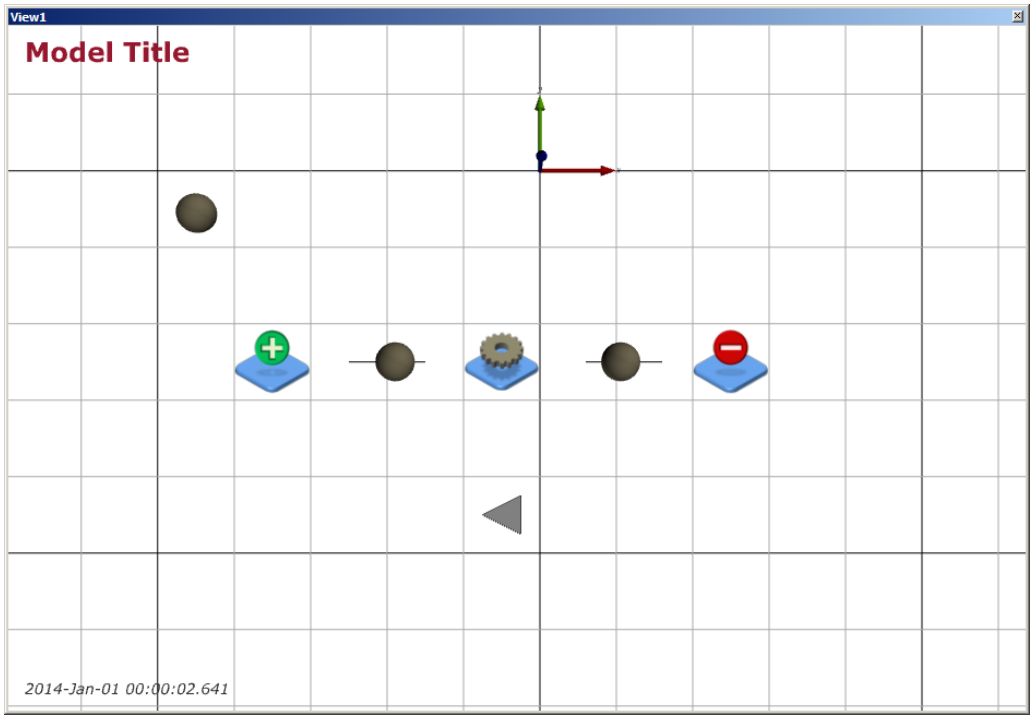
Lastly, configure Gen to produce copies of Proto:

Table 3-7 Gen Inputs

Tab	Keyword	Value
Key Inputs	PrototypeEntity	Proto

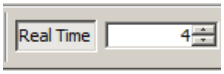
Save the model by selecting **Save As…** from the **File** menu, and press the **Play** button to run the simulation. The model should appear similar to Figure 3-6.

Figure 3-6 Screenshot of Step 2



Note that the **Real Time** button in the **Control Panel** is depressed, as shown in Figure 3-7. This indicates that the model speed is restricted to the Real Time speed-up factor shown in the text box to the right of the Real Time button. The Real Time speed-up factor controls how fast the simulated time elapses in the model. The default Real Time speed-up factor is 1, meaning that each real (wall-clock) second corresponds to one simulated second. It can be changed by entering the desired Real Time speed-up factor into the text box, or by pressing the up or down arrow buttons (which double or halve the Real Time speed-up factor respectively).

Figure 3-7 Real Time Controls

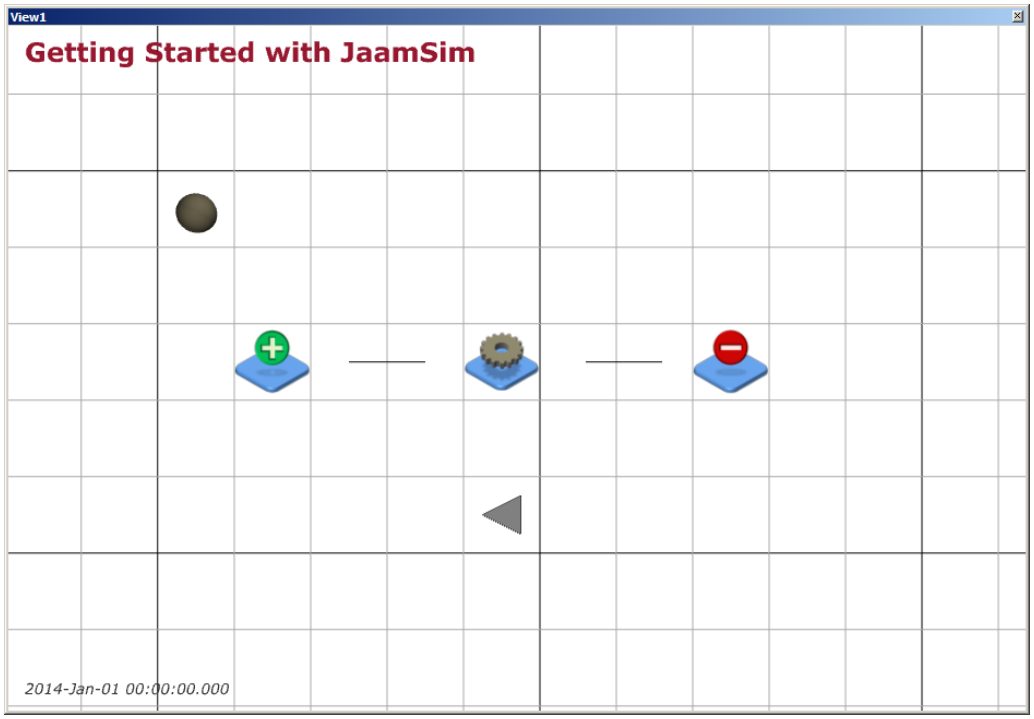


To continue working on the model, **Pause** or **Reset** the model by selecting the corresponding buttons on the Control Panel. Pausing the model allows the simulation to later resume starting from paused time, while resetting the model forces the model to start from time zero.

3.3 Step 3: Changing Model Graphics

In this step, some graphical adjustments will 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 more complex models. At the end of this step, the model should look similar to Figure 3-8.

Figure 3-8 Screenshot of Step 3



In addition to moving objects, **CTRL + Left Click + Drag** can also resize objects. If a handle is selected during this process, the object will be resized, otherwise it will move following the cursor. These two techniques are illustrated using a cube in Figure 3-9 and Figure 3-10.

Figure 3-9 Moving an Object

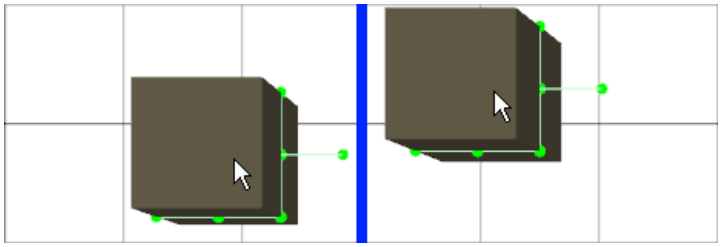
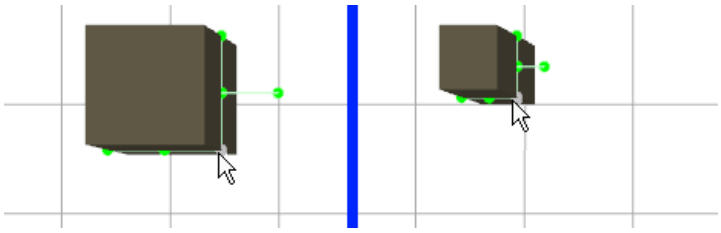


Figure 3-10 Resizing an Object



Since the XYZ-Axis is not useful in 2D viewing, hide it by un-checking **Show Axes** from the **Options** menu on the Control Panel.

Change the title of the model in the View window by returning to the Object Selector and expanding the OverlayText palette as shown in Figure 3-11. Select the Title object and change the keyword value listed in Table 3-8.

Figure 3-11 Changing the Model Title

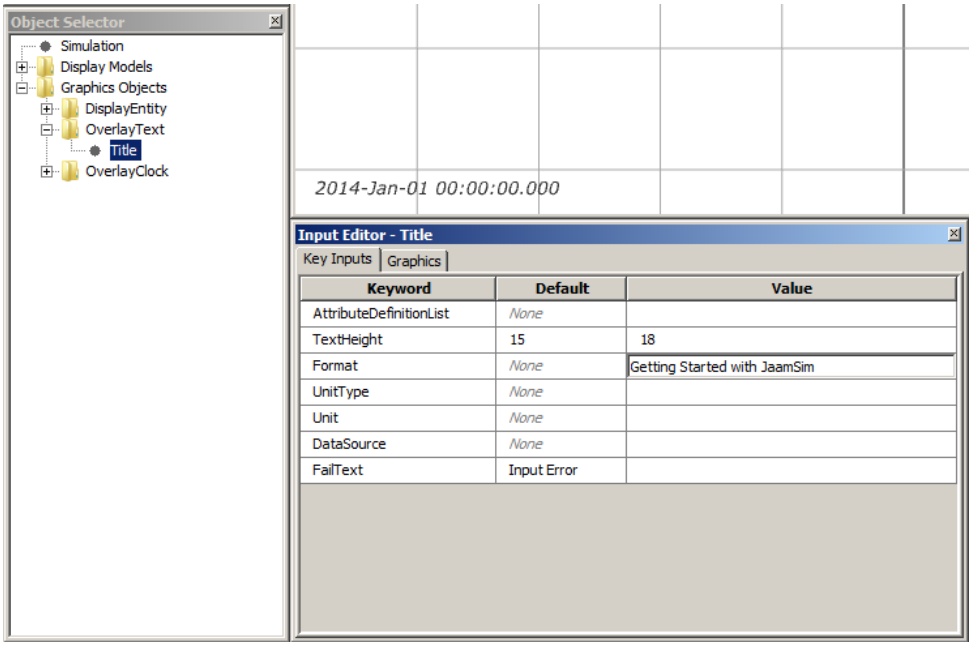


Table 3-8 Title Inputs

Object	Keyword	Value
Title	Format	'Getting Started with JaamSim'

Play the model to observe the effects of these changes, and reset the simulation when ready to proceed.

3.4 Step 4: Adding a Probability Distribution

In this step, dynamic variability is added to the model by including a probability distribution to control the inter-arrival time of Proto entities.

Create an exponential distribution object as listed in Table 3-9.

Table 3-9 Object to Create for Step 4

Model Builder Palette	Object Type	Name
Probability Distributions	ExponentialDistribution	GenIATDist

Set the keyword values for GenIATDist as shown in Table 3-10.

Table 3-10 GenIATDist Inputs

Tab	Keyword	Value
Key Inputs	UnitType	TimeUnit
Key Inputs	MinValue	0 s
Key Inputs	MaxValue	10 s
Key Inputs	Mean	2 s

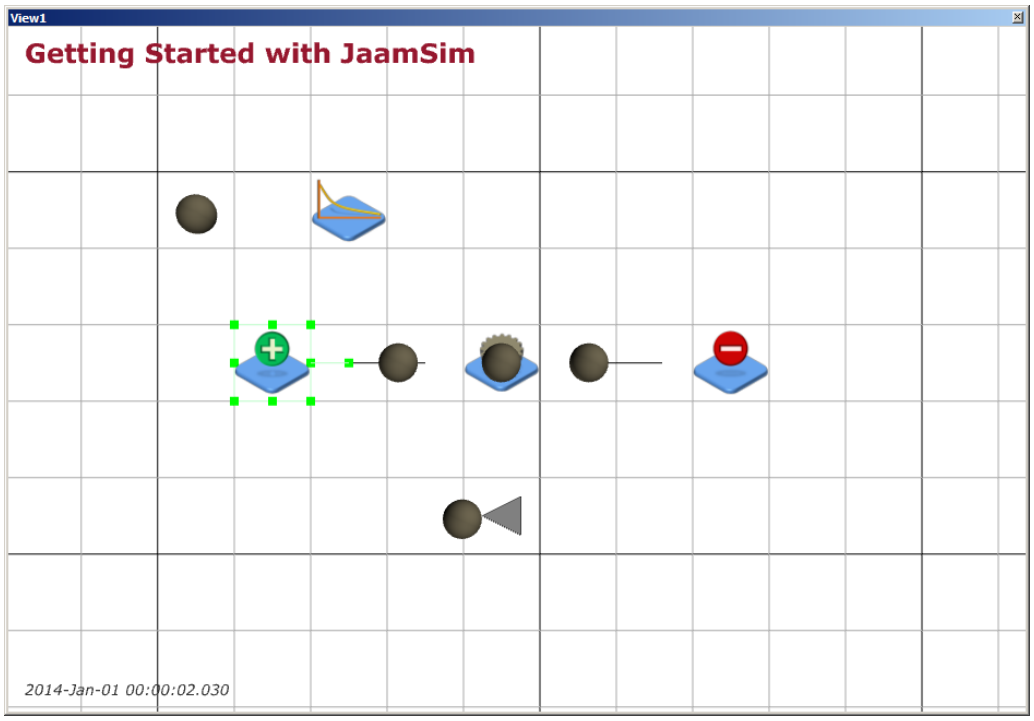
Update the Gen object to sample the GenIATDist distribution for its interarrival time:

Table 3-11 Gen Inputs

Tab	Keyword	Value
Key Inputs	InterArrivalTime	GenIATDist

Save the model again and press **Play**. Observe that the Proto entities are now generated randomly, and that there are now times when Proto entities are waiting in ServQueue to be processed, as seen in Figure 3-12.

Figure 3-12 Screenshot of Step 4



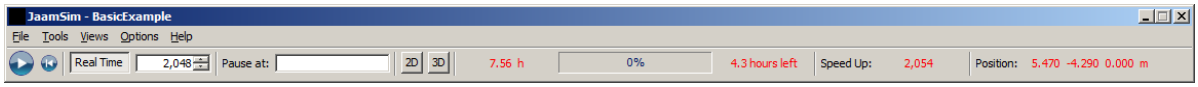
4 Graphical User Interface

The graphical user interface (GUI) consists of the Control Panel, one or more View windows, the Model Builder, the Object Selector, the Input Editor, and the Output Viewer. These are described in the following sub-sections.

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.

Figure 4-1 JaamSim Control Panel



The Control Panel is divided into two rows, consisting of the Menu Bar and the Tool Bar:

4.1.1 Menu Bar

File Menu

The File entry displays a menu with actions related to saving and loading model input configuration files.

Table 4-1 File Menu

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 one or more 3D models or images. Creates both the ColladaModels/ImageModels containing the graphics and the corresponding DisplayEntities.
Print Input Report	Prints the present inputs in a standard file format (.inp).
Exit	Closes all windows and exits JaamSim.

Tools

The Tools entry displays a menu that provides options for showing the windows of the JaamSim graphical user interface:

Table 4-2 Tools Menu

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

The Views entry displays a menu containing a list of currently defined View windows and provides the ability to create new Views. A View window shows a graphical 3D representation of the model.

Table 4-3 Views Menu

Menu Entry	Description
View1	Opens the window for View1, the default View. Does nothing if the window has already been opened.
Define new View	Creates a new View object and displays its window.

Options

The Options entry in the menu bar contains the following entries:

Table 4-4 Options Menu

Menu Entry	Description
Snap to Grid	If checked, an object being dragged with the mouse will automatically position itself to the nearest grid point.
Show Axes	If checked, coordinate axes are displayed at the origin (0, 0, 0) of the coordinate grid.
Show Grid	If checked, the coordinate grid is displayed on the xy-plane.
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

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

4.1.2 Tool Bar

The left side of the Tool Bar contains controls for manipulating the simulation run and the 3D view for the active View window. The right side shows the status of the simulation run with parameters such as the elapsed simulation time.

Run Controls

The following controls are provided for starting, pausing, and resetting a simulation model and for controlling its execution speed.

Table 4-5 Run Controls

Tool Bar Item	Description
Run/Pause Simulation	Starts, pauses, and resumes the simulation run.
Reset Simulation	Stops the model, clears any generated entities, and sets simulation time to zero.
Real Time Mode	If pressed, the simulation speed is held to a constant multiple of wall-clock time.
Speed Multiplier	The ratio of simulated time to wall-clock time that is used when the Real Time mode is set.
Pause at:	The time at which the software automatically pauses the simulation. Accepts numbers with time units (e.g. 500 h, 1 y) or date/time format (yyyy-mm-dd hh:mm:ss.s).

View Controls

Two buttons are provided to adjust the camera position:

Table 4-6 View Controls

Tool Bar Item	Description
2D	Moves the camera for the active View window to a bird's eye view directly above the xy-plane of the simulation.
3D	Moves the camera for the active View window to give the viewer a sense of depth in all dimensions.

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

Run Status

The right side of the Tool Bar consists of indicators to illustrate the progress and status of the simulation run:

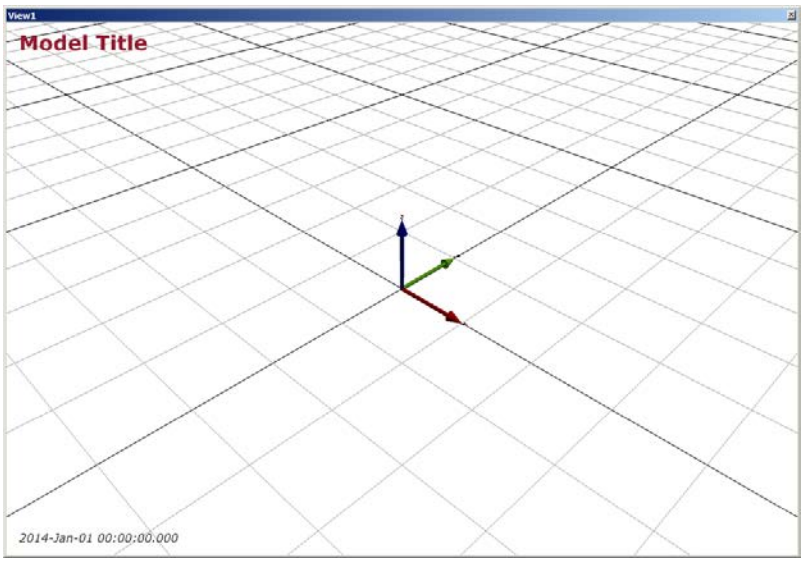
Table 4-7 Run Status

Tool Bar Item	Description
Simulation Time	The present simulation time, displayed in the Preferred Unit for time.
Progress Bar	Percentage of the simulation run that has been completed.
Time Remaining	Wall-clock time remaining until the simulation run is completed.
Speed Up	The ratio of elapsed simulated time to wall-clock time.
Position	Location of the mouse cursor in the active View window expressed in (x, y, z) coordinates.

4.2 View Windows

View windows 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. A screenshot of the default View window is shown in Figure 4-2.

Figure 4-2 Default View Window



4.2.1 Default Graphical 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, Title, and Clock.

The XY-Grid and XYZ-Axis objects are intended as visual aids for placing new objects, and as such they 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 two objects are a placeholder for the model title and a clock to show simulation time. These objects are overlay objects (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. The XY-Grid and the XYZ-Axis can be turned on or off through the Options item in the Menu Bar.

4.2.2 Camera Movement

The basic camera movements are zoom, pan, and orbit. Scrolling the mouse wheel zooms the camera in and out. Clicking and dragging the mouse cursor pans the camera 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 in Table 4-8, along with various other useful camera manipulations.

Table 4-8 View Camera Controls

Mouse/Keyboard Action	Effect
Left Click	Selects the point of interest. The point on the surface of the object under the cursor becomes the point of interest. If no object is under the cursor, 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	Pans the xy-plane. The camera is moved in the xy-plane from its present position but the cursor stays fixed on the same point in the View. If no object is under the cursor, then the point on the xy-plane is used. The point of interest is reset to the cursor position.
Shift + Left Drag	Pans the z-axis. The camera is moved along the z-axis from its present position so that the cursor stays fixed on the same point in the View. If no object is under the cursor, then the point on the xy-plane is used. The point of interest is reset to the cursor position.
Right Drag	Orbits the camera. 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.

4.2.3 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 hold 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 holding 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 Table 4-9.

Table 4-9 Moving and Resizing Objects

Mouse/Keyboard Action	Effect
Left Click	Selects the object. The object under the cursor is selected for input/output viewing and for re-positioning, re-sizing, or rotating. The selected object is indicated by a green rectangle bordering 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, while holding its z-coordinate constant, following the cursor.
Shift + Control + Left Drag	Moves the object parallel to the z-axis, while holding its x-coordinate and y-coordinate constant, following the cursor.
Control + Left Drag on a Handle	Resizes/rotates the object. The selected object is re-sized or rotated using the selected handle.

4.2.4 Moving and Reshaping Linear Objects

In addition to the standard controls described above, linear objects, such as Arrow objects, can be reshaped by adding and removing points and by moving individual points.

The actions for linear objects are described in Table 4-10:

Table 4-10 Moving and Reshaping Linear Objects

Mouse/Keyboard Action	Effect
Left Click on the Line	Selects the line. The line under the cursor 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 point 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 cursor.
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 cursor.
Control + Left Drag on a Handle	Moves location of the point in the xy-plane. The lines on either side of the point adjust to following the point.
Shift + Control + Left Drag on a Handle	Moves the location of the point along the z-axis.
Alt + Control + Left Click on the Line	Adds a new point and handle to the line.
Shift + Alt + Control + Left Click on a Handle	Deletes the point and re-connects the points on either side of the deleted point.

4.2.5 Context Menu

Right-clicking in the View window displays a menu listing all objects whose graphical representation lies under the mouse cursor. When a single object is selected, either by selecting an object from the menu if multiple objects are under the cursor or if there is only a single object under the cursor, a submenu will be displayed with the entries listed in Table 4-11:

Table 4-11 Context Menu Entries

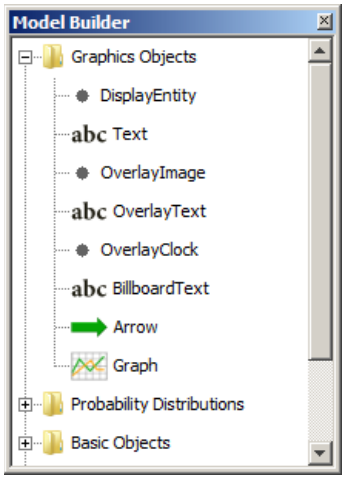
Menu Item	Description
Input Editor	Selects the object and opens its Input Editor window.
Output Viewer	Selects the object and opens its Output Viewer window.
Property Viewer	Selects the object and opens its Property Viewer window.
Duplicate	Creates a copy of the selected object in the current View window.
Delete	Deletes the selected object.
Change Graphics	Opens a dialog box to select a new DisplayModel graphical representation for the selected object.
Show Label	Creates an EntityLabel object that displays the present name of the selected object. The object's name can be changed by double-clicking on the EntityLabel and editing the displayed name.
Set RelativeEntity	Allows the position of the entity to be set relative to the position of a second entity. If the second entity is moved, the first entity moves with it.
Set Region	Allows the position of the entity to be set in a local coordinate system defined by the region. If the region is moved or rotated, the entities in the region move with it.
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.

Figure 4-3 shows the Model Builder with the Graphical Objects palette expanded.

Figure 4-3 Model Builder



The Display Models palette contains objects that do not have a graphical representation and cannot be dragged and dropped by the user. It appears in the Object Selector, but not in the Model Builder.

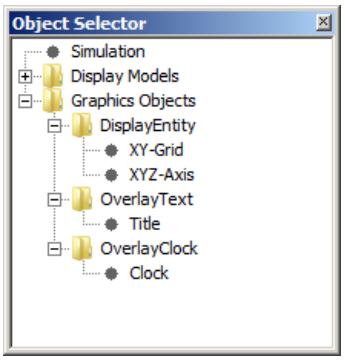
Applications built as add-ons to JaamSim will include additional palettes of objects that were created specifically for that application.

4.4 Object Selector

The Object Selector contains all 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.

Figure 4-4 shows the Object Selector with the Graphics Objects nodes expanded.

Figure 4-4 Object Selector



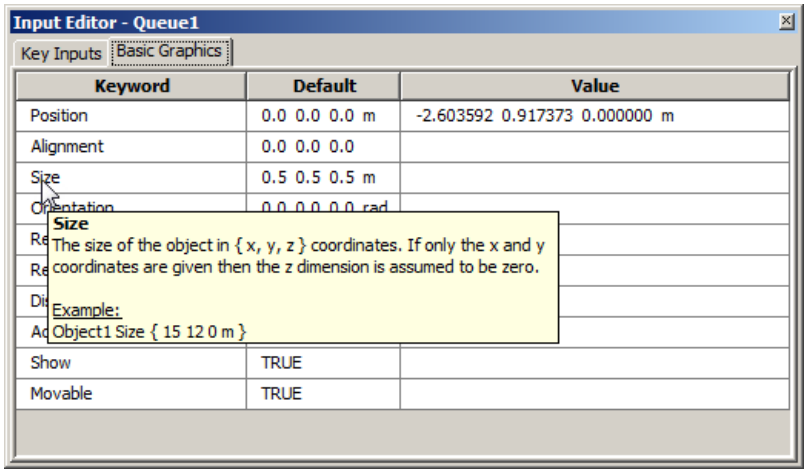
Objects that have been created using the Model Builder can be renamed or deleted using the Object Selector. Once an object has been selected, it can be renamed by triple-clicking on its entry in the Object Selector or by pressing F2, similar to the convention in Windows. It can be deleted by pressing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

4.5 Input Editor

The Input Editor allows the user to modify inputs for existing objects or assign inputs to new 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 cursor over a keyword will display a tooltip containing a brief description of the keyword and an example input. Figure 4-5 shows the Input Editor with the tooltip for Size.

Figure 4-5 Input Editor



The example input in the tooltip is given in the format used when editing the configuration file. The entry to place in the Input Editor is the text appearing between the braces, which in this case would be: `15 12 0 m`.

4.5.1 Entering and Editing Keyword Inputs

Keywords are modified by clicking the Value column and entering a new value with the appropriate units from the Units Palette. 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, which are outlined in Table 4-12 Input Data Types.

Table 4-12 Input Data Types

Data Type	Description
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.
Time Formats	Times can be entered as a number and unit (e.g. '5.2 s'), in a year/month/day format (e.g. '2015-02-28'), or in an hour/minute/second format (e.g. '01:35:20.5').
Expressions	Specified using object outputs and mathematical operators, e.g. '1 + 2* [Entity1].OutputB'.
Booleans	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	Specified by an object name. For example View1 indicates the View window View1, created automatically by JaamSim.

The use of time formats and Expressions are described in the 'Advanced Topics' section of this manual.

Braces are used to delineate distinct entries in a list. For example, a list of two points would be entered as `{ 0.0 1.0 0.0 m } { 1.0 1.0 0.0 m }`. When an input has only one set of inner braces, it can be entered without the inner braces. For instance, `{ 1 2 3 }` can be entered as `1 2 3`.

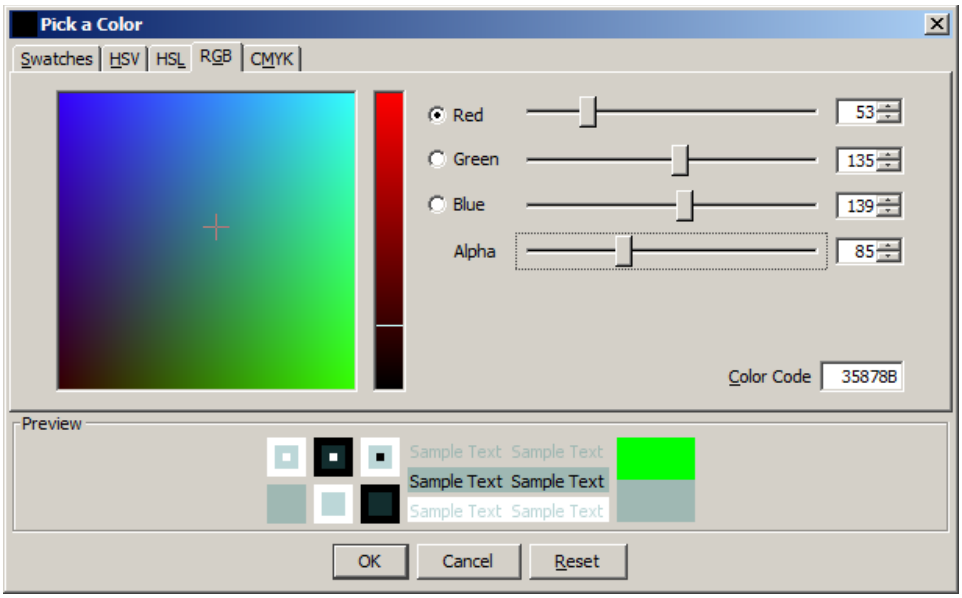
A drop-down menu is available for many types of inputs. For Boolean inputs, the drop-down menu offers the choice of TRUE or FALSE. For an object input, the drop-down menu lists all the objects of the appropriate type.

4.5.2 Colour Selection

For readability, it is often preferable to specify a colour input using a colour keyword instead of an RGB value, for instance ‘pink’ instead of ‘255 192 203’. A table of standard colour keywords is provided in the Named Colours section of this manual.

Additionally, the Input Editor provides a pop-up menu containing a graphical palette, shown in Figure 4-6, which allows for colour selection from a predefined table or through adjusting the RGB values via slider controls.

Figure 4-6 Colour Selection Pop-up Window



4.6 Output Viewer

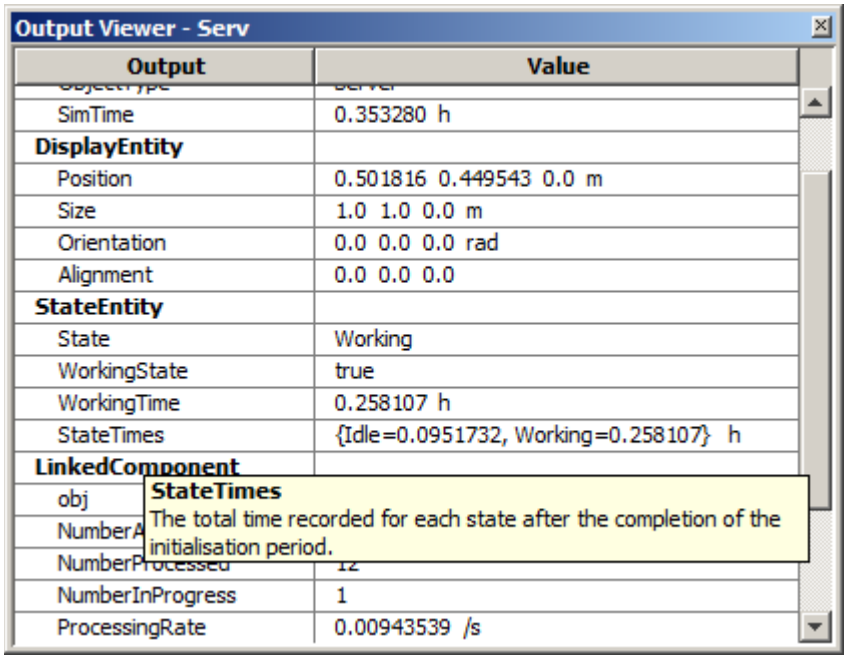
The Output Viewer displays the available outputs that for the selected object, as well as any user-defined Attributes. The values in the Output Viewer are updated continuously as the simulation progresses.

Hovering the cursor over an output name will display a tooltip pop-up containing a brief description of the output.

Numerical outputs are displayed in SI units unless otherwise specified using the DisplayedUnit keyword under the Simulation object.

Figure 4-7 shows the Output Viewer for a Queue object.

Figure 4-7 Output Viewer



5 Units

5.1 Unit Types

JaamSim performs all internal calculations in SI units (meters, kilograms, seconds, etc.). However, as it can sometimes be more convenient to specify quantities using other unit systems, JaamSim natively supports a number of commonly used units, shown in Table 5-1.

Table 5-1 Supported Unit Types and Units

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	/min, /h, /d, /w, /y, /ms, /us, /ns

Units are mandatory for most numerical inputs with the exception of pure numbers and ratios. Inputs that are pure numbers are indicated by the DimensionlessUnit type.

Advanced users can define new units for use in their models. See Section 13.8.

5.2 Changing Units for Model Outputs

Model outputs are normally shown in the appropriate SI unit in the Output Viewer, but it can be more convenient to view outputs in units other than the default units. The DisplayedUnit keyword under the Simulation object allows for the output units to be adjusted. For instance, to have TimeUnit outputs in hours instead of seconds and DistanceUnit outputs in kilometres instead of metres, the DisplayedUnit keyword should be set to `{ h km }`.

6 Simulation Object

The Simulation object is used to store inputs defining basic parameters of the model, such as run duration. The Simulation object is created automatically when a new model is started, and thus does not appear in the Model Builder.

Table 6-1 Simulation Key Inputs

Keyword	Description
RunDuration	Duration of the simulation run in which statistics will be recorded.
InitializationDuration	Duration of the initialization period from which run statistics are discarded before the simulation is run for the time specified in RunDuration, allowing the model to reach steady-state before statistics are recorded. The total length of the simulation run is the sum of InitializationDuration and RunDuration.
PauseCondition	An optional expression that pauses the run to pause when TRUE is returned.
ExitAtPauseCondition	If TRUE, the simulation run will be terminated when the PauseCondition expression returns TRUE.
ExitAtStop	If TRUE, JaamSim will close all windows and exit when the simulation run is finished.
GlobalSubstreamSeed	Global seed that sets the substream for each probability distribution. Must be an integer >= 0. GlobalSubstreamSeed works together with each Probability Distribution's RandomSeed keyword to determine its random sequence. It allows the user to change all the random sequences in a model with a single input.
PrintReport	If TRUE, a summary report will be written to an output file named <run name>.rep.
ReportDirectory	The directory where output files will be written. The default location is the directory of the input configuration file.
UnitTypeList	The unit types for the selected outputs for the simulation run. Use DimensionlessUnit for a text output.
RunOutputList	One or more selected outputs to be printed at the end of each simulation run. Each output is specified by an expression. In script mode (-s tag), the selected outputs are printed to the command line (standard out). Otherwise, they are printed to the file <configuration file name>.dat.
TickLength	The smallest time increment for JaamSim's internal integer-based time keeping. The default value of 1 microsecond will support simulation runs of more than 100 years.

7 Graphics Objects

Graphical Objects are used to create 3D objects, pictures, text, graphs, arrows, and other graphical components needed to visualise and monitor a simulation. The following objects are included in the Graphics Objects palette.

Table 7-1 Graphics Objects Palette

Object	Description
Region	Local coordinate system.
DisplayEntity	Graphical object displaying either a 3D shape or a 2D picture.
Text	Text that appears in the 3D model universe.
EntityLabel	Text that labels another object.
InputBox	Text that provides an input value for the model.
OverlayImage	Picture that appears in a fixed position in the display window.
OverlayText	Text that appears in a fixed position in the dispay window.
OverylayClock	Time and date display that appears in a fixed position in the display window.
BillboardText	Text that follows a 3D position but is always upright on the screen.
Arrow	Line that terminates in an arrow head.
Graph	Chart that shows the values of model outputs as they change in time.
View	Display window showing view of the 3D model universe.
VideoRecorder	Object that makes a video recording of the model.

7.1 Region

The Region object is used to define a local coordinate system. When a Region is specified for an object, its inputs for position and orientation are relative to the position and orientation of its Region. The global coordinate system is the default for objects that do not reference a specific Region.

The Position and Orientation keywords are used to define the origin for the coordinate system and the angles for its coordinate axes. Once these inputs have been set, it is advised to set the inputs for Show and Movable to FALSE so the Region object is no longer visible and cannot be moved accidentally.

Table 7-2 Region Key Inputs

Keyword	Description
Position	The origin of the local coordinate system.
Orientation	The angles of the local coordinate axes relative to the global coordinate axes.
Show	If TRUE, then the Region object is displayed as three unit vectors in the directions of the local coordinate axes. If FALSE, the Region object is hidden from view.
Movable	If TRUE, then the Region object can be positioned by dragging with the mouse. If FALSE, the Region cannot be moved with the mouse.

7.2 DisplayEntity

The DisplayEntity is the basic 3D graphical object in JaamSim. All JaamSim objects that have graphics are subclasses of DisplayEntity.

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 an ImageModel. The inputs for the 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 (a 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.

The easiest way to create a new DisplayEntity for a 3D object or an image is to use the Import function provided in the Control Panel under File. This method creates a new DisplayEntity that is connected to a new DisplayModel with the imported 3D object or image.

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.

Table 7-3 DisplayEntity Key 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.
Region	The name of the local coordinate system used to specify the position of the object. If blank, the global coordinate system is used.
RelativeEntity	An object with respect to which the Position keyword is referenced. If that object is moved, any object connected to it 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.
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.

7.3 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 output 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 keyword and specifying the value to be displayed with the DataSource keyword. If the variable text is a number with units, the value can be converted from SI to a specified unit through the Unit keyword.

The easiest way to modify the contents of a Text object is to edit it in directly in the view window. Double-click on the object to place it in edit mode. In this mode, text can be entered, deleted, inserted, and highlighted using the same conventions as a typical text editor. Text can be copied from and pasted to the clipboard using Cntrl-C and Cntrl-V keys. Press the Return key or click on another object to save the changes and return the Text object to its normal mode.

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:

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

Table 7-4 Text Key Inputs

Keyword	Description
TextHeight	The size of the font as displayed in the View window.
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 DataSource keyword, include the appropriate Java format in the text. For instance, `%s` will display a text output and `%.6f` will display a number with six decimals of accuracy.
UnitType	The unit type for the numerical value to be displayed as variable text. Set to DimensionlessUnit if the variable text is non-numeric such as the state of a Server.
Unit	The unit in which to express an expression that returns a numerical value.
DataSource	An expression that returns the variable text to be displayed. The expression can return a number that will be formatted as text or it can return text directly, such as the state of a Server.
FailText	The text to display if an error occurs when the text is formatted or when an expression entered to DataSource is evaluated.

7.4 EntityLabel

The EntityLabel object is a special version of a Text object that is used to display the name of a selected object. The only way to create an EntityLabel is to right-click on the object to be labelled and select “Show Label”. It cannot be dragged and dropped from the Model Builder.

An EntityLabel moves automatically with the object and is destroyed when the object is destroyed. An object can have only one EntityLabel at a time.

An object name can be changed by editing its EntityLabel in the view window. Double-click on the EntityLabel to enter edit mode and revise the name as desired. Press the Return key or click on another object to accept the new name.

An EntityLabel has no inputs except for the TextHeight keyword inherited from the Text object.

The appearance of all the EntityLabels in a model can be changed by editing the inputs for the EntityLabelModel (under Display Models > TextModel > EntityLabelModel in the Object Selector). The standard colour, font and style (bold and/or italics) for all EntityLabels can be selected in this way. See Section 13.6.3.

7.5 InputBox

1.0

The InputBox object is a special version of a Text object that is used to allow a user to enter a model input directly in the view window, without using the Input Editor. The input value can be modified by double-clicking on the displayed value to enter edit mode. After editing is complete, press the Return key or click on another object to accept the new value.

The function of an InputBox object is similar to that of an InputValue object (see Section 9.1) except that it can be used to assign an input for any type of keyword, both numeric and non-numeric, while InputValue is restricted to numerical keywords that can accept an expression or an object that returns a number.

Table 7-5 InputBox Key Inputs

Keyword	Description
TextHeight	The size of the font as displayed in the View window.
TargetInput	The names of the Entity and Keyword that will receive the input. For example, to set the ServiceTime keyword for Server1, enter: Server1 ServiceTime.

7.6 Overlay Objects (OverlayImage, OverlayText, OverlayClock)

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 and Clock 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 in Table 7-6.

Table 7-6 Overlay Object and Usage Summary

Overlay Object	Parent Object	Usage
OverlayImage	DisplayEntity	Static image (Logos, other graphics)
OverlayText	Text	Static or 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 key inputs listed in Table 7-7:

Table 7-7 Overlay 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.

7.7 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 space and in this way differs from the OverlayText object. Both static and dynamic text can be displayed by a BillboardText object using the same keywords as Text objects.

The keywords for BillboardText are identical to those for Text objects 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 will be corrected in a future version of the software.

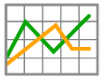
7.8 Arrow

An Arrow object consists of a line, which can be composed of multiple segments, and an arrowhead.

Table 7-8 Arrow 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.
Width	The width of the Arrow line segments in pixels.
ArrowSize	A set of xyz numbers defining the size of the arrowhead in the respective directions at the end of the segments.
Color	The colour of the Arrow, defined using a colour keyword or RGB values.

7.9 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. Figure 7-1 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.

Figure 7-1 Sample Graph with Formatting Keyword Defined Graphically

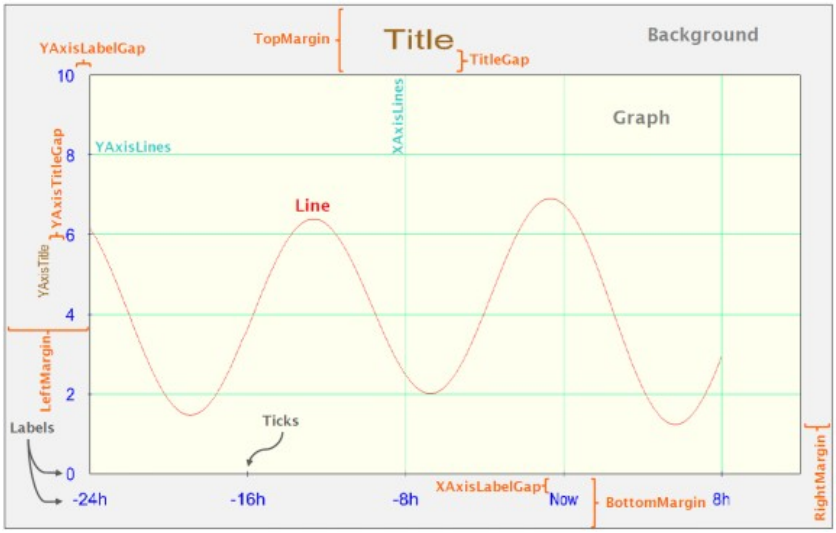


Table 7-9 Graph Key Inputs

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.
UnitType	The type of units for each line specified by the DataSource input. Must be specified before the DataSource input and the Y-Axis inputs.
DataSource	One or more sources of data to be graphed against the primary y-axis. Specified as a series of Expressions, each enclosed by braces.
LineColours	A list of colours for the data series to be displayed. For multiple colours, each colour must be enclosed in braces as each colour can be itself defined as a list of RGB values.
LineWidths	A list of widths, in pixels, for the data series to be displayed.
SecondaryUnitType	The type of units for each line specified by the SecondaryDataSource input. Must be specified before the SecondaryDataSource input and the Secondary Y-Axis inputs.
SecondaryDataSource	One or more sources of data to be graphed against the secondary y-axis. Specified as a series of Expressions, each enclosed by braces.
SecondaryLineColours	A list of colours for the data series to be displayed. For multiple colours, each colour must be enclosed in braces as each colour can be itself defined as a list of RGB values.
SecondaryLineWidths	A list of widths, in pixels, for the data series to be displayed.

Table 7-10 Graph X-Axis Inputs

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 XAxisStart and XAxisEnd where vertical gridlines are inserted.
XLinesColor	Colour of the vertical gridlines, or a list corresponding to the colour of each gridline defined in XLines, defined using a colour name or RGB values.

Table 7-11 Graph Y-Axis Inputs

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 primary y-axis, in units of the DataSource.
YAxisEnd	The maximum value for the primary y-axis.
YAxisInterval	The interval between primary y-axis labels.
YAxisLabelFormat	The Java format to be used for the tick mark values on the primary y-axis.
YLines	A list of values at which to insert horizontal gridlines.
YLinesColor	Colour of the horizontal gridlines, defined using a colour name or RGB values.

Table 7-12 Graph Secondary Y-Axis Inputs

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.

7.10 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.

Table 7-13 View Graphics 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 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.

7.11 VideoRecorder

The VideoRecorder object is used to create a short video of a model in operation.

The AVI file created by the VideoRecorder is encoded using the VP8 codec, which is NOT supported by Windows Media Player. Furthermore, the present encoding algorithm is quite inefficient making the file size much larger than necessary. Both problems can be solved by recoding the video using free open-source software such as HandBrake (www.handbrake.fr).

The original AVI file in VP8 codec can be viewed with the VLC video player (www.videolan.org).

Table 7-14 VideoRecorder Key Inputs

Keyword	Description
CaptureStartTime	Simulation time at which to capture the first frame.
CaptureInterval	Simulation time between captured frames.
CaptureFrames	The total number of frames to capture for the video. The recorded video assumes 30 frames per second. Therefore, if a 2 minute video is required, the number of frames should be set to 120 x 30 = 3600.
CaptureArea	The size of the video/image, expressed as the number of horizontal and vertical pixels. The top left-hand corner of the captured frames will be the same as the top left-hand corner of the image on the monitor. If the specified image size is larger than the monitor resolution, then the image will be extended beyond the bottom and/or right sides of the monitor.
CaptureViews	The list of View windows to be captured.
VideoBackgroundColor	"The background color for the captured frames. Only the 3D view portion of the specified windows will be captured. The remainder of the frame, such as the Control Panel or any gaps between the view windows, will be replaced by the background color.
VideoName	A label to append to the run name when the AVI file is saved. The saved file will be named <run name>_<VideoName>.avi.
SaveImages	If TRUE, an individual PNG file will be saved for each frame.
SaveVideo	If TRUE, an AVI file containing the video will be saved.

8 Probability Distributions

The Probability Distributions Palette provides a standard selection of theoretical Probability Distributions as well as user-defined Probability Distributions.

8.1 Theoretical Probability Distributions

The following standard Probability Distributions are available in JaamSim:

Table 8-1 Supported Probability Distributions

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.
NonStatExponentialDist	Generates samples from a time-varying negative exponential probability distribution. The correct 'non-stationary Poisson process' algorithm is used.
ErlangDistribution	Generates samples from an Erlang probability distribution.
GammaDistribution	Generates samples from a Gamma probability distribution.
BetaDistribution	Generates samples from a Beta probability distribution.
WeibullDistribution	Generates samples from a Weibull probability distribution.
LogNormalDistribution	Generates samples from a Log-Normal probability distribution.
LogLogisticDistribution	Generates samples from a Log-Logistic probability distribution.

The Probability 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)).

8.2 User-Defined Probability Distributions

A custom Probability Distribution can be specified point-by-point, and can be either a continuously varying value or one with a discrete set of values.

Table 8-2 User-Defined Probability Distributions

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

8.3 Keywords and Outputs for Probability Distributions

Each of the Probability Distributions uses the following standard keywords in addition to the ones that are specific to an individual Probability Distribution.

Table 8-3 Probability Distribution Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the distribution, e.g. TimeUnit. To keep the units consistent for other inputs, this input must be set first.
RandomSeed	Seed for the random number generator. Must be an integer >= 0. The RandomSeed keyword works together with the GlobalSubstreamSeed under the Simulation object to determine the random sequence. The GlobalSubsteamSeed keyword allows the user to change all the random sequences in a model with a single input.
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 calculated from the simulation and are standard to all Probability Distributions.

Table 8-4 Probability Distribution Outputs

Output Name	Description
Value	The last value sampled from the distribution. When used in an expression, this output returns a new sample every time the expression is evaluated.
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 Probability Distribution.
SampleMean	The average of the samples returned by the Probability Distribution.
SampleStandardDeviation	The standard deviation of the samples returned by the Probability Distribution.
SampleMin	The minimum of the samples returned by the Probability Distribution.
SampleMax	The maximum of the samples returned by the Probability Distribution.

8.4 Changing the Random Seed

Each Probability Distribution has the RandomSeed keyword, which generates a different pseudo-random sequence of values for each Probability Distribution. Often, a simulation model is executed multiple times, called 'replications', using different random seeds to determine the statistical accuracy of the output parameters.

One way to achieve multiple replications is to change the RandomSeed for each Probability Distribution. However, this can be impractical when large numbers of distributions are used in a model. The GlobalSubstreamSeed keyword for the Simulation object simplifies this process. Changing the input for this keyword causes all Probability Distributions in the model to generate a different random sequence. In effect, the seed for each Probability Distribution is the combination of the inputs to the RandomSeed and GlobalSubstreamSeed keywords.

8.5 Multiple Instances of the Same Distribution

When multiple objects behave according to a Probability Distribution with the same characteristics, it is advisable to have a unique instance of the Probability Distribution for each object. This avoids any cross-correlation effects arising from the order in which a single distribution may be sampled.

9 Basic Objects

The Basic Objects Palette contains a number of objects that can be used in many types of simulation models. The following objects are provided in the Basic Objects palette.

Table 9-1 Basic Objects Palette

Object	Description
InputValue	Provides a way to enter a numerical value directly into the simulation model screen.
TimeSeries	Provides a floating point number that changes in simulated time following a series of input values.
TimeSeriesThreshold	Specifies a range of values from a TimeSeries for which an activity is permitted.
ExpressionThreshold	Specifies a logical condition for which an activity is permitted.
ExpressionLogger	Records the values for one or more expressions to a log file at regular intervals.
EntitlementSelector	Selects an index on the basis of entitlement from a given set of proportions.
ExpressionEntity	Calculates the value for a specified expression.
DowntimeEntity	Provides Breakdown and Maintenance controls.
ValueSequence	Generates a repeating sequence of numerical values.
EventSchedule	Generates a sequence inter-arrival times from a list of event times.

9.1 InputValue

The InputValue object is a special version of a Text object that is used to allow a user to enter a numerical input directly in the view window, without using the Input Editor. The input value can be modified by double-click on the displayed value to enter edit mode. After editing is complete, press the Return key or click on another object to accept the new value.

The present value for an InputValue object can be accessed by the inputs to the keywords for other objects either through an expression, e.g. [InputValue1].Value, or by simply entering the name of the InputValue.

The function of an InputValue object is similar to that of an InputBox object (see Section 7.59.1) except that it is restricted to numerical keywords that can accept an expression or an object that returns a number. In contrast, the InputBox object can be used with any input, both numeric and non-numeric.

Table 9-2 InputValue Key Inputs

Keyword	Description
TextHeight	The size of the font as displayed in the View window.
UnitType	One of the pre-defined unit types within JaamSim. Must be set prior to the Value input.
Value	The present value for the object. Normally, this input would be assigned by editing the value displayed in a view window.

The InputValue object has only one output.

Table 9-3 InputValue Outputs

Output Name	Description
Value	The present value for the object.

9.2 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, a TimeSeries, or an Expression, which makes TimeSeries a powerful and flexible component for building simulation models.

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 input. Once specified, it cannot be changed.

Table 9-4 TimeSeries Key Inputs

Keyword	Description
UnitType	One of the pre-defined unit types within JaamSim. 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, e.g. `{ 0 s 100 m } { 1 s 150 m } { 2 s 50 m }`. The input for UnitType must be set before entering the input for Value.
CycleTime	When the TimeSeries will repeat from the start. By default, the TimeSeries object does not cycle.

Timestamps can be entered in the normal fashion, such as 0 s, 5 h, etc., or in various year/month/day formats. The latter formats are described in the 'Advanced Topics' section.

The first timestamp must be zero seconds (i.e. 0 s) or January 1, 00:00 of an arbitrary year. If a non-zero year is used, for example 2010-01-01, then the TimeSeries considers this date to be time zero of the simulation and all other timestamps are offset accordingly.

Table 9-5 TimeSeries Outputs

Output Name	Description
PresentValue	The value for the TimeSeries at the present simulation time.

The value for the TimeSeries is constant between one timestamp and the next. If the simulation time is greater than the CycleTime, the TimeSeries will repeat from the beginning of the Value list. For example:

```
Define  TimeSeries  { TimeSeries1 }
TimeSeries1   UnitType  {  SpeedUnit  }
TimeSeries1   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  }  }
TimeSeries1   CycleTime  {  14  d }
```

TimeSeries1 would take on a new value on 2014-01-03 (0.76 km/h) and again on 2014-01-11 (0.24 km/h). On 2014-01-15, the TimeSeries would cycle and take on the original value (0 km/h). This 14-day cycle would repeat until the end of the simulation run.

In the above example, the following input for the Value keyword would have been equivalent:

```
TimeSeries1  Value  { {  0 d  0.00 km/h }
                     {  2 d  0.76 km/h }
                     { 10 d  0.24 km/h } }
```

9.3 TimeSeriesThreshold

The TimeSeriesThreshold object varies its state between open and closed depending on the present value for a TimeSeries object.

The TimeSeries to be monitored is specified by the TimeSeries keyword. Trigger levels are determined by the MinOpenLimit and MaxOpenLimit keywords. For the TimeSeriesThreshold to be open, the PresentValue 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 the 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.

Table 9-6 TimeSeriesThreshold Key Inputs

Keyword	Description
UnitType	The unit type for the threshold, e.g. DistanceUnit, TimeUnit, MassUnit.
TimeSeries	The TimeSeries object to be monitored.
MaxOpenLimit	The limit over which the TimeSeriesThreshold is closed.
MinOpenLimit	The limit under which the TimeSeriesThreshold is closed.
LookAhead	The amount of time that the TimeSeriesThreshold 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.

Table 9-7 TimeSeriesThreshold Graphics Inputs

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

Table 9-8 TimeSeriesThreshold Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 9-9.
Open	TRUE if the TimeSeriesThreshold is Open, FALSE if the TimeSeriesThreshold is closed.
OpenFraction	The fraction of total time during which the TimeSeriesThreshold is open.
ClosedFraction	The fraction of total time during which the TimeSeriesThreshold is closed.

Table 9-9 TimeSeriesThreshold State Definitions

State Name	Description
Open	The TimeSeriesThreshold is open.
Closed	The TimeSeriesThreshold is closed.

9.4 ExpressionThreshold

The ExpressionThreshold object varies its state between open and closed depending on the value returned by an Expression.

The Expression to be evaluated is specified by the OpenCondition keyword. It is re-evaluated with every time advance of the model, and on demand when the Open output is used by another Expression.

Expressions that change between TRUE and FALSE based on simulated time output, '`this.SimTime`', must be avoided because the change in state will not be detected until an event occurs. The correct way to obtain this type of behaviour is to use a TimeSeries and a TimeSeriesThreshold. The PendingOpen and PendingClosed states are used to indicate the situation where an ExpressionThreshold is being used inappropriately. These states can only be detected when a View window is open.

Table 9-10 ExpressionThreshold Key Inputs

Keyword	Description
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.
CloseCondition	The logical condition for the ExpressionThreshold to close. If not specified, the CloseCondition defaults to the opposite of the OpenCondition. If the OpenCondition and CloseCondition are both TRUE, then the ExpressionThreshold is set to open.

Table 9-11 ExpressionThreshold Graphics Inputs

Keyword	Description
OpenColour	The colour to display when the ExpressionThreshold is open.
ClosedColour	The colour to display when the ExpressionThreshold is closed.
ShowWhenOpen	If TRUE, the ExpressionThreshold is displayed when it is open. If FALSE, it is hidden when open.
ShowWhenClosed	If TRUE, the ExpressionThreshold is displayed when it is closed. If FALSE, it is hidden when closed.
PendingOpenColour	The colour to display when the when the ExpressionThreshold condition is true, but the ExpressionThreshold object has not be updated yet and its state is still closed.
PendingClosedColour	The colour to display when the ExpressionThreshold condition is false, but the ExpressionThreshold object has not been updated yet and its state is still open.
ShowPendingStates	If TRUE, the states PendingOpen and PendingClosed will be displayed when appropriate.

Table 9-12 ExpressionThreshold Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 9-13.
OpenFraction	The fraction of total time during which the ExpressionThreshold condition is true.
ClosedFraction	The fraction of total time during which the threshold condition is false.
Open	TRUE if the ExpressionThreshold condition is true, FALSE if the ExpressionThreshold condition is false.

Table 9-13 ExpressionThreshold State Definitions

State Name	Description
Open	The ExpressionThreshold condition is true.
Closed	The ExpressionThreshold condition is false.
PendingOpen	The threshold condition is true, but the ExpressionThreshold object has not been updated yet and its state is still closed.
PendingClosed	The threshold condition is false, but the ExpressionThreshold object has not been updated yet and its state is still open.

9.5 ExpressionLogger

The ExpressionLogger object provides the ability to record the value for one or more expressions whenever the object is triggered during the simulation run. An ExpressionLogger can be triggered by one or more of types of events:

- At regular time intervals,
- Whenever an object changes state, and
- Whenever the value of an expression changes.

Logging at regular intervals is handled by the keywords in the Key Inputs tab.

Table 9-14 ExpressionLogger Key Inputs

Keyword	Description
Interval	A fixed interval at which entries will be written to the log file. This input is optional if state tracing or value tracing is specified.
UnitTypeList	The unit types for the quantities being logged. Use DimensionlessUnit for text entries.
DataSource	One or more sources of data to be logged. Each source is specified by an Expression. Also acceptable are: a constant value, a Probability Distribution, TimeSeries, or a Calculation Object.
IncludeInitialization	If TRUE, entries are logged during the initialization period.
StartTime	The simulation time for the first log entry.
EndTime	The latest simulation time at which to make an entry in the log.

State and value tracing is handled by the keywords in the Tracing Inputs tab.

Table 9-15 ExpressionLogger Tracing Inputs

Keyword	Description
StateTraceList	A list of entities whose states will be traced. An entry in the log file is made every time one of the entities changes state. Each entity's state is written automatically to the log file - it is not necessary to add an expression to the DataSource keyword's input.
ValueUnitTypeList	The unit types for the values being traced.
ValueTraceList	One or more sources of data whose values will be traced. An entry in the log file is made every time one of the data sources changes value. Each data source's value is written automatically to the log file - it is not necessary to add an expression to the DataSource keyword's input. Each source is specified by an Expression. Also acceptable are: a constant value, a Probability Distribution, TimeSeries, or a Calculation Object.

9.6 EntitlementSelector

The EntitlementSelector object is similar to the DiscreteDistribution object in that it returns an index in some range, except that instead of returning a random selection based on probabilities it makes its choice using an entitlement algorithm based on proportions. The entitlement algorithm chooses the index that minimizes the difference between the actual number returned for each index and the expected number based on the proportions.

The ProportionList keyword is used to specify the relative proportions for the N choices.

Table 9-16 EntitlementSelector Key Inputs

Keyword	Description
ProportionList	A list of N numbers equal to the relative proportion for each of the N indices. Must sum to 1.0.

Table 9-17 EntitlementSelector Outputs

Output Name	Description
SampleCount	The number of times each of the N indices has been selected.
SampleDifference	The difference between the number of times each index has been selected and the expected number calculated from the proportions.
Value	The last index that was selected.
NumberOfSamples	The total number of times that an index has been selected.

9.7 ExpressionEntity

The ExpressionEntity is used to evaluate an expression. It is useful when a complicated expression is used in several different places. Instead of entering the expression several times, it is better to use an ExpressionEntity to evaluate the expression and then use its output “Value” to pass the result to other expressions.

Table 9-18 ExpressionEntity Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the expression.
Expression	The expression to be evaluated.

The ExpressionEntity object has only one output.

Table 9-19 ExpressionEntity Outputs

Output Name	Description
Value	The present value for the expression.

9.8 DowntimeEntity

The DowntimeEntity object is used to generate planned and unplanned maintenance events for various types of objects. The DowntimeEntity generates the downtime events and their durations, but the objects that use one or more DowntimeEntities must provide their own logic for halting operations.

Table 9-20 DowntimeEntity Key Inputs

Keyword	Description
FirstDowntime	The simulation time for the first downtime event. The value may be a constant or a probability distribution. If a value is not specified, the Interval keyword is sampled to determine the simulation time of the first downtime event.
IntervalWorkingEntity	The object for which to track working time accumulated in order to schedule a downtime event in objects that have this DowntimeEntity defined in their 'DowntimeEntities' keyword. If this keyword is not set, calendar time will be used to determine the IAT.
DurationWorkingEntity	The object for which to track working time accumulated in order to complete a downtime event in objects that have this DowntimeEntity defined in their 'DowntimeEntities' keyword. If this keyword is not set, calendar time will be used to determine when the downtime will be over.
Interval	The time between the start of the last downtime event and the start of the next downtime event.
Duration	The duration of the next or present downtime event.
Type	Three types of downtime events can be modelled: • IMMEDIATE (interrupt the present task and start maintenance without delay) • FORCED (complete the present task before starting maintenance) • OPPORTUNISTIC (complete the present task and any waiting tasks before starting maintenance)

The DowntimeEntity object has only two outputs.

Table 9-21 DowntimeEntity Outputs

Output Name	Description
StartTime	The simulation time for the start of the next downtime event.
EndTime	The simulation time for the end of the next or present downtime event.

9.9 ValueSequence

The ValueSequence object is used to generate a specified sequence of numbers. It can be used instead of a Probability Distribution when a model is to be validated against recorded data. It can also be used to specify an interval or duration for a planned maintenance activity.

The next value in the sequence is returned every time the ValueSequence is referenced. The values are recycled after the last value in the list is returned.

Table 9-22 ValueSequence Key Inputs

Keyword	Description
UnitType	The unit type for the generated values.
Expression	The sequence of numbers to be generated. Note that the appropriate unit for the numbers must be entered in the last position, e.g. 1 2 3 m

The ValueSequence object has only two outputs.

Table 9-23 ValueSequence Outputs

Output Name	Description
Index	The position of the last value returned in the list.
Value	The last value returned by the ValueSequence. When used in an expression, this output returns a new value every time the expression is evaluated.

9.10 EventSchedule

The EventSchedule object is similar to the ValueSequence object in that a specified sequence of values is returned. The difference is that an EventSchedule returns the inter-arrival times for a specified sequence of event times. When applied to the InterArrivalTime keyword for an EntityGenerator, it generates a specified sequence of entities at the simulation times provided to the EventSchedule.

Table 9-24 EventSchedule Key Inputs

Keyword	Description
TimeList	A sequence of monotonically-increasing simulation times at which to generate events. If entered in date format, an input of '0000-01-01 00:00:00' corresponds to zero simulation time.
CycleTime	Defines when the event times will repeat from the start.

The EventSchedule object has only one output.

Table 9-25 EventSchedule Outputs

Output Name	Description
Index	The position of the event time in the list for the last inter-arrival time that was returned.
Value	The last inter-arrival time returned from the sequence. When used in an expression, this output returns a new value every time the expression is evaluated.

10 Process Flow Objects

The Process Flow Palette contains all the objects needed to create process flow type models. These models are characterized by a passive entity that is passed from one object to another following a process flow diagram. These types of models are often used to simulate a manufacturing process where the entities represent parts that are moved between processing stations. The following objects are provided in the Process Flow palette.

Table 10-1 Process Flow Palette

Object	Description
SimEntity	The basic entity for use in a process flow type model.
EntityGenerator	Creates copies of a prototype entity at specified intervals.
EntitySink	Destroys any entity it receives.
Server	Processes a received entity over a specified duration.
Queue	Stores received entities until they are needed.
EntityConveyor	Transports a received entity along a specified path at a fixed speed.
EntityDelay	Delays a received entity by a specified duration.
Resource	Set of identical resource units that can be seized and released by various processes.
Seize	Seizes one or more units of a Resource.
Release	Releases one or more units of a Resource.
Assign	Assigns new values to attributes.
Branch	Directs a received entity to a selected destination.
Duplicate	Sends copies of the received entity to a set of destinations.
Combine	Matches entities from multiple queues. The entity from the first queue is passed on while the other entities are destroyed.
SetGraphics	Changes the appearance of the received entity.
EntityGate	Blocks received entities from progressing further until the EntityGate is opened by one or more Thresholds.
EntitySignal	Opens or closes a specified SignalThreshold when an entity is received.
SignalThreshold	Threshold that is opened and closed directly by an EntitySignal object.
Assemble	Combines sub-components to create an assembled part.
EntityContainer	An entity that can hold one or more other entities.
Pack	Inserts entities in a new EntityContainer.
Unpack	Removes all the entities from an EntityContainer which is subsequently destroyed.
AddTo	Add entities to an existing EntityContainer.
RemoverFrom	Removes some or all of the entities from an EntityContainer.
EntityLogger	Records the outputs and state data for a generated entity in an output log.
Statistics	Collects statistics from the received entities.

10.1 SimEntity

The SimEntity object is the basic entity that is passed through a process flow type model.

The main feature of SimEntity is its State output indicates the present activity of the SimEntity. Its state can be set whenever it enters a new object using the StateAssignment keyword for that object. Note that state names cannot include any blanks or special characters.

A SimEntity tracks the time it spends in each state during the simulation run. This information can be logged for each SimEntity using the EntityLogger object.

Table 10-2 SimEntity Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.

Table 10-3 SimEntity Outputs

Output Name	Description
State	The present user-defined activity being performed by the SimEntity.
WorkingState	Set to true if the present state is one of the pre-defined working states.
WorkingTime	Total time spent in one or more of the working states.
StateTimes	Total time spent in each of the state.

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 an Expression.

Table 10-4 EntityGenerator Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which a generated entity is passed.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity.
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. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
InterArrivalTime	The elapsed time between one generated entity and the next. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
EntitiesPerArrival	The number of entities to be generated for each arrival time. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
PrototypeEntity	The entity to be copied by the EntityGenerator.
MaxNumber	The maximum number of entities to be generated.

Table 10-5 EntityGenerator Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-6.
obj	Not used.
NumberAdded	Not used.
NumberProcessed	The total number of entities that have been generated.
ProcessingRate	The rate at which entities are generated (entities per second).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
NumberGenerated	The total number of entities generated by this EntityGenerator.

Table 10-6 EntityGenerator State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.3 EntitySink

The EntitySink object destroys an incoming entity.

Table 10-7 EntitySink Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.

Table 10-8 EntitySink Outputs

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

10.4 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.

Entities can also be sent directly to the Queue specified by the WaitQueue keyword. Whenever an entity is added to a Queue, all the Servers that specified this Queue as its WaitQueue will be notified. The first Server that is available will then remove the entity for processing.

The rate at which entities are processed is determined by the ServiceTime keyword. This input has units of time and accepts a constant value, a TimeSeries, a Probability Distribution, or an Expression.

The Server can be stopped under various circumstances using 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 Server to operate. However, if a threshold closes while the Server is processing an entity, it will complete its work on that entity before ceasing further operation.

Table 10-9 Server Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the entity being processed relative to the position of the Server object.
WaitQueue	A Queue object in which to hold entities that are waiting to be processed.
Match	An integer value that, if set, restricts the Server to only the entities in the queue that have the same match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
OperatingThresholdList	A list of threshold objects that must all be open for operation to proceed. Operation is stopped if any of the thresholds are closed.
ServiceTime	The time required to process the incoming entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-10 Server Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-11.
WorkingState	Set to true if the present state is one of the pre-defined working states.
WorkingTime	Total time spent in one or more of the working states.
StateTimes	Total time spent in each of the state.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the match variable for the last entity that was processed.

Table 10-11 Server State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.5 Queue

A Queue object defines a location for simulation entities to wait for processing by other entities.

Unlike many other objects in this palette, an entity received by a Queue is not passed automatically to the next object. It must wait in the Queue until it is removed by some other object. Queues are used in this way by the Server, Seize, EntityGate, Assemble, Combine, Pack, Unpack, AddTo, and RemoveFrom objects. Whenever an entity is added to a Queue, all the objects that use this Queue are notified that a new entity is available. The first available object will remove the entity from the Queue and start processing it.

Queued entities can be sequenced by an optional priority value specified by the Priority keyword. In most cases, the Priority is specified by an Expression that is evaluated when the entity first arrives at the Queue. Priority is integer valued, and decimal values will be truncated, which means, for example, that priority values of 3.2 and 3.6 are identical (i.e. truncated to 3).

Entities with the same priority values can be sequenced in either first-in-first-out (FIFO) order, the default, or in last-in-first-out order (LIFO).

Lastly, a queued entity can be provided with an optional identification number using the Match keyword. Objects that use Queues, such as a Server, can request its Queue to provide the first entity with a specified value for the Match keyword. As with the Priority keyword, in most cases the Match value is specified by an Expression that is evaluated when the entity first arrives at the Queue. The Match variable is integer valued, and if a decimal value is provided, it will be truncated.

Table 10-12 Queue Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
Priority	The priority for positioning a received entity in the Queue. Priority is integer valued and a lower numerical value indicates a higher priority. Priority is normally specified by an Expression, however a dimensionless number or an object that returns a dimensionless number such as a TimeSeries or a Probability Distribution can also be used.
Match	An integer value that can be used to label the queued entities. Match is normally specified by an Expression, however a dimensionless number or an object that returns a dimensionless number such as a TimeSeries or a Probability Distribution can also be used.
FIFO	The order in which entities are placed in the queue. TRUE indicates first-in-first-out order (FIFO). FALSE indicates last-in-first-out order (LIFO).
RenegeTime	The time an entity will wait in the queue before deciding whether or not to renege. Evaluated when the entity first enters the queue. A constant value, a distribution to be sampled, a time series, or an expression can be entered.
RenegeCondition	A logical condition that determines whether an entity will renege after waiting for its RenegeTime value. Note that TRUE and FALSE are entered as 1 and 0, respectively. A constant value, a distribution to be sampled, a time series, or an expression can be entered.
RenegeDestination	The object to which an entity will be sent if it reneges.
Spacing	The amount of graphical space between objects in the Queue.
MaxPerLine	Maximum number of objects in each row of the Queue.

Table 10-13 Queue Outputs

Output Name	Description
State	Not used.
NumberAdded	The total number of objects that have been added to the Queue.
NumberProcessed	The total number of objects that have been removed from the Queue.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	Not used.
QueueTimes	The length of time, in seconds, that each entity has spent in the Queue.
PriorityValues	The priority value calculated for each entity in the Queue when it first arrived.
MatchValues	The match value calculated for each entity in the Queue when it first arrived.
QueueLength	The present number of objects in 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.
AverageQueueTime	The average time each entity waited in the Queue. Calculated as total accumulated queue time divided by the number of entities added to the Queue.

10.6 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.

Table 10-14 EntityConveyor Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changes to those for the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
TravelTime	The time required to convey an entity from the start to the end.
Width	The width of the line representing the EntityConveyor in pixels.
Color	The colour of the line representing the EntityConveyor.

Table 10-15 EntityConveyor Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-16.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.

Table 10-16 EntityConveyor State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.

10.7 EntityDelay

The EntityDelay object delays an incoming entity by a variable duration before passing 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 entity's delay is determined by the Duration keyword. This input has units of time and accepts a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-17 EntityDelay Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
Duration	The time required to process the incoming entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
Animation	If TRUE, entities are moved along the specified path to represent their progression through the delay activity.
Width	The width of the line representing the EntityDelay in pixels.
Color	The colour of the line representing the EntityDelay.

Table 10-18 EntityDelay Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in .
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

Table 10-19 EntityDelay State Definitions

State Name	Description
Working	One or more entities are undergoing this delay activity.
Idle	No entities are undergoing this delay activity.

10.8 Resource

The Resource object provides a pool of units that can be seized and released by the Seize and Release objects. The capacity of a Resource is modelled as a dimensionless integer value.

Table 10-20 Resource Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
Capacity	The number of units that can be seized. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression. A decimal input will be truncated to an integer by internal logic.

Table 10-21 Resource Outputs

Output Name	Description
UnitsSeized	The total number of units that have been seized.
UnitsReleased	The total number of units that have been released.
UnitsInUse	The present number of units that are in use.
UnitsInUseAverage	The average number of units that have been in use.
UnitsInUseStandardDeviation	The standard deviation of the number of units that have been in use.
UnitsInUseMinimum	The minimum number of units that have been in use.
UnitsInUseMaximum	The maximum number of units that have been in use.
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.

10.9 Seize

The Seize object allocates one or more units of a specified set of Resources on receiving an incoming entity.

If any of the Resources have insufficient units available, the received entity is directed to a Queue object identified by the WaitQueue keyword. All entities received by the Seize object first pass through this Queue object, even if sufficient units of the Resources are available.

Entities can be sent directly to the Queue specified by the WaitQueue keyword. Whenever an entity is added to the Queue, all the Seize objects that specified this Queue as their WaitQueue will be notified. The first Seize block that has sufficient resource units available will then remove the entity for processing.

Entities waiting for the same Resources at more than one Seize object are processed in order of the priority assigned to them by the input to their Queue's Priority keyword. If the entities in two or more Queues have the same priority value, then the one that has waited the longest is chosen.

All of the specified Resources for the selected entity must be available before any are seized. Once all of them are available, they are seized simultaneously.

Table 10-22 Seize Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
WaitQueue	A Queue object in which to hold entities that are waiting for sufficient Resource units to be available.
Match	An integer value that, if set, restricts the Seize object to only the entities in the Queue that have the same match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
OperatingThresholdList	A list of threshold objects that must all be open for operation to proceed. Operation is stopped if any of the thresholds are closed.
Resource	A list of Resources to be seized.
NumberOfUnits	A list containing the number of units of each Resource to be seized. Each entry can be a constant value, a TimeSeries, a Probability Distribution, or an Expression. A decimal input will be truncated to an integer by the internal logic.

Table 10-23 Seize Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the match variable for the last entity that was processed.

10.10 Release

The Release object de-allocates one or more units of a specified set of Resources on receiving an incoming entity. All of the Resources are released simultaneously.

On release of the Resources, the entities waiting for these Resources are processed in order of the priority assigned to them by the input to their Queue's Priority keyword. If the entities in two or more Queues have the same priority value, then the one that has waited the longest is chosen.

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
Resource	A list of Resources to be released.
NumberOfUnits	A list containing the number of units of each Resource to be released. Each entry can be a constant value, a TimeSeries, a Probability Distribution, or an Expression. A decimal input will be truncated to an integer by the internal logic.

Table 10-24 Release Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

10.11 Assign

The Assign object performs one or more Attribute assignments whenever it receives an incoming entity.

The Assign object is the only place where an Attribute's value can be modified. Attributes can be assigned for the Assign object itself, the received entity, or for any other entity in the model. Once the assignments have been performed, the received entity is passed to the next object without delay.

The Attribute assignments 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 (dimensionless) Attributes of the Assign object, the entity received, and Entity1, respectively.

The right side of each assignment equation is an Expression to be evaluated. The left side identifies the attribute whose value is to be modified, using the same syntax as an expression. Note that only Attributes can be assigned; it is not possible to assign a new value to an output.

The unit type for the Expression must match the unit type for the Attribute that was defined as part of the input to the AttributeDefinitionList keyword. See Section 5 for more information about Units.

Table 10-25 Assign Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
AttributeAssignmentList	The assignment equations to be performed on receipt of an entity.

Table 10-26 Assign Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

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 DiscreteDistribution, a TimeSeries, or an Expression. The use of an Expression allows the choice to be made based on the Attributes of the incoming entity.

Table 10-27 Branch Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
NextComponentList	A list of possible objects to which an incoming entity 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. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-28 Branch Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

10.13 Duplicate

The Duplicate object sends copies of the received entity to one or more objects.

Table 10-29 Duplicate Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
TargetComponentList	A list of the objects to which a copy of the received entity will be sent.

Table 10-30 Duplicate Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

10.14 Combine

The Combine object takes one entity each from multiple queues and passes on a single entity to the next object. Each of the entities must have the same match value calculated by the input to Match keyword for its Queue. Entities are combined when each queue has at least one entity with the same Match value as the other Queues. When a match is found, the entity in the first Queue is passed to the object specified by the NextComponent keyword, while the entities from the other Queues are destroyed.

Table 10-31 Combine Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntiy.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the entity being processed relative to the position of the Combine object.
OperatingThresholdList	A list of threshold objects that must all be open for the operation to proceed. Operation is stopped if any of the thresholds are closed.
ServiceTime	The time required to process the incoming entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
WaitQueueList	A list of Queue objects that hold the entities waiting to be combined.

Table 10-32 Combine Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the match variable for the last set of entities that were combined.

10.15 SetGraphics

The SetGraphic object is used to change the graphical appearance of a specified entity.

Table 10-33 SetGraphics Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
TargetEntity	The entity whose graphics are to be changed. Defaults to the received entity.
GraphicsList	A list of DisplayEntities whose graphics can be copied to the received entity.
Choice	The index (1-N) of the DisplayEntity whose graphics are to be copied to the received entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-34 SetGraphics Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

10.16 EntityGate

The EntityGate object allows incoming entities to be blocked temporarily.

When the EntityGate is open and the Queue is empty, incoming entities pass through without delay. When it is Closed, incoming entities are directed to a Queue where they are held until such time as the EntityGate becomes Open. When the EntityGate enters the Open state, 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 become Open.

Table 10-35 EntityGate Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the entity being processed relative to the position of the EntityGate object.
WaitQueue	A Queue object in which to hold incoming entities that are blocked when EntityGate is closed.
Match	An integer value that, if set, restricts the EntityGate to only the entities in the queue that have the same match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
OperatingThresholdList	A list of threshold objects that determine whether the EntityGate is open or closed.
ReleaseDelay	The time required to remove each entity from the Queue.

Table 10-36 EntityGate Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-37.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the match variable for the last entity that was processed.

Table 10-37 EntityGate State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.17 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.

Table 10-38 EntitySignal Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
TargetSignalThreshold	The SignalThreshold object whose state will be changed by this EntitySignal.
NewState	The state to be set for the SignalThreshold: TRUE if it is open, FALSE if it is closed.

Table 10-39 EntitySignal Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.

10.18 SignalThreshold

The SignalThreshold object varies its state between open and closed when instructed to do so by one or more EntitySignal objects.

SignalThreshold has no internal logic of its own for changing state.

Table 10-40 SignalThreshold Key Inputs

Keyword	Description
InitialState	The initial state for the SignalThreshold at the start of the run. TRUE = open, FALSE = closed.

Table 10-41 SignalThreshold Graphics Inputs

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

Table 10-42 SignalThreshold Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-43.
Open	TRUE if the SignalThreshold has been set to open, FALSE if the SignalThreshold has been set to closed.
OpenFraction	The fraction of total time during which the SignalThreshold is open.
ClosedFraction	The fraction of total time during which the SignalThreshold is closed.

Table 10-43 SignalThreshold State Definitions

State Name	Description
Open	The SignalThreshold is open.
Closed	The SignalThreshold is closed.

10.19 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 must be sent directly to the appropriate queue, not to the Assemble object itself. If necessary, a Branch object can be used to direct incoming entities to the various queues.

The assembly process begins when there is at least one sub-component entity in each of the Queues. When 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.

Table 10-44 Assemble Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a received entity is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the entity being processed relative to the position of the Assemble object.
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. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
WaitQueueList	A set of Queue objects, one for each type of sub-component to be assembled.
NumberRequired	The number of entities required from each Queue for the assembly process to begin. The last value in the list is used if the number of Queues is greater than the number of values.
MatchRequired	If TRUE, the all entities used in the assembly process must have the same Match value. The Match value for an entity is determined by the Match keyword for each Queue. The value is calculated when the entity first arrives at its Queue.
PrototypeEntity	The entity whose copies will represent the assembled part.

Table 10-45 Assemble Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-46.
obj	Entity last received by the Assemble object.
NumberAdded	Number of entities received.
NumberProcessed	The total number of assembled parts that have been created.
ProcessingRate	The rate at which assembled parts have been created (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the Match variable for the last entities that were assembled.

Table 10-46 Assemble State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.20 EntityContainer

The EntityContainer object is used to store a collection of entities for subsequent processing as a unit.

EntityContainer is a sub-class of SimEntity and consequently includes the State output. It can be passed to any object that can accept a SimEntity. The Pack object is used to place entities in generated EntityContainers. The Unpack object is used to remove the entities and to destroy the EntityContainer.

Table 10-47 EntityContainer Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultStateList	A list of states that will appear in the output report, even if no time is recorded for this state.
PositionOffset	The position of the first entity in the container relative to the EntityContainer.
Spacing	The amount of graphical space between entities in the EntityContainer.
MaxPerLine	Maximum number of entities contained in each row of the EntityContainer.

Table 10-48 EntityContainer Outputs

Output Name	Description
State	The present user-defined activity being performed by the EntityContainer.
Count	The present number of entities inside the EntityContainer.

10.21 Pack

The Pack object is used to generate EntityContainers and fill them with incoming entities.

EntityContainers are created automatically on demand by the Pack object. The number of entities to be packed in each EntityContainer is determined by the NumberOfEntities keyword. Once the specified number of entities are available in the Queue, they are packed one by one in the EntityContainer in same order as the Queue. The time to pack each entity is specified by the ServiceTime keyword.

Table 10-49 Pack Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those of the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a packed EntityContainer is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the EntityContainer being processed relative to the position of the Pack object.
OperatingThresholdList	A list of threshold objects which must be open for the packing operation to proceed.
PrototypeEntityContainer	The EntityContainer whose copies will be packed.
WaitQueue	The Queue in which the arriving entities are stored while they wait to be packed.
Match	An integer value that, if set, restricts the Pack object to only the entities in the Queue that have the same Match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
NumberOfEntities	The number of entities to pack into each EntityContainer. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
ServiceTime	The time required to pack each entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-50 Pack Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-51.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of EntityContainers that have been created.
ProcessingRate	The rate at which EntityContainers have been created (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the Match variable for the last entities that were packed.
Container	The EntityContainer currently being filled.

Table 10-51 Pack State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.22 Unpack

The Unpack object is used to remove the entities from incoming EntityContainers.

Entities are unpacked one by one from the EntityContainer in same order as they were packed. The time to unpack each entity is specified by the ServiceTime keyword. The received EntityContainer is destroyed once it has been unpacked.

Table 10-52 Unpack Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those of the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a packed EntityContainer is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the EntityContainer being processed relative to the position of the Unpack object.
WaitQueue	The Queue in which the arriving entities are stored while they wait to be unpacked.
Match	An integer value that, if set, restricts the Unpack object to only the entities in the Queue that have the same Match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
OperatingThresholdList	A list of threshold objects which must be open for the unpacking operation to proceed.
ServiceTime	The time required to unpack each entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 10-53 Unpack Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-54.
obj	Entity last received.
NumberAdded	Number of EntityContainers received.
NumberProcessed	The total number of entities that have been unpacked.
ProcessingRate	The rate at which entities have been unpacked (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the match variable for the last EntityContainer that was unpacked.

Table 10-54 Unpack State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.23 AddTo

The AddTo object is used to add a given number of entities to a received EntityContainer.

The number of entities to be added each EntityContainer is determined by the NumberOfEntities keyword. Once the specified number of entities are available in the Queue and there is an EntityContainer waiting in its Queue, they are added one by one to the EntityContainer in same order as the Queue. The time to add each entity is specified by the ServiceTime keyword.

Table 10-55 AddTo Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those of the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which a processed EntityContainer is passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the EntityContainer being processed relative to the position of the AddTo object.
WaitQueue	The Queue in which the arriving entities are stored while they wait to be added.
Match	An integer value that, if set, restricts the AddTo object to only the entities in the Queue that have the same Match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
OperatingThresholdList	A list of threshold objects which must be open for the adding operation to proceed.
NumberOfEntities	The number of entities to add to each EntityContainer. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
ServiceTime	The time required to add each entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
ContainerQueue	The Queue in which the arriving EntityContainers are stored while they wait for processing.

Table 10-56 AddTo Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-57.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of EntityContainers that have been created.
ProcessingRate	The rate at which entities have been unpacked (NumberProcessed/Total Time).
ReleaseTime	The time at which the last EntityContainer was passed to the next object.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the Match variable for the last entities that were added to the EntityContainer.
Container	The EntityContainer to which entities are being added.

Table 10-57 AddTo State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.24 RemoveFrom

The RemoveFrom object is used to remove a given number of entities from an incoming EntityContainer.

Entities are removed one by one from the EntityContainer in same order as they were packed. The time to unpack each entity is specified by the ServiceTime keyword. EntityContainers are passed to another object once the specified number of entities have been unpacked.

Table 10-58 RemoveFrom Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those of the corresponding DisplayEntity.
DefaultEntity	The prototype entity used to generate the stream of entities received by this object.
NextComponent	The next object to which the removed entities are passed.
StateAssignment	The state to be assigned to a SimEntity when it is first received by this object. The state name can be chosen freely by the user. The SimEntity's state is unchanged if the value for this keyword is blank.
ProcessPosition	The position of the EntityContainer being processed relative to the position of the RemoveFrom object.
OperatingThresholdList	A list of threshold objects which must be open for the entity removal operation to proceed.
WaitQueue	The Queue in which the arriving EntityContainers are stored while they wait to be processed.
Match	An integer value that, if set, restricts the RemoveFrom object to only the EntityContainers in the Queue that have the same Match value. Normally, the Match input will be an Expression, but a constant value or any object that returns a dimensionless number such as a TimeSeries or a Probability Distribution will also be accepted.
ServiceTime	The time required to remove each entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
NumberOfEntities	The number of entities to remove from a received EntityContainer. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
NextForContainers	The object to which a processed EntityContainer will be passed.

Table 10-59 RemoveFrom Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Definitions for the various states are given in Table 10-60.
obj	Entity last received.
NumberAdded	Number of EntityContainers received.
NumberProcessed	The total number of entities that have been removed.
ProcessingRate	The rate at which entities have been removed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last EntityContainer was passed to the next object.
Working	Returns TRUE if entities are being processed.
Open	Returns TRUE if all the thresholds specified by the OperatingThresholdList keyword are open.
MatchValue	The value of the Match variable for the last EntityContainer that was unpacked.

Table 10-60 RemoveFrom State Definitions

State Name	Description
Working	Performing its normal service.
Idle	Available for service, but has no work to perform.
Stopped	Service has been suspended by the closure of one or more thresholds.
Clearing_while_Stopped	The present service activity is being completed, before ceasing further operation.

10.25 EntityLogger

The EntityLogger object records the outputs, attributes, and state data for each entity that it receives.

The output log file is created automatically when the simulation run begins. The output file is named <configuration file name>-<EntityLogger name>.log to ensure that it is unique for the simulation run. For example, if the configuration file is named "run1.cfg" and the EntityLogger's name is EntityLogger1, then the name of the log file will be "run1-EntityLogger1.log". A pre-existing file with this name will be overwritten once the simulation run is started.

Table 10-61 EntityLogger Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
NextComponent	The next object to which a received entity is passed.

Table 10-62 EntityLogger Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
LogTime	The simulation time at which the last entity was logged.

10.26 Statistics

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

The quantity to be tracked is specified using the SampleValue keyword, which accepts an Expression. Some example inputs 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

Table 10-63 Statistics Key Inputs

Keyword	Description
AttributeDefinitionList	User-defined outputs to be added to this entity.
DefaultEntity	The prototype entity used to generate the entities received by the Statistics object.
NextComponent	The next object to which a received entity is passed.
UnitType	The unit type for the variable for which statistics will be collected.
SampleValue	An Expression that returns the sample value for which statistics are to be collected.

Table 10-64 Statistics Outputs

Output Name	Description
State	Not used.
obj	Entity last received.
NumberAdded	Number of entities received.
NumberProcessed	The total number of entities that have been processed.
ProcessingRate	The rate at which entities have been processed (NumberProcessed/Total Time).
ReleaseTime	The time at which the last entity was released.
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 Calculation Objects

The Calculations Objects Palette contains objects for building continuous type models and mixed discrete-event/continuous models. As the name suggests, a continuous model changes state continuously as time advances. JaamSim is able to model this type of behaviour very efficiently using outputs, Attributes, and Expressions. The following objects are provided in the Calculation Objects palette.

Table 11-1 Calculation Objects Palette

Object	Description
Controller	Signals the updating of each component in the specified sequence.
WeightedSum	Calculates the weighted sum of the input values.
Polynomial	Evaluates a polynomial function of the input value.
Integrator	Integrates the input value over time.
Differentiator	Differentiates the input value over time.
PIDController	Proportional-Integral-Differential controller.
Lag	Calculates the LAG operation for the input value.
MovingAverage	Calculates a moving average of the input value over a specified range of time.
SineWave	Generates a sinusoidal wave.
SquareWave	Generates a square wave.
UnitDelay	Delays the input value by one Controller time step.

11.1 Controller

The Controller object generates the update signals for the individual objects in a continuous type model that are managed by this Controller.

Table 11-2 Controller Key Inputs

Keyword	Description
SamplingTime	Interval between update signals to the objects managed by this Controller.

Table 11-3 Controller Outputs

Output Name	Description
SimTime	The present simulation time.

11.2 WeightedSum

The WeightedSum object adds two or more input values. A set of dimensionless constants can be provided to multiply each input value prior to addition.

$$y = C_1*x_1 + C_2*x_2 + \ldots + C_N*x_N, \text{ if the coefficients } C_i \text{ are specified}$$
$$= x_1 + x_2 + \ldots + x_N \quad , \text{ if the coefficients } C_i \text{ are not specified}$$

where:

y = present output value for the weighted sum
x_i = present value for the i^{th} input to the weighted sum
C_i = i^{th} entry in the **CoefficientList** input
N = number of inputs to the weighted sum

The value returned by the WeightedSum is calculated on demand.

Table 11-4 WeightedSum Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the calculation.
InputValueList	The list of inputs to the weighted sum. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
CoefficientList	The list of dimensionless multiplicative factors to be applied to the values provide by the inputs.

Table 11-5 WeightedSum Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.3 Polynomial

The Polynomial object evaluates a specified polynomial for the present input value:

$$y = C_0 + C_1*x + C_2*x\text{^}2 + \ldots + C_N*x\text{^}N$$

where:

y = present output value for the polynomial
x = present input to the polynomial
C_i = i^{th} entry in the **CoefficientList** input

The value returned by the Polynomial is calculated on demand.

Table 11-6 Polynomial Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the calculation.
InputValue	The input value for the present calculation. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
CoefficientList	The list of dimensionless coefficients for the polynomial function. The number of coefficients determines the order of the polynomial.

Table 11-7 Polynomial Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.4 Integrator

The Integrator object integrates the input value with respect to time using the trapezoidal rule:

```
y = Y + (t – T)*0.5*(x + X)
```

where:

```
y = present output value for the integrator
x = present input to the integrator
t = present simulation time
Y = output value for the integrator at the last update time
X = input to the integrator at the last update time
T = simulation time at the last update
```

The value returned by the Integrator is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 11-8 Integrator Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object. If a Controller is not specified, the output value is calculated asynchronously, on demand.
UnitType	The unit type for the value returned by the calculation.
InputValue	The input value for the present calculation. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
InitialValue	The output value at zero simulation time.

Table 11-9 Integrator Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.5 Differentiator

The Differentiator object calculates the time derivative of the input value:

```
y = (x – X)/(t – T), for t > T
  = Y              , for t = T
```

where:

```
y = present output value for the differentiator
x = present input to the differentiator
t = present simulation time
Y = output value for the differentiator at the last update time
X = input to the differentiator at the last update time
T = simulation time at the last update
```

The value returned by the Integrator is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 11-10 Differentiator Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object. If a Controller is not specified, the output value is calculated asynchronously, on demand.
UnitType	The unit type for the value returned by the calculation.
InputValue	The input value for the present calculation. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.

Table 11-11 Differentiator Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.6 PIDController

The PIDController object simulates the Proportional-Integral-Differential (PID) type controller that is widely used in electronic control systems.

```
error = SetPoint – x
integral = INTEGRAL + (error – ERROR)*(t – T)
derivative = (error – ERROR)/(t – T)

y’ = ProportionalGain/SetPointScale * (error + integral/IntegralTime +
                 derivative*DerivativeTime)

y = min( max( y’, OutputLow ), OutputHigh )
```

where:

```
y = present output value for the PID controller
x = present ProcessVariable input to the PID controller
t = present simulation time
Y = output value for the PID controller at the last update time
X = ProcessVariable input to the PID controller at the last update time
T = simulation time at the last update
ERROR = value for error at the last update time
INTEGRAL = value for integral at the last update time
```

The value returned by the PIDController is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, T, ERROR, and INTEGRAL used in the calculation.

Table 11-12 PIDController Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object. If a Controller is not specified, the output value is calculated asynchronously, on demand.
UnitType	The unit type for the value returned by the calculation.
SetPoint	The set point for the PIDController. The unit type for the set point is given by the UnitType keyword. The set point can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
ProcessVariable	The process variable feedback to the PIDController. The unit type for the process variable is given by the UnitType keyword. The process variable can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
ProcessVariableScale	A constant with the same unit type as the process variable and the set point. The difference between the process variable and the set point is divided by this quantity to make a dimensionless variable.
OutputUnitType	The unit type for the output from the PID controller.
ProportionalGain	The coefficient applied to the proportional feedback loop. The unit type for the proportional gain is given by the OutputUnitType keyword.
IntegralTime	The coefficient applied to the integral feedback loop.
DerivativeTime	The coefficient applied to the differential feedback loop.
OutputLow	The lower limit for the output signal.
OutputHigh	The upper limit for the output signal.

Table 11-13 PIDController Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.
ProportionalValue	The proportional component of the output value.
IntegralValue	The integral component of the output value.
DerivativeValue	The derivative component of the output value.

11.7 Lag

The Lag object calculates the lag operation used in electronic control systems.

```
y = Y + (t – T)*(x – Y)/LagTime
```

where:

```
y = present output value for the Lag object
x = present input to the Lag object
t = present simulation time
Y = output value for the Lag object at the last update time
X = input to the Lag object at the last update time
T = simulation time at the last update
```

The value returned by the Lag object is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 11-14 Lag Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object. If a Controller is not specified, the output value is calculated asynchronously, on demand.
UnitType	The unit type for the value returned by the calculation.
InputValue	The input value for the present calculation. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
LagTime	A value with units of time used in the lag calculations.

Table 11-15 Lag Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.
Error	The value for (InputValue - Value).

11.8 MovingAverage

The MovingAverage object calculates the average value of the input over a given time interval.

$$y = \{ x + \sum_{(i = M \text{ to } M-N-2)} X(i) \} / N$$

where:

y = present output value for the MovingAverage object
x = present input to the MovingAverage object
N = **NumberOfSamples**
M = number of updates that have been performed previously
X(i) = input to the MovingAverage object at the i^{th} update time

The value returned by the MovingAverage object is calculated on demand. The update signal received from the Controller is used only to record the values X(i) used in the calculation.

Table 11-16 MovingAverage Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object. If a Controller is not specified, the output value is calculated asynchronously, on demand.
UnitType	The unit type for the value returned by the calculation.
InputValue	The input value for the present calculation. The inputs can be a constant value or any entity that returns a number, such as a calculation object, a TimeSeries, a Probability Distribution, or an Expression.
NumberOfSamples	The number of previous values over which to take the average.

Table 11-17 MovingAverage Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.9 SineWave

The SineWave object generates a sine wave output:

$$y = \mathbf{Offset} + \mathbf{Amplitude} * \sin(2\pi\ t/\mathbf{Period} + \mathbf{PhaseAngle})$$

where:

```
y = present output value for the SineWave object
t = present simulation time
```

The value returned by the SineWave is calculated on demand.

Table 11-18 SineWave Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the calculation.
Amplitude	The height of the SineWave measured from zero.
Period	The time interval over which the SineWave repeats.
PhaseAngle	The angle by which the SineWave is shifted in time.
Offset	An absolute shift of the SineWave up or down.

Table 11-19 SineWave Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.10 SquareWave

The SquareWave object generates a square wave output:

$$y = \mathbf{Offset} + \mathbf{Amplitude}, \text{ if } \sin(2\pi\ t/\mathbf{Period} + \mathbf{PhaseAngle}) >= 0$$
$$= \mathbf{Offset} - \mathbf{Amplitude}, \text{ if } \sin(2\pi\ t/\mathbf{Period} + \mathbf{PhaseAngle}) < 0$$

where:

```
y = present output value for the SquareWave object
t = present simulation time
```

The value returned by the SquareWave is calculated on demand.

Table 11-20 SineWave Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the calculation.
Amplitude	The height of the SquareWave measured from zero.
Period	The time interval over which the SquareWave repeats.
PhaseAngle	The angle by which the SquareWave is shifted in time.
Offset	An absolute shift of the SquareWave up or down.

Table 11-21 SineWave Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

11.11 UnitDelay

The UnitDelay object holds the value from the last update time. It is used to prevent an infinite loop from occurring when a modelled calculation includes a feedback loop.

$$y = \mathbf{X}$$

where:

```
y = present output value for the UnitDelay object
X = input to the UnitDelay at the last update time
```

Table 11-22 UnitDelay Key Inputs

Keyword	Description
UnitType	The unit type for the value returned by the calculation.
InitialValue	The value for the UnitDelay function at zero simulation time.

Table 11-23 UnitDelay Outputs

Output Name	Description
SimTime	The present simulation time.
Value	The present output value for this calculation object.

12 Fluid Objects

The Fluid Objects Palette contains objects for building dynamic models of fluid flow involving tanks, pipes, pumps, etc. The following objects are provided in Fluid Objects palette.

Table 12-1 Fluid Objects Palette

Object	Description
Fluid	Defines a specific fluid and its properties.
FluidFlow	Computed flow of a specified fluid between a source and a destination.
FluidFixedFlow	Constant flow to/from a specified tank.
FluidTank	Cylindrical vessel for storing fluid.
FluidPipe	Cylindrical conduit for transporting fluid.
FluidCentrifugalPump	Type of pump with a rotating impeller.

12.1 Fluid

The Fluid object defines the basic properties of the fluid.

Table 12-2 Fluid Key Inputs

Keyword	Description
Density	The density of the Fluid. Defaults to the density of water.
Viscosity	The dynamic viscosity of the Fluid. Defaults to the viscosity for water.
Colour	The colour used to represent the Fluid.
Gravity	The acceleration of gravity to be used in the fluid flow calculations.

12.2 FluidFlow

The FluidFlow object calculates the flow rate between a source and destination by solving the unsteady Bernoulli equation.

Table 12-3 FluidFlow Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before the ones with higher values.
Fluid	The Fluid moved by the flow.
Source	The source object for the flow.
Destination	The destination object for the flow.

Table 12-4 FluidFlow Outputs

Output Name	Description
SimTime	The present simulation time.
FlowAcceleration	The time derivative of the volumetric flow rate.
FlowInertia	The sum of Density*Length/FlowArea for the hydraulic components in the flow.

12.3 FluidFixedFlow

The FluidFixedFlow object moves Fluid between a source and destination at a fixed volumetric flow rate.

Table 12-5 FluidFixedFlow Key Inputs

Keyword	Description
Controller	The Controller object that provides the update signal for this object.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before the ones with higher values.
Fluid	The Fluid being moved by the flow.
Source	The source object for the flow.
Destination	The destination object for the flow.
FlowRate	The constant volumetric flow rate from the source to the destination.
Points	The coordinates for the multi-segment line representing the flow path.
Width	The width, in pixels, of the multi-segment line representing the flow path.
Colour	The colour of the multi-segment line representing the flow path.

Table 12-6 FluidFixedFlow Outputs

Output Name	Description
SimTime	The present simulation time.
FlowRate	The volumetric flow rate.

12.4 FluidTank

The FluidTank object represents a cylindrical storage tank containing Fluid. FluidTanks are modelled as large diameter FluidPipes so that the velocity and inertia of the Fluid are preserved in calculations.

Table 12-7 FluidTank Key Inputs

Keyword	Description
Previous	The upstream flow component that feeds the FluidTank.
Diameter	The hydraulic diameter of this component.
Capacity	The volumetric capacity of the FluidTank.
InitialVolume	The volume of Fluid in the FluidTank at the start of the simulation run.
AmbientPressure	The atmospheric pressure acting on the surface of the Fluid in the FluidTank.
InletHeight	The height of the flow feeding the FluidTank. Measured relative to the bottom of the FluidTank.

Table 12-8 FluidTank Outputs

Output Name	Description
SimTime	The present simulation time.
FlowArea	Cross-sectional area of the FluidTank.
Velocity	The velocity of the Fluid in the FluidTank.
InletPressure	The static pressure at the inlet to the FluidTank.
DynamicPressure	The dynamic pressure for the Fluid in the FluidTank.
ReynoldsNumber	The Reynolds number for the Fluid in the FluidTank.
OutletPressure	The static pressure at the outlet from the FluidTank.

12.5 FluidPipe

The FluidPipe object represents a cylindrical pipe for transporting Fluid.

Table 12-9 FluidPipe Key Inputs

Keyword	Description
Previous	The upstream flow component that feeds this component.
Diameter	The hydraulic diameter of this component.
Length	The length of the FluidPipe.
HeightChange	The change in elevation between the FluidPipe 's inlet and outlet.
Roughness	The roughness height for the inside surface of the FluidPipe. Used to calculate the Darcy friction factor.
PressureLossCoefficient	The pressure loss coefficient, or K-factor, for the FluidPipe. The factor multiplies the dynamic pressure and is applied as a loss at the FluidPipe 's outlet.

Table 12-10 FluidPipe Outputs

Output Name	Description
SimTime	The present simulation time.
FlowArea	Cross-sectional area of the FluidPipe.
Velocity	The velocity of the Fluid in the FluidPipe.
InletPressure	The static pressure at the inlet to the pipe.
DynamicPressure	The dynamic pressure for the Fluid in the FluidPipe.
ReynoldsNumber	The Reynolds number for the Fluid in the FluidPipe.
OutletPressure	The static pressure at the outlet from the FluidPipe.

12.6 FluidCentrifugalPump

The FluidCentrifugalPump object models the performance of a centrifugal pump.

Table 12-11 FluidCentrifugalPump Key Inputs

Keyword	Description
Previous	The upstream flow component that feeds the FluidCentrifugalPump.
Diameter	The hydraulic diameter of this component.
MaxFlowRate	The maximum volumetric flow rate that the FluidCentrifugalPump can generate.
MaxPressure	The maximum static pressure that the FluidCentrifugalPump can generate, which occurs at zero flow rate.
MaxPressureLoss	The maximum static pressure loss for the FluidCentrifugalPump, which occurs at the maximum flow rate.
SpeedController	A calculation entity that returns relative speed for the pump: 0.0 = zero speed, 1.0 = maximum speed.

Table 12-12 FluidCentrifugalPump Outputs

Output Name	Description
SimTime	The present simulation time.
FlowArea	Cross-sectional area of the FluidCentrifugalPump's outlet.
Velocity	The velocity of the Fluid in the FluidCentrifugalPump.
InletPressure	The static pressure at the inlet to the FluidCentrifugalPump.
DynamicPressure	The dynamic pressure for the Fluid in the FluidCentrifugalPump.
ReynoldsNumber	The Reynolds number for the Fluid in the FluidCentrifugalPump.
OutletPressure	The static pressure at the outlet from the FluidCentrifugalPump.

13 Advanced Topics

A number of more advanced features of JaamSim are discussed in this section.

13.1 Time Formats

Inputs of time can be entered in the normal fashion, e.g. 5 s, 12 h, etc., or they can be entered in a year/month/day format. Both the ISO 8601 standard and the full RFC8601 format are supported.

Table 13-1 Supported Time Formats

Format	Example	Description
YYYY-MM-DD	2013-08-25	August 25, 2013
YYYY-MM-DDTHH:mm:SS	2013-08-25T04:25:00	4:25 AM on August 25, 2013
'YYYY-MM-DD HH:mm:SS'	'2013-08-25 04:25:00'	4:25 AM on August 25, 2013
YYYY-MM-DDTHH:mm:SS.dddddd	2013-08-25T04:25:00.1	0.1 seconds past 4:25 AM on August 25, 2013
HHHHHHHH:mm:SS.dddddd	50000:00:00.000001	1 microsecond past 50,000 hours

January 1 of year 0 (0000-01-01) is taken to be zero simulation time. For example, an input of '0001-01-01' is equivalent to 8760 h.

13.2 Expressions

Expressions are mathematical statements that are evaluated by JaamSim. Many keywords that expect a number can also accept an object that returns a number, such as a Probability Distribution, a TimeSeries, and an Expression.

13.2.1 Basics

Mathematical expressions are entered using the following syntax:

- An expression is enclosed in single quotes if it contains any spaces
- Object names are enclosed by square brackets
- Spaces in an expression are optional
- Outputs and Attributes of an object are referenced using a dot notation, e.g. [ObjectName].OutputName
- Outputs and Attributes can be chained when necessary using the format [ObjectName].OutputName1.OutputName2
- Outputs that return an array of values can used in an expression by specifying an index, e.g. [ObjectName].OutputName(Index). The index value can be either a constant or an expression that returns a dimensionless number. A non-integer value for the index will be truncated.
- The reserved string 'this' is used to refer to the object evaluating the expression
- Rules for mathematical order of operation are respected
- Units are enclosed by square brackets

- All calculations must respect Units. For example, two values must have the same UnitType if they are to be added.
- Unit conversions are performed automatically. For example, a distance divided by a time results in a speed.

The following are examples of syntactically valid expressions. They assume that the present object has a dimensionless output, OutputA, and another entity, Entity1, has a dimensionless output, OutputB.

```
'1 + 2*3'
'1 + 2 * 3^2'
'1 + 2*this.OutputA'
'1 + 2*[Entity1].OutputB'
'[Simulation].RunIndex(2)'
'1[m] + 2[km]'
'2[m] / 4[s]'
```

13.2.2 Functions

The following mathematical functions can be used in expressions:

Table 13-2 Basic Mathematical and Logical Functions

Function	Description	Example	Result
PI	Mathematical constant 'pi'	`PI()`	`3.14159…`
E	Mathematical constant 'e'	`E()`	`2.71828…`
min	Smallest of a list of values	`'min(1[s], 2[s])'`	`1[s]`
max	Largest of a list of value	`'max(1[s], 2[s])'`	`2[s]`
indexOfMin	Position of the minimum in a list of values	`'indexOfMin(1[s], 2[s])'`	`1`
indexOfMax	Position of the maximum in a list of values	`'indexOfMax(1[s], 2[s])'`	`2`
abs	Absolute value	`'abs( -1[s] )'`	`1[s]`
ceil	Smallest (closest to negative infinity) integer that is greater than or equal to the argument	`'ceil( 5.2[s] )'`	`6[s]`
floor	Largest (closest to positive infinity) integer that is less than or equal to the argument	`'abs( 5.2[s] )'`	`5[s]`
signum	zero if the argument is zero, 1.0 if the argument is greater than zero, and -1.0 if the argument is less than zero.	`'signum( 5.2[s] )'`	`1`
sqrt	Square root	`'sqrt( 4.0 )'`	`2.0`
cbrt	Cube root.	`'cbrt( 8.0 )'`	`2.0`
%	Modulus (remainder) operator. Used as an operator, not a function.	`'11.5 % 4'`	`3.5`
choose	Selects from a list using an index	`'choose(2, 1[s], 2[s])'`	`2[s]`

Table 13-3 Exponential Functions

Function	Description	Example	Result
exp	Exponential function	`'exp( 1 )'`	`2.71828…`
ln	Natural logarithm	`'ln( 2.71828 )'`	`0.999999`
log	Base-10 logarithm	`'log( 100 )'`	`2.0`

Table 13-4 Trignometric Functions

Function	Description	Example	Result
sin	Sine of an angle or dimensionless number	`'sin( 30[deg] )'`	`0.5`
cos	Cosine of an angle or dimensionless number	`'cos( 60[deg] )'`	`0.5`
tan	Tangent of an angle or dimensionless number	`'tan( 45[deg] )'`	`1`
asin	Arcsine function	`'asin( 0.5 )'`	`30[deg]`
acos	Arccosine function	`'acos( 0.5 )'`	`60[deg]`
atan	Arctangent function	`'atan( 1.0 )'`	`45[deg]`
atan2	Two-argument arctangent function. For Cartesian coordinates x and y, atan2(x,y) returns the angle for the corresponding polar coordinates.	`'atan2(1.0, -1.0)'`	`135[deg]`

Examples of valid Expressions that use functions are:

```
'1 + 2*max(3,4)'
'1 + 2 * max(1+this.OutputA, 3*[Entity1].OutputB)'
```

It is often helpful to use an ExpressionEntity to test the behaviour of an expression that is being developed.

13.2.3 Logical Operations

Logical operations can also be performed in Expressions. All non-zero dimensionless values are interpreted as TRUE, while zero is interpreted as FALSE. The following are examples of syntactically valid logical Expressions:

```
'this.OutputA >= 1'
'(this.OutputA >= 1) && ([Entity1].OutputB == 0)'
```

The following logical operators are supported:

Table 13-5 Logical Operators

Operator	Description	Example	Result
==	Equal to	`'4.2 == 4.2'`	1
!=	Not equal to	`'3.5 != 4.2'`	1
<	Less than	`'3.5 < 4.2'`	1
<=	Less than or equal to	`'3.5 <= 3.5'`	1
>	Greater than	`'4.2 > 3.5'`	1
>=	Greater than or equal to	`'3.5 >= 3.5'`	1
&&	Logical AND operation	`'1 && 1'`	1
||	Logical OR operation	`'1 || 0'`	1
!	Logical NOT operation	`'! 0'`	1

13.2.4 Conditionals

Conditional operations are implemented with the ternary operator, '?', using the following syntax:

```
<condition_expression> ? <true_expression> : <false_expression>
```

Examples of the conditional operator are:

```
'2>1 ? 5 : 4'
'this.OutputA >= 2 ? [Entity1].OutputB + 1 : 0'
```

Complex logical expressions can be constructed by chaining a series of ternary operators to form an "if, else-if" type structure, e.g.:

```
'this.OutputA <= 2 ? 2 : (this.OutputA <= 4 ? 4 : 6)'
```

Note that brackets were added to this expression to improve readability – they are not mandatory.

13.3 Attributes

In addition to the pre-programmed outputs, users can create custom outputs using Attributes, for individual objects. Attributes appear in the Output Viewer immediately after being defined through the AttributeDefinitionList keyword, and as such can be referenced through the [ObjectName].OutputName format, except replacing the OutputName with the desired Attribute name.

An entity's attributes are defined using the AttributeDefinitionList keyword using the following format:

```
{ <Attribute1> <InitialValue1> <Unit1> } ... { <AttributeN> <InitialValueN> <UnitN> }
```

For example, the inputs to create two new Attributes, AttributeA and AttributeB, with initial values of 1 km and 9 (dimensionless) are:

```
{ AttributeA 1 km } { AttributeB 9 }
```

Attributes can be used in Expressions in the same way as an output. For instance, if the above AttributeA was defined for Entity1, it can be referenced by using:

```
[Entity1].AttributeA
```

Once an Attribute has been defined, only the Assign object can modify its value.

13.4 Custom Outputs

Users can add their own outputs to an object by entering one or more expressions to the CustomOutputList keyword. Custom outputs appear in the Output Viewer next to the attributes.

The CustomOutputList keyword has the following format:

```
{ <Output1> <expression1> <UnitType1> } ... { <OutputN> <expression2> <UnitTypeN> }
```

For example, the following input creates a custom output called TwiceSimTime, whose value is equal to two times the present simulation time:

```
{ TwiceSimTime  '2 * this.SimTime'  TimeUnit }
```

Unlike an attribute, whose value can only be changed by an Assign object, the value for a custom output is calculated on demand by evaluating the specified expression.

13.5 Performing Multiple Simulation Runs

Multiple simulation runs can be executed automatically, one after another, using the keywords under the Multiple Runs tab for the Simulation object. The Simulation outputs RunNumber and RunIndex are used to change selected inputs between simulation runs. By default, the RunNumber output starts at 1 and is incremented by one with each simulation run that is performed. This output can be used to vary one or more inputs by referencing [Simulation].RunNumber in an expression. For example, setting the ServiceTime input for a Server to the expression

```
'1[s] + 0.1[s]*[Simulation].RunNumber'
```

would set the service time to 1.1 s, 1.2 s, 1.3 s, etc., as the run number is incremented over multiple runs.

The RunIndex output is used when there are multiple inputs to test. This output contains an array of integers that are each incremented from 1 to N, where a separate value for N can be specified for each index. The number of run indices and the ranges over which they are incremented are determined by the RunIndexDefinitionList keyword for Simulation (see Table 13-6 below).

For example, suppose there are two Servers and service times of 1.1 s, 1.2 s, and 1.3 s are to be tested for Server1 and service times of 2.1 s and 2.2 s are to be tested for Server2. In this case, two run indices would be defined by entering the values 3 2 to the RunIndexDefinitionList keyword. This input indicates that RunIndex(1) would be incremented over the range 1 to 3 and RunIndex(2) would be incremented over the range 1 to 2. The inputs to the ServiceTime keyword for Server1 would be:

```
'1[s] + 0.1[s]*[Simulation].RunIndex(1)'
```

While the input for Server2 would be:

```
'2[s] + 0.1[s]*[Simulation].RunIndex(2)'
```

With these inputs, a total of six runs would be performed with the run indices incremented in the following sequence: 1-1, 1-2, 2-1, 2-2, 3-1, 3-2. The notation i-j indicates run indices RunIndex(1) = i and RunIndex(2) = j.

It is not necessary to perform all the simulation runs defined by the run indices. The Simulation keywords StartingRunNumber and EndingRunNumber can be used to determine the runs that will be performed.

Run indices are related to the run number by a mathematical equation that performs the necessary transformation. In the example given above, the RunNumber increases from 1 to 6, at the same time as the run indices increase from 1-1 to 3-2. Run numbers and run indices in the i-j-…-k notation can be used interchangeably for the StartingRunNumber and EndingRunNumber inputs.

Table 13-6 Simulation Keywords for Multiple Runs

Keyword	Description
RunIndexDefinitionList	Defines the number of run indices and the maximum value N for each index. When making multiple runs, each index will be iterated from 1 to N starting with the last index. One run will be executed for every combination of the run index values. For example, if three run indices are defined with ranges of 3, 5, and 10, then at total of 3*5*10 = 150 runs will be executed.
StartingRunNumber	The first run number to be executed. The value can be entered as either an integer or as the equivalent combination of run indices. For example, if there are three run indices with ranges of 3, 5, and 10, then run number 22 can be expressed as 1-3-2 because 22 = (1-1)*5*10 + (3-1)*10 + 2.
EndingRunNumber	The last run number to be executed. The value can be entered as either an integer or as the equivalent combination of run indices. For example, if there are three run indices with ranges of 3, 5, and 10, then run number 78 can be expressed as 2-3-8 because 78 = (2-1)*5*10 + (3-1)*10 + 8.

When making multiple runs, the output reports for the runs are appended one after another in a single file. When large numbers of runs are to be made, it is best to use the RunOutputList keyword for Simulation to specify the key outputs needed to analyse the results from the runs. These outputs are collected in a single output file <configuration file name>.dat, with one row for each simulation run that was performed. The file is tab delimited and can be imported directly into Excel for analysis. The RunOutputList report can be generated with or without the main report, as specified by the PrintReport keyword.

13.6 Display Models

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 and TextModel objects 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, colour, etc.) is determined by the TextModel. In this case, the TextModel plays the same role as a text style in word processing software. Users 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 DisplayModel that can be used for this purpose. Duplicate an existing DisplayModel by right-clicking its entry in the Object Selector and selecting Duplicate, then edit its keyword values and name as necessary.

A selection of DisplayModels can be found in the Display Models Palette in the Object Selector. Each type of DisplayModel is intended for specific types of DisplayEntities.

Table 13-7 Types of DisplayModel

DisplayModel	Description	Usage
ColladaModel	An imported 3D object	DisplayEntity and all its sub-classes.
ImageModel	An imported picture.	DisplayEntity and all its sub-classes.
TextModel	A text style (font, colour, etc.).	Text object and its sub-classes.
PolylineModel	Style for a series of line segments connected by nodes (width and colour)	Selected sub-classes of DisplayEntity such as EntityConveyor and EntityDelay.
ArrowModel	Similar to PolylineModel except that the last line terminates in an arrowhead.	Arrow object.
GraphModel	Formatting data for a graph.	Graph object.
ShapeModel	A flat geometric object such as a circle or rectangle	DisplayEntity and all its sub-classes.

All of these various DisplayModel objects have the same Basic Graphics keywords that control optional rendering and scaling at different drawing ranges.

Table 13-8 DisplayModel Common Inputs

Keyword	Description
VisibleViews	A list of Views for which this DisplayModel 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.

13.6.1 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 (see the Display Models Palette section).

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

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 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.

Table 13-9 ColladaModel Key Inputs

Keyword	Description
ColladaFile	A file path to the DAE, OBJ, 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.

13.6.2 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.

Table 13-10 ImageModel 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.

13.6.3 TextModel

TextModel objects specify the general appearance of Text objects and of overlay objects that display text, such as 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.

Table 13-11 TextModel Key Inputs

Keyword	Description
FontName	The font face to be used for this TextModel, selectable via a drop-down menu.
FontColour	The colour of the text, defined by a colour 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 colour keyword or RGB values.
DropShadowOffset	The spacing between the text and its drop shadow, specified in (x, y, z) coordinates.

13.6.4 Polyline Model

The PolylineModel is used to determine the graphics for entities that appear as a single-segment 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.

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

13.6.5 ArrowModel

The ArrowModel is used to determine the graphics for the Arrow object. It has no inputs other than the standard ones for DisplayModels.

13.6.6 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.

Table 13-12 GraphModel 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-axis 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, colour, 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, colour, and style for the x-axis 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, colour, and style for the labels next to the x-axis and y-axis tick marks. The dropshadow settings are ignored. Black Verdana text is used if this input is left blank.
GraphColor	The colour of the Graph background.
BackgroundColor	The colour of the outer pane background.
BorderColor	The colour of the Graph border.

13.6.7 ShapeModel

ShapeModel objects are used to display a 2D object whose geometry and colour can adjusted.

Table 13-13 ShapeModel Key Inputs

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

13.7 Editing the Configuration File

The easiest way to create a simple model in JaamSim is to use the Graphical User Interface. For more complex models, it is often easier to create or edit the configuration file (CFG file) in a text editor. The configuration file is saved in plain text and has been designed be human readable.

There are many advantages to a readable input file in plain text:

- Complex models often have thousands of inputs. The only way to have confidence in the accuracy of the simulation results is to perform a careful audit of the model inputs by someone who is not part of the project team.
- Standard software for change control such as GIT can be used to track model inputs. With two or more people a project team, it can be difficult to ensure that everyone is working on the latest version of the model. Even with only one simulation analyst, it can be challenging to document all the changes to a model and the reason for each change.
- Software for performing simulation experiments and optimisation can be developed by third-parties in other programming languages such as Python. There is no need for these tools to be built into the simulation software itself, causing clutter in the user interface and locking users into a restricted set of options.

We recommend the use of Notepad++, an open-source editor available for download at www.notepad-plus-plus.org.

13.7.1 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.

13.7.2 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.

13.7.3 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 braces are used for keywords that accept multiple input values.

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

13.7.4 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 when only a few inputs are varied across many simulation runs. Include statements can be used to 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.

13.7.5 Groups

Group objects bundle multiple objects together to simplify inputs. Instead of referring to a long list of objects, a single Group can be 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.

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.

Table 13-14 Group Key Inputs

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

The following example demonstrates the use of Groups:

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

In this example, a Group of three Arrow objects is created and each Arrow is set to the colour black.

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

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

13.7.6 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.

13.8 Defining a New Unit

In addition to the predefined units in JaamSim, new units can be specified using a Unit object. A new Unit object can be created by entering the appropriate Define statement in the configuration file. For example if a new TimeUnit called 'Fortnight' is required, then the define statement would be:

```
Define  TimeUnit  {  Fortnight  }
```

All Unit objects have the same input keyword:

Table 13-15 Unit 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.

For example, for the 'Fortnight' time unit (two weeks) defined above, the ConversionFactorToSI keyword should be set as follows:

```
Fortnight  ConversionFactorToSI  (  1209600  1  }
```

Once the new unit is defined in this way, it can be used with any input requires that type of unit.

13.9 Launching JaamSim from the Command Line

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

```
jaamsim.exe config1.cfg -tags
```

or, when using the .jar file:

```
java -jar jaamsim.jar config1.cfg -tags
```

Here, config1.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:

Table 13-16 Batch Mode Run Tags

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).
-s or -script	Directs JaamSim to accept configuration file input piped to JaamSim through standard-in and to direct its output specified by the RunOutputList keyword to standard-out. The .jar file (jaamsim.jar) must be used with this feature, not the executable (jaamsim.exe).

It is also possible to load two or more configuration files into a single model using the following command:

```
jaamsim.exe config1.cfg config2.cfg -tags
```

or,

```
java -jar jaamsim.jar config1.cfg config2.cfg -tags
```

Multiple 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.

JaamSim can be interact with other software packages using the -s (script) tag. For example, the following command instructs JaamSim to load the configuration file config.cfg and then accept additional configuration file inputs from program1. The outputs specified by the RunOutputList keyword for Simulation are then directed as inputs to program2.

```
program1.exe | java -jar JaamSim.jar config.cfg -s -b | program2.exe
```

At present, the script tag is supported only for the .jar file version of JaamSim - it does not work correctly with the .exe file version.

Note that “java -jar” must be used in this command for standard-in and standard-out to be connected correctly to JaamSim. Without the "java -jar" portion of the command, the java virtual machine sets both standard-in and standard-out to null.

13.10 ScriptEntity

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

Table 13-17 ScriptEntity Key Inputs

Keyword	Description
Script	Path to the file containing the scripting instructions to be loaded

The script file contains sets of model inputs preceded by the Time keyword under 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 toggle 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.

Table 13-18 Key Inputs for .scr file

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

For 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 }
```

14 Named Colours

This section provides the names and respective RGB values for the pre-defined colours built into JaamSim.

Colour Name	Colour	R	G	B
lavenderblush		255	240	245
pink		255	192	203
lightpink		255	182	193
palevioletred		219	112	147
hotpink		255	105	180
deeppink		255	20	147
violetred		208	32	144
mediumvioletred		199	21	133
raspberry		135	38	87
thistle		216	191	216
plum		221	160	221
orchid		218	112	214
violet		238	130	238
magenta		255	0	255
purple		128	0	128
mediumorchid		186	85	211
darkorchid		153	50	204
darkviolet		148	0	211
blueviolet		138	43	226
indigo		75	0	130
mediumpurple		147	112	219
lightslateblue		132	112	255
mediumslateblue		123	104	238
slateblue		106	90	205
darkslateblue		72	61	139
ghostwhite		248	248	255
lavender		230	230	250
blue		0	0	255
darkblue		0	0	139
navy		0	0	128
midnightblue		25	25	112
cobalt		61	89	171

Colour Name	Colour	R	G	B
chocolate		210	105	30
rawsienna		199	97	20
sienna		160	82	45
brown		138	54	15
lightsalmon		255	160	122
darksalmon		233	150	122
salmon		250	128	114
lightcoral		240	128	128
coral		255	114	86
tomato		255	99	71
orangered		255	69	0
red		255	0	0
crimson		220	20	60
firebrick		178	34	34
indianred		176	23	31
burntumber		138	51	36
maroon		128	0	0
sepia		94	38	18
white		255	255	255
gray99		252	252	252
gray98		250	250	250
gray97		247	247	247
gray96		245	245	245
gray95		242	242	242
gray94		240	240	240
gray93		237	237	237
gray92		235	235	235
gray91		232	232	232
gray90		229	229	229
gray89		227	227	227
gray88		224	224	224
gray87		222	222	222

Colour Name	Colour	R	G	B
royalblue		65	105	225
cornflowerblue		100	149	237
lightsteelblue		176	196	222
lightslategray		119	136	153
slategray		112	128	144
dodgerblue		30	144	255
aliceblue		240	248	255
powderblue		176	224	230
lightblue		173	216	230
lightskyblue		135	206	250
skyblue		135	206	235
deepskyblue		0	191	255
peacock		51	161	201
steelblue		70	130	180
darkturquoise		0	206	209
cadetblue		95	158	160
azure		240	255	255
lightcyan		224	255	255
paleturquoise		187	255	255
cyan		0	255	255
turquoise		64	224	208
mediumturquoise		72	209	204
lightseagreen		32	178	170
manganeseblue		3	168	158
teal		0	128	128
darkslategray		47	79	79
turquoiseblue		0	199	140
aquamarine		127	255	212
mintcream		245	255	250
mint		189	252	201
seagreen		84	255	159
mediumspringgreen		0	250	154
springgreen		0	255	127
emeraldgreen		0	201	87
mediumseagreen		60	179	113
cobaltgreen		61	145	64
darkseagreen		143	188	143

Colour Name	Colour	R	G	B
gray86		219	219	219
gray85		217	217	217
gray84		214	214	214
gray83		212	212	212
gray82		209	209	209
gray81		207	207	207
gray80		204	204	204
gray79		201	201	201
gray78		199	199	199
gray77		196	196	196
gray76		194	194	194
gray75		191	191	191
gray74		189	189	189
gray73		186	186	186
gray72		184	184	184
gray71		181	181	181
gray70		179	179	179
gray69		176	176	176
gray68		173	173	173
gray67		171	171	171
gray66		168	168	168
gray65		166	166	166
gray64		163	163	163
gray63		161	161	161
gray62		158	158	158
gray61		156	156	156
gray60		153	153	153
gray59		150	150	150
gray58		148	148	148
gray57		145	145	145
gray56		143	143	143
gray55		140	140	140
gray54		138	138	138
gray53		135	135	135
gray52		133	133	133
gray51		130	130	130
gray50		127	127	127

Colour Name	Colour	R	G	B
honeydew		240	255	240
palegreen		152	251	152
lawngreen		124	252	0
greenyellow		173	255	47
limegreen		50	205	50
forestgreen		34	139	34
sapgreen		48	128	20
green		0	128	0
darkgreen		0	100	0
darkolivegreen		85	107	47
olivedrab		107	142	35
olive		128	128	0
ivory		255	255	240
lightyellow		255	255	224
lightgoldenrodyellow		250	250	210
cornsilk		255	248	220
lemonchiffon		255	250	205
beige		245	245	220
yellow		255	255	0
khaki		240	230	140
lightgoldenrod		255	236	139
palegoldenrod		238	232	170
darkkhaki		189	183	107
banana		227	207	87
gold		255	215	0
goldenrod		218	165	32
darkgoldenrod		184	134	11
brick		156	102	31
floralwhite		255	250	240
seashell		255	245	238
oldlace		253	245	230
linen		250	240	230
antiquewhite		250	235	215
papayawhip		255	239	213
blanchedalmond		255	235	205
eggshell		252	230	201
bisque		255	228	196

Colour Name	Colour	R	G	B
gray49		125	125	125
gray48		122	122	122
gray47		120	120	120
gray46		117	117	117
gray45		115	115	115
gray44		112	112	112
gray43		110	110	110
gray42		107	107	107
gray41		105	105	105
gray40		102	102	102
gray39		99	99	99
gray38		97	97	97
gray37		94	94	94
gray36		92	92	92
gray35		89	89	89
gray34		87	87	87
gray33		84	84	84
gray32		82	82	82
gray31		79	79	79
gray30		77	77	77
gray29		74	74	74
gray28		71	71	71
gray27		69	69	69
gray26		66	66	66
gray25		64	64	64
gray24		61	61	61
gray23		59	59	59
gray22		56	56	56
gray21		54	54	54
gray20		51	51	51
gray19		48	48	48
gray18		46	46	46
gray17		43	43	43
gray16		41	41	41
gray15		38	38	38
gray14		36	36	36
gray13		33	33	33

Colour Name	Colour	R	G	B
moccasin		255	228	181
navajowhite		255	222	173
wheat		245	222	179
peachpuff		255	218	185
tan		210	180	140
burlywood		222	184	135
melon		227	168	105
sandybrown		244	164	96
cadmiumyellow		255	153	18
carrot		237	145	33
orange		255	128	0
flesh		255	125	64
cadmiumorange		255	97	3

Colour Name	Colour	R	G	B
gray12		31	31	31
gray11		28	28	28
gray10		26	26	26
gray9		23	23	23
gray8		20	20	20
gray7		18	18	18
gray6		15	15	15
gray5		13	13	13
gray4		10	10	10
gray3		8	8	8
gray2		5	5	5
gray1		3	3	3
black		0	0	0

15 Example Configuration File

The configuration file for the Basic Example described in Section 3 is given below.

```
RecordEdits

Define ColladaModel { Grid100x100  Axis }
Define DisplayEntity { XY-Grid  XYZ-Axis }
Define View { View1 }
Define TextModel { TitleTextModel  ClockTextModel }
Define OverlayText { Title }
Define OverlayClock { Clock }
Define SimEntity { Proto }
Define EntityGenerator { Gen }
Define EntityConveyor { GenToServ  ServToSink }
Define Server { Serv }
Define EntitySink { Sink }
Define Queue { ServQueue }
Define ExponentialDistribution { GenIATDist }

GenIATDist UnitType { TimeUnit }

Simulation Description { 'Simulation run control inputs' }
Simulation SnapToGrid { TRUE }
Simulation RealTime { TRUE }
Simulation RealTimeFactor { 2048 }
Simulation PauseTime {  }
Simulation ShowModelBuilder { TRUE }
Simulation ShowObjectSelector { TRUE }
Simulation ShowInputEditor { TRUE }
Simulation ShowOutputViewer { TRUE }
Simulation ShowPropertyViewer { FALSE }
Simulation ShowLogViewer { FALSE }

Grid100x100 ColladaFile { <res>/shapes/grid100x100.dae }

XY-Grid Description { 'Grid for the X-Y plane (100 m x 100 m)' }
XY-Grid Size { 100  100  m }
XY-Grid DisplayModel { Grid100x100 }
XY-Grid Movable { FALSE }

Axis ColladaFile { <res>/shapes/axis_text.dae }

XYZ-Axis Description { 'Unit vectors' }
XYZ-Axis Alignment { -0.4393409  -0.4410096  -0.4394292 }
XYZ-Axis Size { 1.125000  1.1568242  1.1266404  m }
XYZ-Axis DisplayModel { Axis }
XYZ-Axis Show { FALSE }
XYZ-Axis Movable { FALSE }

View1 Description { 'Default view window' }
View1 ViewCenter { -0.299610  -2.582932  2.866546  m }
View1 ViewPosition { -0.299610  -2.582933  11.526800  m }
View1 ShowWindow { TRUE }
View1 SkyboxImage { <res>/images/sky_map_2048x1024.jpg }

TitleTextModel Description { 'Text style for the Title' }
TitleTextModel FontColour { 150  23  46 }
TitleTextModel FontStyle { BOLD }

ClockTextModel Description { 'Text style for the Clock' }
```

```
ClockTextModel FontColour { 51  51  51 }
ClockTextModel FontStyle { ITALIC }

Title Description { 'Title for the simulation model' }
Title TextHeight { 18 }
Title Format { 'Getting Started with JaamSim' }
Title Position { 0.000000  0.000000  0.000000  m }
Title DisplayModel { TitleTextModel }
Title ScreenPosition { 15  15 }

Clock Description { 'Simulation date and time (no leap years or leap seconds)' }
Clock TextHeight { 10 }
Clock StartingYear { 2014 }
Clock DateFormat { 'yyyy-MMM-dd HH:mm:ss.SSS' }
Clock DisplayModel { ClockTextModel }
Clock ScreenPosition { 15  15 }
Clock AlignBottom { TRUE }

Proto Position { -4.400000  -0.600000  0.000000  m }
Proto Alignment { 0.0  0.0  -0.5 }

Gen NextComponent { GenToServ }
Gen InterArrivalTime { GenIATDist }
Gen PrototypeEntity { Proto }
Gen Position { -3.500000  -2.500000  0.000000  m }

GenToServ NextComponent { Serv }
GenToServ TravelTime { 1  s }
GenToServ Position { -2.500000  -2.500000  0.000000  m }
GenToServ Points { {  -2.500  -2.500  0.000  m  }  {  -1.500  -2.500  0.000  m  } }

Serv NextComponent { ServToSink }
Serv WaitQueue { ServQueue }
Serv ServiceTime { 1  s }
Serv Position { -0.500000  -2.500000  0.000000  m }

ServToSink NextComponent { Sink }
ServToSink TravelTime { 1.5  s }
ServToSink Position { 0.600000  -2.500000  0.000000  m }
ServToSink Points { {  0.600  -2.500  0.000  m  }  {  1.600  -2.500  0.000  m  } }

Sink Position { 2.500000  -2.500000  0.000000  m }

ServQueue Position { -0.500000  -4.500000  0.000000  m }

GenIATDist RandomSeed { 1 }
GenIATDist MaxValue { 10  s }
GenIATDist Mean { 2  s }
GenIATDist Position { -2.500000  -0.500000  0.000000  m }
```