Extend™

Performance modeling for decision support

Demonstration version of

Extend 3.0, Extend+BPR 3.0, and Extend+Manufacturing 3.0

Imagine That!™

Imagine That, Inc. • 6830 Via Del Oro, Suite 230 • San Jose, CA 95119-1353 USA
Telephone 408-365-0305 • FAX 408-629-1251

Contents

Introduction

Extend graphically simulates and validates your ideas quickly and easily. This advanced simulation tool for decision support enables you to develop models of real-life processes in a wide variety of fields. Use it to create models from building blocks, explore the processes involved, and see how they relate. Then change assumptions to arrive at an optimum solution. Extend and your imagination are all you need to create professional models that meet all your business and academic needs.

About this demo

This demo is a limited working version of the Extend, Extend+BPR, and Extend+Manufacturing simulation packages. Although you cannot print models or save any of the models you build as you can in a full version of the program, in this demo version you can:

- Run simulations.
- Change parameters within blocks.
- Build models that contain up to 25 blocks.
- Build new blocks and save them in libraries.

This demo application is a universal version that works on all machines, including the Power Macintosh. The full version of Extend ships with both a universal version of the program and an FPU (floating point unit or math co-processor) version. The FPU version runs faster than the universal version. However, the FPU version only runs on computers that have FPUs (such as the MacII, MacIIfx, or Quadra).

The full versions of the Extend family packages allow you to save changes, print, and build models of any size, restricted only by the limitations of your system. The full versions also include additional sample models, extensive libraries, and (of course) complete documentation.

The Extend demo requires a Macintosh computer with a hard disk drive with at least 4 megabytes of RAM. If you are using System 7, a color monitor, or are running large models, you may need more RAM. Extend family packages are MultiFinder, A/UX, and System 7 compatible. While these programs support System 7, they also work fine in System 6 (you must be running Macintosh operating system 6.0.7 or later).

This demo is a limited version of Extend, distributed for purposes of evaluating the Extend family packages. It has been assumed the user has a basic understanding in the operation of a Macintosh. If you have any questions during or after examining this demo, please call Imagine That, Inc. at 408-365-0305 or fax us at 408-629-1251.

Modeling and simulation defined

A model is a logical description of how a system performs. Simulation involves designing and building a model of a system and carrying out experiments on it. For example, the board game Monopoly is a model of a system in which you strategically buy and sell properties and buildings. When you play Monopoly, you are *simulating* the real system or the purchasing process. Simulation with Extend means that instead of interacting with a real system, you create a model which corresponds to it in certain aspects.

Models can be used to describe how a real-world activity will perform. One of the principal benefits of using a model to replicate a process is that you can begin with a simple approximation of the process and gradually refine it as your understanding of the process improves. This “stepwise refinement” enables you to achieve good approximations of very complex problems surprisingly quickly. As you add refinements, your model becomes more and more accurate.

Models enable you to test your hypotheses without having to carry them out, saving you thousands, even hundreds of thousands of dollars! For example, if you are a factory designer, you can use Extend to simulate how a new factory layout works without actually setting up the factory.

What can simulation do for me?

Simulation is a powerful tool for decision support. It provides a method for checking your understanding of the world around you and helps you produce better results faster. Simulating with Extend enables you to:

- Predict the course and results of certain actions.
- Understand why observed events occur.
- Identify problem areas before implementation.
- Explore the effects of modifications.
- Confirm that all variables are known.
- Evaluate ideas and identify inefficiencies.
- Gain insight and stimulate creative thinking.
- Communicate the integrity and feasibility of your plans.
- Comprehend interactions between system components.

Extend models

Build Extend models simply by connecting pre-built blocks together. This powerful simulator feature means you can build models quickly, without entering equations or programming. When you need more flexibility than Extend's blocks allow, you can easily access the built-in ModL language to create your own custom blocks. Because Extend is both a simulator and a simulation language, it can be used at three levels:

- Build models using the extensive libraries of blocks supplied with Extend. Much like using building blocks, Extend's blocks allow you to quickly build models without any programming.
- If the blocks supplied with Extend do not meet all your needs or if you want to combine the function of several blocks into one, you can enter equations (including control statements) into the Equation block in the

Generic library.

- For full power and flexibility, you can use Extend's built-in scripting environment to build your own blocks with custom icons, dialogs, animation, and help.

Overview of features

With Extend, you get all the ease-of-use and capability you need to quickly model any system or process.

- A full array of building blocks that allow you to build models rapidly
- Animation of the model for enhanced presentation
- A customizable graphical interface showing the relationships in the system you are modeling
- Hierarchical modeling to make even complex systems easy to build and understand
- Sensitivity analysis so you can investigate how a parameter change impacts the pattern of behavior for the entire model.
- A full-featured, built-in authoring environment for simplifying interaction with models
- Dialogs and notebooks for changing model values, so you can quickly try out assumptions and test your model
- Customizable reports for in-depth analysis and presentation
- The ability to adjust settings (or parameters) while the simulation is running
- Direct interfacing with Excel through Apple Events as your model runs

- Full connectivity with other programs and platforms through Copy/Paste, import/export, System 7 Publish/Subscribe, text files, XCMDs, and so on
- Monte Carlo, batch-mode, and design-of-experiments for optimizing systems

Extend's integrated environment combines all the advanced features of the most powerful simulation systems.

Multi-purpose simulation—Extend is a multi-application environment so you can model continuous, discrete event, linear, and non-linear dynamic systems.

Built-in programming environment and dialog editor—Modify Extend's blocks, build your own for specialized applications, and add customized animation to your model with Extend's language and dialog editor.

Library based—The blocks you build can be saved in libraries and easily reused in other models.

Over 300 built-in functions—Directly access functions for integration, statistics, queuing, animation, IEEE math, matrix, sounds, arrays, FFT, debugging, XCMDs, string and bit manipulation, I/O, and so on. You can also define your own functions.

Message sending capabilities—Blocks can send messages to other blocks interactively for subprocessing.

Global functions---Write your own library of functions in a block and call them from other blocks in the model using block message calls.

Sophisticated data-passing capabilities—Pass values, arrays, or structures composed of arrays.

Huge models—Model size is limited only by the limits of your system.

On-line help

Extend's on-line help is accessible through the Help command in the Apple menu (marked with an □) and is available any time you are using Extend.

The help from any libraries which are open is automatically included with Extend's on-line help topics. Help can also be accessed through a button labeled “Help” in each block’s dialog.

Chapter 1: Basic Model Operation

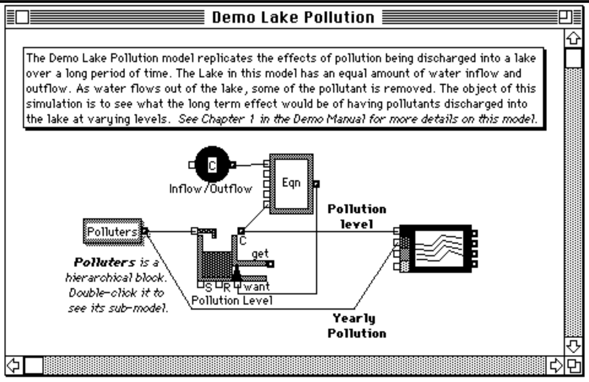
In this chapter you will open a model, run it, and familiarize yourself with some of the features that have helped put Extend at the top of its class.

Opening a model

After installing the Extend Demo version, open the folder labeled “Extend 3.0 Demo” and double-click on the “Extend Demo” icon. Then, click once on the start-up screen to start the Extend Demo application. An “Untitled” window appears. We’re going to introduce you to Extend by looking at a good general model called the Demo Lake Pollution model. This model replicates the effects of pollution being discharged into a lake over a long period of time. The Lake in this model has an equal amount of water inflow and outflow. As water flows out of the lake, some of the pollutant is removed. The object of this simulation is to see what the long term effect would be of having pollutants discharged into the lake at varying levels.

There are two ways to open the Demo Lake Pollution model.

- In the File menu, scroll down to Open model... In the window that appears, open the Tutorial folder. Then, select the Demo Lake Pollution file and open it.
- Or, use the Finder. Double-click on the Tutorial folder in the Extend 3.0 Demo folder. Then, find the Demo Lake Pollution model, and open it by double-clicking on it. This model appears on your screen:



Demo Lake Pollution

The Demo Lake Pollution model replicates the effects of pollution being discharged into a lake over a long period of time. The Lake in this model has an equal amount of water inflow and outflow. As water flows out of the lake, some of the pollutant is removed. The object of this simulation is to see what the long term effect would be of having pollutants discharged into the lake at varying levels. *See Chapter 1 in the Demo Manual for more details on this model.*

Polluters *is a hierarchical block. Double-click it to see its sub-model.*

Demo Lake Pollution model

This is how a typical model window looks. The window that the model is in is just like other Macintosh document windows, allowing you to scroll, resize, zoom, and close.

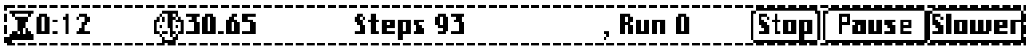
The Demo Lake Pollution model was built using the blocks supplied in Extend's Generic and Plotter libraries. It illustrates features such as animation, hierarchy, notebooks, drawing tools, and cloning dialog items. (These special features are discussed in more detail in Chapter 3.)

Each icon, or block, in the model represents some aspect of the situation being modeled. The tank-like container in the lower center of the model keeps track of the current level of pollution in the lake. There is a source of pollution (the “Polluters” located at the left of the model) and a calculation which determines the outflow of water and pollutant from the lake (the two blocks situated upper center). As seen by the lines connected to the Plotter on the right side of the model, this model is constructed to graph both the lake’s pollution level over time and the amount of pollutant being dumped each year.

Running a model

Before investigating the parts of this model, let's run it to see what happens. Just choose Run Simulation from the Run menu (you may also use the keyboard shortcut, ⌘-R). That's all there is to it. When the simulation ends, as specified by default in the Preferences command, Extend will beep.

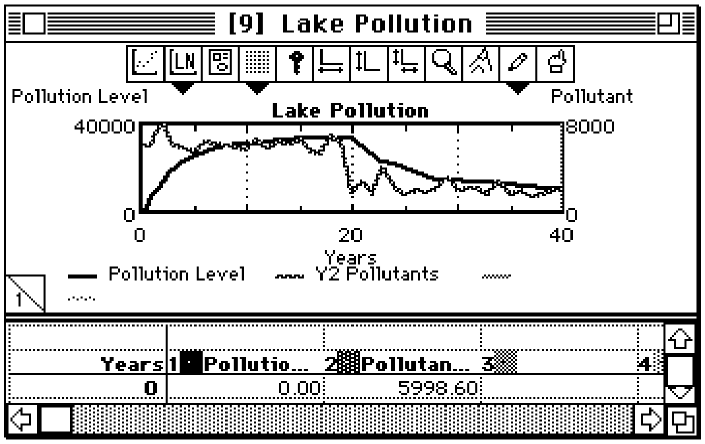
While the simulation runs, Extend shows a small status bar at the bottom of your screen:



Status bar

This bar shows you the time left in the simulation, the current simulation time, the number of simulation steps or calculations, and the number of the Run. These values are determined by the entries in the Simulation Setup dialog, as described in Chapter 2. You can also use the buttons on the right end of the status bar to stop, pause, or slow down (to view animation more closely) a simulation in progress.

When you run a simulation, Extend shows the plotter on the screen (if there is one in the model). The plotter for the Demo Lake Pollution model looks like:



Years	1 Pollutio...	2 Pollutan...	3	4
0	0.00	5998.60		

Demo Lake Pollution plotter

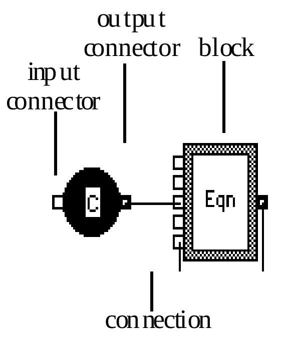
This is an example of a plot with two lines and two value axes. The blue (solid black) line and the left axis show the total amount of pollution in the lake; the red (dotted gray) line and the right axis show the amount added each year. Note that the scales for the two lines are different.

You can observe the values by moving the cursor along the traces in the plotter. The corresponding values are displayed at the top of the plotter's data table. The bottom of the window shows the data points which produce the line. You can scroll down this list to see the numerical values for the two lines.

This simulation shows what happens to the lake over a 40 year period. In the plotter, you can see that the pollution level increases for about 20 years, then starts to decrease. The amount of pollutant dumped into the lake also decreases after 20 years. The settings in the blocks that make up this model determine what happens in the simulation. (Chapter 3 describes how to run this model showing animation.)

Model basics

Now that you have run the Demo Lake Pollution model, look at the components of the model more closely. You can see from the Demo Lake Pollution example that there are many parts in Extend models. The most important parts of a model are the *blocks* , the *libraries* that hold the blocks, the *connectors* on each block, the *connections* between the blocks, and the *dialogs* associated with each block.



Parts of a model

Extend models are built by connecting blocks together in a logical sequence. The simulation itself is a series of calculations and actions which proceed along the path of the connections over and over. Each calculation of the entire model is called a *step* or *event*. After one *step*, the simulation repeats itself. You tell Extend how long (in simulation time) you want your model to run. It can run for as long as you want. You can also change variables and repeat the entire simulation as many times as you want to explore alternatives.

Blocks

Blocks are the basic building components of Extend models. You can think of an Extend block like a block in a block diagram. Each block in the diagram is used to represent a specific function or process within a system. However, unlike this typical block diagram, in an Extend block information comes in and is processed by the program that is in the block. The block then transmits information out to the next block in the simulation.

Each block in an Extend model has a unique icon that shows how that block relates to the process it represents and to other blocks in the model. There are five blocks visible in the Demo Lake Pollution model (you will see later that there are four more blocks inside the hierarchical “Polluters” block). You can have more than one copy of the same block in a model. If you build your own blocks or build hierarchical blocks (discussed later), you can put custom icons on your blocks. You can do this using Extend’s drawing program or by copying pictures into Extend.

There is nothing fundamentally different about the structure of the different blocks. Any block may create, modify, or present information, and many blocks perform more than one of these functions. Chapter 4 describes block internals in much more detail.

Libraries

Libraries are repositories for blocks. The entire *definition* for a block (its program, icon, dialog, and so on) is stored in the library. If you change the definition of a block in a library, all models that use that block are automatically updated. For example, if you change a block’s icon, all instances of that block in existing models are automatically updated with the new icon. Because all Extend blocks come with source code (with the exception of the blocks in the BPR and Manufacturing libraries in this demo), you can change their definition to suit your need, as discussed in Chapter 4.

When you add a block to a model, the block itself is not put into the model. Instead, a reference to the block information in the library is placed there. The data that you enter in the block's dialog in the model is stored with the model, not in the library. This allows each instance of a block in a model to be configured differently. For instance, you can have two Constant blocks in a model with two different constant values entered in their dialogs.

Because Extend saves the names of the blocks in a model as well as the libraries' locations, it automatically opens the needed libraries when you open a model. For example, when you opened the Demo Lake

Pollution model, Extend opened two libraries: Demo Generic Lib and Demo Plotter Lib. To verify this, pull down the Library menu and note the two libraries at the bottom of the menu.

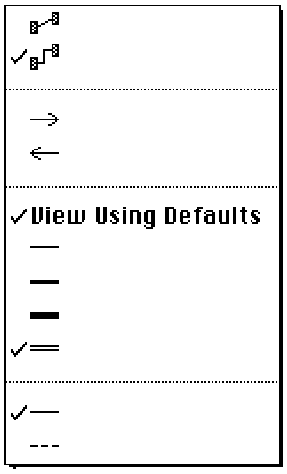
To find out what blocks are in the Demo Generic Library, go to the Library menu and scroll down to Demo Generic Lib. Move your mouse towards the arrow on the right to access a hierarchical menu that contains a list of all the blocks in the library. To see what each block looks like, select Open Library Window. Release the mouse and a window appears along the left of your screen that shows each block and its name. Please note, the libraries in this Demo version contain only a sampling of the blocks found in the full versions of Extend, Extend+BPR, and Extend+Manufacturing.

Connectors

Blocks in Extend have *pre-defined* input and output connectors, the small squares attached to the sides of the block. As you might expect, information flows into a block at input connectors (the white squares) and out of the block at output connectors (the black squares). A block might have many input and/or output connectors; some blocks have none. Each connector has a unique meaning to the block, as described in the block's Help. Having *pre-defined* connectors means that Extend blocks know what to do with the data when you connect blocks together, so you don't have to enter equations. Connector details are in Chapter 2.

Connection lines

Connection lines hook blocks together. These lines (called *connections*) show the flow of information from block to block through the model. They can be formatted by using the Connection Lines command from the Model menu. When building models, select this command and choose an option from its hierarchical menu. This allows you to change the line style of the connections when you draw a new connection line between blocks. For example, you can draw lines of various thicknesses or hollow lines. Or you can draw lines that always use right angles (these are the only lines that can have arrows or be dashed lines).

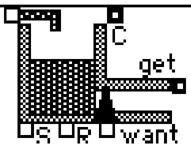


Connection Lines command

Dialogs

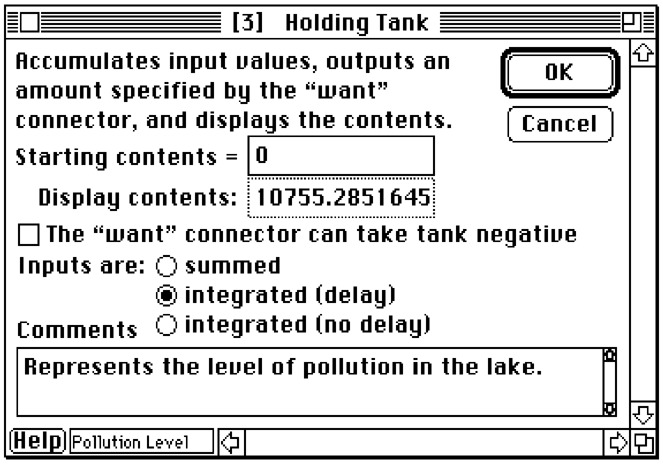
Every non-hierarchical block has a dialog associated with it. Dialogs are used to enter values and settings for running your simulations. Dialogs can also provide information about the status of the model. Extend's dialogs act just like dialogs in other Macintosh programs so it is easy to enter numbers and values. To open a block's dialog, simply double-click on the block's icon.

For example, look in the Holding Tank block that represents the pollution in the Lake. The icon is:



Holding Tank icon

When you double-click the icon, the dialog for that block opens:


[3] Holding Tank

Accumulates input values, outputs an amount specified by the "want" connector, and displays the contents.

Starting contents = 0

Display contents: 10755.2851645

☐ The "want" connector can take tank negative

Inputs are: summed [✓ integrated (delay)] integrated (no delay)

Comments

Represents the level of pollution in the lake.

Pollution Level

Holding Tank dialog

This dialog lets you make settings, such as setting the starting contents and indicating how the inputs are added. The "Display contents" field tells you the current value of the contents (in this case, how much pollutant is currently in the lake). You can get more information about how the block works by clicking on the Help button at the bottom left corner of the dialog.

You can leave dialogs open while you run your model, although this slows down the simulation a bit. Some dialogs report values from the model, so you can use dialogs to show values that you want to watch during the simulation.

Chapter 2: Building a Model

Now that you know how to open and run a model, let's build one. This chapter describes the steps in creating a model with blocks from the libraries that come with Extend.

Model overview

This chapter gives you step by step instructions on building a model that will represent people going to a bank. It shows a simple queue of people waiting for a free teller at the bank. Customers arrive, wait in line until a teller is free, complete their transactions, then leave the bank. The output is a simple plot showing the number of customers processed and the length of the waiting line.

In Extend, this model will not only appear to be simple, but *is* simple. However, try setting it up in a spreadsheet like Excel. No doubt you'll find it very difficult to do. If you do manage to set it up in a spreadsheet, just try changing assumptions. Give up? Now, see how easy it is with Extend!

Note: We've already built this model and provided it as an example in the Tutorial folder. It's the one labeled "Demo Bank Line". But don't cheat! First, follow these instructions and build one yourself, then compare it to the one we've built. You'll see how quickly you can become an Extend expert modeler!

Building the model

To start building a new model, choose New Model from the File menu. Extend opens a new model window. (If you just started Extend, you don't need to use the New Model command because Extend starts with a new, untitled model window.)

We're going to walk you through these four simple steps you'll follow each time you add a block to your model worksheet:

- Open the library, if necessary
- Add the block to the model
- Move the block to its desired position
- Connect it to other blocks

Let’s get building!

Opening the libraries

In order to copy a block into a model, the library in which that block resides must be open. For now, all you need is to have the Demo Discrete Event and Demo Plotter libraries open so that you can get blocks from them.

To open the Demo Discrete Event library:

- Choose Open Library from the Library menu.
- In the dialog, locate and select the “Demo Discrete Event Lib”. Unless you moved it, it is in the Demo Libraries folder in the Extend 3.0 Demo folder .
- Click Open to open the Demo Discrete Event Lib library.

Use the same steps to open the Demo Plotter library, which is also in the Demo Libraries folder.

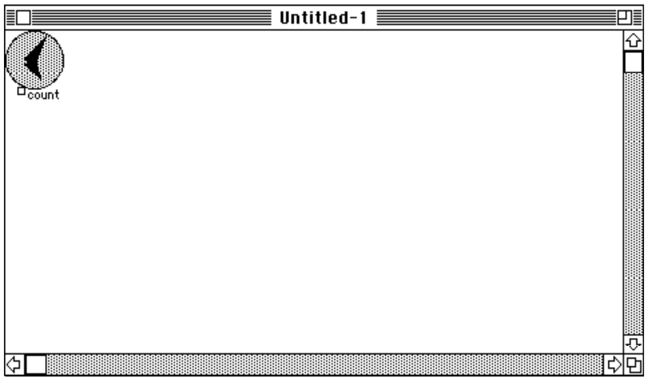
Adding a block to the model

When you add a block to a model, you will follow these steps:

- Click in the Library menu.
- Drag down to the name of the library that holds the desired block. When the library is selected, the names of all the blocks in the library appear in a hierarchical menu to the right.
- Drag to the right and then down the list to the name of the block you want.
- Let go of the mouse button.

This puts a copy of the block in the upper left corner of the window and selects it.

Every discrete event model must have an Executive block positioned to the left of the other blocks in the model. To start constructing the Demo Bank Line model, add an Executive block from the Demo Discrete Event library to the model following the instructions above. The block is selected when you add it to a model; to deselect the block, click anywhere else in the window. Your model worksheet should look like:

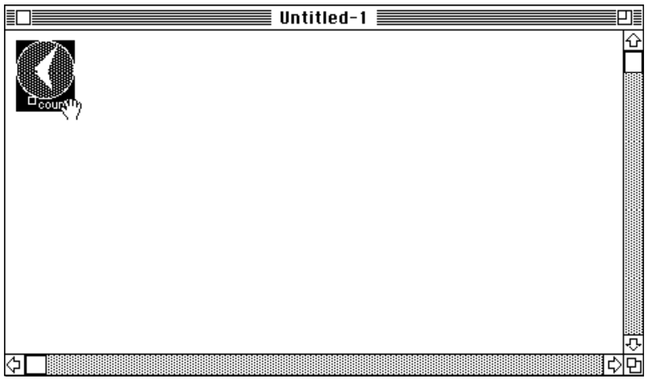


Executive block added

Moving blocks

To move a block, click on the block and drag it with the mouse. When the cursor is over a block, it turns to a drag hand, ☝. Click on the block, drag it to the desired position, and let go. This is just like moving items in drawing programs such as Illustrator, FreeHand, SuperPaint or MacDraw.

Move the Executive block that you just added down a bit in the window:

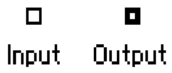


Executive block moved

Connecting blocks

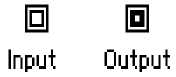
Blocks are hooked together through their connectors by connection lines (the lines that you see between the blocks in the model window). These lines allow information to flow between the blocks. You can think of them like telephone lines connecting one phone to another.

Extend has several types of connectors. Many blocks use value input and output connectors to pass values:



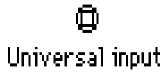
Value input and output connectors

Discrete event blocks use item input and output connectors to pass items:



Item input and output connectors

Universal input connectors can be connected to either value or item connectors.



Universal input connector

In the Demo Lake Pollution model from the previous chapter, the blocks use value input and output connectors to pass values. The blocks we're going to use in this chapter to build our bank line example use the item input and output connectors to pass items; some of these blocks also use value connectors which pass values.

Value connectors provide numeric information about the model. Values are numbers which reflect the state of the model at any particular time, such as the length of a line or the amount of delay. Item connectors pass entities or "items" from one part of the model to another. Because each item is a unique entity, you can assign properties (such as color, quality, or priority) to it, then manipulate it and track it in the model based on those properties. If you want to track information about the individual entities in a model, you use the Discrete Event library. When "View Using Defaults" is selected under the Connection Lines command in the Model menu, connections between value connectors appear as a solid line and connections between item connectors appear as a hollow line.

Output connectors can be connected to more than one input connector. You cannot, however, connect more than one output to a single input. This difference makes sense because the information flowing out of an output connector can be useful in many places, but an input connector can only have one source of information. Blocks that need to have many kinds of input have one input connector for each piece of information.

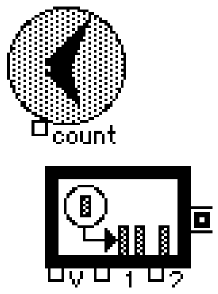
Extend makes sure that you connect the right types of input with the right types of outputs. For example, if you try to connect an item output to a value input, Extend will stop you.

In most cases, in order for a block to pass information to other blocks, it must be connected to those blocks. Connecting the blocks is as easy as playing "connect the dots". Basically, here's the steps you'll follow to connect an output connector of one block (say Block A) to the input connector of another (say block B):

- Move the cursor to the output connector of block A. The cursor changes from an arrow to a technical drawing pen, .
- Click in the output connector, then drag the line to the input connector on block B. You can tell when you are over the connector because the line you draw becomes thicker.
- Let go of the mouse button.

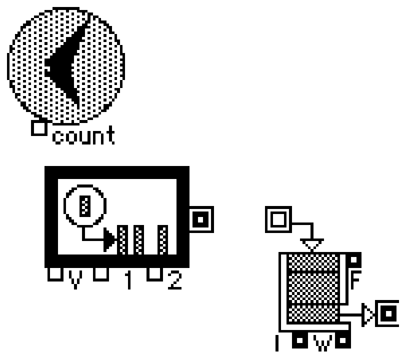
Now, let's actually connect some blocks. First, you'll have to add blocks to the Demo Bank Line model: a Generator block to specify customer arrival rates and a Queue FIFO block to represent the line in which they'll be waiting for a teller. After you add the blocks, we'll show you how you can change their settings to specify model parameters.

Let's start by adding a Generator block. Since Extend always puts new blocks at the last place you clicked in the model window, you can click in the vicinity of where you would like the block to be placed before you add the block. Click on the worksheet to the right of the Executive block and slightly below it. Add a Generator block from the Demo Discrete Event library. (Use the same steps you followed on page 12 when you added the Executive block to your worksheet.) Move it so it looks like:



Position of the first two blocks

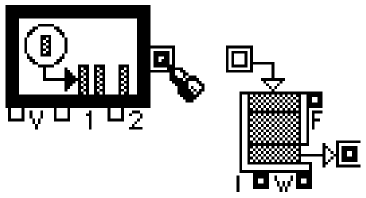
Then add a Queue FIFO block (also from the Demo Discrete Event Lib) to the right of the Generator block:



First three blocks

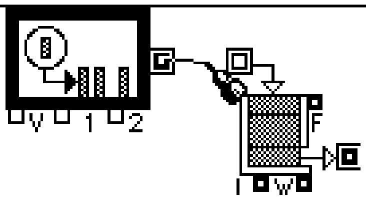
Select "View Using Defaults" under the Connection Lines command in the Model menu. Now connect the output of the Generator block to the top input connector on the Queue FIFO block.

- Move the cursor to the output connector of the Generator block:



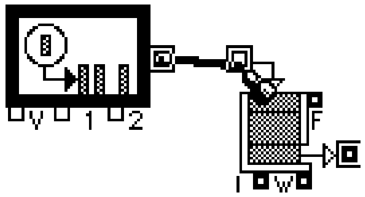
Pen over output connector

- Click in the connector, then drag the line towards the Queue FIFO block:



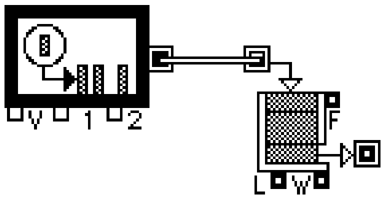
Creating the connection

- Move the cursor to the input connector on the Queue FIFO block. The line becomes thicker.



The thick line

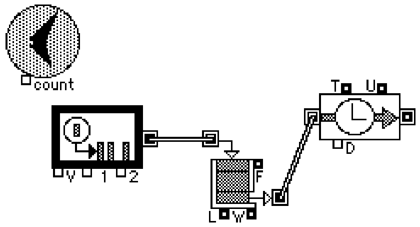
- Let go of the mouse button, but not before you see the line thicken. The line you just made should become hollow to show you the connection was successful. If a connection was not made, the connection line will be a dashed line. If you didn't successfully complete your connection, delete the line you just drew by selecting it (click on it), so that it thickens again (this time indicating that it has been selected), then press the Delete or Backspace key to get rid of it. Then try your connection again.



The connection made

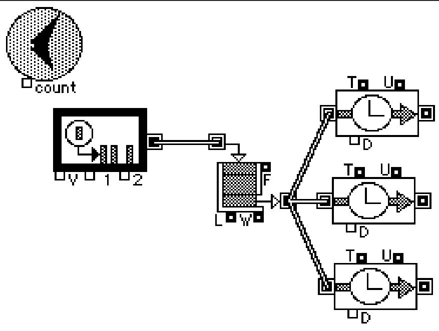
Tellers

In a bank, meeting with a teller takes time. Since you are simulating the way people enter and leave a bank, the teller will delay the customer for the amount of time the transaction activity takes. The Activity Delay block is perfect for this. Add an Activity Delay block (from the Demo Discrete Event library) to the right and above the Queue FIFO block and connect the output of the Queue FIFO to the input on the new block:



Teller added

Our bank has three tellers, so you need to have one Activity Delay block to represent each teller. Either add each block directly from the Demo Discrete Event library or simply duplicate the one already on your worksheet by using either familiar Copy/Paste commands or selecting Duplicate from the Edit menu. Connect the output of the Queue FIFO to the input on each of the new tellers. Remember that you can connect an output to many inputs but not vice versa. Your model should now look like this:

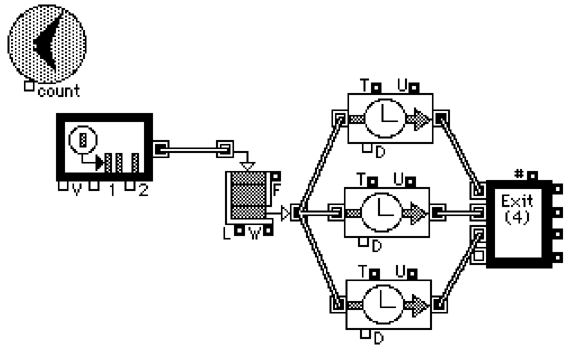


Three tellers added

Exit

You now need a way for the customers to leave the bank. The Exit (4) block counts the customers as they leave the bank after they have finished with the tellers.

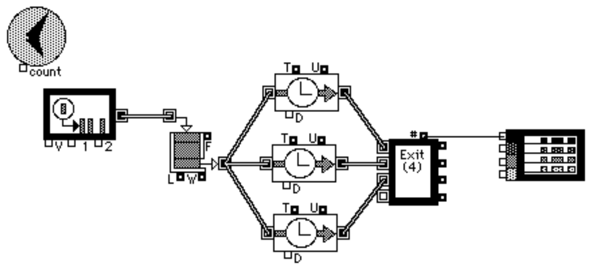
- Add an Exit (4) block from the Demo Discrete Event library to the right of the tellers.
- Connect the output of each teller to separate inputs of the Exit (4) block:



Exit (4) added

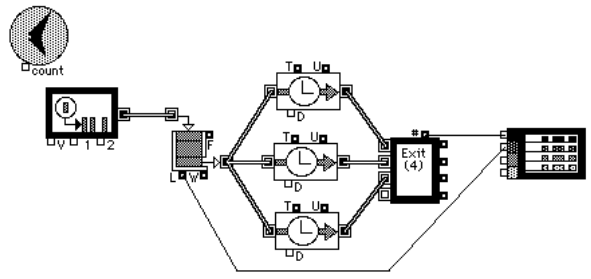
Plotter

- Add a Plotter Discrete Event block to the right side of the model. This block is in the Demo Plotter library.
- Connect the # output connector of the Exit (4) block (that is, the total number of customers who have exited) to the top input of the plotter:



Model with plotter

- Then connect from the L connector on the Queue FIFO to the second input of the plotter:

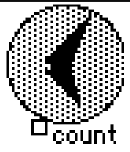
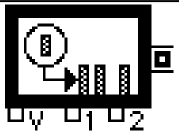
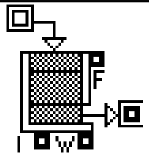
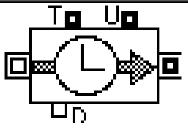
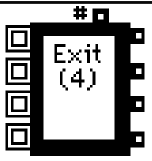
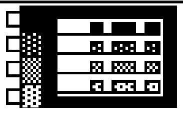


Length connector connected

You do this so that the plotter will also show the length (L) of the waiting line over time.

Examining blocks

Now, all that's left to complete your model is to set the values for it; for instance, to indicate how long each teller takes to process a transaction. Before doing that, let's spend a minute to take a closer look at the blocks you've placed in your model. As mentioned earlier, each block has a specific purpose, as indicated by its icon and dialog. The functions of the blocks in the model are:

Block	Description
	The Executive block is a special block that is the heart of every discrete event model and *must* be placed further left than the left edge of any other block in the model. Generally, you will have no reason to change the default values in the dialog as it just controls discrete timing.
	The Generator block is used to generate items (in this case, customers coming in the door) at arrival times specified through the dialog. In the next section, you will set the dialog so that one customer arrives approximately every 39 seconds.
	The Queue FIFO block is used to simulate waiting in line ("FIFO" stands for "first-in, first-out"). When customers enter the bank, they wait in line. If a teller is free, the customer automatically leaves the queue and goes to the teller. If no teller is free, the customers wait in the queue until a teller is ready. Through its connectors, the block reports how long the line is (*L*) and how long the wait is (*W*). If you leave the dialog open when you run the model you can see other information, such as the average wait time for each customer. This block automatically holds and releases customers; you do not need to change any values in this block.
	The customer goes to the first available teller, represented by an Activity Delay block. The customer is then delayed at the teller by the amount of time it takes to complete a transaction. You can either specify a static delay or one that changes dynamically as the model progresses. In the next section, we will set the delay time in the Activity Delay block to reflect a 2 minute transaction time.
	After the delay, the customer goes to the Exit (4) block which takes the customer out of the simulation. The connector at the top outputs the total number of customers that have entered the block. This block automatically counts customers as they leave; you do not need to change any values in this block.
	The Discrete Event plotter, from the Demo Plotter library, shows the results of the simulation. By connecting from value output connectors to the plotter, you determine what values the plotter will show.

Setting values in blocks

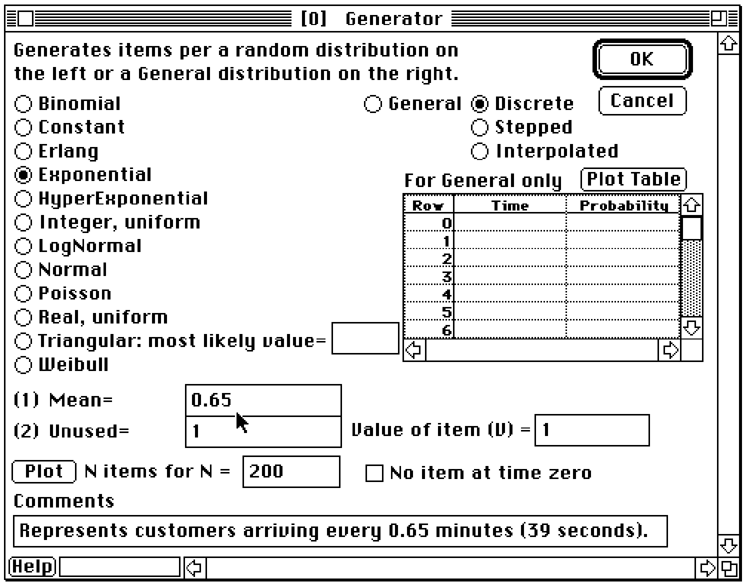
Now that you've built your model and have a basic understanding of each block's functionality, you need to specify the values that will make the model work the way you want. As seen above, many of Extend's blocks perform a pre-determined task. For those blocks, you do not need to change any values. For example, the Queue FIFO block automatically releases customers on a first-in-first-out basis. If you want customers released on a last-in-first-out basis, you would use a Queue LIFO block (not included in this demo version, but is in the full version).

In our Bank Line model, the frequency of customers entering the bank must be specified as well as the length of time a teller takes with each customer. Therefore, you only need to set values in the Generator (incoming customers) and Activity Delay (tellers) blocks.

Source of customers

The Generator block can be used to generate items at a constant rate or with a random distribution. Since your bank customers arrive sporadically (sometimes in groups, sometimes none at all) use an exponential distribution of customers. (To determine which distribution you need, click on the Help button on the bottom left of the dialog box for distribution definitions.)

- Open the dialog for the Generator block by double-clicking on its icon:



[0] Generator

Generates items per a random distribution on the left or a General distribution on the right.

Binomial [✓ Exponential] Constant Erlang HyperExponential Integer, uniform LogNormal Normal Poisson Real, uniform Triangular: most likely value= Weibull

General [✓ Discrete] Stepped Interpolated

For General only

Row	Time	Probability
0		
1		
2		
3		
4		
5		
6		

(1) Mean= 0.65

(2) Unused= 1

Value of item (V) = 1

N items for N = 200

No item at time zero

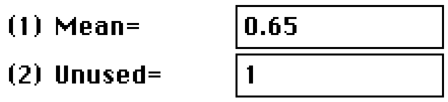
Comments

Represents customers arriving every 0.65 minutes (39 seconds).

Generator dialog

You can specify the type of distribution you want by clicking on the appropriate button: either a random distribution from the column of buttons on the left or a General distribution on the right. The labels on the two entry boxes near the bottom left will change depending on the type of distribution you choose. It is in these entry boxes that you specify the arguments for your chosen distribution function.

- Select the Exponential distribution button by clicking once on the button.
- Then, enter a mean value of .65 in the top argument entry box to represent arrival rate:



(1) Mean= 0.65

(2) Unused= 1

Numeric values

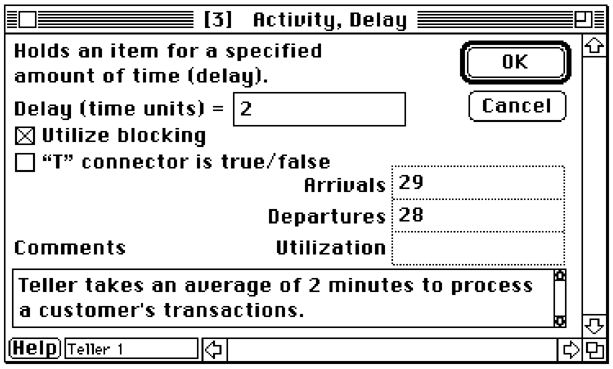
Since an exponential distribution only has a mean argument, the second argument is "Unused". An exponential inter-arrival rate of 0.65 means that a customer will enter the bank approximately every 0.65 minutes or about one every 39 seconds.

- Now, click the OK button to save your changes.

Interacting with the tellers

- Open the dialog for the top Activity Delay block:
- Change the value in the entry box next to "Delay (time units) =" from "1" to "2". This indicates that it takes the teller 2 minutes to handle each customer's transactions. *(This block has the capability of excepting a dynamic random delay time as well. Random delay*

examples are discussed in Chapter 5.)



Holds an item for a specified amount of time (delay).
Delay (time units) = 2
☒ Utilize blocking
☐ "T" connector is true/false
Arrivals 29
Departures 28
Utilization
Comments
Teller takes an average of 2 minutes to process a customer's transactions.
Teller 1

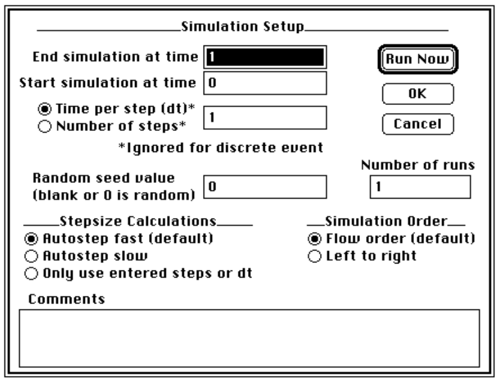
Activity Delay dialog

- Your dialog should look like this. If it does, click OK.
- Follow the same procedure to change the delay values for each of the other two tellers to "2" also.

Simulation setup and running

Simulation Setup dialog

The Simulation Setup command from the Run menu lets you specify how the simulation will run and for how long. The dialog looks like:



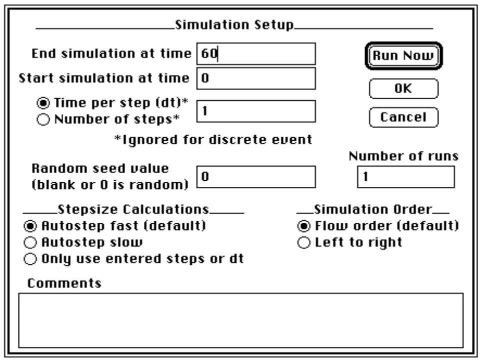
Simulation Setup
End simulation at time 1
Start simulation at time 0
[● Time per step (dt)*] ○ Number of steps* 1
*Ignored for discrete event
Random seed value (blank or 0 is random) 0
Number of runs 1
Stepsize Calculations
[● Autostep fast (default)]
○ Autostep slow
○ Only use entered steps or dt
Simulation Order
[● Flow order (default)]
○ Left to right
Comments

Simulation Setup dialog

Each time you run a simulation, Extend uses the same dialog values you specified in the Simulation Setup dialog. Thus, you will usually only use the dialog the first time you run each model.

Generally, the only setting you will need to change in the dialog is the end time. For most purposes, you want the simulation to start at the beginning, so you would use the default start time of 0. You would only change the "Number of runs" option if you want to repeat the simulation and look at how results can vary randomly over many runs. Also, as you'll note, delta time is ignored for discrete event models because discrete simulations progress based on when events occur in the model, rather than the change in time.

Let's see what happens in our bank over the course of one hour. Set the end time to 60 and use the default start time of 0. This means that the model will simulate 60 minutes of time. Please keep in mind, it is important to keep the time units consistent. Since we started using minutes, we will continue using minutes throughout the entire model.



Simulation Setup

End simulation at time	60
Start simulation at time	0
[✓ Time per step (dt)*] Number of steps*	1
Random seed value (blank or 0 is random)	0
Number of runs	1

*Ignored for discrete event

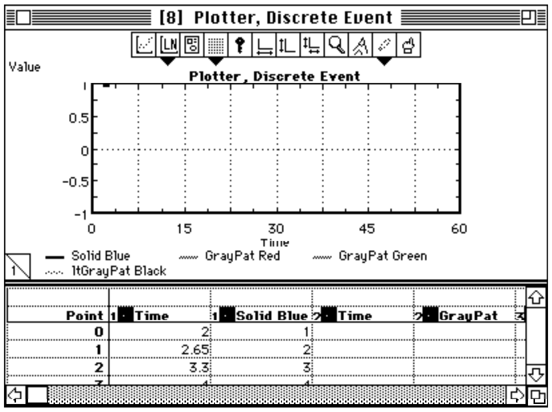
Stepsize Calculations: [✓ Autostep fast (default)] Autostep slow / Only use entered steps or dt

Simulation Order: [✓ Flow order (default)] Left to right

Comments

Simulation Setup dialog for 40 steps

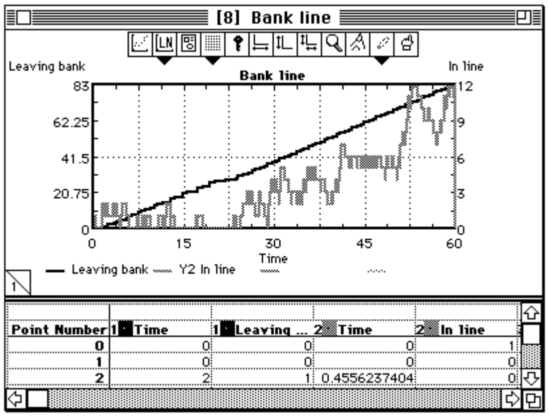
Click the Run Now button. Notice that the choices in the Status bar at the bottom of the screen are based on the settings you designated in the Simulation Setup dialog. The plotter appears as your simulation runs and begins to report information:



Point	1 Time	1 Solid Blue	2 Time	2 GrayPat
0	2	1		
1	2.65	2		
2	3.3	3		

Initial plotter output

Unfortunately, this doesn't tell you anything because the Y axis can't handle the large numbers you're plotting. Click on the tool near the top of the plotter window to scale the Y axis to the data. Now the plot looks more reasonable (we've added a few labels to ours to denote specific items):



Point Number	1 Time	1 Leaving ...	2 Time	2 In line
0	0	0	0	1
1	0	0	0	0
2	2	1	0.4556237404	0

Scaled plotter output

You can scroll through the data table at the bottom of the window to see the values that correspond to the line in the plot. You can also scan the plotter simply by moving the cursor over the plot. As you do, you will see the coordinates where the cursor crosses the trace. Play with the tools in the plotter's tool bar for varying effects.

If you run this model more than once, you’ll notice your results vary slightly. That’s because, as you’ll recall, we have a varying number of customers entering the bank (the random seed varies). Therefore, the number exiting will change as well. If you would like to get the same results each run, change the "Random seed value" in the Simulation Setup to a whole number, such as "1". This keeps the randomness consistent on each run. Change it to "2", and you'll get a different random sequence.

Experimenting with the model

Before you go to Chapter 3, experiment with this model:

- Watch the "customers" progress through the "bank" by running the simulation with Animation turned on. To do this, click on Show Animation in the Run menu (you'll know it's selected if there is a check next to it). Now, run the simulation. Each block's animation is unique and provides information about the status of the items in the model. What the animation means is described fully in a block's help.
- Try changing parameters in the model, then run it again. For instance, change the delay of one or more tellers and see the effect on the waiting line in the plotter. (You can compare a current run to a prior run in this plotter by turning the pages on the bottom left corner of the plot window. Pages are numbered from 1 to 4. In the full version of Extend, there is a block called the MultiSim plotter which can simultaneously show the results of up to 4 runs.)

We told you building an Extend model would be easy. Now do you believe us? Try building some models on your own or just play with the ones we've built for you in this demo version of Extend (they're also described in Chapter 5)!

Now, let's get a little more in depth and find out what else Extend can do...

Chapter 3: Some of Extend 's Advanced Features

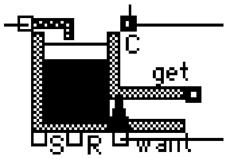
Extend has many features that can help make your models easier to use, more aesthetically pleasing, and more informative for when you want to communicate what is happening in the model. Many of these advanced features add to the professional look of your Extend model.

Animation

Many of the blocks in the Generic, Discrete Event, BPR, and Manufacturing libraries have animation built into them. To see animation, select Show Animation (or Show Movies for QuickTime) from the Run menu so that the command is checked, then run the simulation. Animation is not shown all the time because it tends to slow simulations due to the redraw time.

The Holding Tank in the Demo Lake Pollution model from the Tutorial folder is a good example of an animated block. So that you can see the animation without interference from the plotter, just move the plotter window to the side as the simulation runs . You can also set the plotter to not open, but for now, just move it.

To see the Holding Tank block in action, choose the Show Animation command so that it is checked, then run the model. You will see the level go up and down as it fills and empties. On the first simulation run, the level will go from 0 to an estimate of the maximum value; later runs will use an average of the preceding run's maximum as the level's maximum value.



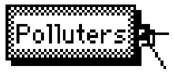
Holding Tank block with animation

Using blocks in Extend's Animation library, you can add customized animation to any model without programming. You can also easily add animation to the blocks you build using special animation functions.

Hierarchy

A hierarchical block contains other blocks that are connected like they would be in a model. When you open a hierarchical block, you see a group of blocks nested inside of it. These blocks represent a portion of the model, or a *subsystem*. You can build hierarchical blocks using menu commands; this allows you to nest subsystems for top-down or bottom-up modeling. Hierarchical blocks can be saved in libraries, and they can have custom icons and animation. You can even nest hierarchical blocks within other hierarchical blocks, up to 32,000 levels deep.

A hierarchical block can contain simple blocks, other hierarchical blocks, or both. The blocks in a hierarchical block are connected just like other blocks in a model, and each hierarchical block has input and output connectors like regular blocks. A hierarchical block looks slightly different from a standard block in that it has a shadow. The Polluters block in the Demo Lake Pollution model is an example of a hierarchical block:

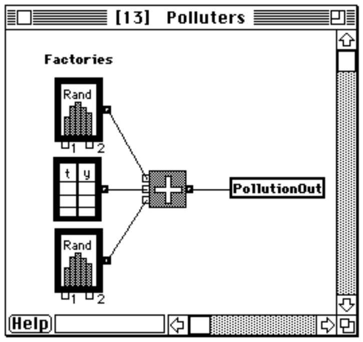


Polluters block

There are many uses for hierarchical blocks. Although you do not have to use them as you build models, you will find them very handy.

- If you have a complex model with dozens of blocks, you can use hierarchical blocks for model simplification. Just select a group blocks, choose Make Selection Hierarchical under the Model menu, and follow the screen prompts to make a new hierarchical block. Hierarchical blocks can be saved in libraries and reused in other models without having to reproduce all of the connections.
- Instead of showing all the detail, you can present your model as a few simple steps. To reveal the subsystems within a step, just double-click on the hierarchical block.
- As you develop a new model, you can go from the simplest assumptions to more complex ones by creating more and more levels of hierarchy. This helps you structure your thinking and makes your models easier to follow as they become more complex.

You open an existing hierarchical block by double-clicking on it. For example, when you double-click on the Polluters block you see:



Open hierarchical block

Hierarchical blocks' connections with the rest of the model are shown as connection text boxes (named connections with borders around them). In this example, "PollutionOut" is the block's connector, as you can tell from the text at the right of the hierarchical window.

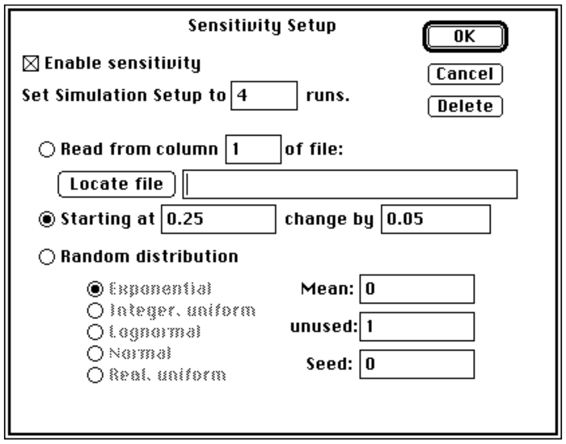
To change the settings in one of the blocks in the hierarchical block, simply open the hierarchical block and double-click on the desired icon. For example, you might want to change one of the dialogs for the factories. You can make any changes in these blocks just as you would in blocks on the model worksheet. You can also clone dialog items from these blocks, as discussed later.

Sensitivity analysis

Sensitivity analysis allows you to conduct experiments and investigate the effects of changes to your model in a structured, controlled manner. You do this by running simulations many times, changing the value of a specific variable or numeric parameter each time the simulation is run. For example, if you run the same simulation a hundred times while varying a parameter with each run, you can see how much of an impact that parameter has on model results.

Extend's sensitivity analysis feature gives you the ability to explicitly specify individual parameters to change and provides several methods for changing them: incremental, random, or ad hoc. For example, the lynx birth rate in the "Ecosystem" model in the "Extend" folder is set to increase by 0.05 for each simulation run. As you run this simulation, the MultiSim plotters show the effect that varying the lynx birth rate has on the lynx and hare populations.

To use sensitivity in a model, double-click a block to access its dialog. Then hold down the Command (⌘) key and click once on the dialog parameter you want to sensitize. When you do this, the Sensitivity Setup dialog appears. You can use this dialog to specify how the sensitized parameter will change, set the number of simulation runs, and even temporarily disable the settings for the parameter. For example, the Sensitivity Setup dialog for the Lynx Birth Rate is:



Sensitivity Setup

☒ Enable sensitivity

Set Simulation Setup to 4 runs.

○ Read from column 1 of file:

Locate file

● Starting at 0.25 change by 0.05

○ Random distribution

● Exponential ○ Integer, uniform ○ Lognormal ○ Normal ○ Real, uniform

Mean: 0

unused: 1

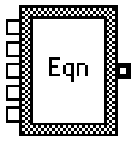
Seed: 0

Sensitivity Setup dialog

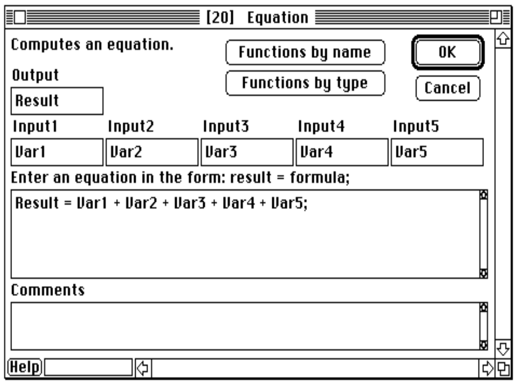
Equation editor

Extend's libraries are tool kits of blocks that allow you to build models quickly. However, this may not be sufficient for all your needs. For example, there may not be a block that provides the function or equation that you want. Or, you may want to combine the function of several blocks into one.

The Equation block allows you to type an equation directly into its dialog. The equation must be of the form *output = formula*. The equation will automatically be compiled when you click OK.



Equation Block



[20] Equation

Computes an equation.

Output: Result

Input1: Var1 Input2: Var2 Input3: Var3 Input4: Var4 Input5: Var5

Enter an equation in the form: result = formula;

```
Result = Var1 + Var2 + Var3 + Var4 + Var5;
```

Comments

Equation Block Dialog

The Equation block is similar in many ways to the formula bar of a spreadsheet. Most of the usual components (operators, values, functions, and so on) are the same. There are two differences: instead of a cell reference, the equation block has input and output value connectors that you identify by name, and the equation block outputs its results to the connectors of other blocks.

The Equation block has five inputs and one output. In its dialog, you can type in an equation that uses any of Extend's built-in functions and operators in combination with the values from the connectors. Your equation can be as simple as performing mathematical operations on an input value or as complex as a full programming segment.

To access the values from the inputs and to output the results of the equation, you must use the *names* of the connectors in the equation. The dialog has default names (such as "result" or "var1"), but you can enter names which have more relevance to your model. You do not have to use all the named connectors in your equation, but you do have to use any connected input. Extend will warn you if you use a connector that is not named or is not connected.

An example of Equation block usage is in the Demo Lake Pollution model where it is used to calculate the outflow of water (and pollutant) from the lake.

Drawing tools

Extend gives you a set of drawing tools so you can draw pictures for block icons and backgrounds for blocks in a model, should you want to highlight or separate sections of the model. The tools available when a worksheet is active are:

Tools in the toolbar

The first four tools let you select items in your model. Since you will probably use the *block/text selection* tool most of the time, it is the default tool. The other selection tools are useful for selecting an item if both a block and a drawing object are near each other in your model. For example, if you have a drawing object behind a block and you want to just move the drawing object, use the drawing selection tool.

The next seven tools let you add text and drawing objects to the model. You use these tools to make your models easier to read or to make them more aesthetically pleasing.

The most common drawing object to add to models is text for labels. To do this, you start a *text box* by either selecting the text tool and clicking on your model where you want to add text or by simply double-clicking anywhere in the model window. Then type in the text you want. To stop entering text, press the Enter key or click anywhere else in the model window. You can add formatting and color to text using the menu commands.

The six tools after the text tool add shapes or lines to your worksheet. For example, to add a rectangle, select the rectangle tool, click in your model where you want one of the rectangle's corners, and drag to the diagonally opposite corner. You can also add colors and patterns to the shapes and lines you draw.

Cloning dialog items

Block information is conveniently stored in dialog boxes associated with each specific block. But in some cases, such as in very large models, having all your choices only in dialogs can be a disadvantage. For example, you may want easy access to parameters for several blocks that are scattered throughout the model. Extend overcomes this dilemma by giving you freedom to copy or "clone" dialog items and place them in another, more convenient location, like the model window or into a notebook, as discussed below.

You can clone any dialog item, including data tables and plotter graphs. You can clone an item to more than one location, such as to two parts of the model window. Every clone acts exactly like the original: if you change the original or any clone, all instances are updated immediately.

To clone dialog items, simply open the desired dialog and select the dialog clone tool, ; note that all the dialog items are now outlined. Click and drag the desired item. For multiple items, select by dragging a frame or holding down the Shift key as you select more items. Then close the dialog by clicking the close box or choosing the Close command from the File menu.

For example, in the Demo Lake Pollution model, go to the Model menu and select Show Notebook. We've made a clone of the plot and pollution level so they can easily be viewed while the model runs. Now, try cloning the constant value from the Constant block to the model window. While you still have the clone tool selected, resize the clone by clicking once near the center of the cloned item so the resizing handles appear. Then click and drag on a handle to change the size and shape of the clone. Then change to the block/text selection tool to change the value of the Constant value clone in the model window. If you keep the Constant block's dialog open when you make the change, you can see that the value in the dialog changes at the same time.

Notebooks

Each model has a *notebook* which can be used for controlling the model parameters, reporting simulation results, and documenting the model. You can create and view the notebook for a model by choosing Show Notebook from the Model menu. You can enter text in the notebook, draw shapes, and import pictures.

Notebooks are a natural place for cloned items. If you do not want dialog items on your model but still want easy access to them, you can move them into the model's notebook. You can use notebooks for both input and output values. For example, you may want to clone buttons into the notebook and change them as the simulation progresses. Or, you might want to use the notebook mostly for looking at the various outputs of your model, such as plots. The notebook is also handy for combining all of your output into one spot so you can copy it for a report.

For example, open the notebook for the Demo Lake Pollution model. Note that it has some text, a clone of the contents from the Holding Tank with its label, and a clone of the plotter.

Other Authoring Features

- Excel Interfacing -- Use Apple Events to communicate to and from Excel spreadsheet cells while your simulation runs.
- Customized Reporting -- Concentrate your reports for in-depth analysis and presentation. (Reports are editable text files that can be read in by any word processing or spreadsheet program.)
- Control Blocks -- Add interactive control directly to your model by adding a Slider, Switch, or Meter. Each of these blocks can control other blocks and show values directly as the simulation runs.
- User Messaging -- Monitor specific parameters and alert the user when critical values are attained.
- Lockable Models -- Put models in run-only mode to protect the original model.
- Stationery Files -- Create model templates for common modeling situations.

Chapter 4: Extend's Built-In Scripting Environment

Up to this point, you have only seen how to use Extend with the blocks that are provided in this package. The full versions of Extend, Extend+BPR, and Extend+Manufacturing come with ***extensive*** libraries of blocks. However, you may still want to modify one of these blocks or create a new one.

One of the most unique features about Extend is its built-in scripting environment that allows you to customize dialogs, icons, and block behavior. All the blocks supplied in the full Extend family packages include their source code, so you can easily modify them to suit your needs. (In this demo version, you can access the source code of blocks in all the libraries except the Demo BPR Lib and the Demo Manufacturing Lib.) Or, build your own with Extend's built-in, compiled scripting language, ModL. Once you understand how blocks work, it is very easy to modify existing blocks and create blocks of your own. Extend is fully extensible, so there are virtually no limits to what you can model. To see just how powerful and flexible Extend's scripting environment is, examine the models in the Models Using Custom Blocks folder and their descriptions in Chapter 5.

This chapter describes the internal parts (structure) of standard blocks. If you haven’t done so already, please read Chapters 1 and 2 before reading this chapter. Many of the concepts in building blocks are based on your understanding of how blocks are used in models.

Block structure

Every block has five parts:

Dialog	The dialog is what you see when you double-click on the block’s icon. This is where you enter and change parameters for your model as well as view a block’s status. Extend’s built-in dialog editor let’s you specify all the buttons, text, and entry boxes that go into the dialog to make it unique.
Icon	The icon you see in the model. You can draw the icon with Extend’s drawing environment, your own painting program, or copy clip art and paste it in. You can also add custom animation to the icon using Animation functions.
Connectors	The input and output connectors on the block. These appear in the icon and transmit information to and from the script.
Help text	The text that appears when you call up the Help command. Information is included that explains how the block works, possible block usage, and connector functions. Help can be accessed through the Help button in a block’s dialog and, when a library is open, through the Help command from the Apple () menu.
Script	Extend has a built-in, compiled, C-like language called ModL. It is the ModL program that makes a block work. The program reads information from the connectors, dialog, and the model environment and produces output that can be used by other blocks.

These parts are interconnected. For example, the script reads information from the connectors, the help text is displayed through the dialog, and so on.

If you were to create a new block, you would:

- Choose Build New Block from the Define menu.

If you were to edit an existing block or simply just wanted to look at its internals, you would:

- Hold down both the Option key and double-click on the block’s icon in a model, ***or***

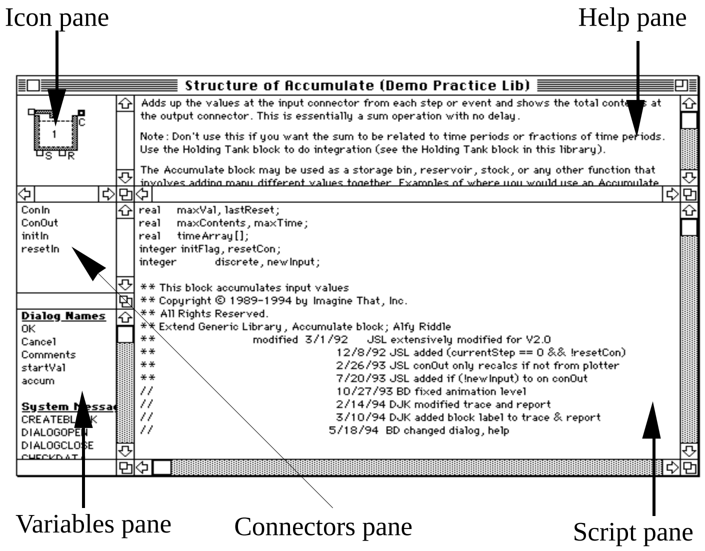
- Double-click on the block's icon in the library window. You can open the library window by choosing Open Library Window at the top of each library's menu.

When you look at a block's internals, you see two windows. The first window, the *structure window*, holds the icon, connector information, help text, and script. The second window, the *dialog window*, shows just the block's dialog.

Let's use the Accumulate block to examine basic block structure. As you'll recall from Chapter 1, any change you make to a block in a library affects every model in which this block appears. Thus, it is very important that you don't make any changes to a block unless you are very sure you want those changes. Just to be safe, let's use the Accumulate block from the Demo Practice library (*not* the Demo Generic library).

- Choose Open Library from the Library menu. In the dialog, select and open the Demo Practice library (*not* the Demo Generic library). It is located in the Demo Libraries folder.
- Drag down in the Library menu to the Demo Practice library and choose Open Library Window from the top of the hierarchical menu. This opens the library window.
- Double-click on the Accumulate block's icon in the library window.

You should see the two internal windows overlapping on your screen. By default, the structure window is in front. The title bar of the structure window displays the block name and the name of its library:



Adds up the values at the input connector from each step or event and shows the total contents at the output connector. This is essentially a sum operation with no delay.

Note: Don't use this if you want the sum to be related to time periods or fractions of time periods. Use the Holding Tank block to do integration (see the Holding Tank block in this library).

The Accumulate block may be used as a storage bin, reservoir, stock, or any other function that involves adding many different values together. Examples of where you would use an Accumulate

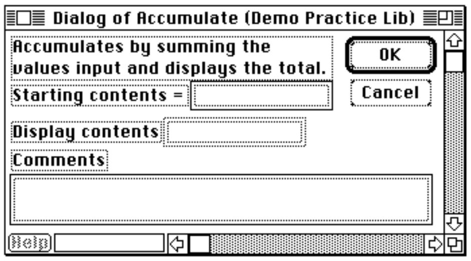
```
ConIn
ConOut
initIn
resetIn
```

```
Dialog Names
OK
Cancel
Comments
startVal
accum

System Messa
CREATEBLOCK
DIALOGOPEN
DIALOGCLOSE
CHECKDAT
```

```
real    maxVal, lastReset;
real    maxContents, maxTime;
real    timeArray[];
integer initFlag, resetCon;
integer         discrete, newInput;

** This block accumulates input values
** Copyright © 1989-1994 by Imagine That, Inc.
** All Rights Reserved.
** Extend Generic Library, Accumulate block; Alfy Riddle
**                  modified  3/1/92    JSL extensively modified for V2.0
**                              12/8/92 JSL added (currentStep == 0 && !resetCon)
**                              2/26/93 JSL conOut only recalcs if not from plotter
**                              7/20/93 JSL added if (!newInput) to on conOut
//                              10/27/93 BD fixed animation level
//                              2/14/94 DJK modified trace and report
//                              3/10/94 DJK added block label to trace & report
//                             5/18/94  BD changed dialog, help
```

Structure window for Accumulate block

The dialog window, behind the structure window, looks like:



Accumulates by summing the values input and displays the total.

Starting contents =

Display contents

Comments

Dialog window for Accumulate block

You can change the size of any of the panes by dragging any of the ⧉ icons. You can also change the font used to display the script through the dialog of the Preferences command in the Edit menu.

To close these windows, choose Close from the File menu or click on the close box in the upper left corner of either window.

Dialog

Every block has a dialog in which information is entered and/or displayed. A dialog may be as simple as some text with OK and Cancel buttons, or it may be quite complex. Extend prompts you on how to design your block's dialog.

A dialog contains *dialog items* and a *Help* button. Each item has its own definition. The items in a dialog depend on how the block was defined. For example, the Accumulate dialog has nine dialog items:

- Four text labels:

  "Accumulates by summing the values input and displays the total."

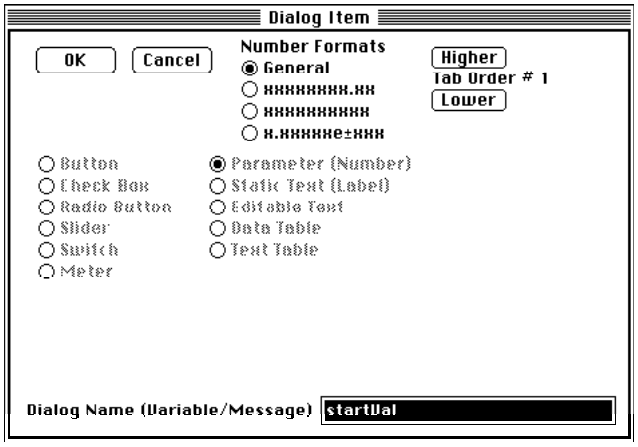
  "Starting contents ="

  "Display contents "

  "Comments"
- Three entry boxes. Two of these boxes are *parameter* items for entering and displaying numbers and one is for entering *text*. The top box is for entering the starting value. The second one is for viewing the total accumulation, and the third is for comments.
- An OK button
- A Cancel button

The Help button and block label fields are always located to the left of the bottom scroll bar.

To see the definition of a dialog item, double-click on that item in the dialog window. For example, when you double-click on the entry box next to "Starting contents =", you see:



Number Formats: [✓ General] xxxxxxxx.xx xxxxxxxxxx x.xxxxxe±xxx

Tab Order # 1

Button, Check Box, Radio Button, Slider, Switch, Meter, [✓ Parameter (Number)], Static Text (Label), Editable Text, Data Table, Text Table

Dialog Name (Variable/Message) startVal

Definition of the top entry box

There are 11 types of dialog items you can choose from in Extend.

Type	Description
Button	A button that can be clicked, like an OK button or Cancel button. Buttons have *titles*that appear as text inside the button.
Check box	Square buttons that have an "X" in them when they are selected and are empty when they are not. Check boxes have *titles* that appear as text to the right of the button.

Radio button	Round buttons that appear in groups. Only one button in the group can be selected. When you select one button from a group, every other button in that group becomes deselected. Radio buttons have a group number and have *titles* that appear as text to the right of the button.
Slider	A control that resembles a slider on a stereo. You can drag the knob to change a value, or your block can move the knob to show a value.
Switch	A switch that resembles a light switch. It has two values, 0 and 1.
Meter	An output-only item that shows a needle in a meter.
Parameter (Number)	Entry box that takes a number. You can specify the numeric format of the number as it appears in the box: General, currency (2 decimal places), integer, or scientific.
Static Text (Label)	Text that appears in the dialog.
Editable text	Entry box that takes text.
Data table	A two-dimensional table (similar to a spreadsheet) for holding numbers. A table can have up to 428 columns and 2700 rows (limited to a total of 3200 cells). The table comes with scroll bars for moving around in the table. You define the number of rows, number of columns, number format, and headings for the columns.
Text table	A two-dimensional table (similar to a spreadsheet) for holding text. A table can have up to 428 columns and 2000 rows (limited to a total of 2000 cells). The table comes with scroll bars for moving around in the table. You define the number of rows, number of columns, and headings for the columns.

As in other Macintosh programs, you can move between entry boxes by pressing the Tab key. Each parameter and editable text box in a dialog window has a tab order number. To change the order, click the Higher and Lower buttons in the definition box for that item. When you change the tab order for a dialog item, the tab order for the other dialog items is automatically adjusted.

All dialog items have *dialog names* (for labels, the dialog name is optional). These dialog names are variable names and message names used by the script to interact with the dialog. Some dialog items have *titles* or text that appear in the dialog.

Once you have placed a dialog item in a dialog window, you can easily move it by selecting it and dragging it to a new location. If you want to move multiple dialog items, you can select all the items at once by holding down the Shift key as you click on each one or by dragging a frame around them. You can also resize any item in a dialog window by selecting it and dragging its handle (the black square in the lower right corner of the item). When a dialog item is placed in Extend, it is put in an invisible grid. To position them off the grid, hold down the Option key as you drag it. If you add a new item to a dialog window, you will:

- Choose New Dialog Item from the Define menu.
- Choose the type of item you want from the dialog.
- Drag the item to the desired location.

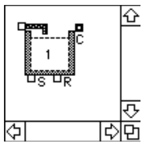
Icon

A block's icon is the most apparent aspect of a block since it appears in the model window. You can see the icon in the upper left pane of the structure window. The icon consists of a drawing or group of drawn objects and the block's connectors.

There are two ways you can draw an icon: either use Extend's drawing tools or paste drawings from the Clipboard. If you have a drawing program such as SuperPaint or MacDraw, you might want to use it to draw or edit your icon, then paste it into the icon pane. Or you can paste in clip art or other pictures. Otherwise, use Extend's drawing tools.

Extend automatically gives you a grid for placing objects. If you want to use fine placement for your objects, hold down the Option key before selecting the object to override the grid as you move objects in the pane.

The icon pane looks like:



Icon pane

Connectors

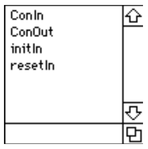
Chapter 2 described connectors and connections in detail. There, you saw three types of connectors: value, item, and universal. Note that these three types of connectors are shown in the toolbar when the structure window is the front-most window:

Structure window toolbar

Find the tool on the palette. It adds animation objects to the icon. The next four icons add connectors: value (), item (), universal (), and diamond(). To add a connector, click on one of these tools, then click in the icon pane at the desired position.

Note: The fourth type of connector, the diamond connector, does not have any "special" properties, and is provided for your convenience. For example, if you are designing a new set of blocks and want to be sure that users only connect those blocks to each other, you can use the diamond connector. If the user tries to connect a diamond connector to a value or item connector, Extend will not let them.

You add connectors to the icon pane using the toolbar. Every connector has a unique name. The name of the connector defines whether it is an input or output connector. The names are shown in the connector pane. For example, the names of the connectors in the Accumulate block are:



ConIn
ConOut
initIn
resetIn

Connector pane

You can change the name to whatever you want, but the name must end in "In" or "Out". If you were to change the name of a connector, you would want to:

- Select the connector name in the connector pane. Extend highlights that connector in the icon pane so you can identify it.
- Type a new name or edit the name.
- Press the Enter key or click anywhere else in the connector pane to save the edited name.

When you add connectors to the icon, they are all initially input connectors. To make one of these connectors an output connector, change its name to something that ends with "Out".

If you selected the wrong type of connector (such as a value connector when you wanted an item connector), you can easily change its type:

- Select the connector on the icon pane by clicking on it.

- Click on the correct connector type in the toolbar.

Help text

You can change the help text in the upper right pane to anything you want. Simply edit the text in the window. You can also add formatting to the text in the help window by selecting it and giving commands from the Text menu. The help text appears when you click the Help button in the lower left corner of the dialog and is also available in the Help command of the Apple () menu.

Information included in the Help gives a brief description of the block itself and its usage, defines dialog items and explains their usage, indicates the connectors' use, and tells what animation (if any) the block has.

Script

The block's script is in the lower right pane of the structure window. You can edit and view in this pane like you can in other Macintosh programs. Scroll through the script to get a feeling for how it looks.

If you are familiar with the C programming language, you are probably very comfortable with Extend's language, ModL. It is essentially C with a few extensions and changes. If you are not familiar with C but know some other programming language such as BASIC or FORTRAN, you can probably follow the general logic of the script and get a feeling for how it works.

The first lines of the script are the declaration of the types of variables used in the script. The next few lines, the ones that begin with "**" or "//", are comments. After the comments are lines of programming code that are grouped into sections. Each section is either a *message handler* or a procedure. Message handlers begin with a line "on *xxx*". Procedures begin either with "procedure xxx" or "*type xxx*." For example, a message handler you see in every block begins "on Simulate" and the message handler script starts and ends with curly braces ({ and }).

The "on *xxx*" message handlers tell Extend what to do in various circumstances. For instance, the lines in the "on Simulate" message handler are executed for every step in the simulation. The lines in the "on InitSim" message handler are only executed once, at the beginning of the simulation. When you create your own blocks, you can add message handlers that are executed at defined times, such as when the dialog for the block is opened (so you can initialize the dialog's contents), when the simulation is stopped, or when a dialog button was clicked.

Extend's ModL language

- Extend's built-in language, ModL, is a C-like language with object oriented extensions. To see more about the language syntax, look in the Help command of the Apple () menu.
- Extend has a built-in compiler and editor. Blocks are compiled once, when they are created. You do not recompile blocks when the model runs.
- Extend's XCMD and XFCN functions allow you to execute code

created outside of the Extend language environment.

- The ModL language has extensive error checking to protect the development environment.

Chapter 5: Sample Models

Extend

Extend is a general-purpose simulation package mostly used by engineers, scientists, and those who want to build their own libraries of blocks. The Extend package comes with a tutorial, manual, example models, and the Generic, Discrete Event, Engineering, and Plotter libraries. It is most commonly used for economics, electronic and mechanical engineering, biology, chemistry, and simple discrete event modeling.

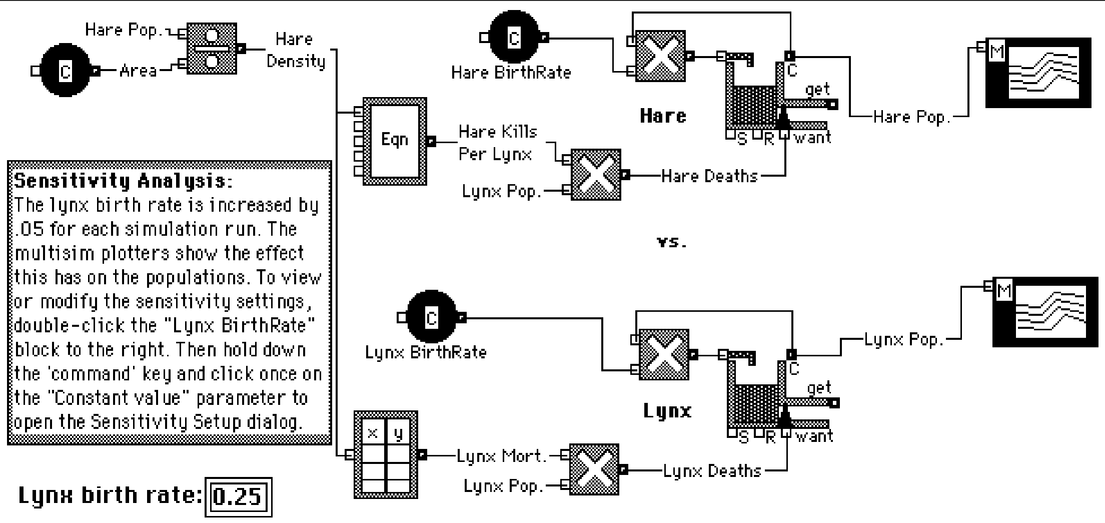
Ecosystem

We've modeled a typical predator-prey interaction in an average ecosystem represented by the hare and lynx. Each population has a direct effect on the other. Because the lynx feed on hare, the hare population begins to diminish. Because their food source is disappearing, the lynx begin to die off.

With less lynx in our ecosystem, the hare have a chance to propagate, increasing their population. Because their food source has been replenished, the lynx propagate and, in turn, begin depleting the hare population again. This cycle repeats over and over again.

Particulars:

- This model is built using Extend's Generic and Plotter libraries.
- Our 100 hectares ecosystem initially contains 6000 hare and 125 lynx.
- On the average, hare produce 1.25 offspring each per year and lynx, 0.25 each per year. Note: this model utilizes Extend's sensitivity analysis feature by varying the birth rate of the lynx. In the initial run, the birth rate is 0.25. That rate increases by 0.05 each run in a series of 4 runs. After running the model, see how the change in this one parameter impacts the pattern of behavior for the entire model.



Sensitivity Analysis:
The lynx birth rate is increased by .05 for each simulation run. The multisim plotters show the effect this has on the populations. To view or modify the sensitivity settings, double-click the "Lynx BirthRate" block to the right. Then hold down the 'command' key and click once on the "Constant value" parameter to open the Sensitivity Setup dialog.

What ifs

- What if our ecosystem started with more lynx, say 500? Double-click on the Holding Tank to the right of "**Lynx**". Under "Starting contents =", change "125" to "500". Click "OK". What effect did this change have on the life cycle curves in the plotter?

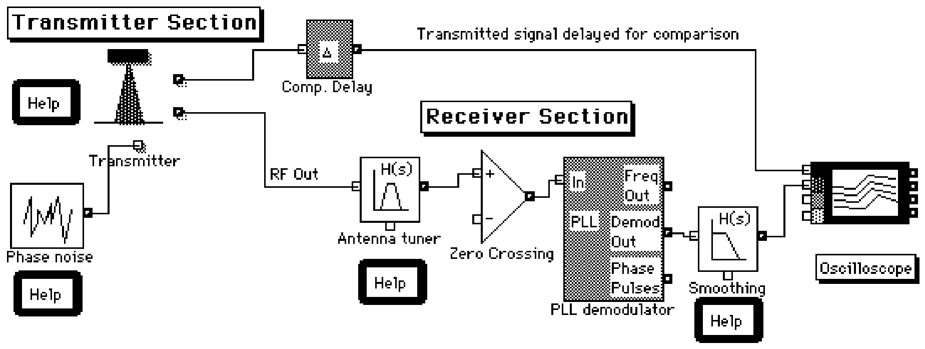
- What if we lost 30 hectares of the area of our ecosystem effecting the density of hare? Open the Constant block on the left of the model next to “Area”. Change “100” to “70”. Click “OK”. Run the simulation again to see how it effects the overall system.

Noisy FM System

Just how does a receiver in a digital FM transceiver system perform within a noisy environment? Before investing your time and money in building a prototype and carrying out testing on it, model the system in Extend. This way, you can experiment with the system’s performance faster, less expensively, and with more exacting results.

Particulars:

The strategy in this model is to generate a typical radio transmission of a known repetitive signal, add some noise, and feed it into the receiver. By plotting both the delayed transmitted signal and the noise-distorted wave-form from the receiver, you can determine the inaccuracies in the signal. The Transmitter section, represented by a single hierarchical block, consists of several VCO's connected together to produce a digital stream of periodic short and long pulses. This is used to modulate another VCO used as the FM carrier generator. Double click on the "Transmitter Section" block to view the VCO's. The Noise section adds phase noise to the FM carrier so that its output simulates a signal that has traveled through the atmosphere. The Receiver section contains a phase-lock loop which tracks the signal, demodulating the transmission back into a digital bitstream. This model was built using blocks from Extend’s Engineering and Plotter libraries.



What ifs:

- What if you were to increase the level of noise in the Noise block. Double-click on the block labeled “Phase Noise”. Adjust the “Amplitude” and/or “Mean”. Click “OK”. Does the Receiver still track the signal when you run the simulation?

Extend+BPR

Extend+BPR (Business Process Reengineering) bundles Extend with a library of blocks optimized for modeling business processes. It is used to describe, analyze, and reengineer procedures in the insurance, real estate, banking/finance, software, and other industries. It is also helpful for modeling departmental and divisional operations in business and government.

Business systems are composed of real-world objects which interact when specific events occur. Extend+BPR simulates these systems using blocks which mimic business processes and timing that corresponds to actual events. These processes can then be reengineered by making appropriate changes to the process model, rather than committing time and resources to altering the process itself.

Extend+BPR blocks directly correspond to the activities, queues, delays, and transformations that comprise business processes. The BPR library also includes high-level reengineering capabilities such as batching, cycle timing, and conditional routing.

Cycle Timing

In order to reengineer business processes, it is important to first model the process “as is” and take measurements of important factors. Once you have measured all strategic variables, you can model the potentially improved process and see how the new measurements compare to the benchmarks.

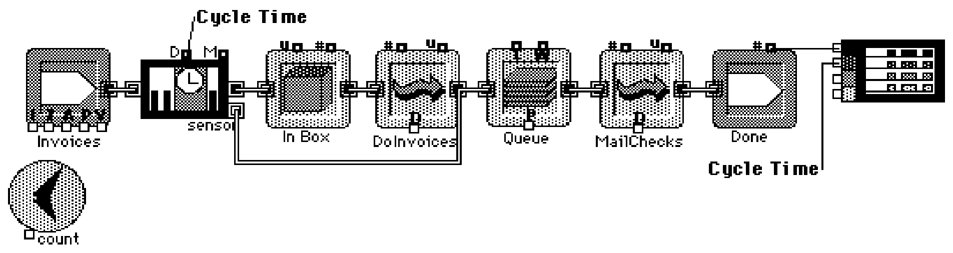
One of the most important factors to measure and improve is cycle timing. Cycle time is a measurement of the time from when an item is first received until it is finally sent out of the process. For instance, the cycle time for an order starts when the order is received and ends when the customer receives the goods ordered. Cycle time is not the same as processing time, in fact, it is almost always more than processing time. For example, although an order entry clerk can input an order a minute, the cycle time in the order entry department may be days or even weeks if orders arrive faster than they are processed.

Particulars

This Cycle Timing model illustrates a two-activity model where invoices are received and checks are generated and mailed.

- Invoices arrive approximately one every 6 minutes and take 7.5 minutes to process.
- It takes .25 minutes to print and mail the checks once invoices are approved.
- There are 6 invoices already awaiting approval at the start of this business day.
- This model runs for a simulated 8 hour day or 480 minutes.

The Timer block automatically keeps track of the cycle times between any two points in a model. In this model, it tracks the time from when invoices are received until the check is ready to be mailed. Results of this simulation show that although it takes 7.5 minutes to process each invoice, the cycle time for invoice processing increases to about 140 minutes by the end of the day!



What ifs

- What if we’ve reengineered this process in such a way that it now takes only 5 minutes to process each invoice. Double-click on the first Transaction block (labeled “DoInvoices”) and change the “Transaction time =” to “5”. Click “OK”. Look inside the Plotter block as you run the simulation to see how this reengineered process effects the cycle time. Compare this to the previous run (where it took 7.5 minutes to process an invoice) by clicking in the upper triangle of the square on the bottom left corner of the plot pane.

- What if we want to compare the cycle and processing times over the course of one week. Under Simulation Setup in the Run menu, change “End simulation time” to “2400”. Click “Run Now”. How does the plot look now? Is this what you expected?

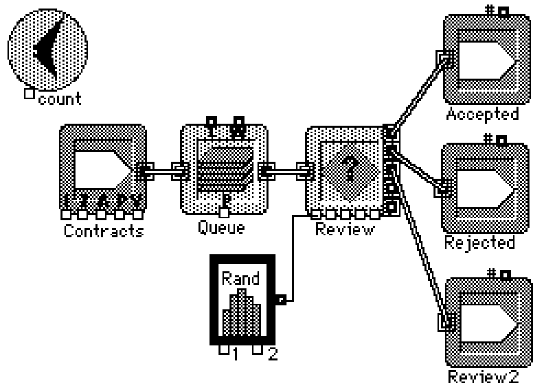
Decision Process

Decision making occurs every day in business. In any corporation, workers will prioritize tasks, review applications or claims, and route paperwork for approval. This Decision Process model shows the initial review of contracts by a company's legal department. After a period of time, the processor determines that the contracts are acceptable, unacceptable, or require more review.

Particulars

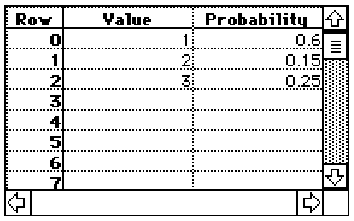
- Contracts arrive about one every hour.
- It takes 1.25 hours to review a contract.
- 75% of the contracts are accepted, 10% are rejected, and 15% require more review.
- This model runs for one simulated business week or 40 hours.

The block labeled Contracts generates the contracts for review. The Review process pulls contracts from the Queue block and processes them one at a time. The Input Random Number block (labeled RAND) provides a convenient table for specifying how many contracts pass and how many fail or need more review.



What ifs

- What if business picks up and contracts arrive two per hour, but it still takes 1.25 hours to review each one. Open the Input block (labeled "Contracts"). Next to "Mean =", change the "1" to a "0.5". Click "OK". What happens when you run the simulation? Open the block labeled "Queue" and look at the backlog of contracts that are stacking up.
- What if acceptance criteria becomes more stringent so that only 60% of the contracts pass this review stage while 25% need additional review and 15% are rejected. To change the percentages, open the Input Random Number block (labeled "Rand"). Change the values under Probability as follows:



Row	Value	Probability
0	1	0.6
1	2	0.15
2	3	0.25
3		
4		
5		
6		
7		

Save these changes by clicking "OK". Run the simulation again. See how easy it is to change and test assumptions in Extend!

Extend+Manufacturing

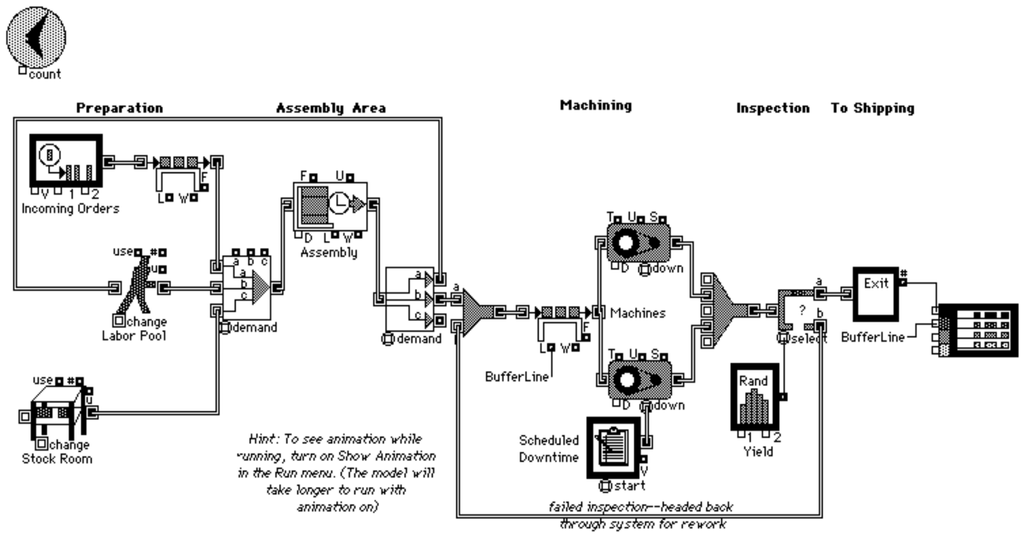
For modeling discrete industrial and commercial processes such as manufacturing systems, networks, distribution systems, service industries, paper flow, etc., use Extend+Manufacturing. It is heavily used by industrial engineers, operations researchers, and manufacturing systems analysts.

Assembly/Rework

This model represents an assembly/manufacturing system where components and orders are combined with labor to produce a product. Let's say we're shipping computer systems. Each system consists of three parts: the monitor, the CPU, and the keyboard. For each order, the three parts are taken from the stock room to the assembly area where a laborer assembles the parts. After assembly, the laborer sends the assembled system with its order to the machining area and the laborer goes back to assemble more systems. During the machining process, the systems are boxed for shipping. Sometimes the boxes get crushed or the systems may simply not be packed correctly and must be returned to be repackaged. Those passing inspection continue on with their affiliated order to the shipping area.

Particulars:

- Orders for systems are placed an average of three a minute.
- There are a total of six workers available. All six can work in the assembly area at one time.
- There are 5001 parts in stock for assembly.
- When one order is received and one of each part has been delivered to the batching area, they must wait to move on to the assembly area until a laborer is available. (Note: the ability to have processing be constrained by scarce resources is a very important feature of discrete event modeling.)
- It takes 0.5 minutes to assemble a unit.
- Systems go to the first available machine for processing which takes 0.75 minutes. One machine is shut down for maintenance for one minute every five minutes.
- 85% pass inspection while 15% are sent back for rework.


count
Preparation
Assembly Area
Machining
Inspection
To Shipping
Incoming Orders
Labor Pool
change
demand
Assembly
Machines
BufferLine
down
Scheduled Downtime
start
Rand
Yield
select
Exit
Stock Room
Hint: To see animation while running, turn on Show Animation in the Run menu. (The model will take longer to run with animation on)
failed inspection--headed back through system for rework

What ifs

- What if you wanted to watch the flow of parts and labor through the model. Go to the Run menu and select Show Animation. Run the model again.

- What if we decide to add another machine to our process in an effort to increase productivity. Click on the top Machine block. Select Duplicate from the Edit menu. Connect this new machine between the Buffer block and Combine block just like the other Machines. (This new Machine will also take 0.75 minutes for processing since it is a duplicate of the first, unless you wish to change it.) Now run the simulation and see if it would indeed be beneficial to purchase a new machine for the plant.
- What if we lay off one laborer. Double-click on the Labor Pool block and change the “Initial labor =” value from “6” to “5”. Click “OK”. Run this simulation again and see if the line runs as efficiently or if that sixth laborer is a necessary part of the operation.

Job Shop Operations

Often in a system, you have more than one type of item being processed in the same operation. Processing for each part varies depending on the type of part it is. The Job Shop Operations model shows just such a system.

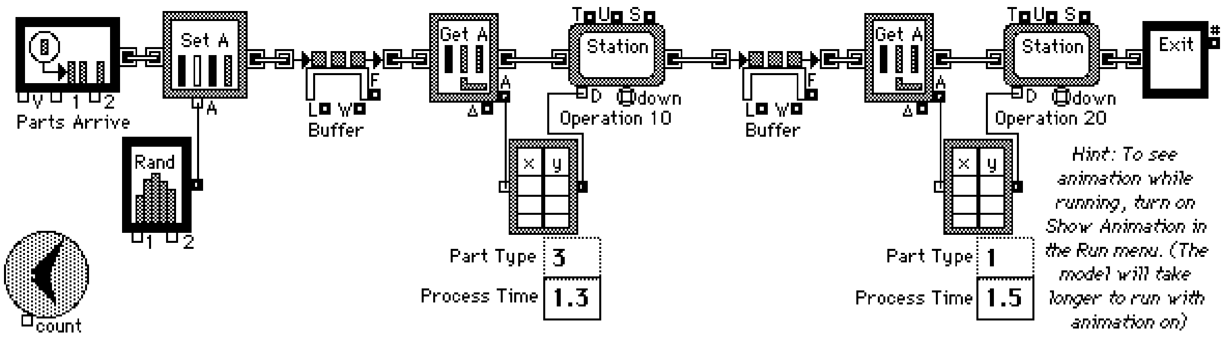
In this model, we assign an attribute, Part Type, to incoming items. Each Part Type has a different processing time in each of two processing areas, Operation 10 and Operation 20. The attribute is recalled during processing and the item is processed for the amount of time required by its Part Type attribute.

Particulars

- Items arrive sporadically at an average of about one every two minutes.
- Upon arrival, each item is assigned a Part Type: 50% will be Part Type 1, 30% Part Type 2, and 20% Part Type 3.
- Processing times for each Part Type at each stage of processing are:

Part Type	Operation 10	Operation 20
1	2.0 minutes	1.5 minutes
2	1.6 minutes	2.0 minutes
3	1.3 minutes	1.8 minutes

- This model runs for 60 minutes.

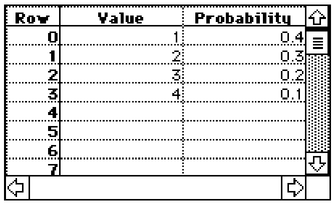


What ifs

- What if items enter the Job Shop at a rate of one every minute. Open the Generator block.

Change the “2” to a “1” in the entry box next to “Mean =”. Click “OK”. Watch the backlog of item in each Buffer block as you run the simulation.

- What if due to variations in 10% of Part Type 1 items we now have a fourth Part Type, Part Type 4. It takes 3.4 minutes for the processing of Part Type 4 during Operation 10 and 2.6 minutes for processing during Operation 20. Open the dialog box on the Input Random Number block (labeled “RAND”). In the table on the right, enter “4” as a Part Type and its probability of occurring to “0.1”. Then, change Part Type 1’s probability to “0.4”. Your table should now look like:



Row	Value	Probability
0	1	0.4
1	2	0.3
2	3	0.2
3	4	0.1
4		
5		
6		
7		

Click “OK”. Now, set the processing time for Part Type 4 in Operation 10 by opening the first Conversion Table (the block with the “x” and “y” on it). In the same manner you set the frequency of Part Type 4 to appear, enter Part Type “4” in the “x in” column of the table with its processing time of “3.4” in the “y out” column. Click “OK”. Then, follow the same procedure to set the processing time for Operation 20. What happens when you run the simulation? Can you still process as many items in an hour as before?

Models using custom blocks

There are virtually no limits to what you can model in Extend. With Extend’s scripting environment you can build custom blocks each with its own specialized behavior, icon, and dialog. Custom blocks make it easy for others to comprehend a model and interact with it.

These models use blocks created with Extend’s built-in language and dialog editor. They illustrate Extend’s unlimited flexibility and power.

Fish Pond

Basically, the Fish Pond model is similar to the Ecosystem model in the Extend folder, except that this model uses custom blocks created within Extend’s built-in scripting environment. This two creature ecosystem shows how a single block design can model many types of creatures depending on the parameters entered. The main block in this model is a fish which can simulate many different kinds of fish in an ecosystem. This Fish block is used to represent two different species: carrion-eating fish and a natural predator (in this case, the piranha). Each Fish block added to the model represents another species and creates a more complex ecosystem.

Most of the action in this model happens in the left and right connectors on the Fish block. The connector at the left finds out how much potential food is available. The connector at the right tells how many live fish there are that can be eaten. The “Carrion” connector tells how many fish have died and are thus available to be eaten by the Carrion-Eaters.

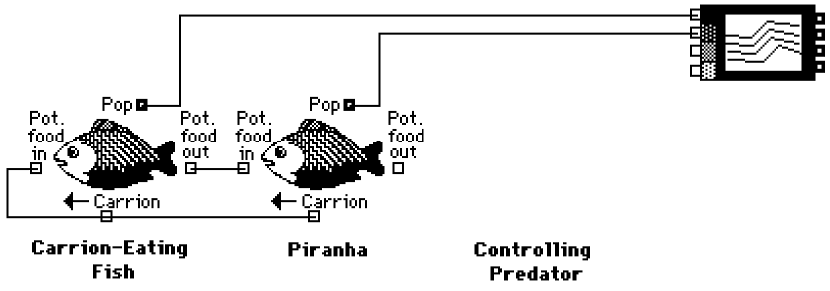
Carrion-Eating Fish eat other fish that have died in the pond. The Piranha eats the Carrion-Eating Fish. The pond has an interesting property in that there are lots of places to hide, so that even if one species’ population gets down to one breeding couple, they can find a place to hide and mate. This keeps them from dying out completely.

Although this model may appear simple, the actual underlying calculations are quite complex. It illustrates Extend's ability to represent a complex system with a few high level constructs.

Particulars

- This is a continuous model built with custom blocks created in Extend's scripting environment.
- Initially, there are 50 Carrion-Eating Fish and 2 Piranha.
- Though their gestation period is the same (9 days), 7 Piranha are reproduced each breeding period while only 5 Carrion-Eating Fish are reproduced.

- Carrion-Eating Fish require 1 prey every 6 days. Piranha require 3 Carrion-Eating Fish every 2 days.
- Each dead Carrion-Eater supplies 2 carrion food equivalents with 10 being supplied for each dead Piranha.
- This model runs for 200 days.



What ifs

- What if, in an attempt to balance this ecosystem, we were to add another predator to control the Piranha population? Add another Fish block from the Demo Custom Blocks Lib and place it to the right of the Piranha block over the label "Controlling Predator". Connect the "Pot. food out" connector of the Piranha block to the "Pot. food in" connector of the Controlling block to show how they feed on the Piranha. Connect the "Carrion" connector in line with the others to show how the dead Controlling Predator become a part of the food pool for the Carrion-Eaters. Finally, connect the "Pop." output to the third input down on the plotter. Using the default parameters in the Controlling Predator block, run the simulation. Now that there is some control on the Piranha population, are all species in the pond propagating in cycles in a more balanced manner? Examine the results in the Plotter to find out.
- What if we start with more Piranha in the pond, but, because of disease, they don't reproduce as quickly. Double-click on the Piranha and increase the "Initial # of animals" and decrease the "Mean # offspring". Click "OK". What happens when you run the simulation now?
- What if we wanted to use this model to represent another type of ecosystem, say one that involves Hare and Lynx as shown earlier in this chapter. You can change the appearance of a block by option-double-clicking on one of the Fish blocks. Click on the fish in the icon pane to select it, then delete in. Now, either paste in a picture from another drawing program, some clip art, or use Extend's drawing tools to draw a new icon. Now, choose Save Block as... from the File menu. In the window that appears, rename the block and select the library you want the block in before clicking on Install. For example, if you have used a picture of a Hare, call the block Hare and install it in the Custom Blocks Lib. Follow the same procedure for the Lynx block. Now, you can open a new model window to build an identical model using the new blocks you have created. Then, adjust parameters within each block to build your own ecosystem.

Three Body Problem

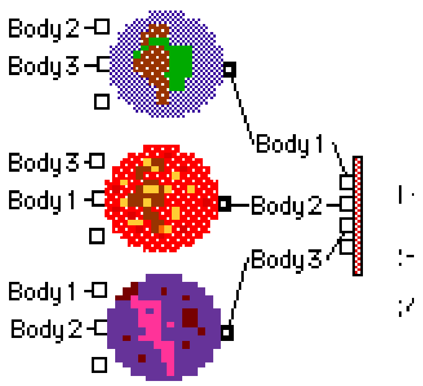
This model shows gravitational interactions between three distinct bodies, which are represented here by planets. It contains two types of blocks, the planet blocks and the planet plotter, both created using Extend's built-in language, ModL. The planet blocks contain information about the individual planets and calculate the gravitational forces. The Planet Plotter displays the positions of the planets and animates the motions.

Particulars:

This chart shows each body (planet) and its characteristics in this model:

Body	Planet	Mass	Density	Initial X	Initial Y	Δ X	Δ Y
Body 1	Earth	50	5	0	0	0.1	0
Body 2	Mars	20	4	-5	20	-0.3	0
Body 3	Venus	5	1	7	20	1	-0.1

When you run the simulation, each body is placed on an invisible plotter at the Initial X and Y locations designated in the block. It moves based on its specific mass and density and according the change in X and Y indicated. Notice that animation occurs outside of the icons and that the planets can even leave the field for periods of time.



What ifs

- What if you wanted to watch gravitational attractions around the Sun. Open any of the planet blocks and change its planet name by clicking on the Sun button. When you click OK, look at the new icon.
- What if a meteor has moved one of these planetary bodies to a different initial position. Open the dialog of the planet that was moved, and enter new parameters for “Init X” and “Init Y”. Click “OK”. What does the simulation look like now?

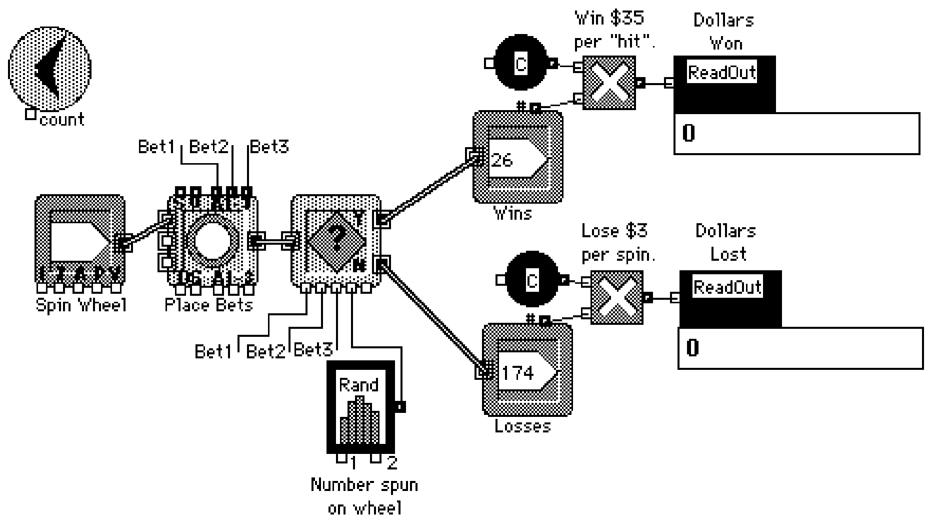
Recreational model

Roulette

What are the odds of your numbers coming up? Give this simulation a whirl and see for yourself.

Particulars

- This model uses blocks from the BPR library to illustrate a casual loop (although action *A* might reach outcome *B*, it may also result in the opposite of *B*).
- Bets can be placed on three numbers between one and 38.
- One dollar is bet on each number each spin of the roulette wheel.
- Since the odds are 35-1, you collect $35 each time one of your numbers hits. When no numbers hit on a spin, you lose the $3 you bet.



What ifs

- What if you want to bet on three different numbers. Open the Operation block (labeled "Place Bets") and enter new numbers in the entry boxes next to "bet1", "bet2", and "bet3". Click "OK". Are these numbers winners for you when you spin the wheel?
- What if you were feeling especially lucky and decided to double your ante. Open the top Constant block (next to "Win $35...") and change the win per spin value to 70. Click "OK". Then, open the other Constant block (next to "Lose $3...) and change the loss per spin value to 6. Click "OK". What happens now?