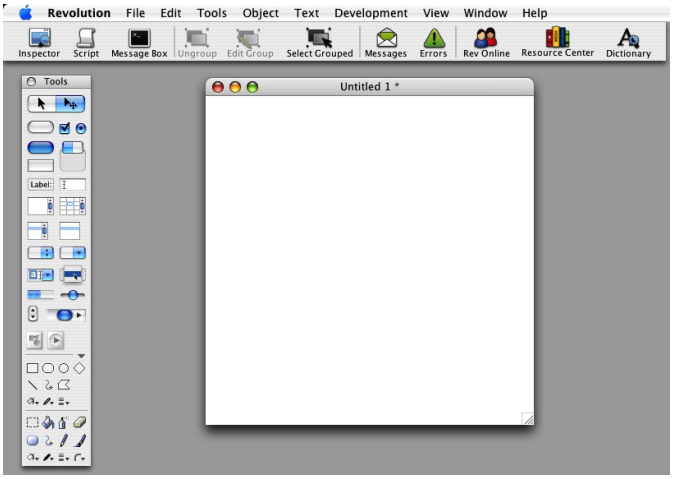
Getting Started with Revolution

This tutorial will give you an overview of the objects and components you will use when building applications. It describes how to place and manipulate objects in your application as well as explaining the inner workings of Revolution. This knowledge will help you build efficient applications quickly.

Stacks, Cards and Objects

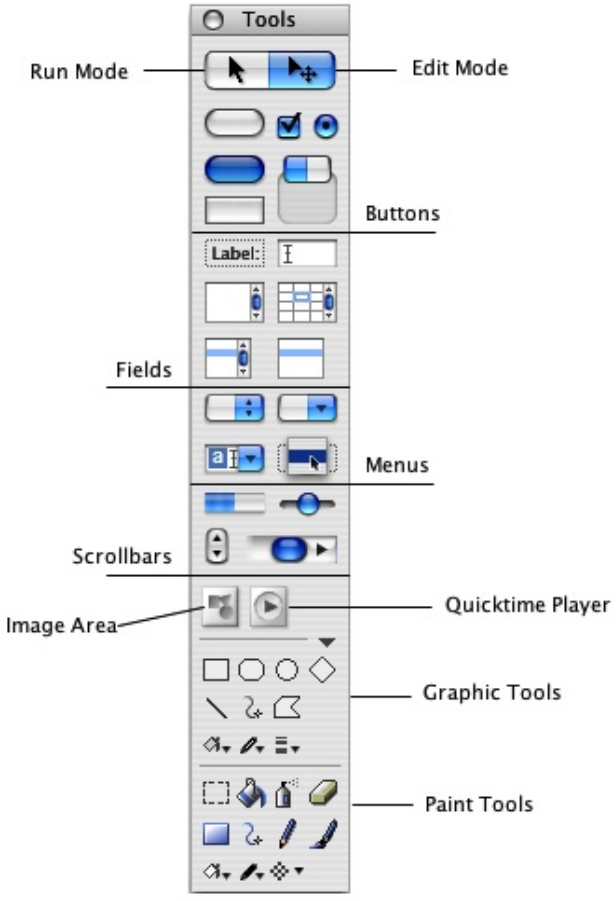
Applications in Revolution are known as stacks and the separate screens which make these up are called cards. A stack can contain any number of cards and may also contain substacks. The substacks will generally provide specialist actions or implement specific parts of your application.



Objects can be placed on cards by dragging them from the Toolbar (which is shown on the left). Once an object has been placed on a card it can be moved around by further drag and drop actions or by using the objects property inspector.

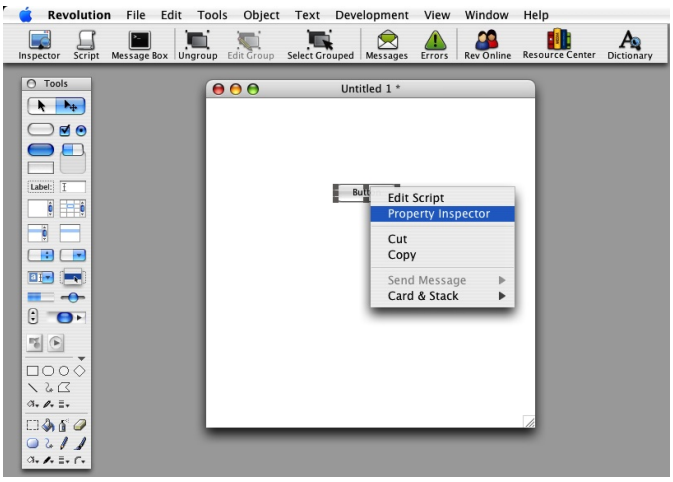
The Toolbar

The toolbar contains all the objects you would normally expect to use in an application, such as buttons, text fields and scrollbars. It also provides the main method of switching between edit and run modes, which are the two modes which Revolution operates in. When in edit mode objects can be placed on the cards, resized and edited. When in run mode no editing can take place but buttons etc can be clicked and results observed.



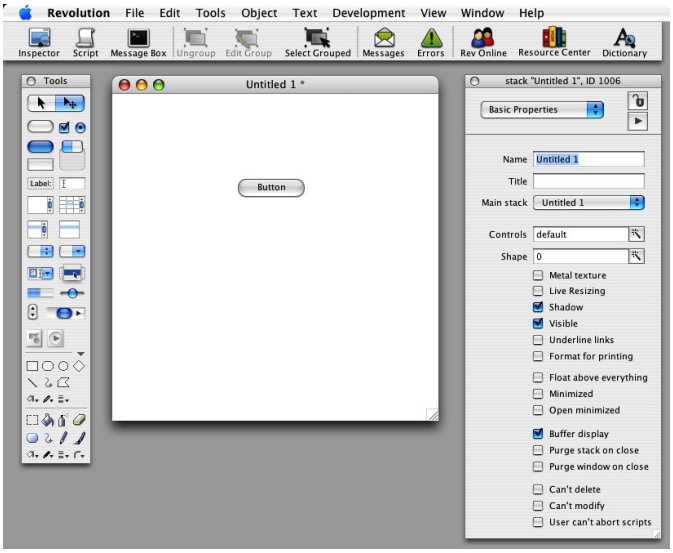
Object Properties

Every object in Revolution has associated properties which specify how it looks and behaves. These can be altered when Revolution is in edit mode and the easiest way to do this is using the Property Inspector. This can be opened by selecting the object and clicking the inspector button on the menu bar, by right clicking the object and selecting Property Inspector from the resulting menu or, alternatively, by double clicking the object itself.



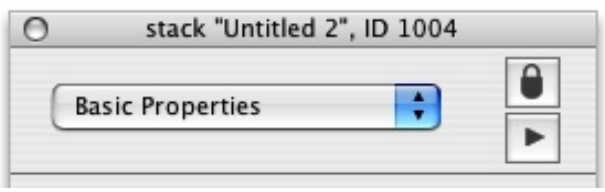
The Property Inspector

The Property Inspector displays all of the object's properties, from it's name and title to it's display settings. The best way to find out what a property does is to create a stack and alter it's settings, observing the results. Some of the more important features however are summarised below.



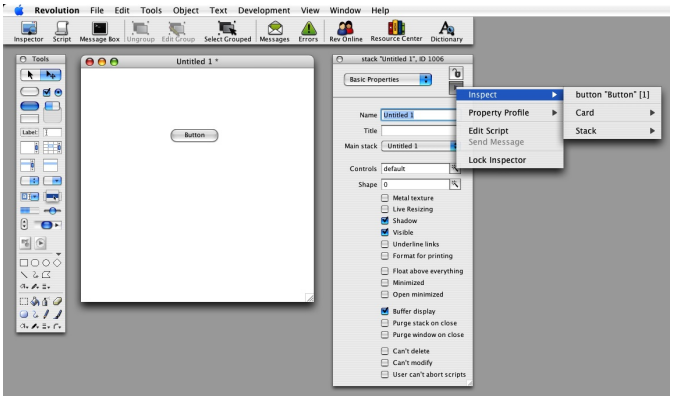
The Lock Palette

This locks the property inspector to the current object. If you request to see a different object's properties a new inspector will be opened, thus allowing you to view multiple object's properties at once.



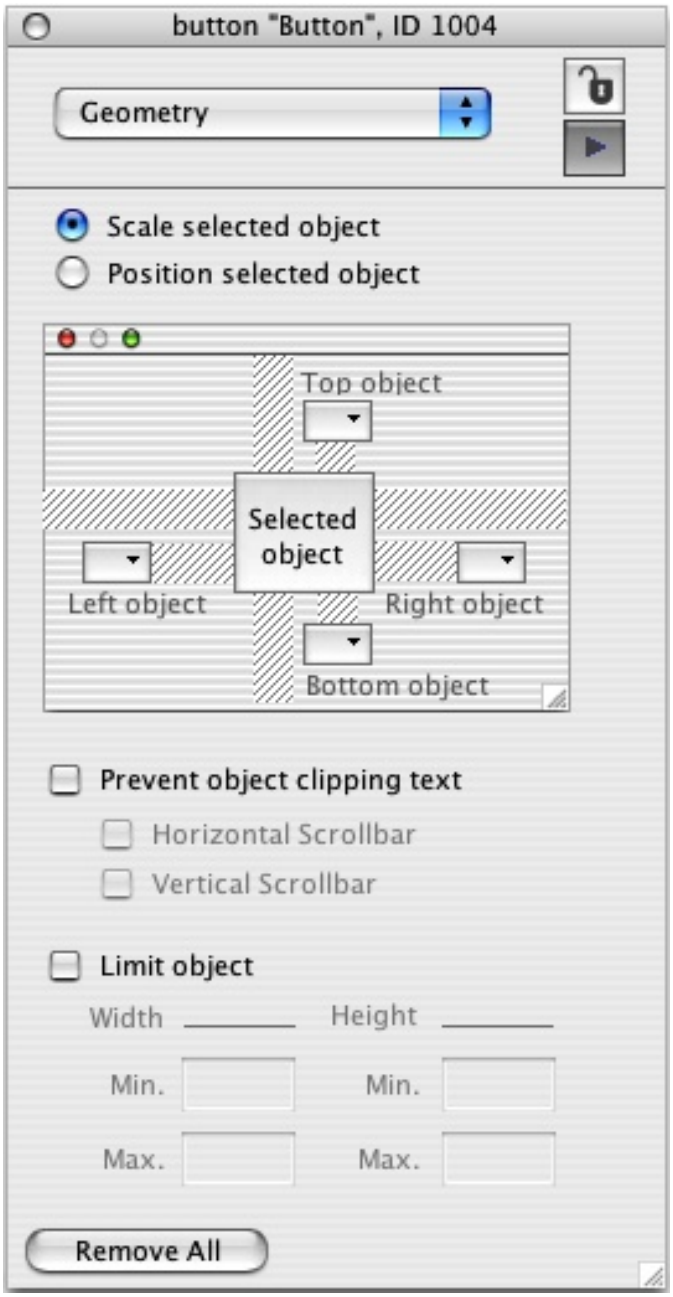
The Selection Menu

The selection menu lets you choose which object's properties you want to view. When clicked it will bring up a list which contains all the objects in your current application. When you select one of these the property inspector will update to show that object's properties.



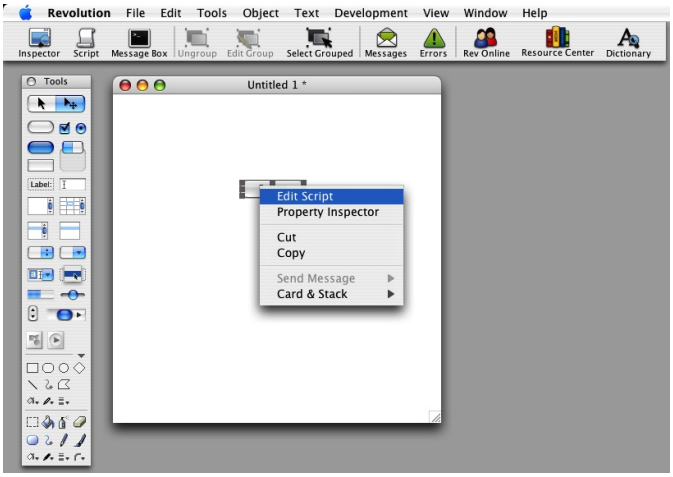
The Geometry Manager

The Geometry Manager is used to specify how objects should act when the stack they are contained within is resized. It lets you specify links between the selected object and either the side of the stack or another object within it. These links control how the selected object will resize and reposition itself when the stack size itself is changed. The links can be either relative or absolute, meaning that the selected object will always be a set number of pixels away from the other object or a set percentage away.



Writing Scripts

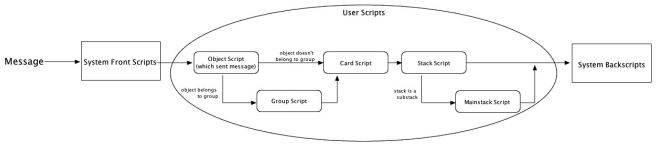
Once an object has been created and assigned properties it's behaviour can be specified by giving it a script.



Scripts in Revolution send and handle messages which control action. Some messages, such as mouseUp, are sent automatically by the system while other, user messages, are sent by making what is equivalent to a method call. The messages an object sends and how it handles them specify how it reacts to events and generally behaves.
Within Revolution all scripts are written in the script editor. The easiest way to view this is to ensure the object whose script you want to edit is in focus and then click the script button on the menubar.
Alternatively the object can be right-clicked and the 'edit script' option can be chosen from the menu which appears.

Messages and the Message Path

All messages sent in Revolution are passed along a message path until a handler for the message is found. The path contains any scripts which have permission to deal with the message being passed and consists of two sets of system scripts and any user scripts which are related to the object which sent the message. The order of the path is fixed to control who/what gets access to the message first and ensures standard behaviour is adhered to.

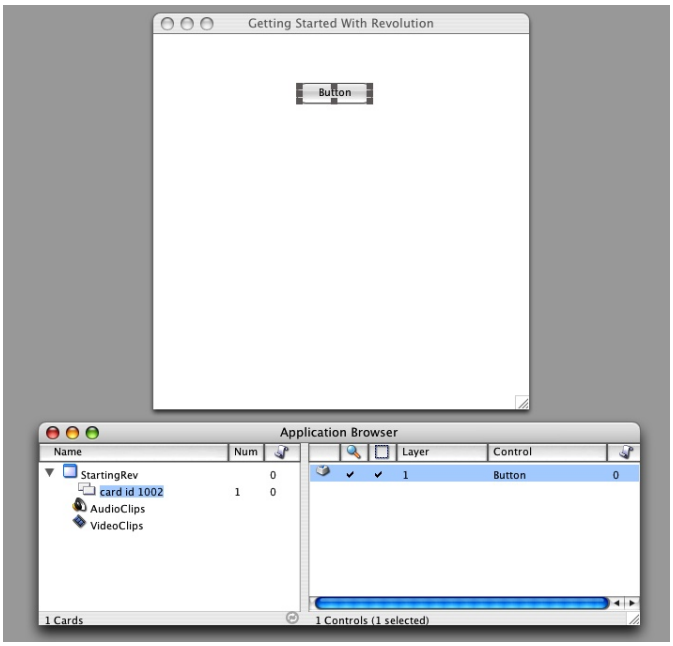


The first stop on the message path is always the system front scripts. These deal with any system messages which the user, typically, should not change the behaviour of. If a handler is not found in the front scripts the script of the object which the message was sent from is checked. If that also doesn't have a handler the message is passed to the script of any group which the object is a member of. If the object is

not in a group, or if no handler was found, the message is next passed to the card script and then the stack script which the object is a member of. If a handler is not found in these the system back scripts are checked. These scripts handle system messages which the user may change the behaviour of if they choose. If, at this point, a handler still hasn't been found it means that it does not exist and an error will be thrown.

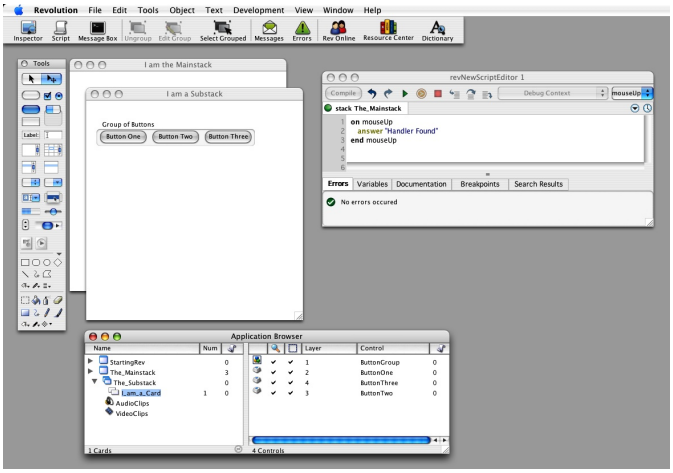
The Application Browser

In order to see the path which a message will take through the user scripts the Application Browser can be used. This displays a list of stacks and any cards, substacks, groups or objects which are placed within them.



All the stacks are displayed in the left-hand column and can be expanded to show the substacks and cards they include. When a card within the stack is clicked it's contents are displayed in the right-hand column of the Application Browser. Every object on the card will be displayed as well as details of what it is, whether it is visible, whether it can be selected, the layer it is on and how many lines of script it has. If you want to view the system scripts as well as the user scripts in the Application Browser you should select Revoltuion UI Elements in List from the View menu.

Message Path Example



Imagine one of the buttons in the screenshot to the right is clicked by a user. This will cause a mouseUp button to be sent by the Revoltuion engine. This message must be handled somewhere in the application. Let's assume that the only handler in the user scripts is the one in the script of The_Mainstack, as shown in the screenshot. The path the message will take through the application is outlined below.

The message will firstly be passed to the Revolution Front Scripts where a handler will not be found. Next it will go to the script of the button which was clicked. We can see from the Application Browser that the scripts of the buttons are all empty so we know that a handler also won't be found here. From the shot of the stack we can see that the buttons are all part of a group, ButtonGroup. The script of this will therefore be the next stop on the message path followed by the card script and then the script of the stack which the button is placed on, in this case The_Substack. At this point a handler has still not been found. Checking the application browser we can see that The_Substack is a substack of The_Mainstack, as it is indented underneath The_Mainstack. Therefore, instead of passing the message to the backscripts it will instead be passed to The_Mainstack. We know this has a handler so the message path will stop here. If The_Mainstack did not have a handler the message would have been passed to the Revolution Back Scripts which contain a mouseUp handler and would have stopped an error being thrown.