Computers & the Humanities 281

Scripts in Revolution

Up to this point, we have focused on the latter half of hypermedia, how to display the various bits of information that we wish to share. We will now turn to that portion of Revolution that allows us to transform our media into hypermedia.

As you may have divined already, all objects in Revolution possess a script, whether that script is populated with handlers or not. However, for the purposes of this lecture, the discussion will initially be limited to scripts for buttons (as most of the scripting you will do will be for buttons anyway). Keep in mind that most (if not all) of the concepts presented here, though applied to buttons, may apply to any other object.

Definitions and Tools

Script

The script of a button is where you put the commands to specify what the button does. To edit a button's script:

- With the object selected, click the **Script** icon in the button bar.
- With the object selected, click the selection menu (circle w/triangle) in the properties inspector and select **Edit Script**.
- Press the Command-option (Mac) or Control-alt (Windows) keys and move the cursor over the button. This key combination works with both the pointer (edit) and browse tools.

Transcript

The commands are written in Revolution's programming language, Transcript. Transcript has the general functionality of a stand-alone programming language like BASIC, Pascal or C, plus specific functionality to control the Revolution environment.

Handler

A Handler is a series of Transcript commands, roughly equivalent to a program or subroutine in traditional programming languages. A handler specifies the commands to "handle" a particular event (usually mouse events). It always starts with `on` *`eventName`* and ends with `end` *`eventName`*, with the transcript commands listed between the two. All these statements together comprise a handler.

```
on userClicksMouse
    do this
    do this and
    do that
end userClicksMouse
```

Fortunately, Revolution's script editor provides helps that speed up the scripting process:

- When you start typing the handler name (e.g., `mouseUp`) a list of standard Transcript terms appears underneath the editing area. Press the Command key (Mac) or the Control key (Windows) and the number corresponding to the handler message you want to use and the handler name will be automatically completed for you ("Autocomplete" in Script Editor section below).
- After you select or type the message name and press return, the "end" statement is automatically entered for you, and the cursor is placed between the two waiting for the commands to be entered.

Note: Sometimes the distinction between *script* and *handler* blurs in casual conversation. Strictly speaking, *handler* is proper term for the individual Transcript programs contained within a script, while *script* is the container where all handlers are defined and reside, i.e., a script can contain several handlers. Speaking metaphorically, a handler is a recipe and the script of an object is a cookbook.

Events and Messages

Events are actions or occurances usually instigated by the user. For buttons, the most common event is a **MouseUp**. This is defined as when the mouse button goes up while the cursor is located over the button. Events generate messages that are then sent to the appropriate object and cause that object to do something. By clicking on a button (a MouseUp event), a MouseUp message is generated and sent to the button. If the button possesses a MouseUp handler in its script, it then executes the commands contained within the handler, and the button is then seen as "doing something."

Perhaps this process could be seen more clearly by extending the cookbook metaphor. A patron arrives at a restaurant and orders biscuits and gravy (the event). The waiter delivers the order (the message) to the cook (the object) who checks his mental cookbook (the script) and finds the appropriate recipe for that order (the handler). The cook then proceeds to make the biscuits and gravy to order (Transcript commands in the handler). Strictly speaking, the cook's actions are instigated by the patron's request, much like a button acts at the user's request.

Script Editor

The script editor is Revolution's special built-in word processor for editing scripts.

- Basic word processing: type/select, cut/copy/paste, undo, etc.
- Find and replace capabilities similar to those found in word processors.
- **Print Script** can be useful for documenting handlers.
- Exit by clicking the close box or choosing **Close Script Editor**, or **Apply Script and Close** from the **File** menu.
- Some simple shortcuts:

- Enter = apply (save) changes
- Enter when no changes to save = close the script editor
- Tab key = format the current handler
- Right click (Windows) or Control-click (Mac) on a specific word to look it up in the Transcript dictionary.

Some of the more useful and commonly used features of the Script Editor are:

- You can change properties of the script editor using the buttons at the top of the editor palette. (Font style, wrap, maximize/minimize.)
- The bottom section of the script editor contains one of three utilities: Find and Replace, Go to Line, or Autocomplete. You can rotate through these utilities using the button at bottom right just above the Apply button.
- Use Comment and Uncomment to insert or remove hyphens or pound signs (depending on what the comment character is set to.)
- Lines are automatically indented to show logical structure.
- Long lines can wrap or not depending on how you've set the wrap setting in the script editor.
- Use option-return to force a line break. (Appears as ¬) Lines may be split anywhere between words except:
  - within a literal text string (i.e. inside quotes)
  - within a comment (i.e. after double hyphen)

Message Palette

While the script editor is useful for writing "permanent" handlers for buttons, it is often handy to be able to execute a Transcript command immediately. This capability is provided with the Message Palette.

- Choose Message from the Tools menu. Command (Apple)-M (Mac) or Control-M (Windows) is the keyboard shortcut.
- Type the command in the top field message box and press Return. If there is a result or error it will show up in the bottom field.

Some Common Transcript Commands

Here are some of the most basic and common commands that you will be using in your scripting. See the **Transcript Dictionary** in the Revolution **Help** menu for more details, or refer to the online documentaion.

Go

The `go` command is used to cause another card to be displayed, i.e., to navigate between cards, either within the current stack or to another stack. To move sequentially through cards, the following commands work:

- `go to next card`
- `go to previous card`

In the preceeding commands, `to` and `card` are optional in the syntax and may be omitted with impunity. To display (or move to) a card not contiguous with the current card, you may refer to a specific card by its name or number, or by using reserved words. For example:

- `go to card "map"`
- `go card 7`
- `go to first card`
- `go last`

As before, `to` is optional, but `card` must be included in order to identify the object to which we're referring.

Visual Effect

The visual effect command specifies a visual transition for Revolution to use as it displays (or goes to) another card. The default plain visual effect causes the next card to appear immediately. There are several provided in Revolution. Examine the Transcript Dictionary or complete details and all the options. Here are a few examples:

- `visual effect dissolve`
- `visual iris close to black`
- `visual effect scroll right`
- `visual effect scroll left`
- `visual effect wipe up`
- `visual effect wipe down`
- `visual effect checkerboard`
- `visual effect venetian blinds`
- `visual effect dissolve to white`
- `visual effect reveal up to inverse`

Visual effects are used in conjunction with a `go` command. They do not happen when you use arrow keys or **View** menu commands. Here is an example script of a visual effect used properly:

```
on mouseUp
    visual effect push right fast
    go next card
end mouseUp
```

The use of visual effects should be limited to conveying information (for a very specific purpose) and not just because "it looks cool." Visual effects used indiscriminately or without purpose are a distraction rather than an enhancement.

Hide/Show

The hide and show commands allow you to make fields, buttons, and other Revolution objects appear or disappear. This command essentially toggles the visible property for an object.

- `hide field "demo field"`
- `show field "demo field"`
- `hide button "demo button"`
- `show group "demo group"`

Hidden objects not only cannot be seen but also cannot be clicked. You can see all invisible objects by choosing **Show Invisible Objects** from the **View** menu.

Put

The put command is used to place text into a field.

- `put "any text" into field "demo"`
- `put empty into field "demo"`
- `put the date into field "demo"`
- `put the long time`

If no field name is specified, the text goes into the message box.

Wait

The wait command causes a delay or pause in a script. It will either wait for a specific amount of time or for a particular event.

- `wait 3 seconds`
- `wait 1 second`
- `wait 20 ticks`
- `wait 500 milliseconds`
- `wait until the mouseClick`

A `tick` is 1/60th of a second. The default time unit is `ticks`. Thus, `wait 60` is the same as `wait 1 second`. A millisecond is 1/1000th of a second. The following handler displays a pop-up field:

```
on mouseUp
    show card field "PopUp"
    wait 3 seconds
    hide card field "PopUp"
end mouseUp
```

Beep

The beep command sounds the "alert" tone of the computer a specified number of times. If no number is specified it beeps once.

- `beep`
- `beep 3`

The "alert" sound is the sound used by the operating system for error or attention situations. The alert sound is set at the system level, external to Revolution. Although the default is a beep sound, it may be customized to any sound, and often is. All applications conventionally use this sound to alert the user to an error or attention condition. A wise Revolution designer will do the same.

Comments

Comments serve to document your scripting. Despite the fact that Transcript reads so much like English that many statements are virtually self-documenting, you should still use comments to mark major sections and clarify anything complicated. Revolution ignores comments, but they can sure help a human understand what's going on.

Any text following two hyphens (--) or a pound sign (#) is treated as a comment, i.e. ignored when the handler is executed. The default comment symbol can be set in the Script Editor preferences. Revolution recognizes either symbol as a comment, even though you may have selected one or the other as default. For example:

- `-- a comment may document handler logic`
- `beep 3 -- or explain a particular command`
- `--beep 3 -- or prevent a line from executing`
- `# beep 3 # this is also a valid comment format`

It is considered good practice to document your scripting with comments. You may be surprised how easily you forget how or why you scripted the way you did, especially when you come back to the script months after the fact. Comments serve as useful and helpful reminders and may prevent many hours of weeping, wailing, and gnashing of teeth.

Scripts Exercise
Course Schedule
Main Page