HyperTalk: The Language for the Rest of Us

Kyle Wheeler

January 18, 2004

Contents

1 Introduction **1**

2 History **1**
- 2.1 The Birth . . . 1
- 2.2 The Life . . . 2
- 2.3 The Death . . . 2
- 2.4 The Legend . . . 2

3 Goals **2**

4 Syntax Semantics **3**
- 4.1 Implementation Notes . . . 3
- 4.2 Objects . . . 3
- 4.3 Messages . . . 4
- 4.4 Handlers . . . 4

5 Bibliography **4**

A BNF **6**
- A.1 Scripts . . . 6
- A.2 Expressions . . . 6
- A.3 Ordinals and Positions . . . 7
- A.4 Chunks and Containers . . . 7
- A.5 Objects . . . 7
- A.6 Commands . . . 8
  - A.6.1 Command Nonterminals . . . 8
  - A.6.2 Commands . . . 9
  - A.6.3 *set* Command Syntax . . . 11
- A.7 Functions . . . 11

Abstract

In 1976, Apple Computer, Inc. released the Apple I and created the personal computing industry. In 1984, Apple released the first Macintosh computer, revolutionizing the personal computing industry. In 1987, Apple released HyperCard and HyperTalk, and tried to revolutionize the personal computing industry again.

1 Introduction

There is, perhaps, no piece of software written by Apple Computer, Inc. more prone to generating extreme emotions in its users than its operating system. Next below that, however, is HyperCard. Designed and released in 1987 by Bill Atkinson [7], HyperCard was an instant success. Leveraging the power and simplicity of its scripting language, HyperTalk, designed by Bill Atkinson and by Dan Winkler [1], HyperCard demystified the art of creating software. The language has a grammar and syntax similar to English, and as such appealed to computer hobbyists, teachers, and the uninitiated alike. The commands HyperTalk uses are similar to those used by the Macintosh Toolbox, the base-level API of Apple's Macintosh operating system, and the logical structure is similar to Pascal and organized in an event-driven manner [8].

2 History

2.1 The Birth

HyperTalk was born as the core scripting language of the HyperCard application, developed by Bill Atkinson[1] for Apple Computer, Inc. in 1987 under the condition that it must be available for free on all Macs. Originally, the application was named "WildCard" (and the language "WildTalk", respectively), however, the name was changed because of legal issues. Atkinson was inspired to explore new interface technologies by Xerox's Palo Alto Research Center and their SmallTalk language.

Quickly, the application and the language became very popular. The language was easy to learn and drew many people into programming computers for all sorts of purposes, from basic animation, to automation, to creating large databases.

[1] Key developer of QuickDraw and MacPaint, an Apple Fellow, and founder of GeneralMagic. Currently a high-resolution nature photographer.

2.2 The Life

Because Apple was under obligation to Bill Atkinson to provide HyperCard for free, the company found it difficult to justify devoting employees to developing HyperCard further. Regardless, HyperCard and HyperTalk became very popular, spawning a bimonthly magazine (HyperLink), and more than a few books. HyperCard "stacks" began to be sold alongside more traditional Macintosh programs in mail-order catalogs.

Eventually, in 1989, the internal political environment of Apple Computer changed under pressure from Kevin Calhoun[2] (a programmer at Apple), and HyperCard and HyperTalk underwent a massive improvement that resulted in HyperCard 2.0 (and a revised and more consistent version of HyperTalk) which was released in 1990. Further improvements, like support for a color interface, were announced as being under development. Third-party vendors developed thousands of applications based on HyperCard, and in addition, thousands of XCMDs (external commands to extend HyperTalk to control additional things or to provide certain functionality) "for everything from HyperTalk compilers, to graphing systems, database access, internet connectivity and practically everything else" [7]. HyperCard was even used, before the introduction of PowerPoint, as a general-purpose presentation generator.

Shortly thereafter, however, Apple Computer reorganized, and spun its software division off to create the Claris company—outsourcing even the Macintosh Operating System. This was a disaster for the company. The OS was returned to Apple, and HyperCard, after some minor updates to fulfill promises of color support, was apparently forgotten.

2.3 The Death

HyperCard was finally rolled into Apple's QuickTime group (as it seemed to be multimedia-related), and began to be developed into a QuickTime development platform under the direction of Kevin Calhoun. The result of this development, HyperCard 3.0, was presented and distributed in 1996 at the annual Apple Word-Wide Developer's Conference as a beta and sneak-preview of things to come. This version of HyperCard/HyperTalk had an impressive array of new features, including internet connectivity, and the ability to be displayed in a web browser or QuickTime viewer (somewhat similar to Flash, by Macromedia). The new version was never released, and the lead-developer, Kevin Calhoun, left Apple in 2001 [7]. Without a champion in Apple, or apparent support from Apple's management, HyperCard and HyperTalk languished and became less popular. HyperCard is still available for sale on Apple's website, but has not received an update since the mid-nineties.

[2]Calhoun left Apple in 2001 to form his own company, 4R Software [20].

2.4 The Legend

HyperCard clones, which also used the HyperTalk scripting language, were developed in the absence of Apple's commanding lead. These clones and descendants included SuperCard (for the Macintosh)[3], Toolbox (for Microsoft Windows)[4], MetaCard (for Windows, Mac, and Unix/X11)[5], WinPlus (for Microsoft Windows), HyperSense (originally for NeXT), FreeCard (an open-source clone), PLUS (for Mac, Windows, and OS/2)[6], HyperStudio (for Mac and Windows)[7], LinkWay (for DOS)[8], and a cross-platform OracleCard from Oracle[9]. [7] The ideas embodied in HyperTalk, and even much of the syntax, was also used by Macromedia in their Director and Authorware products as the Lingo scripting language.

While HyperCard was dead, HyperTalk maintained popularity within the ranks of Apple's engineering core. In 1993, Apple engineers developed a mechanism standard called Apple's Open Scripting Architecture, which standardized a generic way for programs to respond to script calls ("Apple Events"). This allowed the development of a slightly modified HyperTalk language, called AppleScript, that was generic enough to be a cross-application OS-level scripting language (allowing programs from many vendors to be controlled by and accept high-level user commands from the operating system) [10]. The language and basic grammar was even translated into other languages, including English, French, Japanese, and Italian [6]—although the feature was dropped with the introduction of Mac OS 8.5 on October 17, 1998 [11]. AppleScript itself lived on as an popular way to manage work-flow and automate operating system tasks.

3 Goals

> *"HyperCard is a descendant of two ideas. One was the give-away Rolodex program that I wrote just to keep track of my own journal articles. The other was a research project I did on what the new generation computer should [be] … "*
>
> —Bill Atkinson [3]

[3]by Silicon Beach Software, now SolutionsEtcetera[21]
[4]by Asymmetrix, now defunct
[5]now known as Runtime's Revolution[4, 17]
[6]by Format Software in Germany, now defunct [15]
[7]by Roger Wagner Publishing [12]
[8]by Larry Kheriaty and eventually IBM [14]
[9]Later renamed "Oracle Media Objects"[18]

Algorithms aside, what probably first inspired both HyperCard and HyperTalk is the so-called Macintosh dream. As Atkinson says, "The Macintosh dream has really been putting the power of the personal computer into an individual person's hands." While the general applications of the time were getting much easier to use and didn't require memorization of control-characters and command sequences, Atkinson felt that the power of program creation still lay outside the individual person's ken—that building useful and helpful programs still required arcane knowledge of the computer's internals or of some obtuse mathematical constructs. To that end, HyperCard with HyperTalk was an attempt to make programming accessible to anyone. As Atkinson said, "The most exciting thing for me is when I see people amazed and pleased at the newfound power they got from a program—when they say, 'Wow, I can do this!'... It's the original Macintosh dream of making the power of personal computer accessible to individuals. HyperCard is just unfolding another layer of Macintosh." [3]

HyperCard and HyperTalk were particularly important to people invested in "hypertext", a concept that was developed in Stanford in the 1960's as a format for creative information grouping [19]. HyperCard was hailed as a convenient demonstration of the power and utility of linked and grouped information (and more accessible to the common user than older hypertext projects like the Xanadu project), and many of the first HyperCard "stacks" were used for precisely that purpose.

4 Syntax Semantics

The basic design of HyperTalk is as a message-passing language, generating and handling messages (or events) between objects. How an object responds to messages that are sent to it depends on the script attached to it. With the advent of AppleScript and the Open Scripting Architecture, messages (Apple Events) can even be passed to other applications (which are treated as remote objects).

Scripting in HyperTalk, unlike most programming languages, is extremely easy for non-programmers to understand because its syntax is so similar to English. The common example of how readable HyperTalk is is the phrase:

```
put the first word of the third line of field
'hello' into field 'goodbye'
```

...which does exactly what it seems to. In order to achieve this kind of readability and apparent simplicity requires a lot of what is frequently termed "syntactic sugar." For example, numbers have many synonyms: `1` and `2` can be replaced with `one` and `two` or even `first` and `second`. For similar reasons, HyperTalk is untyped, allowing code like this to work [13]:

```
ask "What number do you want to square?"
put it * it into field "Answer"
```

Also, many otherwise complex actions—such as dialing the modem, displaying a file-browser dialog box, or getting information about the system the script is running on—is abbreviated, abstracted, and made available to the script with simple statements. Because of this incredible built-in power and verbosity, the language and the list of keywords in the language is quite vast.

4.1 Implementation Notes

In many ways, HyperTalk—particularly in the beginning—depended heavily on the programming structure of HyperCard. HyperCard is frequently referred to as presenting itself like a stack of index cards. HyperCard projects are called "stacks" of "cards" to encourage that perception. Cards are containers for other objects like buttons, pictures, and text fields. The most basic HyperTalk scripts were used for defining transitions from card to card, and were associated with particular objects, such as a button or a card or a text field. A HyperTalk script or function would be triggered by an event sent to that object, such as a `mouseUp`. As more and more events are added to each object, much more active programming can be accomplished. Actions are mostly accomplished by sending events to other objects, all of which could be named and numbered for easy reference. The commands HyperTalk uses in addition to message passing are similar to those used by the Macintosh Toolbox, the base-level API of Apple's Macintosh operating system, however most common tasks (such as displaying dialog boxes[10]) have been simplified.

The HyperTalk script is normally saved in plain text form in the stack, although HyperCard 2.4 capable of compiling it to a binary executable.

4.2 Objects

Officially, HyperCard supports five kinds of objects: buttons, fields, cards, backgrounds, and stacks—although applications can behave as a sixth kind of object. The distinction between buttons, fields, and backgrounds is only in how they present themselves in the graphic user interface—buttons as uneditable and clickable-looking things, fields as text blocks or graphics, and backgrounds as inert pictures; the HyperTalk capabilities of each and which messages they can receive are roughly identical. [10]

[10] For example: `answer "This is displayed." with "Aha."` or `ask "What is your name?"`

4.3 Messages

Messages come in two flavors: system messages and commands. System messages are defined by the environment, and are generated in response to user actions such as mouse clicks and key presses or environmental changes such as the time. Commands are arbitrarily named messages that are defined by the HyperTalk script-writer [10]. Messages are generally sent in the following manner:

`send ⟨message⟩ to ⟨object⟩`

4.4 Handlers

There are two kinds of execution blocks, or handlers. The first is a message handler, which is executed whenever the object the script is attached to receives a message of the corresponding name. In the script, The other kind of handler is the function handler [10]. Message handlers are defined like so:

```
on ⟨messageName⟩
   script statements
end ⟨messageName⟩
```

Function handlers are similar:

```
function ⟨functionName⟩
   script statements
end ⟨functionName⟩
```

5 Bibliography

References

[1] HyperTalk from FOLDOC, December 2003. http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?HyperTalk.

[2] Steve Collins. International HyperCard User Group (iHUG). Web, December 2003. http://www.ihug.org/index.html.

[3] Quick Connect. The Genius behind HyperCard: Bill Atkinson. *Known Users*, November 1987. http://www.savetz.com/ku/ku/quick_genius_-behind_hypercard_bill_atkinson_the_november_-1987.html.

[4] MetaCard Corporation. MetaCard: Cross Platform Multimedia Authoring and Application Development, December 2003. http://www.metacard.com/.

[5] Peter Flecks and Pantechnicon. HyperCard FAQ, December 2003. http://pan.uqam.ca/cgi-bin/usemod/wiki.pl?HyperCard_FAQ.

[6] Wikimedia Foundation. AppleScript. In *Wikipedia: The Free Encyclopedia* [9]. http://en2.wikipedia.org/wiki/AppleScript.

[7] Wikimedia Foundation. HyperCard. In *Wikipedia: The Free Encyclopedia* [9]. http://en2.wikipedia.org/wiki/HyperCard.

[8] Wikimedia Foundation. HyperTalk. In *Wikipedia: The Free Encyclopedia* [9]. http://en2.wikipedia.org/wiki/HyperTalk.

[9] Wikimedia Foundation. *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation, December 2003. http://en2.wikipedia.org/wiki/Main_Page.

[10] Apple Computer Inc. *HyperCard: Script Language Guide*. Apple Computer Inc., 20525 Mariani Avenue, Cupertino, CA, 2nd edition, 1993.

[11] Cobweb Publishing Inc. Apple Timeline, 1996-2000. Web, December 2003. http://www.lowendmac.com/time/1996-00.shtml.

[12] Roger Wagner Publishing Inc. HyperStudio 4 Product Information, December 2003. http://www.hyperstudio.com/hs4/index.html.

[13] Jeanne A. E. DeVoto (jaed@jaedworks.com). HyperTalk BNF. Web, December 2003. http://www.jaedworks.com/hypercard/scripts/hypertalk-bnf.html.

[14] Christopher Keep, Tim McLaughlin, and Robin Parmar. LinkWay. In *The Electronic Labyrinth* [16]. http://www.iath.virginia.edu/elab/elab.html.

[15] Christopher Keep, Tim McLaughlin, and Robin Parmar. PLUS. In *The Electronic Labyrinth* [16]. http://www.iath.virginia.edu/elab/hfl0024.html.

[16] Christopher Keep, Tim McLaughlin, and Robin Parmar. *The Electronic Labyrinth*. University of Washington, December 2003. http://www.iath.virginia.edu/elab/elab.html.

[17] Rod McCall. Runtime Revolution - User-Centric Software Development. Web, December 2003. http://www.runrev.com/index_uk.html.

[18] Inc. Oracle. Oracle FAQ: Oracle Media Objects. Web, December 2003. http://www.orafaq.com/faqomo.htm.

[19] Brenton R. Schlender. New Software Beginning to Unlock The Power of Personal Computers. *Wall Street Journal*, (2):27, November 16 1987. http://www.ihug.org/WSJ87.html.

[20] M.S.R.F. Schonewille. Re: [HyperCard] Kevin Calhoun, June 2001. http://groups.yahoo.com/group/HyperCard/message/6353.

[21] SolutionsEtcetera.com. SuperCard 4. Web, December 2003. http://www.supercard.us/.

[22] Derrick Story. The Death of HyperCard? Web, March 2001. http://www.macdevcenter.com/pub/a/mac/2001/03/29/mac_-dev.html.

[23] Dan Winkler, Scot Kamins, and Jeanne DeVoto. *HyperTalk 2.2: The Book*. Random House, 1994.

A BNF

The BNF description of the HyperTalk language was published in *HyperTalk 2.2: The Book* [23], and describes the language thusly[11], as cited by [13].

A.1 Scripts

⟨**script**⟩ = ⟨script⟩ ⟨handler⟩ | ⟨handler⟩

⟨**handler**⟩ = ⟨return⟩ ⟨handler⟩ | on ⟨messageKey⟩ ⟨return⟩ ⟨stmntList⟩ eol end ⟨messageKey⟩ ⟨return⟩

⟨**ifBlock**⟩ = if ⟨logical⟩ [⟨return⟩] then {⟨singleThen⟩ | ⟨return⟩ ⟨multiThen⟩}

⟨**singleThen**⟩ = ⟨stmnt⟩ [[⟨return⟩] ⟨elseBlock⟩]

⟨**multiThen**⟩ = ⟨stmntList⟩ { end if | ⟨elseBlock⟩ }

⟨**elseBlock**⟩ = else { ⟨stmnt⟩ | ⟨return⟩ ⟨stmntList⟩ end if }

⟨**repeatBlock**⟩ = repeat [forever | ⟨duration⟩ | ⟨count⟩ | with ⟨identifier⟩ = ⟨range⟩] ⟨return⟩ ⟨stmntList⟩ end repeat

⟨**duration**⟩ = until ⟨logical⟩ | while ⟨logical⟩

⟨**count**⟩ = [⟨for⟩] ⟨unsigned⟩ [times]

⟨**range**⟩ = ⟨integer⟩ [down] to ⟨integer⟩

A.2 Expressions

⟨**expr**⟩ = ⟨source⟩ | - ⟨expr⟩ | not ⟨expr⟩ | ⟨expr⟩ ⟨op⟩ ⟨expr⟩ | (⟨expr⟩) | ⟨chunk⟩ ⟨expr⟩ | there is { a | an | no } ⟨expr⟩

⟨**op**⟩ = + | - | * | / | & | && | ^ | = | < | > | <> | ≠ | <= | >= | ≤ | ≥ | and | or | contains | div | mod | is | is not | is in | is not in | is within | is not within | is a[n] | is not a[n]

⟨**source**⟩ = ⟨literal⟩ | ⟨constant⟩ | ⟨simpleContainer⟩ | [⟨adjective⟩] ⟨function⟩ | [⟨adjective⟩] ⟨property⟩ of {⟨object⟩ | ⟨window⟩ | ⟨menuItem⟩ of ⟨menu⟩ | ⟨chunk⟩ ⟨field⟩ }

⟨**literal**⟩ = “quoted string” | unquotedToken

⟨**constant**⟩ = down | empty | false | formFeed | lineFeed | pi | quote | space | tab | true | up | zero | one | two | three | four | five | six | seven | eight | nine | ten

⟨**adjective**⟩ = long | short | abbrev | abbr | abbreviated

⟨**window**⟩ = [the] { card | pattern | tool | scroll } window | ⟨messageBox⟩

⟨**menuItem**⟩ = ⟨ordinal⟩ menuItem | menuItem ⟨expr⟩

⟨**menu**⟩ = ⟨ordinal⟩ menu | menu ⟨expr⟩

⟨**function**⟩ = the ⟨theFunc⟩ | [the] ⟨theFunc⟩ of ⟨oneFuncArg⟩ | ⟨identifier⟩ (⟨funcArgs⟩)

[11] There may be some omissions due to the breadth of the language.

⟨**theFunc**⟩ = abs | annuity | atan | average | charToNum | clickChunk | clickH | clickLine | clickLoc | clickText | clickV | cmdKey | commandKey | compound | cos | date | diskSpace | exp | exp1 | exp2 | foundChunk | foundField | foundLine | foundText | heapSpace | length | ln | ln1 | log2 | max | menus | min | mouse | mouseClick | mouseH | mouseLoc | mouseV | number | numToChar | offset | optionKey | param | paramCount | params | programs | random | result | round | screenRect | seconds | selectedButton | selectedChunk | selectedField | selectedLine | selectedLoc | selectedText | shiftKey | sin | sound | sqrt | stacks | stackSpace | sum | systemVersion | tan | target | ticks | time | tool | trunc | value | windows

⟨**property**⟩ = address | autoHilite | autoSelectautoTab | blindTyping | botRight | bottom | bottomRight | brush | cantAbort | cantDelete | cantModify | cantPeek | centered | checkMark | cmdChar | commandChar | cursor | debugger | dialingTime | dialingVolume | dontSearch | dontWrap | dragSpeed | editBkgnd | enabled | environment | family | filled | fixedLineHeight | freeSize | grid | height | highlight | highlite | hilight | hilite | icon | id | itemDelimiter | language | left | lineSize | loc | location | lockErrorDialogs | lockMessages | lockRecent | lockScreen | lockText | longWindowTitles | markChar | marked | menuMessage | menuMsg | messageWatcher | multiple | multipleLines | multiSpace | name | numberFormat | owner | partNumber | pattern | polySides | powerKeys | printMargins | printTextAlign | printTextFont | printTextHeight | printTextSize | printTextStyle | rect | rectangle | reportTemplates | right | script | scriptEditor | scriptingLanguage | scriptTextFont | scriptTextSize | scroll | sharedHilite | sharedText | showLines | showName | showPict | size | stacksInUse | style | suspended | textAlign | textArrows | textFont | textHeight | textSize | textStyle | titleWidth | top | topLeft | traceDelay | userLevel | userModify | variableWatcher | version | visible | wideMargins | width | zoomed

A.3 Ordinals and Positions

⟨**ordinal**⟩ = [the] { last | mid | middle | any | first | second | third | fourth | fifth | sixth | seventh | eigth | ninth | tenth }

⟨**position**⟩ = this | [the] prev | [the] next

A.4 Chunks and Containers

⟨**simpleContainer**⟩ = ⟨variable⟩ | ⟨part⟩ | ⟨menu⟩ | ⟨messageBox⟩ | [the] selection

⟨**container**⟩ = ⟨chunk⟩ ⟨simpleContainer⟩ | ⟨simpleContainer⟩

⟨**messageBox**⟩ = [the] msg [box | window]

⟨**chunk**⟩ = [{⟨ordinal⟩ char | char ⟨expr⟩ [to ⟨expr⟩]} of] [{⟨ordinal⟩ word | word ⟨expr⟩ [to ⟨expr⟩]} of] [{⟨ordinal⟩ item | item ⟨expr⟩ [to ⟨expr⟩]} of] [{⟨ordinal⟩ line | line ⟨expr⟩ [to ⟨expr⟩]} of]

A.5 Objects

⟨**object**⟩ = [12] HyperCard | me | [the] target | ⟨button⟩ | ⟨field⟩ | ⟨card⟩ | ⟨bkgnd⟩ | ⟨stack⟩

⟨**button**⟩ = {button id ⟨unsignedFactor⟩ | button ⟨factor⟩ | ⟨ordinal⟩ button} [of ⟨card⟩]

⟨**field**⟩ = {field id ⟨unsignedFactor⟩ | field ⟨factor⟩ | ⟨ordinal⟩ field} [of ⟨card⟩]

⟨**part**⟩ = ⟨button⟩ | ⟨field⟩ | {part id ⟨unsignedFactor⟩ | part ⟨factor⟩ | ⟨ordinal⟩ part} [of ⟨card⟩]

⟨**card**⟩ = recent card | back | forth | card id ⟨unsigned⟩ | card ⟨expr⟩ | card ⟨endLine⟩ | ⟨ordinal⟩ card | ⟨position⟩ card} [of ⟨bkgnd⟩] | ⟨ordinal⟩ marked card | ⟨position⟩ marked card | marked card ⟨expr⟩

⟨**bkgnd**⟩ = bkgnd id ⟨unsigned⟩ | bkgnd ⟨expr⟩ | bkgnd ⟨endLine⟩ | ⟨ordinal⟩ bkgnd | ⟨position⟩ bkgnd

⟨**stack**⟩ = this stack | stack ⟨expr⟩ | stack ⟨endLine⟩

[12] Note: “card field 1” is a field and “card (field 1)” is a card.

A.6 Commands

A.6.1 Command Nonterminals

⟨**dateItems**⟩ = ⟨unsigned⟩, ⟨unsigned⟩, ⟨unsigned⟩, ⟨unsigned⟩, ⟨unsigned⟩, ⟨unsigned⟩, ⟨unsigned⟩

⟨**date**⟩ = ⟨unsigned⟩ | ⟨dateItems⟩ ⟨humanDate⟩ [⟨humanTime⟩] | ⟨humanTime⟩ [⟨humanDate⟩]

⟨**dateFormat**⟩ = [⟨adjective⟩] {seconds | dateItems | date | time}

⟨**dayOfWeek**⟩ = Sunday | Sun | Monday | Mon | Tuesday | Tue | Wednesday | Wed | Thursday | Thu | Friday | Fri | Saturday | Sat

⟨**dest**⟩ = { ⟨card⟩ | ⟨bkgnd⟩ } [of ⟨stack⟩] | ⟨stack⟩ | { ⟨card⟩ | ⟨bkgnd⟩ } of [⟨stack⟩] ⟨exprOrLine⟩

⟨**duration**⟩ = until ⟨logical⟩ | while ⟨logical⟩

⟨**humanDate**⟩ = [⟨dayOfWeek⟩ ,] ⟨month⟩ ⟨unsigned⟩ , ⟨unsigned⟩ | ⟨unsignedFactor⟩ {/ | -} ⟨unsignedFactor⟩ {/ | -} ⟨unsignedFactor⟩

⟨**humanTime**⟩ = ⟨unsigned⟩ : ⟨unsigned⟩ [: ⟨unsigned⟩] [am | pm]

⟨**month**⟩ = January | Jan | February | Feb | March | Mar | April | Apr | May | June | Jun | July | Jul | August | Aug | September | Sep | October | Oct | November | Nov | December | Dec

⟨**point**⟩ = {⟨integer⟩ , ⟨integer⟩}

⟨**preposition**⟩ = before | after | into

⟨**rect**⟩ = {⟨integer⟩ , ⟨integer⟩ , ⟨integer⟩ , ⟨integer⟩}

⟨**springKeys**⟩ = ⟨springKeys⟩ , ⟨springKey⟩ | ⟨springKey⟩

⟨**springKey**⟩ = shiftKey | optionKey | commandKey

⟨**style**⟩ = transparent | opaque | rectangle | roundrect | shadow | checkBox | radioButton | scrolling | oval | popup

⟨**textAlign**⟩ = right | left | center

⟨**textStyleList**⟩ = ⟨textStyleList⟩ ⟨textStyle⟩ | ⟨textStyle⟩

⟨**textStyle**⟩ = plain | bold | italic | underline | outline | shadow | condense | extend | group

⟨**visEffect**⟩ = ⟨visKind⟩ [[very] {slow | slowly | fast}] [to ⟨visSrc⟩]

⟨**visKind**⟩ = barn door {open | close} | cut | plain | dissolve | venetian blinds | checkerboard | iris {open | close} | scroll {left | right | up | down} | wipe {left | right | up | down} | zoom {open | out | close | in } | shrink to {top | bottom | center } | stretch from {top | bottom | center } | push {left | right | up | down}

⟨**visSrc**⟩ = card | black | white | gray | inverse

⟨**window**⟩ = {card | pattern | tool | scroll | fatBits} window | ⟨messageBox⟩

A.6.2 Commands

add ⟨arith⟩ to ⟨container⟩

answer ⟨expr⟩ [with ⟨factor⟩ [or ⟨factor⟩ [or ⟨factor⟩]]] | file ⟨expr⟩ [of type ⟨factor⟩ [or ⟨factor⟩ [or ⟨factor⟩]]] | program ⟨expr⟩ of type ⟨factor⟩

arrowkey left | right | up | down

ask { password | file } ⟨expr⟩ [with ⟨expr⟩ | ⟨line⟩]

beep [⟨unsigned⟩]

choose tool ⟨unsigned⟩ | { browse | button | field | select | lasso | pencil | brush | eraser | line | spray [can] | rect | round rect | bucket | oval | curve | text | reg poly | poly } tool

click at ⟨point⟩ [with ⟨springKeys⟩]

close file ⟨exprOrLine⟩ | printing | application ⟨exprOrLine⟩ | ⟨window⟩

commandKeyDown ⟨expr⟩

controlkey ⟨unsigned⟩

convert { ⟨container⟩ | ⟨date⟩} [from ⟨dateFormat⟩ [and ⟨dateFormat⟩]] to ⟨dateFormat⟩ [and ⟨dateFormat⟩]

copy template ⟨expr⟩ to ⟨stack⟩

create stack ⟨expr⟩ [with ⟨bkgnd⟩] [in [a] new window] | menu ⟨expr⟩

debug hintBits | pureQuickDraw { true | false } | checkPoint | maxmem | sound { on | off }

delete ⟨chunk⟩ ⟨simpleContainer⟩ | [⟨menuItemExpr⟩ { of | from }] ⟨menuExpr⟩ ⟨part⟩

dial ⟨expr⟩ [with modem | with [modem] ⟨expr⟩]

disable [⟨menuItem⟩ of] ⟨menu⟩ | ⟨button⟩

divide ⟨container⟩ by ⟨float⟩

domenu ⟨exprOrLine⟩ | ⟨expr⟩ [, ⟨expr⟩] [without dialog]

drag from ⟨point⟩ to ⟨point⟩ [with ⟨springKeys⟩]

edit [the] script of ⟨object⟩

enable [⟨menuItem⟩ of] ⟨menu⟩ | ⟨button⟩

enterInField

enterKey

export paint to file ⟨expr⟩

find [whole | string | words | word | chars | normal] [international] ⟨expr⟩ [in ⟨field⟩] [⟨ofOnly⟩ marked cards] functionkey ⟨unsigned⟩

get ⟨expr⟩ | [the] ⟨property⟩ [of { ⟨window⟩ | ⟨object⟩ | [⟨menuItem⟩ of] ⟨menu⟩ | ⟨chunk⟩ ⟨field⟩ }]

go [to] {{⟨ordinal⟩ | ⟨position⟩} ⟨endLine⟩ | ⟨dest⟩} [in [a] new window] [without dialog]

help

hide menuBar | picture of ⟨object⟩ | { card | bkgnd } picture | ⟨window⟩ | ⟨part⟩

import paint from file ⟨expr⟩

keyDown ⟨expr⟩

lock screen | messages | error dialogs | recent

mark all cards | ⟨card⟩ | cards where ⟨expr⟩ | cards by finding [whole | string | words | word | chars | normal] [international] ⟨expr⟩ [in ⟨field⟩]

multiply ⟨container⟩ by ⟨arith⟩

open [report] printing [with dialog] | file ⟨exprOrLine⟩ | ⟨expr⟩ [with ⟨exprOrLine⟩] | ⟨exprOrLine⟩

play stop | ⟨expr⟩ [[tempo ⟨unsigned⟩] ⟨exprOrLine⟩]

pop card [⟨preposition⟩ ⟨container⟩]

print ⟨expr⟩ with ⟨exprOrLine⟩ | ⟨unsigned⟩ cards | all cards | marked cards | ⟨card⟩ | ⟨field⟩ | ⟨expr⟩

push ⟨dest⟩

put ⟨expr⟩ [⟨preposition⟩ [⟨container⟩ | [⟨menuItem⟩ of] ⟨menu⟩ [with menuMessage[s] ⟨expr⟩]]

read from file ⟨expr⟩ {until ⟨expr⟩ | for ⟨unsigned⟩}

reply ⟨expr⟩ [with keyword ⟨expr⟩] error ⟨expr⟩

request ⟨expr⟩ { of | from } ⟨expr⟩ { ae | appleEvent } { class | ID | sender | returnID | data [{ of | with } keyword ⟨expr⟩] }

reset paint | menubar | printing

returnInField

returnKey

save { [this] stack | stack ⟨expr⟩ } as [stack] ⟨expr⟩

select [before | after] text of | ⟨chunk⟩ ⟨field⟩ | ⟨message⟩ | ⟨part⟩ | ⟨emptyExpr⟩

set [13] [the] ⟨property⟩ [⟨ofOnly⟩ {⟨window⟩ | ⟨object⟩ | ⟨menuItem⟩ of ⟨menu⟩ | ⟨chunk⟩ ⟨field⟩}] to ⟨propVal⟩

show menuBar | picture of ⟨object⟩ | { card | bkgnd } picture | { ⟨window⟩ | ⟨part⟩ } [at ⟨point⟩] | [all | marked | ⟨unsigned⟩] cards

sort { [cards of] { this stack | ⟨bkgnd⟩ } | marked cards } [ascending | descending] [text | numeric | international | dateTime] by ⟨expr⟩ [{ lines | items } of] ⟨container⟩ by ⟨expr⟩

start using ⟨stack⟩

stop using ⟨stack⟩

subtract ⟨arith⟩ from ⟨container⟩

tabKey

type ⟨expr⟩ [with ⟨springKeys⟩]

unlock screen [with [visual [effect]] ⟨visEffect⟩] | error dialogs | recent | messages

unmark all cards | ⟨card⟩ | cards where ⟨expr⟩ | cards by finding [whole | string | words | word | chars | normal] [international] ⟨expr⟩ [in ⟨field⟩]

[13] See notes on *set*.

visual [effect] ⟨visEffect⟩

wait ⟨duration⟩ | ⟨count⟩ [ticks | tick | seconds | second | sec]

write ⟨expr⟩ to file ⟨exprOrLine⟩

A.6.3 *set* Command Syntax

⟨**style**⟩ = transparent | opaque | rectangle | roundrect | shadow | checkBox | radioButton | scrolling | oval | popup

⟨**textAlign**⟩ = right | left | center

⟨**textStyleList**⟩ = ⟨textStyleList⟩ ⟨textStyle⟩ | ⟨textStyle⟩

⟨**textStyle**⟩ = plain | bold | italic | underline | outline | shadow | condense | extend | group

⟨**propVal**⟩ = ⟨exprOrLine⟩ | ⟨integer⟩ | ⟨unsigned⟩ | ⟨logical⟩ | ⟨point⟩ | ⟨rect⟩ | ⟨style⟩ | ⟨textAlign⟩ | ⟨textStyleList⟩

exprOrLine commandChar, cursor, debugger, environment, itemDelimiter, language, markChar, menuMessage, messageWatcher, name, numberFormat, owner, printTextFont, reportTemplates, script, scriptEditor, scriptingLanguage, scriptTextFont, stacksInUse, textFont, variableWatcher, version

integer top, bottom, left, right, width, height

unsigned brush, dialingTime, dialingVolume, dragSpeed, family, freeSize, icon, ID, lineSize, multiSpace, partNumber, pattern, polySides, printTextHeight, printTextSize, scriptTextSize, scroll, size, textHeight, textSize, titleWidth, traceDelay, userLevel

logical autoHilite, autoSelect, autoTab, blindTyping, cantAbort, cantDelete, cantModify, cantPeek, centered, checkMark, dontSearch, dontWrap, editBkgnd, enabled, filled, fixedLineHeight, grid, hilite, lockErrorDialogs, lockMessages, lockRecent, lockScreen, lockText, longWindowTitles, marked, multiple, multipleLines, powerKeys, sharedHilite, sharedText, showLines, showName, showPict, suspended, textArrows, userModify, visible, wideMargins, zoomed

point loc, topLeft, botRight, bottomRight, scroll (of window)

rect rect, printMargins

style style

textAlign textAlign, printTextAlign

textStyleList printTextStyle, textStyle

A.7 Functions

Note: ⟨funcArth⟩, ⟨funcFloat⟩, ⟨funcExpr⟩, and ⟨funcUnsigned⟩ all take expressions where they're called with parentheses, but factors otherwise.

abs ⟨funcArith⟩

annuity ⟨float⟩ , ⟨float⟩

atan ⟨funcFloat⟩

average ⟨arithList⟩

charToNum ⟨funcExpr⟩

clickChunk

clickH

clickLine

clickLoc

clickText

clickV

cmdKey

commandKey

compound ⟨float⟩ , ⟨float⟩

cos ⟨funcFloat⟩

date

diskSpace

exp ⟨funcFloat⟩

exp1 ⟨funcFloat⟩

exp2 ⟨funcFloat⟩

foundChunk

foundField

foundLine

foundText

heapSpace

length ⟨funcExpr⟩

ln ⟨funcFloat⟩

ln1 ⟨funcFloat⟩

ln2 ⟨funcFloat⟩

max ⟨arithList⟩

menus

min ⟨arithList⟩

mouse

mouseClick

mouseH

mouseLoc

mouseV

number cards [in ⟨bkgnd⟩] | bkgnds | [card | bkgnd] { buttons | fields | parts } | { chars | words | items | lines } in ⟨funcExpr⟩ | ⟨object⟩ | menus | menuItems { in | of } ⟨menu⟩ | marked cards | windows

numToChar ⟨funcUnsigned⟩

offset ⟨string⟩ , ⟨string⟩

optionKey

param ⟨funcUnsigned⟩

paramCount

params

random ⟨funcUnsigned⟩

result

round ⟨funcFloat⟩

screenRect

seconds

selectedButton [card | bkgnd] family ⟨funcUnsigned⟩

selectedChunk

selectedField

selectedLine

selectedLoc

selectedText

shiftKey

sin ⟨funcFloat⟩

sound

sqrt ⟨funcFloat⟩

stacks

stackSpace

sum ⟨arithList⟩

systemVersion

tan ⟨funcFloat⟩

target

ticks

time

tool

trunc ⟨funcFloat⟩

value ⟨funcExpr⟩

windows