HyperTalk: The Language for the Rest of Us

Kyle Wheeler

January 18, 2004

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 Atkinson1 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.

1Key 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 (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.

2.4 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)\(^2\), Toolbox (for Microsoft Windows)\(^3\), MetaCard (for Windows, Mac, and Unix/X11)\(^4\), WinPlus (for Microsoft Windows), HyperSense (originally for NeXT), FreeCard (an open-source clone), PLUS (for Mac, Windows, and OS/2)\(^5\), HyperStudio (for Mac and Windows)\(^6\), LinkWay (for DOS)\(^7\), and a cross-platform OracleCard from Oracle\(^8\). 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 appealing 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]

---

\(^2\)Calhoun left Apple in 2001 to form his own company, 4R Software [20].
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.”

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:

```plaintext
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]:

```plaintext
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) 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


A  BNF

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

A.1  Scripts

\[
\langle \text{script} \rangle = \langle \text{script} \rangle \langle \text{handler} \rangle | \langle \text{handler} \rangle
\]

\[
\langle \text{handler} \rangle = \langle \text{return} \rangle \langle \text{handler} \rangle | \text{on} \langle \text{messageKey} \rangle \langle \text{return} \rangle \langle \text{stmtList} \rangle \text{eol} \langle \text{messageKey} \rangle \langle \text{return} \rangle
\]

\[
\langle \text{ifBlock} \rangle = \text{if} \langle \text{logical} \rangle [ \langle \text{return} \rangle ] \text{then} \{ \langle \text{singleThen} \rangle | \langle \text{return} \rangle \langle \text{multiThen} \rangle \}
\]

\[
\langle \text{singleThen} \rangle = \langle \text{stmt} \rangle [ \langle \text{return} \rangle ] \langle \text{elseBlock} \rangle
\]

\[
\langle \text{multiThen} \rangle = \langle \text{stmtList} \rangle \{ \text{end if} | \langle \text{elseBlock} \rangle \}
\]

\[
\langle \text{elseBlock} \rangle = \text{else} \{ \langle \text{stmt} \rangle | \langle \text{return} \rangle \langle \text{stmtList} \rangle \text{end if} \}
\]

\[
\langle \text{repeatBlock} \rangle = \text{repeat} [ \text{forever} ] | \langle \text{duration} \rangle | \langle \text{count} \rangle | \text{with} \langle \text{identifier} \rangle = \langle \text{range} \rangle | \langle \text{return} \rangle \langle \text{stmtList} \rangle \text{end repeat}
\]

\[
\langle \text{duration} \rangle = \text{until} \langle \text{logical} \rangle | \text{while} \langle \text{logical} \rangle
\]

\[
\langle \text{count} \rangle = [ \langle \text{for} \rangle ] \langle \text{unsigned} \rangle [ \text{times} ]
\]

\[
\langle \text{range} \rangle = \langle \text{integer} \rangle [ \text{down} ] \text{to} \langle \text{integer} \rangle
\]

A.2  Expressions

\[
\langle \text{expr} \rangle = \langle \text{source} \rangle | - \langle \text{expr} \rangle | \text{not} \langle \text{expr} \rangle | \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle | ( \langle \text{expr} \rangle ) | \langle \text{chunk} \rangle \langle \text{expr} \rangle | \text{there is} \{ \text{a} | \text{an} | \text{no} \} \langle \text{expr} \rangle
\]

\[
\langle \text{op} \rangle = + | - | * | / | \& | \&\& | ^ | = | < | > | <> | \neq | <= | >= | \leq | \geq | \text{and} | \text{or} | \text{contains} | \text{div} | \text{mod} | \text{is} | \text{is not} | \text{is in} | \text{is not in} | \text{is within} | \text{is not within} | \text{is a[n]} | \text{is not a[n]}
\]

\[
\langle \text{source} \rangle = \langle \text{literal} \rangle | \langle \text{constant} \rangle | \langle \text{simpleContainer} \rangle | \{ \langle \text{adjective} \rangle \} \langle \text{function} \rangle | \{ \langle \text{adjective} \rangle \} \langle \text{property} \rangle \text{of} \{ \langle \text{object} \rangle | \langle \text{window} \rangle | \langle \text{menuItem} \rangle \text{of} \langle \text{menu} \rangle | \langle \text{chunk} \rangle \langle \text{field} \rangle \}
\]

\[
\langle \text{literal} \rangle = \text{“quoted string”} | \text{unquotedToken}
\]

\[
\langle \text{constant} \rangle = \text{down} | \text{empty} | \text{false} | \text{formFeed} | \text{lineFeed} | \text{pi} | \text{quote} | \text{space} | \text{tab} | \text{true} | \text{up} | \text{zero} | \text{one} | \text{two} | \text{three} | \text{four} | \text{five} | \text{six} | \text{seven} | \text{eight} | \text{nine} | \text{ten}
\]

\[
\langle \text{adjective} \rangle = \text{long} | \text{short} | \text{abbrev} | \text{abbr} | \text{abbreviated}
\]

\[
\langle \text{window} \rangle = [ \text{the} ] \{ \text{card} | \text{pattern} | \text{tool} | \text{scroll} \} \text{window} | \langle \text{messageBox} \rangle
\]

\[
\langle \text{menuItem} \rangle = \langle \text{ordinal} \rangle \langle \text{menuItem} \rangle | \langle \text{menuItem} \rangle \langle \text{expr} \rangle
\]

\[
\langle \text{menu} \rangle = \langle \text{ordinal} \rangle \langle \text{menu} \rangle | \langle \text{menuItem} \rangle \langle \text{expr} \rangle
\]

\[
\langle \text{function} \rangle = \text{the} \langle \text{theFunc} \rangle | [ \text{the} ] \langle \text{theFunc} \rangle \text{of} \langle \text{oneFuncArg} \rangle | \langle \text{identifier} \rangle ( \langle \text{funcArgs} \rangle )
\]

\[\text{There may be some omissions due to the breadth of the language.}\]
A.3 Ordinals and Positions

\[ \langle \text{ordinal} \rangle = [ \text{the} ] \{ \text{last} | \text{mid} | \text{middle} | \text{any} | \text{first} | \text{second} | \text{third} | \text{fourth} | \text{fifth} | \text{sixth} | \text{seventh} | \text{eighth} | \text{ninth} | \text{tenth} \} \]

\[ \langle \text{position} \rangle = \text{this} | [ \text{the} ] \text{prev} | [ \text{the} ] \text{next} \]

A.4 Chunks and Containers

\[ \langle \text{simpleContainer} \rangle = \langle \text{variable} \rangle | \langle \text{part} \rangle | \langle \text{menu} \rangle | \langle \text{messageBox} \rangle | [ \text{the} ] \langle \text{selection} \rangle \]

\[ \langle \text{container} \rangle = \langle \text{chunk} \rangle \langle \text{simpleContainer} \rangle | \langle \text{simpleContainer} \rangle \]

\[ \langle \text{messageBox} \rangle = [ \text{the} ] \langle \text{msg} \rangle [ \text{box} | \text{window} ] \]

\[ \langle \text{chunk} \rangle = [\{ \langle \text{ordinal} \rangle \text{char} | \text{char} \langle \text{expr} \rangle [ \text{to} \langle \text{expr} \rangle ] \} \text{of}] \{ [\{ \langle \text{ordinal} \rangle \text{word} | \text{word} \langle \text{expr} \rangle [ \text{to} \langle \text{expr} \rangle ] \} \text{of}] [\{ \langle \text{ordinal} \rangle \text{item} | \text{item} \langle \text{expr} \rangle [ \text{to} \langle \text{expr} \rangle ] \} \text{of}] [\{ \langle \text{ordinal} \rangle \text{line} | \text{line} \langle \text{expr} \rangle [ \text{to} \langle \text{expr} \rangle ] \} \text{of}] \} \text{of} \}

A.5 Objects

\[ \langle \text{object} \rangle = \text{HyperCard} | \text{me} | [ \text{the} ] \langle \text{target} \rangle | \langle \text{button} \rangle | \langle \text{field} \rangle | \langle \text{card} \rangle | \langle \text{bkgn}\rangle | \langle \text{stack} \rangle \]

\[ \langle \text{button} \rangle = [ \text{button} \langle \text{id} \rangle \langle \text{unsignedFactor} \rangle ] | \langle \text{button} \langle \text{factor} \rangle ] | \langle \text{ordinal} \rangle \langle \text{button} \rangle ] | \langle \text{of} \langle \text{card} \rangle \rangle | \langle \text{of} \langle \text{card} \rangle \rangle \]

\[ \langle \text{field} \rangle = [ \text{field} \langle \text{id} \rangle \langle \text{unsignedFactor} \rangle ] | \langle \text{field} \langle \text{factor} \rangle ] | \langle \text{ordinal} \rangle \langle \text{field} \rangle ] | \langle \text{of} \langle \text{card} \rangle \rangle \]

\[ \langle \text{part} \rangle = [ \text{button} | \langle \text{field} \rangle ] | [ \text{part} \langle \text{id} \rangle \langle \text{unsignedFactor} \rangle ] | \langle \text{part} \langle \text{factor} \rangle ] | \langle \text{ordinal} \rangle \langle \text{part} \rangle ] | \langle \text{of} \langle \text{card} \rangle \rangle \]

\[ \langle \text{card} \rangle = \text{recent card} | \text{back} | \text{forth} | \text{card} \langle \text{id} \rangle \langle \text{unsigned} \rangle | \langle \text{card} \langle \text{expr} \rangle ] | \langle \text{card} \langle \text{endLine} \rangle ] | \langle \text{ordinal} \rangle \langle \text{card} \rangle ] | \langle \text{position} \rangle \langle \text{card} \rangle ] | \langle \text{of} \langle \text{bkgn}\rangle \rangle | \langle \text{of} \langle \text{bkgn}\rangle \rangle | \langle \text{position} \rangle \langle \text{marked card} \rangle ] | \langle \text{position} \rangle \langle \text{marked card} \rangle ] | \langle \text{marked card} \rangle \langle \text{expr} \rangle ]

\[ \langle \text{bkgn}\rangle = \langle \text{bkgn} \langle \text{id} \rangle \langle \text{unsigned} \rangle ] | \langle \text{bkgn} \langle \text{expr} \rangle ] | \langle \text{bkgn} \langle \text{endLine} \rangle ] | \langle \text{ordinal} \rangle \langle \text{bkgn} \rangle ] | \langle \text{position} \rangle \langle \text{bkgn} \rangle ]

\[ \langle \text{stack} \rangle = \text{this stack} | \text{stack} \langle \text{expr} \rangle ] | \text{stack} \langle \text{endLine} \rangle \]

\[ ^{12}\text{Note: } \text{“card field 1” is a field and “card (field 1)” is a card.} \]
A.6 Commands

A.6.1 Command Nonterminals

\[\text{dateItems} = \text{(unsigned), (unsigned), (unsigned), (unsigned), (unsigned), (unsigned)}\]

\[\text{date} = \text{(unsigned)} \mid \text{(dateItems) \{humanDate\}} \mid \text{\{humanTime\}} \mid \text{\{humanDate\}}\]

\[\text{dateFormat} = \text{[ (adjective) \{ seconds \mid dateItems \mid date \mid time\}}\]

\[\text{dayOfWeek} = \text{Sunday | Sun | Monday | Mon | Tuesday | Tue | Wednesday | Wed | Thursday | Thu | Friday | Fri | Saturday | Sat}\]

\[\text{dest} = \{ \{\text{card} \mid \text{bkgnd}\} \text{ of \{stack\}} \mid \{\text{stack}\} \mid \{ \{\text{card} \mid \text{bkgnd}\} \text{ of \{stack\}} \text{\{exprOrLine\}}\]

\[\text{duration} = \text{until \{logical\}} \mid \text{while \{logical\}}\]

\[\text{humanDate} = \text{[ \{\text{dayOfWeek}\} \mid \{\text{month}\} \{\text{unsigned}\} \mid \{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor}\} \{/ \mid -\}\{\text{unsignedFactor\}}\}

\[\text{humanTime} = \text{(unsigned) : (unsigned) \{ (unsigned) \{ am \mid pm \}}\]

\[\text{month} = \text{January | Jan | February | Feb | March | Mar | April | Apr | May | May | June | Jun | July | Jul | August | Aug | September | Sep | October | Oct | November | Nov | December | Dec}\]

\[\text{point} = \{ \{\text{integer} \mid \{\text{integer}\}}\]

\[\text{preposition} = \text{before | after | into}\]

\[\text{rect} = \{ \{\text{integer} \mid \{\text{integer}\} \mid \{\text{integer}\} \mid \{\text{integer}\}}\]

\[\text{springKeys} = \{\text{springKeys} \mid \{\text{springKey}\} \mid \{\text{springKey}\} \mid \{\text{springKey}\} \mid \{\text{springKey}\} \mid \{\text{springKey}\} \mid \{\text{springKey}\} \mid \{\text{springKey\}}\]

\[\text{springKey} = \text{shiftKey | optionKey | commandKey}\]

\[\text{style} = \text{transparent | opaque | rectangle | roundrect | shadow | checkBox | radioButton | scrolling | oval | popup}\]

\[\text{textAlign} = \text{right \mid left \mid center}\]

\[\text{textStyleList} = \text{(textStyleList) \{textStyle\} \mid \{textStyle\}}\]

\[\text{textStyle} = \text{plain | bold | italic | underline | outline | shadow | condense | extend | group}\]

\[\text{visEffect} = \text{(visKind) \{ [ very \{ slow \mid slowly \mid fast \}\} \text{ to \{visSrc\}}\]

\[\text{visKind} = \text{barn door \{ open \mid close\} \mid cut \mid plain \mid dissolve | venetian blinds \mid checkerboard | iris \{ open \mid close\} \mid scroll \{ left \mid right \mid up \mid down\} \mid wipe \{ left \mid right \mid up \mid down\} \mid zoom \{ open \mid out \mid close \mid in\} \mid shrink to \{ top \mid bottom \mid center\} \mid stretch from \{ top \mid bottom \mid center\} \mid push \{ left \mid right \mid up \mid down\} \mid limit \{ left \mid right \mid up \mid down\} \mid wrap \{ left \mid right \mid up \mid down\} \mid close \{ left \mid right \mid up \mid down\} \mid open \{ left \mid right \mid up \mid down\}\]

\[\text{visSrc} = \text{card | black | white | gray | inverse}\]

\[\text{window} = \{\text{card} \mid pattern \mid tool \mid scroll \mid fatBits\} \text{ window} \mid \text{\{messageBox\}}\]
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⟩

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

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

dispatch [ 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⟩

9
import paint from file \(\langle\text{expr}\rangle\)

key\(\text{Down}\) \(\langle\text{expr}\rangle\)

lock screen | messages | error dialogs | recent

mark all cards | \(\langle\text{card}\rangle\) | cards where \(\langle\text{expr}\rangle\) | cards by finding \{ whole | string | words | word | chars | normal \} \[ international \] \(\langle\text{expr}\rangle\) \[ in \(\langle\text{field}\rangle\) \]

multiply \(\langle\text{container}\rangle\) by \(\langle\text{arith}\rangle\)

open \{report\} printing \[ with dialog \] \| file \(\langle\text{exprOrLine}\rangle\) \| \(\langle\text{expr}\rangle\) \[ with \(\langle\text{exprOrLine}\rangle\) \] \| \(\langle\text{exprOrLine}\rangle\)

play stop \| \(\langle\text{expr}\rangle\) \[ [ tempo \langle\text{unsigned}\rangle\] \[ \langle\text{exprOrLine}\rangle\] \]

pop card \[ \langle\text{preposition}\rangle \langle\text{container}\rangle \]

print \(\langle\text{expr}\rangle\) with \(\langle\text{exprOrLine}\rangle\) \| \(\langle\text{unsigned}\rangle\) cards \| all cards \| marked cards \| \(\langle\text{card}\rangle\) \| \(\langle\text{field}\rangle\) \| \(\langle\text{expr}\rangle\)

push \(\langle\text{dest}\rangle\)

put \(\langle\text{expr}\rangle\) \[ \langle\text{preposition}\rangle \langle\text{container}\rangle \] \| \[ \langle\text{menuItem}\rangle\] of \(\langle\text{menu}\rangle\) \[ with menuMessage[s] \langle\text{expr}\rangle \]

read from file \(\langle\text{expr}\rangle\) \{until \(\langle\text{expr}\rangle\) \} \| for \(\langle\text{unsigned}\rangle\)

reply \(\langle\text{expr}\rangle\) \[ with keyword \(\langle\text{expr}\rangle\) \] error \(\langle\text{expr}\rangle\)

request \(\langle\text{expr}\rangle\) \{ of \} \| from \(\langle\text{expr}\rangle\) \{ ae \} \| appleEvent \} \{ class \} \| ID \| sender \| returnID \| data \[ \{ of \} \] \[ with \} \text{keyword} \langle\text{expr}\rangle \}

reset paint \| menubar \| printing

return\(\text{InField}\)

return\(\text{Key}\)

save \{ [ this ] stack \} \| stack \(\langle\text{expr}\rangle\) \] as \[ stack \] \(\langle\text{expr}\rangle\)

select \[ before \] \| after \} text of \[ \langle\text{chunk}\rangle \langle\text{field}\rangle \] \| \(\langle\text{message}\rangle\) \| \(\langle\text{part}\rangle\) \| \(\langle\empty\text{Expr}\rangle\)

set \[\text{[1]} \text{[the ]} \langle\text{property}\rangle \] \[ \langle\of\text{Only}\rangle\} \{ \langle\text{window}\rangle \} \| \langle\text{object}\rangle\} \| \langle\text{menuItem}\rangle\} of \langle\text{menu}\rangle\} \| \langle\text{chunk}\rangle \langle\text{field}\rangle\} \] to \langle\text{propVal}\rangle

show menuBar \| picture of \langle\text{object}\rangle\} \| \{ card \} \| bk\(\text{gnd}\) \} picture \[ \{ \langle\text{window}\rangle \} \| \langle\text{part}\rangle\} \] \[ at \langle\text{point}\rangle\} \] \| \[ all \} \| marked \} \langle\text{unsigned}\rangle \} \} cards

sort \{ cards of \} \[ this stack \} \| \langle\text{bk}\(\text{gnd}\rangle\} \} \| marked cards \} \[ ascending \} \| descending \} \} text \| numeric \} \| international \} \| \langle\text{dateTime}\rangle\} \} by \langle\text{expr}\rangle \} \{ lines \} \| \} items \} \} \} \} \of \} \langle\text{container}\rangle \} \langle\text{expr}\rangle

start using \langle\text{stack}\rangle

stop using \langle\text{stack}\rangle

subtract \(\langle\text{arith}\rangle\) from \langle\text{container}\rangle

\(\text{tabKey}\)

type \langle\text{expr}\rangle \{ with \langle\text{springKeys}\rangle\} \]

unlock screen \[ with \} \langle\text{visual} \langle\text{effect}\rangle\} \] \| error dialogs \| recent \| messages

unmark all cards | \(\langle\text{card}\rangle\) | cards where \(\langle\text{expr}\rangle\) | cards by finding \{ whole | string | words | word | chars | normal \} \[ international \] \(\langle\text{expr}\rangle\) \[ in \langle\text{field}\rangle\]

\[13\text{See notes on set.}\]
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, printTextStyle, reportTemplates, script, scriptEditor, scriptingLanguage, scriptTextStyle, stacksInUse, textField, variableWatcher, version

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

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

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

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

rect rect, printMargins

style style

textAlign textAlign, printTextStyleAlign

textStyleList printTextStyle, textStyle

A.7 Functions

Note: ⟨funcArith⟩, ⟨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
numToChar \langle\text{funcUnsigned}\rangle
offset \langle\text{string}\rangle, \langle\text{string}\rangle
optionKey
param \langle\text{funcUnsigned}\rangle
paramCount
params
random \langle\text{funcUnsigned}\rangle
result
round \langle\text{funcFloat}\rangle
screenRect
seconds
selectedButton \text{[card | bkgnd] family}\langle\text{funcUnsigned}\rangle
selectedChunk
selectedField
selectedLine
selectedLoc
selectedText
shiftKey
sin \langle\text{funcFloat}\rangle
sound
sqrt \langle\text{funcFloat}\rangle
stacks
stackSpace
sum \langle\text{arithList}\rangle
systemVersion
tan \langle\text{funcFloat}\rangle
target
ticks
time
tool
trunc \langle\text{funcFloat}\rangle
value \langle\text{funcExpr}\rangle
windows
Perspectives on HyperTalk

The HyperTalk Zoo

Advanced HyperTalk Topics

HyperTalk Language

Style

Tools for Development

Protecting Stacks

Debugging.

Please note that this chapter was written in 1989, against HyperCard version 1.22. This chapter does not attempt to explain all of the HyperTalk language, nor does it discuss basic topics such as how to edit a script or what a message is. Rather, it attempts to present some techniques to help stack development, some ways to do new things in HyperTalk and work around its syntactic limitations, and some guidelines for the perplexed on what constitutes "good HyperTalk programming". This chapter is for those of you who already know the basics, but are looking for more. HyperTalk was not a great form for a programming language, and it wasn't really English either. But a HyperTalk script "looked like something you could understand and use, and modify when needed". This cogistibility was the deep key in end-user-land. They make an entire world for the rest of us to witness. That is the nature of Art and Artists.

999108. Some parts of the HyperTalk language, especially those related to obsolete operating system or hardware capabilities (like AppleTalk), are missing. Getting started. Getting started is easy. What is it that you're interested in doing? I want to download and play with this. Apple called HyperCard "programming for the rest of us." Steve Wozniak called it "the best program ever written". Released in 1987 and included in the box with every Macintosh sold during the late 1980's and '90s. HyperCard was a software Erector Set: part programming language, part paint program, part database. With HyperCard, you could draw a user interface with MacPaint-like tools, then apply scripts and behaviors to it with an expressive syntax that mimicked natural English. Need more? HyperTalk is a high-level programming language created in 1987 by Dan Winkler and used in conjunction with Apple Inc.'s HyperCard hypermedia program by Bill Atkinson. The main target audience of HyperTalk was beginning programmers, hence HyperTalk programmers were usually called authors, and the process of writing programs was called "scripting". HyperTalk scripts are fairly similar to written English, and use a logic structure similar to the Pascal language.