-

Egyptian Hieroglyphics and LogoWriter

by

Thomas F. Trocco

-

-

Computer Department Chair
Computer Teacher, Grades 2 - 9
St. Hilda's & St. Hugh's School
New York, NY

Acknowledgments: Thanks to Michael Tempel, Ellen Siegel, David Ross, and Laura Allen for their inspiration, comments, and suggestions.

Special thanks to Adaya Alston, Tyler Barnes, Tracy Blount, Sandy Cid, Vanessa Cook, Matthew Graves, Madeline Kende Howe, Kristina Negron, Nadir Quadir, Maritza Shaeffer, Saliou Seck, Michael Simon, Austin Shau, Laura Stickney, Etienne Van Harte, Marcos Antonio Vega, and Mark Verma for their excitement, cooperation, and hard work all semester long.

* Correspondence can be sent to:
Thomas F. Trocco
The Calhoun School
433 West End Avenue
New York, NY 10024

Logo Foundation
250 West 57th Street, Suite 2228
New York, NY 10107-228
Telephone: (212) 765-4918
FAX: (212) 765-4789

The Logo Foundation is a nonprofit educational organization incorporated in New York State.

-

The Setting

Students at St. Hilda's and St. Hugh's School practice keyboarding and use educational software in the preschool with their classroom teachers. Beginning in second grade, students work with LogoWriter in the computer lab. The students meet with me once or twice a week for programming projects. Each class is forty minutes long, with a five minute break between periods. I usually begin each unit by introducing a new concept or command, or by posing a question or problem. Then the students work on experimentation, application, and testing throughout the duration of the unit, which varies from one period to one semester. Often, there are a few interrelated concepts and questions being worked on simultaneously. I meet with half of the second grade (9 students) at a time. From third through ninth grade, I meet with the entire class (up to 18 students). These classes include word processing and book-making in the second grade; simple animation in the third grade; microworld "town" construction with coordinate plotting in the fourth grade; ancient Egypt-based projects in the fifth grade; branching, animated adventure stories based on Arthurian legend in the sixth grade; and random number projects (involving math, language, and graphics) in the seventh grade. Class time is divided between individual work, work in pairs, and work in cooperative groups of four. Many classes also use the lab for Writers' Workshops using LogoWriter with their language arts teachers. During the Egyptian Hieroglyphics project, the lab contained fourteen MSDOS computers running LogoWriter 2.0 and seven Commodore 64 computers running LogoWriter 1.1.

Introduction

The seventeen Fifth Graders and their Social Studies teacher began their study of ancient history with Egypt during the Fall, 1991 semester. At this time, I showed the students Fun With Hieroglyphs.[*] This kit contains a book, a stamp pad, and 24 rubber stamps, each representing one Hieroglyphic. The students immediately saw a connection between these Hieroglyphic shapes and the shapes in LogoWriter, and decided that it would be fun to try to do something similar in LogoWriter.

---

* Roehrig, Catharine (1990) Fun With Hieroglyphs. The Metropolitan Museum of Art, New York, and Viking, A Division of Penguin Books USA Inc., 375 Hudson Street, New York, NY 10014. item #04-01130-0 $19.95

To begin the unit the students were assigned reading to familiarize them with Hieroglyphics. They read Catharine Roehrig's booklet as well as other sources assigned by their social studies teacher. Then the students typed what they had learned into their LogoWriter journals. This included:

• Hieroglyphics began as pictures which represented objects.
• It is easy to create a system where symbols stand for objects, but difficult if ideas or verbs have to be symbolized.
• Most Hieroglyphics later came to represent sounds, though some remained as symbols for specific objects, such as the sun, and some came to represent syllables.
• The Rosetta stone, discovered in 1799, was the key to translating Hieroglyphics.
• There is no one-to-one correspondence between Egyptian and English: there are sounds in Egyptian not found in English and vice-versa.
• Hieroglyphics can be written right to left, left to right, or top to bottom.
• Very few people in ancient Egypt could read or write. Those who could had the important profession of scribe.

A discussion followed on how to apply what they had learned in LogoWriter. The students decided to begin by creating a LogoWriter shape for each Hieroglyphic. In LogoWriter, the turtle can be changed to a different image by using the setsh command, followed by the number of the shape (e.g. setsh 23). In MSDOS LogoWriter, ninety different shapes can be created. A turtle that is assigned one of these shapes can be moved around. The shape may be stamped on the screen or used to shade (tile) an enclosed area on the screen. Even though the shape may no longer be that of a turtle, it is still referred to as a turtle. The students were given paper with grids of 16 X 16 boxes, corresponding to the 16 X 16 grid of the LogoWriter Shape Editor, so they could each design one Hieroglyphic for homework.*

The students decided that they would use only the Hieroglyphics which represented individual sounds so that their 'code' could be more easily translated into English. They felt that if they also used shapes to symbolize words, it would be confusing. There were problems with a few letters where there is not a one-to-one correspondence between English and Hieroglyphics. In spoken Egyptian, the R and L sound, the O and W sound, and the F and V sound were the same, so each pair of (English) letters is designated by one Hieroglyphic. The students immediately identified with this situation: one Asian-American student commented on the similarity of R and L in Japanese and one Latina student

---

* See translation table in Appendix I

commented on the similarity of V and B in Spanish. For three English letters, Q, U, and X, more than one Hieroglyphic had to be used (C + I + O; I + O; and C + S, respectively).*

I took the shapes created by the students and cut and pasted them onto a single scrapbook disk. I gave a copy to each student. Students were allowed to experiment with these new shapes. Many used the Turtle Move key (learned in Second Grade) to move and stamp messages. This involved the following steps:

1. Press the turtle move key (F9 in MSDOS).
2. Use the arrow keys to move the turtle to the desired spot.
3. Press the escape key to exit from turtle move mode.
4. Set the color with setc <number>.
5. Type stamp to stamp the turtle's image on the screen.

Some students animated their Hieroglyphics. The bird (shape 76 representing the letter O) was set in motion with commands such as

repeat 999 [fd 2 wait 1]

Then a second copy of the bird with wings in the up position was made as shape 85. The bird could then be made to fly with the instruction

repeat 999 [setsh 85 fd 1 wait 1 setsh 76 fd 1 wait 1]

After some experimentation, a class discussion was called to decide what could be done with these new shapes. Students decided by consensus that a program which automatically translated English into Hieroglyphics would be an improvement over using Turtle Move and Stamp. Some students began this work individually, while others worked in pairs. (Pairing was more often due to the lack of sufficient computers, than a matter of choice.)

Development of the Program

The students wanted to type a letter and have the turtle's shape change. A new primitive, readchar, was introduced. The students were asked to type readchar in the command center to see what it did. Most of them responded that nothing was happening. Some students then noticed that the cursor had disappeared. They pressed keys at this point and saw LogoWriter messages such as

I don't know what to do with a.

or

I don't know what to do with z.

---

* See Program Listing.

depending upon which key was pressed.
A class discussion ensued. When else had this message appeared? Some students remembered seeing this sort of message when they'd typed in numbers alone, or typed in an arithmetic expression such as 3 + 4. This message would also result from typing some Logo primitives, like heading, shape, or color alone. It was explained that this was because these procedures (including +, which is a procedure) were “reporters”. A reporter is a kind of procedure that reports something to another procedure. If there is no procedure to receive the report, Logo complains with a message. On the other hand,

```
print heading
show 3 + 4
setc color + 1
```

are all correct and don’t generate messages. Readchar is a reporter that waits for a key to be pressed and then reports the character to another procedure. Once this was clear, a procedure was written:

```
to translate
if equal? readchar "a [setsh 62]
end
```

It was obvious that although this would work for the letter “A”, nothing would happen for other letters.

The concept of variables had been introduced earlier. They were being presented in the context of graphics procedures so the students could more easily see the effects of making changes. One procedure was:

```
to square :side
repeat 4 [fd :side rt 90]
end
```

This procedure was put to use when combining the concepts of variables and readchar:

```
to start
type [Enter a number between 0 and 9 to make a square.]
square readchar
end

to square :side
repeat 4 [fd 10 * :side rt 90]
end
```

The behavior of readchar and the concepts of input and output were discussed. Students acted out the start procedure. The student playing the part of the readchar procedure passed a piece of paper with a keyboard character on it
to the square procedure, where it was given the name "side" and passed to the * procedure.
-

The translate procedure was then modified. One line for each letter was added. *

```
to translate :letter
if equal? :letter "a [setsh 62]
if equal? :letter "b [setsh 63]
.
.
.
.
.
.
if equal? :letter "z [setsh 82]
end
```

For this program, the following shapes were used:

61-c
62-a
63-b
64-c
65-s
67-d
68-y
69-f/v
71-j
72-h
73-l/r
74-m
75-n
76-o/w
79-t
82-z
83-k
84-i

The number 61 was chosen as a starting point because shapes 61-90 had not been previously used by any students. There are reasons why the alphabetical and numerical orders do not coincide. The shapes were created in the order listed in Fun With Hieroglyphics. The first two shapes (61 and 62) were for different sounds of the letter A. Later it was decided that the first was close enough to E to use that one. Likewise, the fourth and fifth (64 and 65) were for different sounds of the letter C. Therefore, the soft C (the S sound) appears in this sequence.

The students wanted translate to continue translating; not just translate one letter and stop. After some debate, one student suggested that the last line of the procedure should be the procedure

---

* See the Program Listing.

name itself; a student discovery of recursion. The procedure was again modified. This new last line was added:

translate readchar

Now they realized that this procedure didn't stamp, so it was modified again. Stamp was added after each setsh:

if equal? :letter "a [setsh 62 stamp]

Now the students saw that all shapes would be stamped on top of each other, so the stamp command was changed to the stamper procedure:

if equal? :letter "a [setsh 62 stamper]

The amount for fd in the stamper procedure varied. Some students preferred Hieroglyphics as close to each other as possible (fd 16), while some others spread them out quite a bit (fd 30):

```
to stamper
pd stamp pu
fd 20
end
```

A second stamper procedure, with a smaller fd, was used for Q, U, and X, where more than one shape had to be stamped:

```
to stamper.small
pd stamp pu
fd 10
end
```

A stamper procedure which accepts inputs for the distance moved between symbols could have been used instead of two separate stampers:

```
to stamper :distance
pd stamp pu
fd :distance
end
```

Now the stamping worked, but the symbols appeared vertically because the turtle’s heading was zero. A setup procedure was written:

```
to setup
rg
cc
ef
-
```

~~ht~~
~~pu~~
~~getshapes~~
~~rt 90~~
~~end~~

~~Some students discovered that they were using the wrong shapes. LogoWriter will use whatever shapes are currently in memory. In this case, those were the shapes on the original LogoWriter program disk. Getshapes causes LogoWriter to use the custom shapes on the student's scrapbook disk.~~

~~The students had been introduced to ef (for erase front) when they began programming on the flip side back in Third Grade, and they now put it on ALL pages.~~

~~to ef~~
~~if not front? [flip]~~
~~et~~
~~end~~

~~They have learned, some the hard way, never to use et. They have seen the result of using et when working in the command center while on the flip side of the page: all the procedures are erased.~~

~~Next the students wanted to put some finishing touches on the program. Most of the students wanted to begin stamping in the upper left corner of the screen. The concept of setpos had been introduced earlier in the year. During the Hieroglyphic project, the class was spending one day a week experimenting with new concepts such as setpos and one day a week on Hieroglyphics. From this work, the following line was added to the setup procedure:~~

~~setpos [-150 80]~~

~~Now the students wanted to be able to make spaces between words, skip to the next line, and exit from the recursive procedure. Each student tried this short procedure and learned the ASCII codes of the characters they wanted to test:~~

~~to ascii.codes~~
~~show ascii readchar~~
~~ascii.codes~~
~~end~~

~~The following lines were added to translate:~~

~~if equal? :letter char 32 [fd 30]~~

~~(If the spacebar is pressed, move fd 30.)~~

-

if equal? :letter char 13 [setx -150 sety ycor - 20]

(If the enter key is pressed, move the turtle to the left edge and 20 steps below where it was.)

if equal? :letter char 27 [cc stop]

(If the escape key is pressed, clear the command center and stop.)

The only difficulty was with sety ycor - 20. The concept of relative distance (20 steps below where you currently are) was difficult for some students, even though it is basically a back 20. The students stood on the tiled floor in the computer lab. Each student moved two tiles below where they were each time a "keyboard" student passed a char 13 to the "readchar" student who passed it to the "translate" student. An alternate way of moving to the next line might be:

if equal? :letter char 13 [setx -150 seth 0 bk 20 seth 90]

Because the students had learned setx and sety earlier in the year, it seemed to be a good way of reinforcing this concept in a simple and elegant way without the problem of changing the heading of the turtle twice.

Instructions were then added:

```
to instructions
type [Type any word to be translated into hieroglyphics.
  Hit spacebar for a space, enter for a new line,
  and esc to stop.]
char 13
end
```

Once the students were satisfied with their programs, a great deal of time was provided for creating messages in Hieroglyphics. These messages fell into two categories: those that could be posted for parents to see, and those that were private. The private messages included Egyptian Hieroglyphic Valentine's Day cards, and even more private messages. Students were allowed to pass notes as long as they were created with this program.*

Successes

Since this was an almost totally student-designed and directed project, there has been more interest and focus than in most other long-term projects. New concepts were introduced as they were needed by the students. Student mastery of these concepts was more rapid and more complete due to the their need to know, their sense of ownership of the project, and the way in which it reinforced their work in social studies class. These new concepts and commands include ASCII codes, equal?, if, output, passing of variable values, phonetics, readchar, recursion, setpos, similarities and dissimilarities in sounds of languages, symbols, the importance of written

---

* See examples in Appendix I.

language, translation, and variables. Previously learned concepts have been reinforced. The students have had the opportunity to work individually, in pairs, in teams, and as an entire class. My role has been that of consultant rather than leader.

Problems

This was a semester-long project. By the time we had completed our program, the Fifth Grade was studying the Middle Ages. Some students had trouble keeping up their interest throughout the entire project. Although invited, the Fifth Grade social studies teacher was not able to visit and see the students' work until two months after the project was completed. Numerous short visits by

-

subject teachers during a project would better integrate computers into other subjects and allow both teachers to see each other's ideas and styles of guiding students.

Proposed Additions

Students have asked if they could add:

• a backspace /erase key to erase stamped shapes.

This could be accomplished by the following:

if equal? :letter char 8 [pu setx xcor - 10 setsh 11 pe stamp pu]

Shape 11 is the solid square, which will erase everything under it if pe (pen erase) is used. The amount to move back (10) should be a fraction of the number of steps moved in stamper (20), stamper.small (10), and in spacing over (30), so it can be pressed once, twice, or thrice to erase a mistake.

• an automatic printscreen key.

This could be accomplished using the when command. Add this line to startup:

when "p [printscreen]

And in translate, change

if equal? :letter char 27 [cc stop]

to

if equal? :letter char 27 [pr.inst stop]

Now add this procedure:

```
to pr.inst
cc
type [Type ctrl-p to print your message.]
type char 13
end
```

• shapes for numbers and punctuation marks.

The students could create shapes for English punctuation and Arabic numbers. Many students have already created these for use in other procedures.

If they wish the students will work on these additions during in-class or after-class free time.

Suggestions

Some things which have occurred to me for the future are:

1.These procedures could be applied to any symbolic language: Mayan Hieroglyphics, Ancient Runes, Klingon letters, student-designed Hieroglyphics, etc.

2.Each shape could be stored under the number corresponding to the ASCII code of its letter. For example, the Hieroglyphic shape for A would be shape 65, B would be 66, Z would be 90, etc. After students have mastered the concept, their translate procedure could be greatly simplified:

```
to translate :letter
ifelse (ascii :letter) > 90
 [setsh (ascii :letter) - 32 stamper]
 [setsh ascii :letter  stamper]
translate readchar
end
```

This would be a good way to show the students how program code can often be simplified when simple rules are used. This would also introduce the concept of ifelse, and the fact that upper and lower case letters are 32 characters apart in ASCII code. Testing for the spacebar, the enter key, and the esc key would have to be done differently.

-

Program Listing

```
to start
setup
instructions
translate readchar
end


to setup
rg
cc
ef
ht
pu
getshapes
setpos [-145 50]
rt 90
end


to ef
if not front? [flip]
ct
end
to instructions
type [Type any word to be translated into hieroglyphics.
  Hit spacebar for a space, enter for a new line,
  and esc to stop.]
type char 13
end


to stamper
pd stamp pu
fd 20
end


to stamper.small
pd stamp pu
fd 10
end
```

-

```
to translate :letter
if equal? :letter "a [setsh 62 stamper]
if equal? :letter "b [setsh 63 stamper]
if equal? :letter "c [setsh 64 stamper]
if equal? :letter "d [setsh 67 stamper]
if equal? :letter "e [setsh 61 stamper]
if equal? :letter "f [setsh 69 stamper]
if equal? :letter "g [setsh 70 stamper]
if equal? :letter "h [setsh 72 stamper]
if equal? :letter "i [setsh 84 stamper]
if equal? :letter "j [setsh 71 stamper]
if equal? :letter "k [setsh 83 stamper]
if equal? :letter "l [setsh 73 stamper]
if equal? :letter "m [setsh 74 stamper]
if equal? :letter "n [setsh 75 stamper]
if equal? :letter "o [setsh 76 stamper]
if equal? :letter "p [setsh 77 stamper]
if equal? :letter "q [setsh 64 stamper.small
      setsh 84 stamper.small
      setsh 76 stamper]
if equal? :letter "r [setsh 73 stamper]
if equal? :letter "s [setsh 65 stamper]
if equal? :letter "t [setsh 79 stamper]
if equal? :letter "u [setsh 84 stamper.small
      setsh 76 stamper]
if equal? :letter "v [setsh 69 stamper]
if equal? :letter "w [setsh 76 stamper]
if equal? :letter "x [setsh 64 stamper.small
      setsh 65 stamper]
if equal? :letter "y [setsh 68 stamper]
if equal? :letter "z [setsh 82 stamper]
if equal? :letter char 32 [fd 30]
if equal? :letter char 13 [setx -150 sety ycor - 20]
if equal? :letter char 27 [cc stop]
translate readchar
end
```

-

~~Appendix I –~~
~~Translation Chart and Student Messages~~

-
-

-

-

-

-

-

-
-

-

-

-

Appendix II
Other Versions of Logo

This project was developed using the MSDOS version of LogoWriter. Below are the major differences you will need to take account of when adapting it to other versions of Logo.*

Apple and Macintosh LogoWriter

There are only minor differences between versions of LogoWriter:

LogoWriter for the Apple IIe uses a shape grid that is 10 x 14 pixels; so the shapes will not look quite the same. There are 30 shapes instead of 90. It will be necessary to replace most of the predefined shapes.

LogoWriter for the Macintosh uses a shape grid of 20 x 20 pixels.

On page 10 the suggested "backspace / erase key" depends upon the backspace key having an ASCII number of 8. This is true for MSDOS and Macintosh computers, but on Apple II computers the number is 127.

LCSI Logo II

LCSI Logo II has a 10 x 16 shape grid. Shapes are saved and retrieved in a different manner.

There are Logo language differences. The ef procedure isn't relevant and equal? should be changed to equalp.

Terrapin Logo for the Macintosh

Terrapin Logo for the Macintosh uses a shape grid of 30 x 30 pixels. The process of creating, editing, saving, and loading shapes is different. There are Logo language differences. The ef procedure isn't relevant. You'll need to use substitutes for type, cc, rg, and readchar.

---

* We only discuss versions of Logo that include the capability of creating original turtle shapes. It would be possible to develop this project using a Logo without such capability, but to do so would require a major redesign of the project.

~~Logo Plus~~

~~The Logo language differences mentioned above for Terrapin Logo for the Macintosh apply to Logo Plus, as well. Also, the syntax of the if command is different.~~

~~Shapes in Logo Plus are not limited to a preset grid and may be of any size.~~

~~There are also a number of extinct versions of Logo that allowed the user to create original shapes and are well suited to this project: TILOGO, Sprite Logo, Atari Logo, MSX Logo, and Smart Logo for the Coleco Adam.~~

-