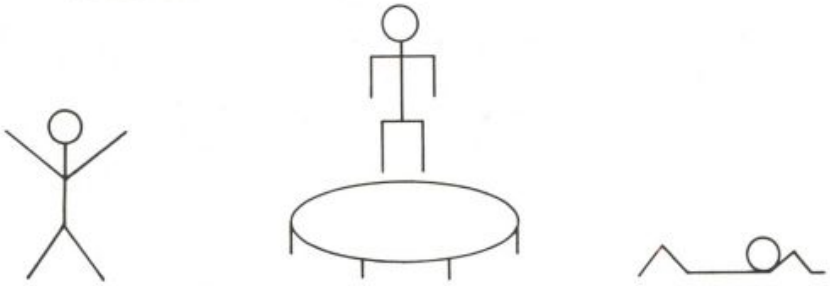
2

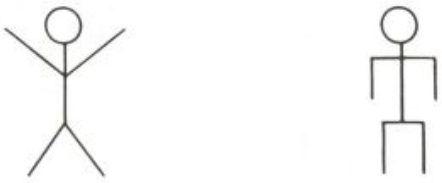
Stories

Exercise

In this project you watch a stick figure go through exercises consisting of jumping jacks and jumping on a trampoline. In the end, the figure collapses from exhaustion.



The idea for this animated story came about when I was playing with the shape editor. I wanted a stick figure to perform a jumping jack. I divided the jumping jack exercise into two parts. In the first part, the figure has its arms extended upward and legs apart; in the second part, the figure has its arms at its side and legs together. I designed the following two shapes in the shape editor to represent these stances.



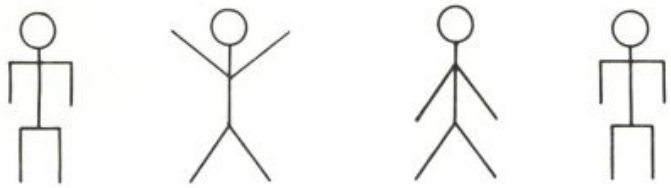
Next I wrote a short procedure to see what these shapes would look like when put together.

```
TO TEST
SETSH 1
WAIT 5
SETSH 2
WAIT 5
TEST
END
```

By Susan Cotten.

The movement was jerky. I made a third shape, which had its arms extended downward and its legs apart. I added this shape to `TEST`. The three shapes, interspersed with `WAIT`, made for a fairly smooth jumping jack.

Here is the sequence of shapes that I use for the jumping jack.

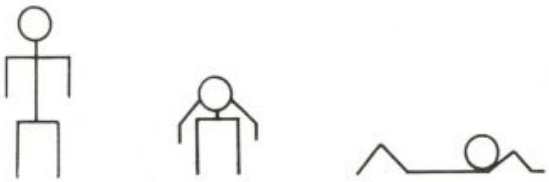


I began to think of other things I could do with my figure. What about jumping on a trampoline? I sketched a trampoline. I saw that the trampoline bed could be drawn using two arcs. After I'd figured out how to draw the trampoline, I put the first stick figure in the sequence on it and wrote a procedure to make the figure jump up and down on the trampoline. I also added a `TOOT` that sounded a *boing* whenever the figure bounced on the trampoline.

I then made a shape for the turtle to wear while getting up on the trampoline.

What next? I recalled how I felt after a good workout. It might be interesting to make the figure collapse on the floor. The figure would begin this sequence standing upright, then would bend over and collapse.

Here are the shapes.



I hadn't yet written the procedures to connect the exercises, but my general plan was to have the figure walk from sequence to sequence. These are the three shapes that I used for the walking motion.

I used nine shapes in my story. I named these shapes FIGURE1, FIGURE2, and so forth. The shapes as seen in the editor and the list of numbers describing them are printed at the end of this write-up.

Putting It All Together

Here are the procedures that make up EXERCISE. They are listed in order of use.

The top-level procedure in this program is EXERCISE.

```
TO EXERCISE
SETUP
JUMPING.JACKS
MOVE.TO.TRAMPOLINE
JUMPING.ON.TRAMPOLINE
GET.OFF.TRAMPOLINE
COLLAPSE
END
```

SETUP readies the screen and turtle for the first exercise sequence.

```
TO SETUP
TELL 0
CS HT FS
SETBG 72
READY.SHAPES
DRAW.TRAMPOLINE
READY.JUMPING.JACKS
END
```

READY.SHAPES takes care of putting the shapes into the shape table.

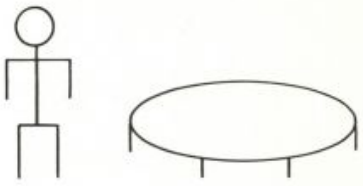
```
TO READY.SHAPES
PUTSH 1 :FIGURE1
PUTSH 2 :FIGURE2
PUTSH 3 :FIGURE3
PUTSH 4 :FIGURE4
PUTSH 5 :FIGURE5
PUTSH 6 :FIGURE6
PUTSH 7 :FIGURE7
PUTSH 8 :FIGURE8
PUTSH 9 :FIGURE9
END
```

DRAW.TRAMPOLINE draws the trampoline in the middle of the screen.

```
TO DRAW.TRAMPOLINE
SETPN 0
SETPC 0 37
TRAMP.BED
TRAMP.LEGS
END
```

```
TO TRAMP.BED
PD
RT 45
ARCR 8 5
RT 90
ARCR 8 5
SETH 0
END

TO ARCR :STEPS :TIMES
REPEAT :TIMES [RT 9 FD :STEPS RT 9]
END

TO TRAMP.LEGS
FRONT.LEG
SETPOS [8.67 -5]
BACK.LEG
SETPOS [28.67 -5]
BACK.LEG
SETPOS [36.67 0]
FRONT.LEG
END

TO FRONT.LEG
PD
BK 8
FD 8
PU
END

TO BACK.LEG
PD
BK 3
FD 3
PU
END
```

READY.JUMPING.JACKS moves the turtle over to the left side of the screen and sets its shape for the first exercise.



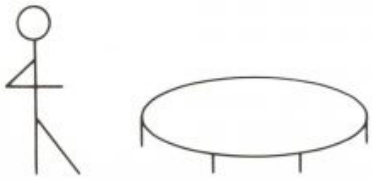
```
TO READY.JUMPING.JACKS
PU SETPOS [-60 0] PD
SETSH 3 SETC 7 ST
END
```

The figure performs six jumping jacks. Whenever the figure touches the ground, Logo makes a sound, a TOOT. At the end of this exercise, the turtle sets its shape to the standing figure.

```
TO JUMPING.JACKS
REPEAT 6 [JUMPING.JACKS1]
SETSH 3
END
```

```
TO JUMPING.JACKS1
SETSH 3 WAIT 20
SETSH 2 TOOT 0 80 10 5 WAIT 20
SETSH 1 WAIT 20
END
```

MOVE.TO.TRAMPOLINE takes care of moving the turtle over to and then up on the trampoline.



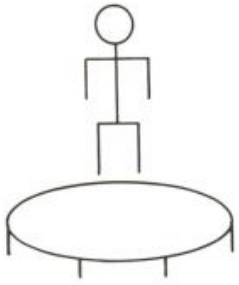
```
TO MOVE.TO.TRAMPOLINE
WALK 10
SETSH 7
SETH 45 FD 12
SETH 90 FD 20
SETH 0 BK 2
SETSH 3
END
```

The input to WALK tells it how far to move, that is, how many times to repeat USE.SHAPES.WALK.

```
TO WALK :DISTANCE
PU SETH 90
REPEAT :DISTANCE [USE.SHAPES.WALK]
END
```

```
TO USE.SHAPES.WALK
SETSH 4 WAIT 5 FD 5
SETSH 6 WAIT 5
SETSH 5 WAIT 5
SETSH 6 TOOT 0 200 15 2 WAIT 5
END
```

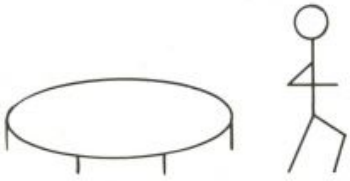
The figure is now standing on the trampoline. JUMPING.ON.TRAMPOLINE makes the figure jump up and down by running JUMP fifteen times.



```
TO JUMPING.ON.TRAMPOLINE
REPEAT 15 [JUMP]
END
```

```
TO JUMP
FD 15 WAIT 2
BK 15 WAIT 10
TOOT 0 80 10 5 WAIT 12
END
```

At this point the figure is through exercising. GET.OFF.TRAMPOLINE moves the figure down off the trampoline and then walks it a short distance.



```
TO GET.OFF.TRAMPOLINE
WALK 3
SETH 0
BK 10
WALK 5
END
```

The figure stops and lets out a musical sigh as it collapses on the floor. COLLAPSE relies on SIGH to do the work. SIGH is given a pitch to play and a list of shapes.

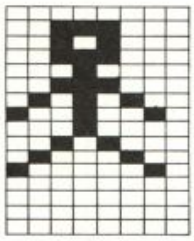
```
TO COLLAPSE
WAIT 35
SIGH 150 [3 3 3 8 8 8 9 9 9 9 9]
END

TO SIGH :NOTE :LIST
IF EMPTYP :LIST [STOP]
SETSH FIRST :LIST
TOOT 0 :NOTE 15 5
WAIT 5
SIGH :NOTE - 8 BF :LIST
END
```

PROGRAM LISTING

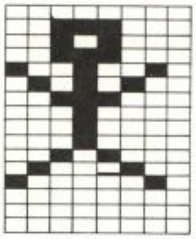
```
TO EXERCISE
SETUP
JUMPING.JACKS
MOVE.TO.TRAMPOLINE
JUMPING.ON.TRAMPOLINE
GET.OFF.TRAMPOLINE
COLLAPSE
END

TO SETUP
TELL 0
CS HT FS
SETBG 72
READY.SHAPES
DRAW.TRAMPOLINE
READY.JUMPING.JACKS
END

TO READY.SHAPES
PUTSH 1 :FIGURE1
PUTSH 2 :FIGURE2
PUTSH 3 :FIGURE3
PUTSH 4 :FIGURE4
PUTSH 5 :FIGURE5
PUTSH 6 :FIGURE6
PUTSH 7 :FIGURE7
PUTSH 8 :FIGURE8
PUTSH 9 :FIGURE9
END

TO DRAW.TRAMPOLINE
SETPN 0
SETPC 0 37
TRAMP.BED
TRAMP.LEGS
END

TO TRAMP.BED
PD
RT 45
ARCR 8 5
RT 90
ARCR 8 5
SETH 0
END

TO ARCR :STEPS :TIMES
REPEAT :TIMES [RT 9 FD :STEPS RT 9]
END

TO TRAMP.LEGS
FRONT.LEG
SETPOS [8.67 -5]
BACK.LEG
SETPOS [28.67 -5]
BACK.LEG
SETPOS [36.67 0]
FRONT.LEG
END
```

```
TO FRONT.LEG
PD
BK 8
FD 8
PU
END

TO BACK.LEG
PD
BK 3
FD 3
PU
END

TO READY.JUMPING.JACKS
PU SETPOS [-60 0] PD
SETSH 3 SETC 7 ST
END

TO JUMPING.JACKS
REPEAT 6 [JUMPING.JACKS1]
SETSH 3
END

TO JUMPING.JACKS1
SETSH 3 WAIT 20
SETSH 2 TOOT 0 80 10 5 WAIT 20
SETSH 1 WAIT 20
END

TO MOVE.TO.TRAMPOLINE
WALK 10
SETSH 7
SETH 45 FD 12
SETH 90 FD 20
SETH 0 BK 2
SETSH 3
END

TO WALK :DISTANCE
PU SETH 90
REPEAT :DISTANCE [USE.SHAPES.WALK]
END

TO USE.SHAPES.WALK
SETSH 4 WAIT 5 FD 5
SETSH 6 WAIT 5
SETSH 5 WAIT 5
SETSH 6 TOOT 0 200 15 2 WAIT 5
END

TO JUMPING.ON.TRAMPOLINE
REPEAT 15 [JUMP]
END

TO JUMP
FD 15 WAIT 2
BK 15 WAIT 10
TOOT 0 80 10 5 WAIT 12
END

TO GET.OFF.TRAMPOLINE
WALK 3
SETH 0
BK 10
WALK 5
END

TO COLLAPSE
WAIT 35
SIGH 150 [3 3 3 8 8 8 9 9 9 9 9]
END

TO SIGH :NOTE :LIST
IF EMPTYP :LIST [STOP]
SETSH FIRST :LIST
TOOT 0 :NOTE 15 5
WAIT 5
SIGH :NOTE - 8 BF :LIST
END


MAKE "FIGURE9 [0 0 0 0 0 0 0 0 0 0 0 ▶
     128 198 166 190 1]
MAKE "FIGURE8 [0 0 0 0 0 0 28 28 28 28 ▶
     54 54 54 54 20 20]
MAKE "FIGURE7 [0 28 20 28 8 24 40 61 ▶
     10 12 16 16 16 16 16 16]
MAKE "FIGURE6 [0 28 20 28 8 24 40 56 ▶
     12 8 20 20 18 18 18 0]
MAKE "FIGURE5 [0 28 20 28 8 24 40 60 8 ▶
     8 20 20 20 20 20 0]
MAKE "FIGURE4 [0 28 20 28 8 24 40 60 8 ▶
     12 18 18 18 34 34 0]
MAKE "FIGURE3 [0 56 40 56 16 56 56 56 ▶
     56 16 40 40 40 40 40 40]
MAKE "FIGURE2 [0 56 40 56 146 84 56 16 ▶
     16 16 40 68 130 0 0 0]
MAKE "FIGURE1 [0 56 40 56 16 56 84 146 ▶
     16 40 68 130 0 0 0 0]
```

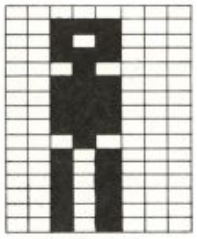
SHAPES



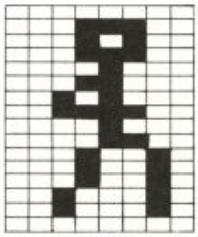
:FIGURE1



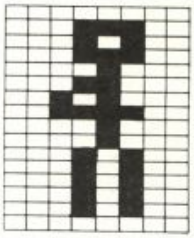
:FIGURE2



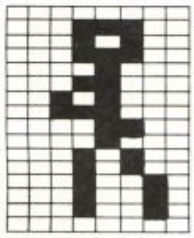
:FIGURE3



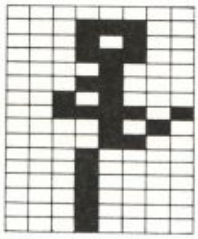
:FIGURE4



:FIGURE5

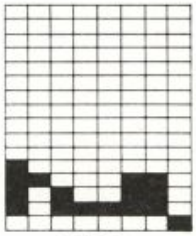


:FIGURE6



:FIGURE7

:FIGURE8



:FIGURE9

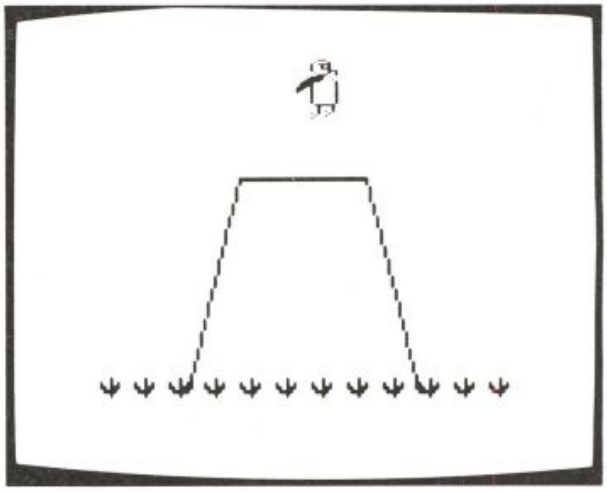
Cartoon

In CARTOON a bird flies around chirping, flapping its wings, and fertilizing the world. Eventually it settles on a mountaintop and lays an egg, whereupon the bird flies off forever. The egg hatches into a little bird, which also flies away.

This project is one that I developed over a long period of time. I started out wanting to explore graphics and dynamics with turtles. As I began to sketch out ideas in the shape editor, I stumbled onto a design that looked like a bird's head and upper body. I used a cartooning technique of drawing in outline a side view of the bird. As soon as the bird emerged as the central character, the rest of the project began to suggest itself. The overall goal became clear: to make a bird and then make it fly.

In the following discussion I try to convey the process I went through in developing this project. If you want to look at the completed program, turn to the listing at the end.

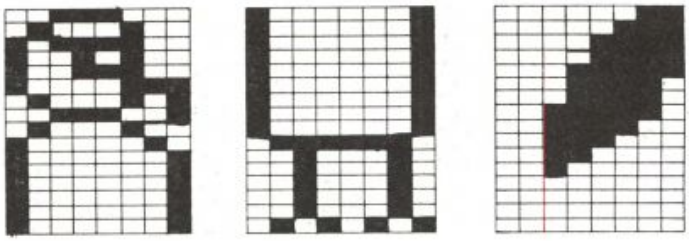
By Erric Solomon.



Making the Bird

Making the bird was a long process of sketching the parts in the shape editor and then testing them out in relation to the other parts. I used one shape for the bird's head and the top of its body and another shape for the bottom of the body and the legs. I also used a separate shape for the wing. Since I decided on a side view, only one wing was needed. I put the first wing shape in shape 3 and used shape 1 for the top of the bird and shape 2 for the bottom of the bird. I chose a yellowish color for the outline of the bird and picked a purple shade for the wing, which I made as a solid figure.

Here is the way these shapes look in the shape editor.



I named the shapes `TOPBIRD`, `BOTTOMBIRD`, and `WING1`.

```
MAKE "TOPBIRD GETSH 1
MAKE "BOTTOMBIRD GETSH 2
MAKE "WING1 GETSH 3
```

I set up the bird giving each of the three turtles a shape and a position. These instructions make up the procedure `BIRD`.

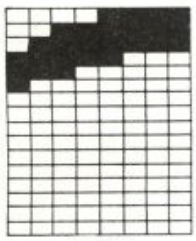
```
TO BIRD
PUTSH 1 :TOPBIRD
PUTSH 2 :BOTTOMBIRD
PUTSH 3 :WING1
TELL 0 SETPOS [-9 -10] SETSH 3 SETC 53
TELL 1 SETPOS [0 0] SETSH 1 SETC 12
TELL 2 SETY -15 SETSH 2 SETC 12
TELL [0 1 2] ST
END
```

Turtle 0 is :WING1 (shape 3), turtle 1 is :TOPBIRD (shape 1), and turtle 2 is :BOTTOMBIRD (shape 2).

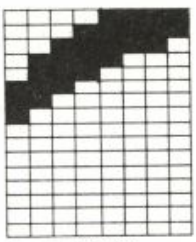
I used a trick here; turtle 0 assumes the role of the wing so that it is visible over the bird's body. (Lower-numbered turtles cover higher-numbered ones.)

Animating the Bird

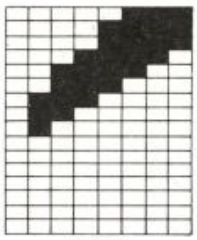
I thought I would make several wing shapes to represent different flapping positions. Making the wings took a lot of experimentation; I made several until I had three that were satisfactory. Here are those three shapes.



:WING2



:WING3



:WING4

I wrote MOVEWING to change the wing by putting new shapes in slot 3 as they were needed for the animation. The input to MOVEWING indicates how long Logo waits before the shape is changed.

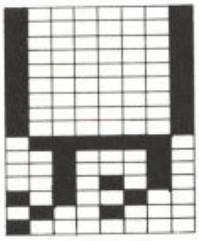
```
TO MOVEWING :DELAY
PUTSH 3 :WING1 WAIT :DELAY
PUTSH 3 :WING4 WAIT :DELAY
PUTSH 3 :WING3 WAIT :DELAY
PUTSH 3 :WING2 WAIT :DELAY
PUTSH 3 :WING3 WAIT :DELAY
PUTSH 3 :WING4 WAIT :DELAY
END
```

At first I was going to put all the wings in the shape table at the same time, but I had made so many shapes I exhausted the shape table slots. Instead I used only slot 3.

I coordinated the speed the bird travels with its wing movement and finally settled on the following:

```
BIRD
SETH 90 SETSP 5
REPEAT 30 [MOVEWING 5]
```

Although I was pleased with the way the wings looked, I didn't like the bird's legs. I wanted them to curl up for flying. So I drew a new bird bottom with the legs up, which I referred to as LEGGSUP.



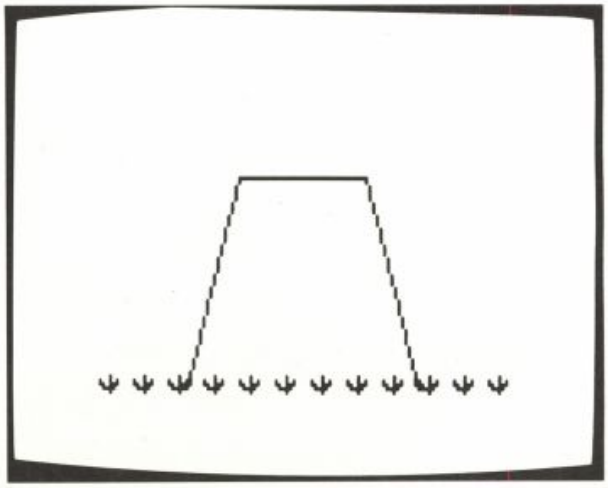
:LEGGSUP

The Scenery

The bird needed a place to fly from, so I made a mountain. In the process I decided to add clumps of grass at the bottom of the mountain.

```
TO MOUNTAIN
PU SETPOS [-65 -90]
PD SETPOS [-35 37]
RT 90 FD 70
SETPOS [65 -90]
END
```

The grass was made in clumps.

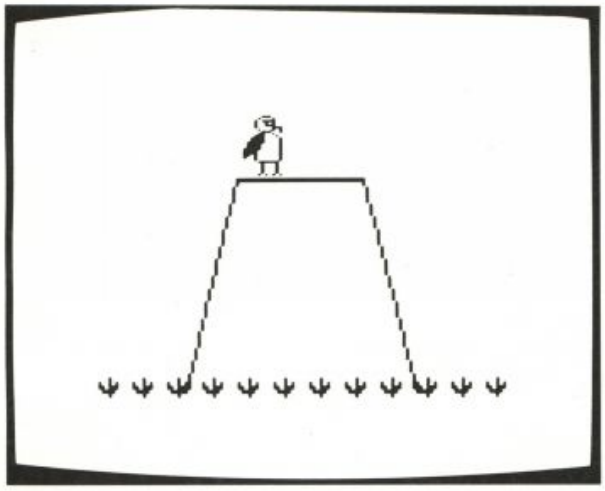


```
TO GRASS
SETH -60 FD 5 BK 5
RT 30 FD 8 BK 8
RT 30 FD 10 BK 10
RT 30 FD 8 BK 8
RT 30 FD 5 BK 5
END
```

I used all four turtles to draw the grass so that this drawing takes place faster.

```
TO DRAWGRASS
TELL [0 1 2 3] SETPN 0 PU
ASK 0 [SETPOS [-110 -90]]
ASK 1 [SETPOS [-90 -90]]
ASK 2 [SETPOS [-70 -90]]
ASK 3 [SETPOS [-50 -90]]
REPEAT 3 [PD GRASS SETH 90 PU FD 80]
END
```

I then changed `BIRD` so that the bird perched on the mountaintop.

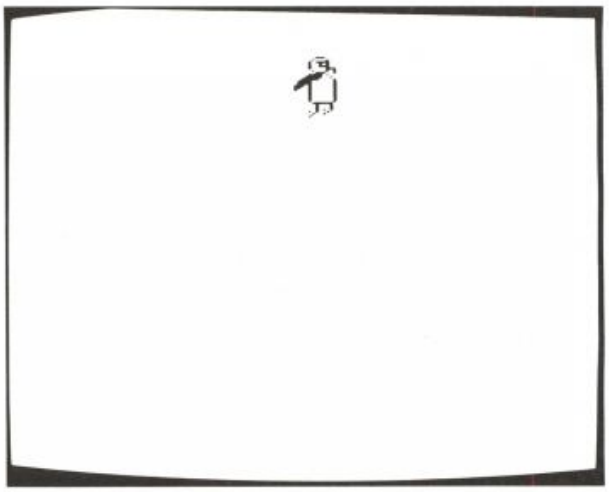


```
TO BIRD
PUTSH 1 :TOPBIRD
PUTSH 2 :BOTTOMBIRD
PUTSH 3 :WING1
TELL 0 SETPOS [-9 -10] SETSH 3 SETC 53
TELL 1 SETPOS [0 0] SETSH 1 SETC 12
TELL 2 SETY -15 SETSH 2 SETC 12
TELL [0 1 2]
EACH [SETPOS SE XCOR - 20 YCOR + 65]
ST
END
```

Making the Bird Fly

After getting the bird to sit on the mountain, I wanted it to take off, fly around, and then land. To produce the animated portion of the story, I anticipated needing several procedures like TAKEOFF, FLAP, and LANDING.

TAKEOFF adjusts the bird's heading and speed for its ascent and leveling off. In the process the bird's legs are raised and its wing flaps.



```
TO TAKEOFF
PUTSH 2 :LEGGSUP
SETH 45
SETSP 5 FLAPFLAP 5 5
SETSP 10 SETH 67
FLAPFLAP 10 5
SETH 90
END
```

FLAPFLAP calls MOVEWING.

```
TO FLAPFLAP :TIMES :DELAY
REPEAT :TIMES [MOVEWING :DELAY]
END
```

(Note that the bigger :DELAY is, the slower the wing flaps.)

Assembling CARTOON

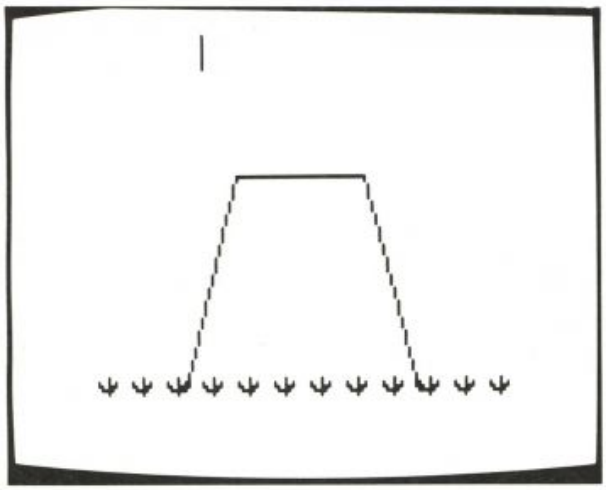
I put all these instructions into a procedure.

```
TO CARTOON
TELL [0 1 2 3] HT CS FS
SETBG 0 SETPN 0 SETPC 0 6
MOUNTAIN
DRAWGRASS
BIRD
TAKEOFF
END
```

Landing the Bird

Once I got the bird to take off, I wanted to make it land. I had to figure out a way to invoke the landing process and then coordinate the bird's heading so that it would land on the mountain. I also had to find a way to stop the bird once it reached the mountaintop.

An obvious way to stop the bird once it landed on the mountain was to set up a demon to watch for turtle 2 colliding with the mountaintop drawn by pen 0. A much harder problem was figuring out how to start the descent. I wanted to use a demon, but I wasn't sure how. I finally happened upon a technique that I found extremely useful in other parts of this project. I made a wall in the sky by drawing a vertical line above the mountain in the background color. Turtle 2 would inevitably collide with this line. I set up a demon to watch for this event and then invoke the landing.



SET.MARKS draws the line. I put SET.MARKS in CARTOON directly after MOUNTAIN draws the mountain using turtle 0 with pen 0. This mark is drawn by turtle 0 with pen 1.

```
TO SET.MARKS
SETPN 1 SETPC 1 BG
PU SETPOS [-55 100]
PD SETY 120 PU
END
```

I put the WHEN instruction in CARTOON immediately following DRAWGRASS.

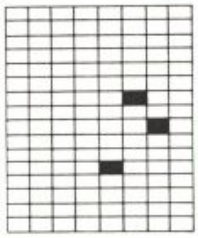
```
WHEN OVER 2 1 [LANDING]
```

LANDING changes the bird's flight direction and sets up a demon to look for a collision between the bird and the mountain.

```
TO LANDING
TELL [0 1 2 3] SETH 130
PUTSH 3 :WING1
WHEN OVER 2 0 [ PUTSH 2 :BOTTOMBIRD SETSP 0]
END
```

Fertilizing the World

While birds fly around they also drop sprinklings of digested matter. This bird is no different. I designed TURD and put it in shape 4.



I wanted TURD to suddenly appear and fall to the ground as the bird continues its flight. I used turtle 3 for this role. Turtle 3 would travel with turtle 2, but invisibly. To camouflage turtle 3, I set its color to the background color until the proper moment. I changed BIRD:

```
TO BIRD
PUTSH 1 :TOPBIRD
PUTSH 2 :BOTTOMBIRD
PUTSH 3 :WING1
PUTSH 4 :TURD
TELL 0 SETPOS [-9 -10] SETSH 3 SETC 53
TELL 1 SETPOS [0 0] SETSH 1 SETC 12
TELL 2 SETY -15 SETSH 2 SETC 12
TELL 3 SETPOS [-9 -15]  SETSH 4 SETC BG
TELL [0 1 2 3]
EACH [SETPOS SE XCOR - 20 YCOR + 65]
ST
END
```

Here is the path I planned for the fertilizer.

```
TELL 3 SETH 180 SETC 7
SETSP 25 ST
```

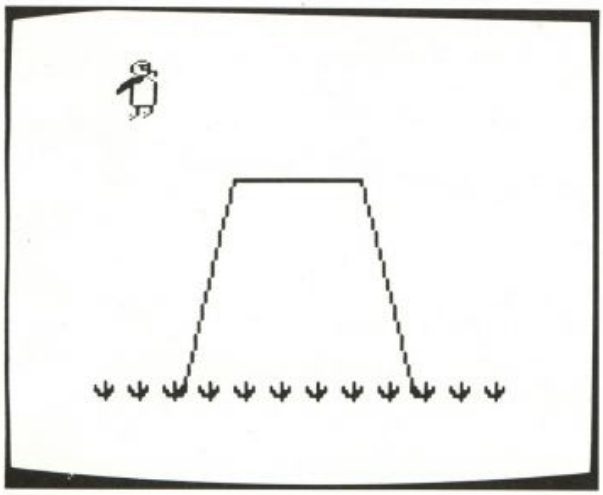
I made one addition; I added sound effects to attract the viewer's attention. I then redirected the turtle commands to the bird turtles.

```
TOOT 1 30 7 20
TELL [0 1 2]
```

I put these instructions in a new procedure called DROP.TURD.

I had to face a couple of problems: when would the bird release the fertilizer and how would turtle 3 know when it hit the ground and so stop?

I concentrated on how to activate DROP.TURD. I wanted the bird to take off and fly for a while before its musical gift descended to earth. The bird would perform this feat after crossing the screen once.



I used the same technique I used for LANDING. To do this, I drew an invisible wall on the screen running vertically at the place where I wanted turtle 3 to begin its visible descent. I used pen 2 to draw the line. The pencolor was the same as the background color; it was invisible to the viewer, but not to a turtle or a demon. I changed SET.MARKS.

```
TO SET.MARKS
TELL 0
SETPN 1 SETPC 1 BG
PU SETPOS [-55 100]
PD SETY 120 PU
SETPN 2 SETPC 2 BG
SETPOS [-50  100]
PD SETY 120 PU
END
```

I set up a demon to watch for turtle 2 (:LEGGSUP) colliding with this line.

```
WHEN OVER 2 2 [DROP.TURD]
```

I had another problem. When does this demon get activated? I wanted this event to happen before the bird starts its landing.

I changed the WHEN instruction so that the demon invokes DROPANDLAND instead of DROP.TURD.

```
WHEN OVER 2 2 [DROPANDLAND]
```

DROPANDLAND dismisses the currently active demon, runs DROP.TURD, and sets up a new demon to invoke LANDING.

```
TO DROPANDLAND
WHEN OVER 2 2 []
DROP.TURD
WHEN OVER 2 1 [LANDING]
END
```

I used the same trick for making turtle 3 as TURD disappear. This time I drew an invisible line at the bottom of the mountain in the background color. Since the visible scenery was drawn with pen 0, I decided to use pen 2 for the line drawn in the background color.

```
SETPN 2 SETPC 2 BG
PU SETPOS [110 -90]
PD SETX -110 PU
```

I added these instructions to SET.MARKS.

I set up a demon to watch for the collision between turtle 3 and pen line 2. The demon invokes a procedure to make the turtle vanish.

Now DROP.TURD looks like this:

```
TO DROP.TURD
WHEN OVER 3 2 [DISAPPEAR]
TELL 3 SETH 180 SETC 7
SETSP 25 ST
TOOT 1 30 7 20
TELL [0 1 2]
END

TO DISAPPEAR
ASK 3 [SETSP 0 HT]
WHEN OVER 3 2 []
END
```

CARTOON *Updated*

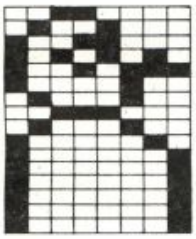
```
TO CARTOON
TELL [0 1 2 3] HT CS FS
SETBG 0 SETPN 0 SETPC 0 6
MOUNTAIN
SET.MARKS
DRAWGRASS
WHEN OVER 2 2 [DROPANDLAND]
RECYCLE
BIRD
TAKEOFF
END
```

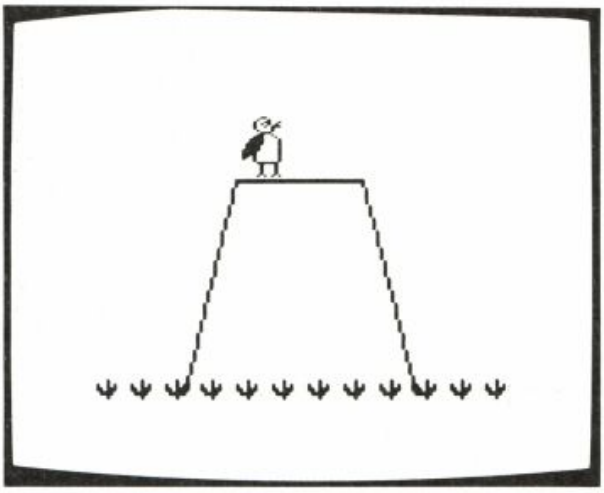
Additional Sound Effects and Graphics

I liked the sound effects in DROP.TURD and decided that the bird should sing before taking off in flight. To do this the bird needed a tune.

```
TO SONG
TOOT 0 1000 7 7
TOOT 0 1100 7 7
TOOT 0 1150 7 7
TOOT 0 1050 7 7
END
```

When birds sing, they open and close their mouths. This action required another shape for the bird.





I refer to this shape as TOPBIRD2. SING alternates TOPBIRD with TOPBIRD2 as it calls SONG.

```
TO SING :TIMES
REPEAT :TIMES [PUTSH 1 :TOPBIRD2 SONG
  PUTSH 1 :TOPBIRD WAIT 20 + RANDOM 30]
END
```

FLY incorporated SING with the bird in flight.

```
TO FLY
TELL [0 1 2 3]
WAIT 60 SING 5 WAIT 60
TAKEOFF SETSP 15
FLAP 7
WAIT 30 SING 5 WAIT 30
END
```

FLY uses TAKEOFF and also FLAP. FLAP stops when the bird lands on the mountain.

```
TO FLAP :DELAY
MOVEWING :DELAY
IF COND OVER 2 0 [STOP]
FLAP :DELAY
END
```

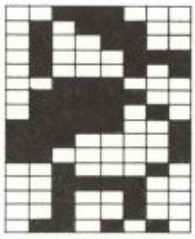
I put FLY in CARTOON in place of TAKEOFF.

More Frills

The bird lays an egg and flies away. The egg hatches and a little bird flies off. First I made a little bird. This bird fits in one turtle shape.

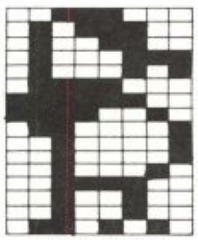
:LITTLEBIRD1

Since I had already made one bird fly, this one was not a problem. I made a total of four shapes for this little bird to show it in various stages of flight.



:LITTLEBIRD2

:LITTLEBIRD3



:LITTLEBIRD4

I used the same technique for this animation as I did for the wing flapping for the big bird.

```
TO FLIP
PUTSH 4 :LITTLEBIRD1 WAIT 10
PUTSH 4 :LITTLEBIRD2 WAIT 10
PUTSH 4 :LITTLEBIRD3 WAIT 10
PUTSH 4 :LITTLEBIRD4 WAIT 8.5
PUTSH 4 :LITTLEBIRD3 WAIT 10
PUTSH 4 :LITTLEBIRD2 WAIT 10
END
```

The bird shapes are put in shape 4 in the shape table. Turtle 3 would now be the little bird.*

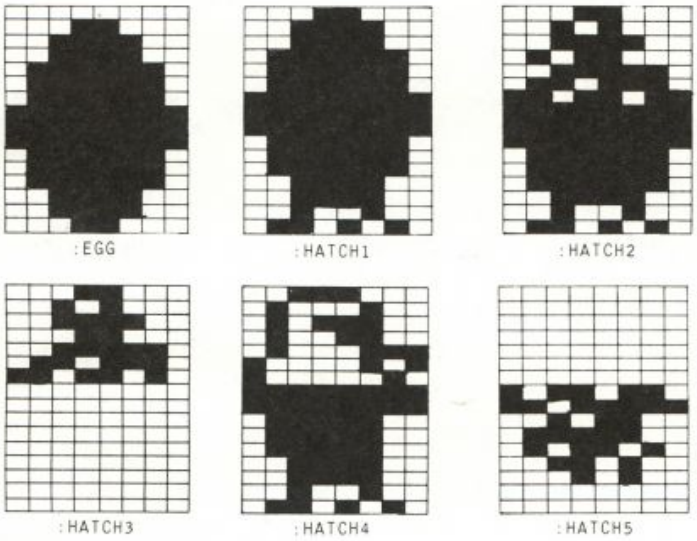
FLYOFF makes the little bird fly away.

```
TO FLYOFF
TELL 3 SETH 45
SETSP 5 WING 3
SETSP 10 SETH 67 WING 5
SETH 90
WING 5 HT
END

TO WING :TIMES
REPEAT :TIMES [FLIP]
END
```

Hatching the Egg

Turtle 3 eventually is changed from an egg to a bird. The egg hatching process consisted of designing shapes and playing around with how they interact with one another.

I drew the egg shape and five more shapes to depict the hatching process.



:HATCH3 and :HATCH5 are fragments of the top and the bottom of the egg. Having these two shapes superimposed on the more complete shapes of :HATCH4 and :LITTLEBIRD1 enhances the hatching animation. It simplifies the pictures I had to draw. This superimposition happens by having turtle

*Notice that the delay for :LITTLEBIRD4 is shorter than the other delays. I did this because the wing in that shape is clipped. It extends beyond the shape's dimensions. With the shorter delay you don't notice the missing wing part, but you do get an uninterrupted sense of a flapping motion.

0 assume these fragments. The shape assumed by turtle 3 shows as well as the added details offered by turtle 0. In fact, this is a frill. Even without turtle 0's participation, the hatching scenerio is quite acceptable.

```
TO HATCH
TELL 0 SET.EGG
SETSH 1
PUTSH 4 :EGG WAIT 30
PUTSH 4 :HATCH1 WAIT 30
PUTSH 4 :HATCH2 WAIT 30
PUTSH 1 :HATCH3 WAIT 30
ST
PUTSH 4 :HATCH4 WAIT 50
HT
PUTSH 1 :HATCH5 WAIT 50
ST
PUTSH 4 :LITTLEBIRD1 WAIT 50
HT
END
```

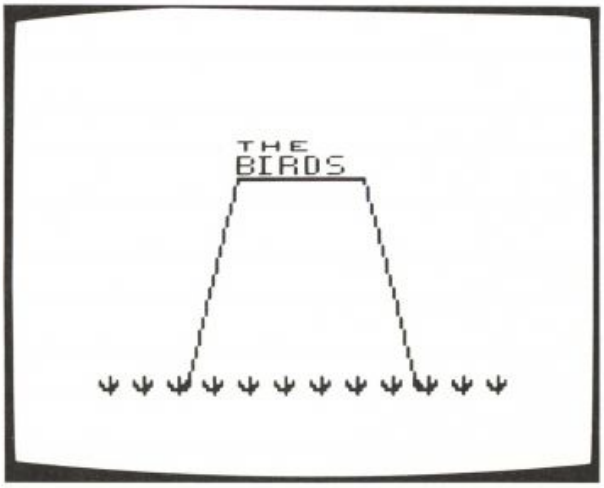
The following instructions prepare for HATCH.

```
CARTOON
TELL 3 SETSP 0
SETPOS [-7 48]
SETC 7
PUTSH 4 :EGG ST
TELL [0 1 2] TAKEOFF
FLAPFLAP 1 15 HT
```

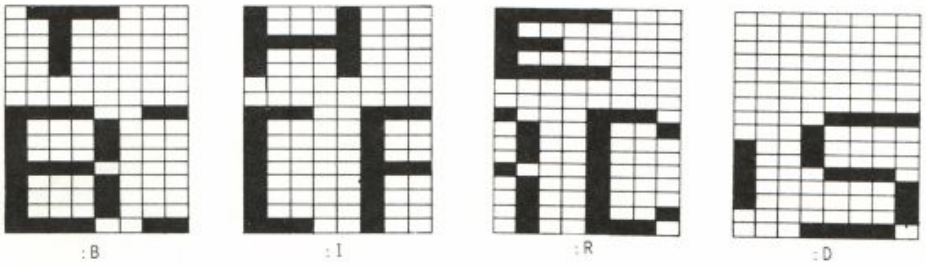
Then

```
HATCH
```

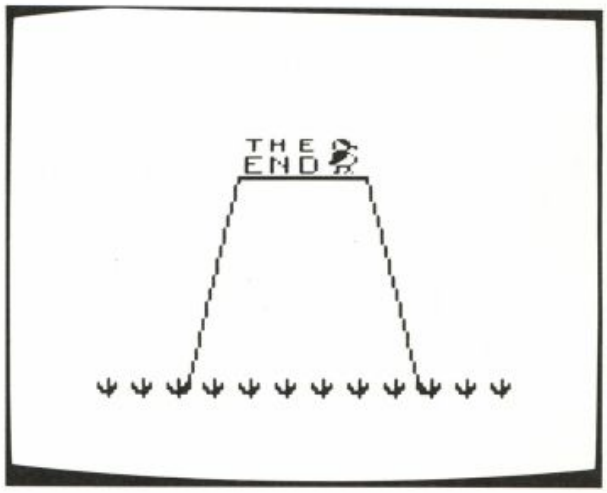
As a final touch I added a title at the beginning and one at the end. These procedures and shapes are included in the following listing.

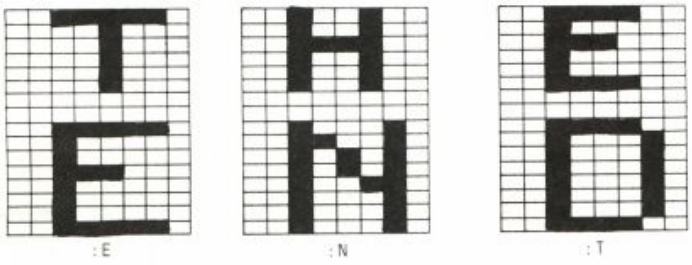



```
TO BIRDS.SIGN
TELL [0 1 2 3] HT
PU SETH 90
SETPOS [-30 50]
ASK 0 [SETSH 1]
PUTSH 1 :B
ASK 1 [FD 15 SETSH 2]
PUTSH 2 :I
ASK 2 [FD 30 SETSH 3]
PUTSH 3 :R
ASK 3 [FD 45 SETSH 4]
PUTSH 4 :D
TELL :ALL SETC BG ST
SETC 56 ST
WAIT 20
REPEAT 4 [SETC COLOR + 1 WAIT 20]
END
```




```
TO THEEND.SIGN
TELL [0 1 2 3] HT
PU SETH 90
SETPOS [-30 50]
ASK 0 [SETSH 1]
PUTSH 1 :E
ASK 1 [FD 15 SETSH 2]
PUTSH 2 :N
ASK 2 [FD 30 SETSH 3]
PUTSH 3 :T
ASK 3 [FD 50 SETSH 4]
PUTSH 4 :LITTLEBIRD1
SETC BG ST
SETC 88 WAIT 20
REPEAT 4 [SETC COLOR + 1 WAIT 20]
SETBG 74 SETPC 1 74 SETPC 2 74
WAIT 60
END
```





I put all these new instructions into CARTOON and made a new procedure SET.EGG.

```
TO SET.EGG
SETSP 0 SETPOS [-7 48]
SETC 7
END
```

Some other nice project ideas for these birds are to make them walk, make one of them find a worm and then fly off, make different background scenery, and so on.

PROGRAM LISTING

```
TO CARTOON
RECYCLE
TELL [0 1 2 3] HT CS FS
SETBG 0 SETPN 0 SETPC 0 6
MOUNTAIN
SET.MARKS
DRAWGRASS
WHEN OVER 2 2 [DROPANDLAND]
BIRDS.SIGN RECYCLE
WAIT 60
BIRD
FLY
ASK 3 [SET.EGG PUTSH 4 :EGG ST]
TELL [0 1 2] TAKEOFF
FLAPFLAP 1 15 HT
HATCH WAIT 30
ASK 3 [FLYOFF]
TELL [0 1 2 3] SETSP 0
THEEND.SIGN
END

TO MOUNTAIN
PU SETPOS [-65 -90]
PD SETPOS [-35 37]
RT 90 FD 70
SETPOS [65 -90]
END

TO SET.MARKS
TELL 0
SETPN 1 SETPC 1 BG
PU SETPOS [-55 100]
PD SETY 120 PU
SETPN 2 SETPC 2 BG
SETPOS [-50 100]
PD SETY 120 PU
SETPOS [110 -90]
PD SETX -110 PU
END

TO DRAWGRASS
TELL [0 1 2 3] SETPN 0 PU
ASK 0 [SETPOS [-110 -90]]
ASK 1 [SETPOS [-90 -90]]
ASK 2 [SETPOS [-70 -90]]
ASK 3 [SETPOS [-50 -90]]
REPEAT 3 [PD GRASS SETH 90 PU FD 80]
END

TO GRASS
SETH -60 FD 5 BK 5
RT 30 FD 8 BK 8
RT 30 FD 10 BK 10
RT 30 FD 8 BK 8
RT 30 FD 5 BK 5
END

TO BIRDS.SIGN
TELL [0 1 2 3] HT
PU SETH 90
SETPOS [-30 50]
ASK 0 [SETSH 1]
PUTSH 1 :B
ASK 1 [FD 15 SETSH 2]
PUTSH 2 :I
ASK 2 [FD 30 SETSH 3]
PUTSH 3 :R
ASK 3 [FD 45 SETSH 4]
PUTSH 4 :D
SETC 56 ST
WAIT 20
REPEAT 4 [SETC COLOR + 1 WAIT 20]
END

TO BIRD
PUTSH 1 :TOPBIRD
PUTSH 2 :BOTTOMBIRD
PUTSH 3 :WING1
PUTSH 4 :TURD
TELL 0 SETPOS [-9 -10] SETSH 3 SETC 53
TELL 1 SETPOS [0 0] SETSH 1 SETC 12
TELL 2 SETY -15 SETSH 2 SETC 12
TELL 3 SETPOS [-9 -15]  SETSH 4 SETC ▸
     BG
TELL [0 1 2 3]
EACH [SETPOS SE XCOR - 20 YCOR + 65]
ST
END

TO DROPANDLAND
WHEN OVER 2 2 []
DROP.TURD
WHEN OVER 2 1 [LANDING]
END

TO DROP.TURD
WHEN OVER 3 2 [DISAPPEAR]
TELL 3 SETH 180 SETC 7
SETSP 25 ST
TOOT 1 30 7 20
TELL [0 1 2]
END
```

```
TO DISAPPEAR
ASK 3 [SETSP 0 HT]
WHEN OVER 3 2 []
END

TO LANDING
TELL [0 1 2 3] SETH 130
PUTSH 3 :WING1
WHEN OVER 2 0 [PUTSH 2 :BOTTOMBIRD ►
     SETSP 0]
END

TO TAKEOFF
PUTSH 2 :LEGGSUP
SETH 45
SETSP 5 FLAPFLAP 5 5
SETSP 10 SETH 67
FLAPFLAP 10 5
SETH 90
SING 1
END

TO FLY
TELL [0 1 2 3]
WAIT 60 SING 5 WAIT 60
TAKEOFF SETSP 15
FLAP 7
WAIT 30 SING 5 WAIT 30
END

TO FLAPFLAP :TIMES :SPEED
REPEAT :TIMES [MOVEWING :SPEED]
END

TO MOVEWING :SPEED
PUTSH 3 :WING1 WAIT :SPEED
PUTSH 3 :WING4 WAIT :SPEED
PUTSH 3 :WING3 WAIT :SPEED
PUTSH 3 :WING2 WAIT :SPEED
PUTSH 3 :WING3 WAIT :SPEED
PUTSH 3 :WING4 WAIT :SPEED
END

TO FLYOFF
SETH 45
SETSP 5 WING 3
SETSP 10 SETH 67 WING 5
SETH 90
WING 5 SONG HT
END

TO WING :TIMES
REPEAT :TIMES [FLIP]
END

TO FLAP :SPEED
MOVEWING :SPEED
IF COND OVER 2 0 [STOP]
FLAP :SPEED
END

TO FLIP
PUTSH 4 :LITTLEBIRD1 WAIT 10
PUTSH 4 :LITTLEBIRD2 WAIT 10
PUTSH 4 :LITTLEBIRD3 WAIT 10
PUTSH 4 :LITTLEBIRD4 WAIT 8.5
PUTSH 4 :LITTLEBIRD3 WAIT 10
PUTSH 4 :LITTLEBIRD2 WAIT 10
END

TO SET.EGG
SETSP 0 SETPOS [-7 48]
SETC 7
END

TO HATCH
TELL [0 1 2 3] SET.EGG
PUTSH 4 :EGG SETSH 1  WAIT 30
PUTSH 4 :HATCH1 WAIT 30
PUTSH 4 :HATCH2 WAIT 30
PUTSH 1 :HATCH3 WAIT 30
TELL 0 ST
PUTSH 4 :HATCH4 WAIT 50
HT
PUTSH 1 :HATCH5 WAIT 50
ST
PUTSH 4 :LITTLEBIRD1 WAIT 50
HT
END

TO THEEND.SIGN
TELL [0 1 2 3] HT
PU SETH 90
SETPOS [-30 50]
ASK 0 [SETSH 1]
PUTSH 1 :E
ASK 1 [FD 15 SETSH 2]
PUTSH 2 :N
ASK 2 [FD 30 SETSH 3]
PUTSH 3 :T
ASK 3 [FD 50 SETSH 4]
PUTSH 4 :LITTLEBIRD1
SETC BG ST
SETC 88 WAIT 20
REPEAT 4 [SETC COLOR + 1 WAIT 20]
SETBG 74 SETPC 1 74 SETPC 2 74
WAIT 60
END
```

```
TO SING :TIMES
REPEAT :TIMES [PUTSH 1 :TOPBIRD2 SONG ►
     PUTSH 1 :TOPBIRD WAIT 20 + RANDOM ►
     30]
END

TO SONG
TOOT 0 1000 7 7
TOOT 0 1100 7 7
TOOT 0 1150 7 7
TOOT 0 1050 7 7
END

MAKE "TOPBIRD2 [48 72 153 170 151 132 ►
     68 56 68 130 129 129 129 129 129 ►
     129]
MAKE "LITTLEBIRD4 [56 92 76 71 69 120 ►
     252 242 97 33 33 34 60 36 36 106]
MAKE "LITTLEBIRD3 [56 92 76 71 69 56 ►
     124 242 193 161 161 34 60 36 36 ►
     106]
MAKE "LITTLEBIRD2 [56 92 76 71 69 56 ►
     124 122 249 225 193 34 60 36 36 ►
     106]
MAKE "HATCH5 [0 0 0 0 0 0 0 182 223 44 ►
     124 90 52 20 0 0]
MAKE "HATCH4 [56 68 92 76 71 133 130 ►
     255 255 126 126 126 60 60 36 106]
MAKE "HATCH3 [24 40 60 88 62 110 218 0 ►
     0 0 0 0 0 0 0 0]
MAKE "HATCH2 [24 40 60 88 62 110 218 ►
     255 255 255 126 126 126 60 36 ►
     106]
MAKE "HATCH1 [24 60 60 126 126 126 255 ►
     255 255 126 126 126 60 60 36 ►
     106]
MAKE "LEGGSUP [129 129 129 129 129 129 ►
     129 129 129 126 34 34 42 164 72 ►
     128]
MAKE "TURD [0 0 0 0 0 0 4 0 2 0 0 8 0 ►
     0 0 0]
MAKE "TOPBIRD [56 68 188 148 156 135 ►
     69 57 68 130 129 129 129 129 129 ►
     129]
MAKE "BOTTOMBIRD [129 129 129 129 129 ►
     129 129 129 129 126 34 34 34 34 ►
     34 85]
MAKE "EGG [24 60 60 126 126 126 255 ►
     255 255 126 126 126 60 60 24 0]
MAKE "WING2 [15 63 127 248 224 128 0 0 ►
     0 0 0 0 0 0 0 0]
MAKE "WING3 [15 31 62 120 112 224 192 ►
     128 0 0 0 0 0 0 0 0]
MAKE "WING4 [7 15 15 30 60 56 112 96 ►
     64 0 0 0 0 0 0 0]
MAKE "WING1 [3 7 15 15 31 30 30 62 60 ►
     56 48 32 0 0 0 0]
MAKE "D [0 0 0 0 0 0 0 0 15 16 144 144 ►
     142 129 129 1 30]
MAKE "R [248 128 224 128 248 0 0 142 ►
     73 72 72 136 72 72 73 78]
MAKE "I [136 136 248 136 136 0 0 231 ►
     132 132 132 135 132 132 132 228]
MAKE "B [248 32 32 32 32 0 0 243 136 ►
     136 136 240 136 136 136 243]
MAKE "LITTLEBIRD1 [56 92 76 71 69 56 ►
     124 122 121 113 225 194 60 36 36 ►
     106]
MAKE "T [60 32 56 32 32 60 0 0 60 34 ►
     34 34 34 34 34 60]
MAKE "N [36 36 60 36 36 36 0 0 34 50 ►
     42 42 38 34 34 34]
MAKE "E [62 8 8 8 8 8 0 0 62 32 32 56 ►
     32 32 32 62]
```

Jack and Jill

This animated story plays freely with two nursery rhymes I knew by heart when I was very young. The rhymes are "The House That Jack Built" and "Jack and Jill." I did not try to retell either of them in any literal sense. Instead, I wanted the story to have in it some of my feelings for these memories of childhood. I wanted the animation to be lively, humorous, and whimsical. I let the story turn on graphics and animation features, and concentrated on how the animation felt.

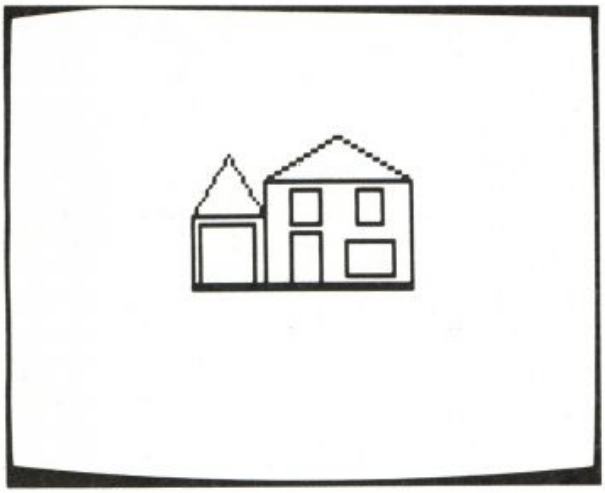
I built this story from a number of smaller projects I had written. I never planned these small projects as parts of a single animated feature, but

By Michael Grandfield.

I realized at some point that they could be used in this way. This write-up describes how the parts were made; it does not offer a description of the final animated story. However, a computer listing of the program is provided at the end of this write-up.

Building a House

Here is a house. Drawing it was the beginning of this whole project.



The funny thing about a picture of a house is that it doesn't let you know anything about how it was drawn. A nice thing about animation is that you can make the way the house is drawn an important, interesting process; and that is what the project was about.

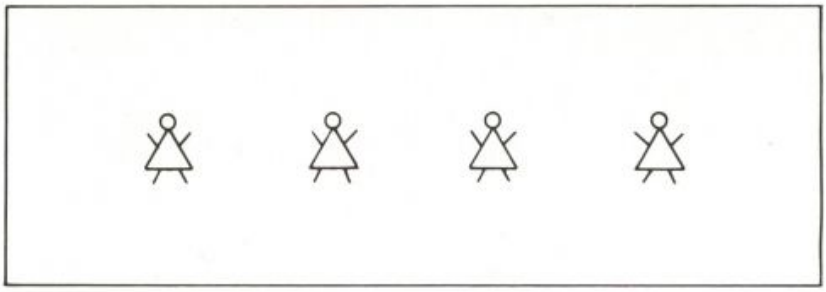
One way to draw this house would be to write individual procedures for each of the shapes that are in the design. The house needs three shapes: a square for the house; a rectangle for the garage, the doors, and the windows; and a triangle for the roofs. Once these procedures are written, you can put together a procedure that uses them all to draw a house.

```
TO DRAW.HOUSE
HOUSE
GARAGE
WINDOWS
ROOFS
END
```

This is a fine way for one turtle to make a drawing. But I wanted to try something different. I chose to play with having all four turtles cooperate in drawing the house. They became the JackBuilt Construction Company.

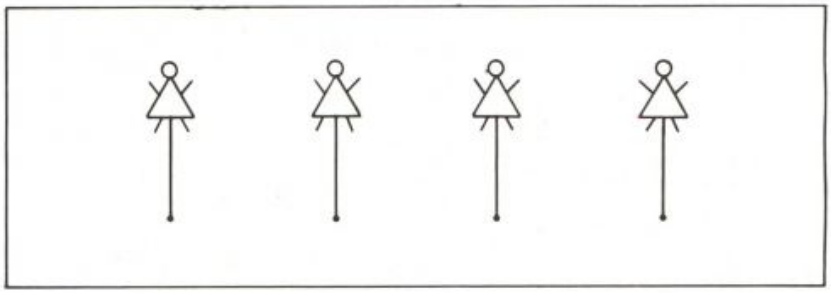
The tricky part of this project was making all the turtles draw different parts of the house at the same time. Here are a series of examples that illustrate the problem and how I decided to solve it.

In this illustration, all of the turtles are visible and placed apart from one other.



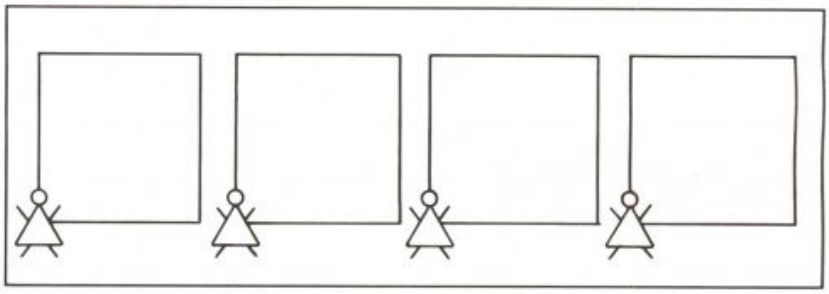
First, let's give the turtles something to do all at once.

```
TELL [0 1 2 3] FD 50
```



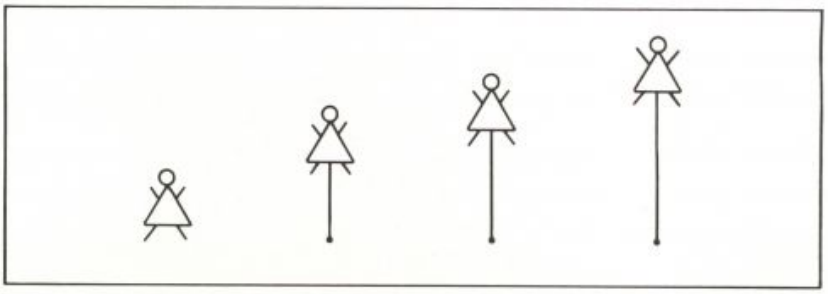
Next, the turtles can simultaneously draw the same figure.

```
SQUARE
```



Now let's have the turtles each do something different.

```
TELL [0 1 2 3] EACH [FD WHO * 20]
```



This command is very different. The turtles are moving one after another in very rapid succession. Yet as long as the instruction given to each turtle is not too long, they will appear to be moving all at once.

This is the key to the project.

The turtles can appear to be drawing a house simultaneously if each instruction to the turtles is pretty short. The procedure needs to explicitly tell each of them to draw each single line. The way I thought about this was to give each turtle at the job site its own work to do. One turtle laid the foundation, another put up walls, another built roofs, and the last made doors and windows. Each turtle had a list of things to do. When each had finished the first item on its list, they all went on to the next item.

```
MAKE "CARPENTER0 [[SETPOS [30 20] SETH 90 PD]
                          [FD 120 SETY 21]
                 [BK 120 SETY 22] [FD 120 SETY 23]
                 [BK 120 SETY 24] [FD 120 PU]
                 [SETPOS [70 85] SETH 90 PD FD 80 PU HOME]]
MAKE "CARPENTER1 [[SETPOS [150 25] PD SETH 0] [FD 60]
                 [LT 60 FD 46 PU]
                 [SETPOS [135 60] SETH 270 PD] [WIN PU]
                 [SETPOS [140 30] PD DOOR] [PU HOME]]
MAKE "CARPENTER2 [[SETPOS [30 25] SETH 0 PD] [FD 40]
                 [RT 30 FD 40] [RT 120 FD 40]
                 [RT 120 FD 40 PU]
                 [SETPOS [35 25] SETH 0 PD]
                 [REPEAT 2 [FD 35 RT 90 FD 30 RT 90]
                 PU HOME]]
MAKE "CARPENTER3 [[SETPOS [70 25] SETH 0 PD] [FD 60]
                 [RT 60 FD 46 PU]
                 [SETPOS [85 80] SETH 90 PD] [WIN PU]
                 [SETPOS [85 55] PD BAY] [PU HOME]]
```

It is important that each of these lists have the same number of items. Also, these lists call three short procedures to draw the door and windows. Here they are.

```
TO DOOR
REPEAT 2 [FD 30 RT 90 FD 15 RT 90]
END

TO WIN
REPEAT 4 [FD 20 RT 90]
END

TO BAY
REPEAT 2 [FD 20 RT 90 FD 30 RT 90]
END
```

Here's a procedure that can use the CARPENTER variables to draw a house.

```
TO BUILD.HOUSE
TELL [0 1 2 3] PU
BUILD.HOUSE1 :CARPENTER0 :CARPENTER1 :CARPENTER2 :CARPENTER3
END
```

```
BUILD.HOUSE uses BUILD.HOUSE1.

TO BUILD.HOUSE1 :LIST0 :LIST1 :LIST2 :LIST3
IF EMPTYP :LIST0 [STOP]
EACH [RUN FIRST THING WORD "LIST WHO TOOT 0 240 15 2]
BUILD.HOUSE1 BF :LIST0 BF :LIST1 BF :LIST2 BF :LIST3
END
```

Getting to the Job Site

Here is the second project. It's about carpenters and their truck.

It seemed important that the turtles look like carpenters and that they arrive at the job site in some believable way. I decided that driving up in a truck was the right thing to do. Here are the shapes for the carpenters and the truck. The listings for these shapes are provided at the end of this section.

As you can see, I decided to use only two shapes for the truck.

It didn't take long for me to discover that I had a bit of a problem. The turtles that carried the truck shapes had to become carpenters as soon as the truck arrived at the job site! This meant that the truck would disappear. I decided to solve this problem by having the carpenters drive up *as* the truck, rather than *in* the truck. When the truck arrived, it would be transformed into the carpenters. When the house was finished, the carpenters would transform back into the truck.

Here are the shapes I made for the transformation.

And here's what that transformation looked like on the screen.

The interesting part of this project was making the shapes and making the animation work effectively.

Here are the procedures.

```
TO SETUP
PUTSH 1 :SHAPE1
PUTSH 2 :SHAPE2
PUTSH 3 :SHAPE3
PUTSH 4 :SHAPE4
PUTSH 5 :SHAPE5
PUTSH 6 :SHAPE6
PUTSH 7 :SHAPE7
MAKE "CARPENTER 1
MAKE "REAR 2
MAKE "FRONT 3
END

TO ENTER.TRUCK
TELL [0 1]
PU SETPOS [-155 20] SETH 90
ASK 0 [SETSH :REAR]
ASK 1 [SETSH :FRONT FD 15]
ST
REPEAT 80 [FD 2 TOOT 0 20 15 2]
END
```

Remember, `TELL [0 1]` is still in effect.

```
TO TRANSFORM
ASK 2 [SETPOS ASK 0 [POS]]
ASK 3 [SETPOS ASK 1 [POS]]
EACH [SETSH SUM WHO 5]
ASK [2 3] [SETSH 3 ST WAIT 30 SETSH 4]
ASK 0 [BK 15]
ASK 1 [FD 15]
RT 90 FD 8
SETSH 3 WAIT 30 SETSH 4
END
```

Now the carpenters are ready to `BUILD.HOUSE`. When they have finished the job, they turn back into their truck and go home.

```
TO TRANSFORM.BACK
SETH 90
EACH [SETPOS LIST SUM 12 WHO * 30 15]
ASK 2 [SETPOS LIST SUM 8 ASK 0 [XCOR] 15]
ASK 3 [SETPOS LIST SUM -8 ASK 1 [XCOR] 15]
EACH [SETSH SUM WHO 5]
ASK 0 [FD 8]
ASK 1 [BK 8]
ASK [2 3] [WAIT 30 SETSH 4 WAIT 30 SETSH 3 WAIT 30 HT]
EACH [SETSH SUM 1 WHO]
END

TO EXIT
REPEAT 62 [FD 2 TOOT 0 20 15 2]
ASK 1 [HT]
REPEAT 8 [FD 2 TOOT 0 20 15 2]
HT
END
```

Here are the listings for the shapes. They are listed in the same order as they appeared in this section.

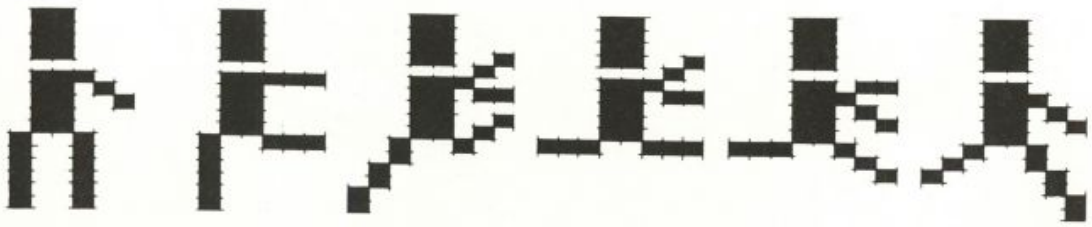
```
MAKE "SHAPE1 [24 24 24 24 0 28 26 26 26
                24 36 36 68 68 132 132]
MAKE "SHAPE2 [0 0 0 0 0 0 255 255
              255 255 255 255 255 255 96 96]
MAKE "SHAPE3 [0 0 0 0 0 0 224 224
              239 239 255 255 255 255 102 102]
MAKE "SHAPE4 [0 0 0 0 0 0 0 0 0 0 0 255 255 255 96 96]
MAKE "SHAPE5 [0 0 0 0 0 0 0 0 0 0 0 239 255 255 102 102]
MAKE "SHAPE6 [0 0 0 0 32 16 139 135 255 255 0 0 0 0 0 0]
MAKE "SHAPE7 [35 19 11 7 6 142 254 12 136 240 0 0 0 0 0 0]
```

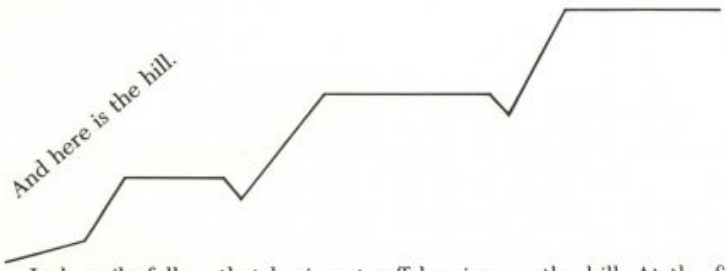
Jack and Jill

This is the third project. It's about animating figures.

I wanted to animate Jack leaping up a hill, meeting Jill, and then Jack and Jill leaping down the hill. I decided to forget about *"Jack fell down and broke his crown/And Jill came tumbling after."* Getting them to leap without falling down was enough to do. I decided, however, that Jack should be a bit clumsy. He would always bump into things before leaping over them.

Here are the leaping shapes. The listings appear in order at the end of this section.





Jack, agile fellow that he is, sets off leaping up the hill. At the first opportunity, he bumps into the rise in the hill. He is saved from a fall by his guardian demon. Every time Jack stubs his toe, this demon stops him and gets him to back up a bit. Here's the demon.

```
WHEN OVER 0 0 [SETSP 0 BK 5]
```

Now let's look at the procedures that bring this animation to life. JACK.AND.JILL is the main procedure.

```
TO JACK.AND.JILL
TELL 0 HT
ASK [1 2 3] [HT]
DRAW.HILL
SETUP
UP.THE.HILL
DOWN.THE.HILL
END

TO DRAW.HILL
PU SETPOS [-155 -50] SETH 90 PD
REPEAT 3 [FD 40 LT 60 FD 30 RT 120 FD 5 SETH 90]
FD 40 PU
END
```

SETUP activates the demon and places Jack and Jill.

```
TO SETUP
PUTSH 1 :SHAPE1
PUTSH 2 :SHAPE2
PUTSH 3 :SHAPE3
PUTSH 4 :SHAPE4
PUTSH 5 :SHAPE5
PUTSH 6 :SHAPE6
PUTSH 7 :SHAPE7
MAKE "JACK 0
MAKE "JILL 1
MAKE "LEAP.UP [1 2 3 4 5 6]
MAKE "LEAP.DN [6 5 4 3 2 1]
TELL :JACK PU SETPOS [-150 -38] SETH 90 SETSH 1 SETC 7 ST
ASK :JILL [PU SETPOS [52 28] SETH 270 SETSH 1 SETC 7 ST]
WHEN OVER 0 0 [SETSP 0 BK 5 MAKE "FREQ 200]
END
```

UP.THE.HILL animates Jack's ascent.

```
TO UP.THE.HILL
SETSP 10
ANIMATE :LEAP.UP
SETSP 0
END
```

ANIMATE stops only when Jack has reached the top of the hill. Until then, it repeatedly invokes ANIMATE1.

```
TO ANIMATE :SHAPES
IF OR XCOR > 50 XCOR < -155 [STOP]
ANIMATE1 :SHAPES
ANIMATE :SHAPES
END
```

ANIMATE1 does all the work.

```
TO ANIMATE1 :SHAPES
IF EMPTYP :SHAPES [STOP]
IF SPEED = 0 [PICK.JUMP]
SETSH FIRST :SHAPES
TOOT 0 440 10 1
ANIMATE1 BF :SHAPES
END
```

Here's the procedure that is invoked when the collision demon has stopped Jack.

```
TO PICK.JUMP
IF WHO = 0 [JUMP.UP] [JUMP.DOWN]
END
```

JUMP.UP is invoked to get Jack up the hill.

```
TO JUMP.UP
SETSH 3 SETH 30
REPEAT 16 [TOOT 1 :FREQ 15 2 FD 2 MAKE "FREQ :FREQ + 200]
SETSH 4 SETH 90
REPEAT 8 [FD 2]
SETSH 1 SETH 150
REPEAT 6 [TOOT 0 :FREQ 15 2 FD 1 MAKE "FREQ :FREQ - 400]
SETH 90 SETSP 10
END
```

JUMP.DOWN is invoked to get Jack and Jill down the hill.

```
TO JUMP.DOWN
WAIT 5
SETH 330 SETSH 3
REPEAT 7 [TOOT 0 :FREQ 15 2 FD 1 MAKE "FREQ :FREQ + 600]
SETH 270 SETSH 4
REPEAT 15 [FD 2]
SETH 210 SETSH 1
REPEAT 14 [TOOT 1 :FREQ 15 2 FD 2 MAKE "FREQ :FREQ - 200]
SETH 270 SETSP 10
END
```

Now Jack and Jill can leap down the hill.

```
TO DOWN.THE.HILL
TELL LIST :JACK :JILL
ASK :JACK [SETPOS [43 28]]
SETH 270 SETSP 10
ANIMATE :LEAP.DN
SETSP 0
END
```

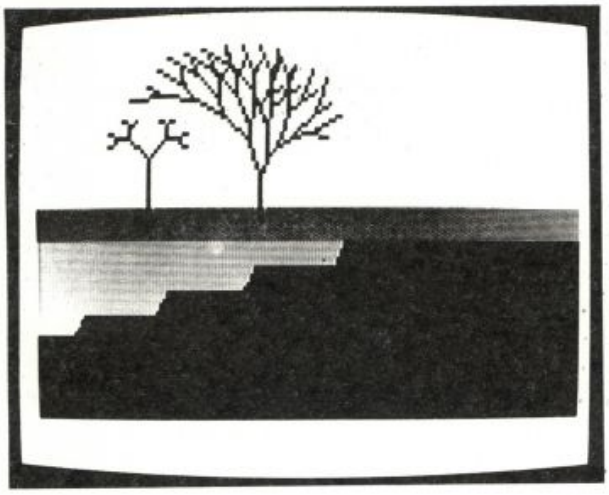
Here are the listings for the shapes.

```
MAKE "SHAPE1 [24 24 24 24 0 28
              26 25 24 24 36 36 36 36 36 36]
MAKE "SHAPE2 [24 24 24 24 0 31
              24 24 24 24 39 32 32 32 32 32]
MAKE "SHAPE3 [24 24 24 25 2 28
              27 24 25 26 36 32 64 64 128 128]
MAKE "SHAPE4 [24 24 24 25 2 28 27 24 24 24 231 0 0 0 0 0]
MAKE "SHAPE5 [24 24 24 24 0 27 28 26 25 24 228 2 1 0 0 0]
MAKE "SHAPE6 [24 24 24 24 0 24 28 26 25 24 36 68 130 2 1 1]
```

Setting the Scene

This is the fourth project. It's about drawing scenery and choosing colors that complement one another.

I wanted to learn about how to write programs for scenery. So I drew lots of different scenes and I learned some very useful techniques that make drawing faster and easier. Here is the scenery I use in Jack and Jill.



SET.SCENE is the main procedure.

```
TO SET.SCENE
SETUP
GRASS
ROAD
HILL
TREES
END
```

SETUP is the setup procedure.

```
TO SETUP
SETBG 80
SETPC 0 7
SETPC 1 4
SETPC 2 61
FS
TELL 0 HT
END
```

In this scene I wanted to fill in a large area of the screen with a single color. This can be a slow process. Here is a technique for overcoming this difficulty.

```
TO GRASS
SETPN 0 SETH 90
PU SETPOS [-160 -105] PD
FD 320
SETH 89.6
REPEAT 2 [FD 7700]
SETH 90
FD 320
PU
END
```

GRASS assumes that the screen is in WRAP. It begins at the bottom of the screen and draws a single line across the screen. Then the interesting part happens. The turtle is turned 0.4 degrees and is instructed to go forward many steps. Try this out.

Now let's look at the rest of SETSCENE1.

```
TO SET.SCENE1
SETUP
GRASS
ROAD
HILL
TREES
END
```

ROAD works in exactly the same way as GRASS.

```
TO ROAD
SETPN 1 SETH 270
PU SETPOS [160 3] PD
FD 320
SETH 270.4
FD 2750
SETH 270
FD 320
PU
END
```

HILL invokes HILL1 to draw the several levels of the hill.

```
TO HILL
SETPN 2 SETH 0
PU SETPOS [-158 -54] PD
HILL1 0 -135
HILL1 0 -85
HILL1 0 -35
HILL1 0 15
PU
END
```

This particular hill was made with fearless Jack in mind.

```
TO HILL1 :A :X
IF :A = 7 [STOP]
SETX :X + :A
SETY SUM YCOR 1
SETX -158
SETY SUM YCOR 1
HILL1 :A + 1 :X
END
```

I decided to add trees to this scene, and I remembered the cover illustration from Harold Abelson and Andrea DiSessa's *Turtle Geometry* (Cambridge, Mass: MIT Press, 1981).

```
TO TREES
SETPN 0 SETH 0
PU SETPOS [-90 22] PD
BRANCH 30 5
PU SETPOS [-25 22] PD
LBRANCH 10 20 5
PU
END
```

Here are the four procedures from *Turtle Geometry*.

```
TO RBRANCH :LENGTH :ANGLE :NUM
FD :LENGTH
NODE :LENGTH :ANGLE :NUM
BK :LENGTH
END

TO LBRANCH :LENGTH :ANGLE :NUM
FD 2 * :LENGTH
NODE :LENGTH :ANGLE :NUM
BK 2 * :LENGTH
END

TO NODE :LEN :ANG :TIMES
IF :TIMES = 0 [STOP]
LT :ANGLE
LBRANCH :LEN :ANG :TIMES - 1
RT 2 * :ANGLE
RBRANCH :LEN :ANG :TIMES - 1
LT :ANGLE
END
```

```
TO BRANCH :LENGTH :TIMES
IF :TIMES = 0 [STOP]
FD :LENGTH
LT 45
BRANCH :LENGTH / 2 :TIMES - 1
RT 90
BRANCH :LENGTH / 2 :TIMES - 1
LT 45
BK :LENGTH
END
```

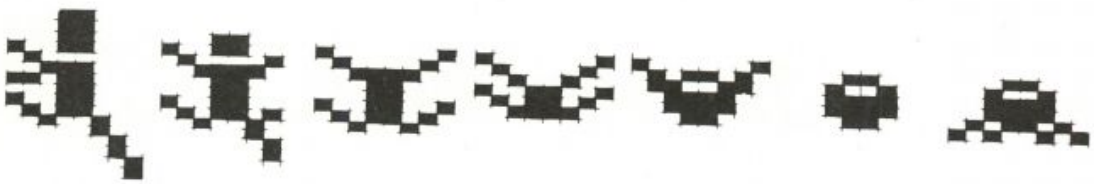
The Whole Ball of Wax

I realized, while I was making up a diskette with all of my projects on it, that I had all of the pieces I needed to make a long animated cartoon. Jack and Jill could become an opus.

Here's the storyline. The JackBuilt Construction Company drives down a tree-lined road that runs along the crest of a terraced hillside. Their truck stops at a house site and the carpenters build a home for Jack and Jill. When the house is built and the carpenters are gone, Jill appears in the doorway and waits for Jack to leap up the hill. Jack sees Jill waiting but rings the doorbell anyway, and Jill comes out. Together they leap off down the road, magically transforming themselves into birds as they leap. The two beautiful birds fly off through the trees, and the story ends.

Animating the birds was the finishing touch to this story. I love magic, and I feel that all children's stories should have some magic in them. I decided to transform Jack and Jill into birds in midleap and have them fly away. This transformation delights your eye and surprises your expectations for the story. It's magic.

Here are the shapes I designed for this transformation sequence. Jack and Jill are traveling from right to left across the screen, so this sequence is also displayed here from right to left. The listings appear in order at the end of this section.



The transformation begins with both figures carrying a leaping shape. `BECOME.BIRDS` accomplishes the actual transformation.

```
TO BECOME.BIRDS
SETSH 1
SETC 46
WAIT 3
PUTSH 1 :BECOME.B1
```

```
SETH 300
WAIT 3
PUTSH 1 :BECOME.B2
SETSP 20
WAIT 3
PUTSH 1 :BECOME.B3
WAIT 3
PUTSH 1 :BECOME.B4
END
```

Notice that throughout the transformation the figures are carrying shape 1. I decided to use PUTSH to change shape 1 several times.

FLY is the final procedure in the story. It animates the birds and synchronizes the sounds they make as they fly away.

```
TO FLY
EACH [IF XCOR < -150 [STOP]]
PUTSH 1 :FLY1
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
PUTSH 1 :FLY2
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
PUTSH 1 :FLY3
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
FLY
END
```

Here are the shapes.

```
MAKE "BECOME.B1 [24 24 24 152 64 56 216 24
                 152 88 36 4 2 2 1 1]
MAKE "BECOME.B2 [0 0 24 152 66 60 24 24 152 90 36 4 2 2 0 0]
MAKE "BECOME.B3 [0 0 0 129 66 60 24 24 153 90 36 0 0 0 0 0]
MAKE "BECOME.B4 [0 0 0 129 66 36 153 90 60 0 0 0 0 0 0 0]
MAKE "FLY1 [0 0 0 129 90 102 60 60 24 0 0 0 0 0 0 0]
MAKE "FLY2 [0 0 0 0 24 36 60 60 24 0 0 0 0 0 0 0]
MAKE "FLY3 [0 0 0 0 24 36 60 60 90 165 0 0 0 0 0 0]
```

PROGRAM LISTING

This program is divided into five files. You only need to load the first file D:JACK; the others are loaded by the program at the appropriate times.

D:JACK

The main procedure is JACK.

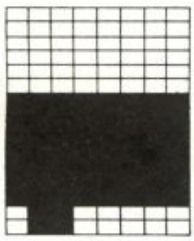
```
TO JACK
SETUP
SET.SCENE
ERALL
LOAD "D:JACK1
CT SS
PR [TYPE "C" TO CONTINUE]
END
```

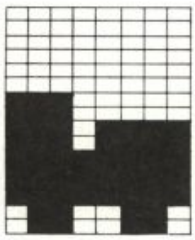
```
TO SETUP
LOAD.SHAPES 1
SETBG 80
SETPC 0 7
SETPC 1 4
SETPC 2 61
SET.DRIVE
FS
ASK [0 1 2 3] [PU HT SETC 39]
END
```

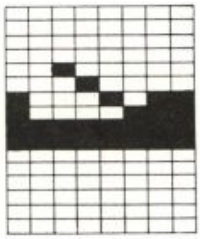
```
TO LOAD.SHAPES :NUM
IF :NUM > 15 [STOP]
PUTSH :NUM THING WORD "SHAPE :NUM
ERN WORD "SHAPE :NUM
LOAD.SHAPES :NUM + 1
END

TO SET.DRIVE
TS CT
SETCURSOR [4 10]
PR [INSERT DEMO DISK IN DRIVE #1,]
SETCURSOR [10 11]
PR [THEN PRESS RETURN]
IF NOT RC = CHAR 155 [SET.DRIVE]
END

TO SET.SCENE
TELL 0
GRASS
ROAD
HILL
TREES
END

TO GRASS
SETPN 0 SETH 90
PU SETPOS [-160 -105] PD
FD 320
SETH 89.6
REPEAT 2 [FD 7700]
SETH 90
FD 320
PU
END

TO ROAD
SETPN 1 SETH 270
PU SETPOS [160 3] PD
FD 320

SETH 270.4
FD 2750
SETH 270
FD 320
PU
END

TO HILL
SETPN 2 SETH 0
PU SETPOS [-158 -54] PD
HILL1 0 -135
HILL1 0 -85
HILL1 0 -35
HILL1 0 15
PU
END

TO HILL1 :A :X
IF :A = 7 [STOP]
SETX :X + :A
SETY SUM YCOR 1
SETX -158
SETY SUM YCOR 1
HILL1 :A + 1 :X
END

TO TREES
SETPN 0 SETH 0
PU SETPOS [-90 22] PD
BRANCH 30 5
PU SETPOS [-25 22] PD
LBRANCH 10 20 5
PU
END

TO RBRANCH :LENGTH :ANGLE :NUM
FD :LENGTH
NODE :LENGTH :ANGLE :NUM
BK :LENGTH
END

TO LBRANCH :LENGTH :ANGLE :NUM
FD 2 * :LENGTH
NODE :LENGTH :ANGLE :NUM
BK 2 * :LENGTH
END

TO NODE :LEN :ANG :TIMES
IF :TIMES = 0 [STOP]
LT :ANGLE
LBRANCH :LEN :ANG :TIMES - 1
RT 2 * :ANGLE
RBRANCH :LEN :ANG :TIMES - 1
LT :ANGLE
END

TO BRANCH :LENGTH :TIMES
IF :TIMES = 0 [STOP]
FD :LENGTH
LT 45
BRANCH :LENGTH / 2 :TIMES - 1
RT 90
BRANCH :LENGTH / 2 :TIMES - 1
LT 45
BK :LENGTH
END

MAKE "SHAPE1 [0 0 0 0 0 0 255 255 255 ▸
     255 255 255 255 255 96 96]
MAKE "SHAPE2 [0 0 0 0 0 0 224 224 239 ▸
     239 255 255 255 255 102 102]
```

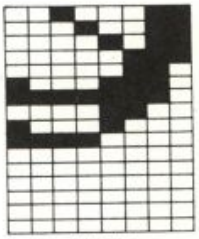
```
MAKE "SHAPE3 [0 0 0 0 32 16 139 135 ▶
     255 255 0 0 0 0 0 0]
MAKE "SHAPE4 [35 19 11 7 6 142 254 12 ▶
     136 240 0 0 0 0 0 0]
MAKE "SHAPE5 [0 0 0 0 0 0 0 0 0 0 0 ▶
     255 255 255 96 96]
MAKE "SHAPE6 [0 0 0 0 0 0 0 0 0 0 0 ▶
     239 255 255 102 102]
MAKE "SHAPE7 [24 24 24 24 0 28 26 26 ▶
     26 24 36 36 68 68 132 132]
MAKE "SHAPE8 [24 24 24 24 0 56 88 152 ▶
     24 24 36 36 36 36 36 36]
MAKE "SHAPE9 [24 24 24 24 0 248 24 24 ▶
     24 24 228 4 4 4 4 4]
```

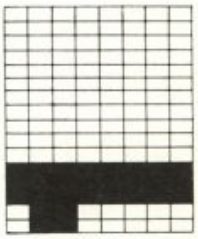
```
MAKE "SHAPE10 [24 24 24 152 64 56 216 ▶
     24 152 88 36 4 2 2 1 1]
MAKE "SHAPE11 [24 24 24 152 64 56 216 ▶
     24 24 24 231 0 0 0 0 0]
MAKE "SHAPE12 [24 24 24 24 0 216 56 88 ▶
     152 24 39 64 128 0 0 0]
MAKE "SHAPE13 [24 24 24 0 36 24 24 24 ▶
     24 36 36 66 66 129 129 129]
MAKE "SHAPE14 [24 24 24 24 0 24 56 88 ▶
     152 24 36 34 65 64 128 128]
MAKE "SHAPE15 [24 24 153 90 36 24 24 ▶
     24 24 24 36 36 66 66 129 129]
```

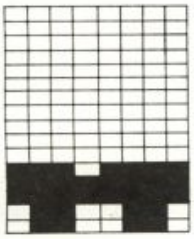

:SHAPE1

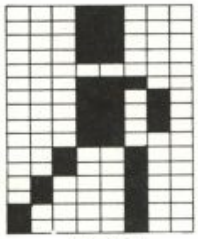

:SHAPE2

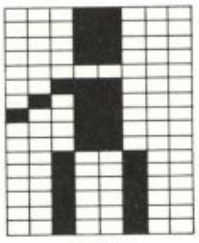

:SHAPE3

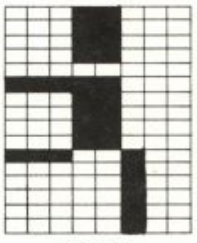

:SHAPE4

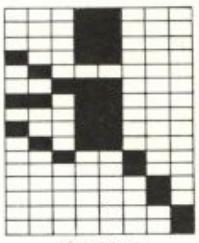

:SHAPE5


:SHAPE6

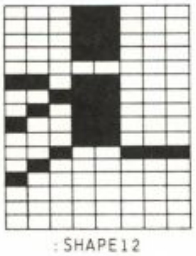

:SHAPE7

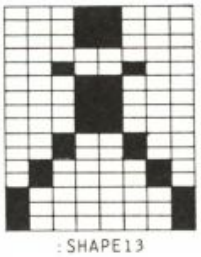

:SHAPE8


:SHAPE9

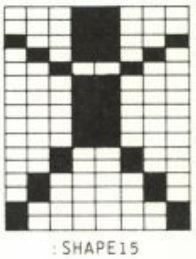

:SHAPE10


:SHAPE11


:SHAPE12


:SHAPE13


:SHAPE14


:SHAPE15

```
D:JACK1

TO C
ER "C
FS
LOAD "D:JACK2
ASK [0 1 2 3] [PU HT]
TELL [0 1]
ENTER.TRUCK
RECYCLE
TRANSFORM
FIRST.CLEANUP
TELL [0 1 2 3]
BUILD.HOUSE
TELL [0 1]
REFORM.AND.EXIT.TRUCK
SECOND.CLEANUP
JACK.AND.JILL
AFTERMATH
END

TO TRANSFORM
ASK 2 [SETPOS ASK 0 [POS]]
ASK 3 [SETPOS ASK 1 [POS]]
EACH [SETSH SUM WHO 5]
ASK [2 3] [SETSH 3 ST WAIT 30 SETSH 4]
ASK 0 [BK 15]
ASK 1 [FD 15]
RT 90 FD 8
ASK [0 1] [SETSH 3 WAIT 30 SETSH 4]
END

TO ENTER.TRUCK
SETH 90
SETPOS [-155 20]
ASK 1 [FD 15]
EACH [SETSH WHO + 1]
ST
REPEAT 80 [FD 2 TOOT 0 20 15 2]
END

TO FIRST.CLEANUP
ER "FIRST.CLEANUP
ER [ENTER.TRUCK TRANSFORM]
RECYCLE
END

TO REFORM.AND.EXIT.TRUCK
SETH 90
EACH [SETPOS LIST SUM 12 WHO * 30 15]
ASK 2 [SETPOS LIST SUM 8 ASK 0 [XCOR] ▸
     15]
ASK 3 [SETPOS LIST SUM -8 ASK 1 [XCOR] ▸
     15]
EACH [SETSH SUM WHO 5]
ASK 0 [FD 8]
ASK 1 [BK 8]
ASK [2 3] [WAIT 30 SETSH 4 WAIT 30 ▸
     SETSH 3 WAIT 30 HT]
EACH [SETSH SUM 1 WHO]
RECYCLE
REPEAT 62 [FD 2 TOOT 0 20 15 2]
ASK 1 [HT]
REPEAT 8 [FD 2 TOOT 0 20 15 2]
HT
END

TO BUILD.HOUSE
SETSH 14
SETPN 2
BUILD :CARP0 :CARP1 :CARP2 :CARP3
BUILD :CARP10 :CARP11 :CARP12 :CARP13
PU
WAIT 30
END

TO SECOND.CLEANUP
ER "SECOND.CLEANUP
ACTIVATE [1 2 3 4 5 6] 65
ERN [CARP0 CARP1 CARP2 CARP3 CARP10 ▸
     CARP11 CARP12 CARP13]
ER [BUILD.HOUSE BUILD DOOR BAY WIN ▸
     REFORM.AND.EXIT.TRUCK ACTIVATE]
RECYCLE
END

TO JACK.AND.JILL
ER "JACK.AND.JILL
SETUP.JJ
UP.THE.HILL
DOOR.BELL
TELL [0 1]
DOWN.THE.HILL?
BECOME.BIRDS
FLY
ASK [0 1] [HT SETSP 0]
END

TO AFTERMATH
TELL 0 ASK [0 1 2 3] [SETPOS [0 0]]
ERALL
LOAD "D:JACK3
LOAD "D:JACK4
CT SS
PR [TYPE "RERUN" TO SEE THIS DEMO ▸
     AGAIN]
PR [TYPE "RESET" TO STOP THIS DEMO]
END
```

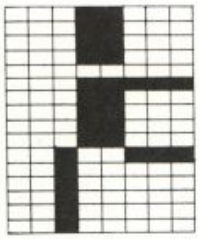
```
TO BUILD :N0 :N1 :N2 :N3
IF EMPTYP :N0 [STOP]
EACH [RUN FIRST ( THING WORD "N WHO ) ►
     TOOT 0 240 15 2]
BUILD BF :N0 BF :N1 BF :N2 BF :N3
END

TO ACTIVATE :LIST :N1
IF EMPTYP :LIST [STOP]
PUTSH FIRST :LIST THING CHAR :N1
ERN WORD " CHAR :N1
ACTIVATE BF :LIST SUM 1 :N1
END

TO DOOR
REPEAT 2 [FD 25 RT 90 FD 20 RT 90]
END

TO BAY
REPEAT 2 [FD 15 RT 90 FD 30 RT 90]
END

TO WIN
REPEAT 2 [FD 15 RT 90 FD 20 RT 90]
END

TO SETUP.JJ
ER "SETUP.JJ
WHEN 0 [JUMP.OVER.RT]
MAKE "NUM 1
TELL 0
SETPOS [-155 -38]
SETH 90 SETSH 1 SETC 7
ASK 1 [SETPOS [92 35] SETH 270 SETSH 8 ►
     SETC 7 ST]
ASK 2 [SETPOS [127 65]]
ASK 3 [SETPOS [127 35]]
ST
RECYCLE
END

TO UP.THE.HILL
SETSP 10
ANIMATE 1 7
SETY 22
END

TO DOOR.BELL
ER "DOOR.BELL
TOOT 0 440 5 15 TOOT 1 660 5 15
TOOT 0 660 5 15 TOOT 1 880 5 15
TOOT 0 440 5 15 TOOT 1 660 5 15
TOOT 0 880 5 60 TOOT 1 1320 5 60
ER [UP.THE.HILL JUMP.OVER.RT]
WHEN 0 []
RECYCLE
END

TO DOWN.THE.HILL?
PUTSH 1 GETSH 9
SETY 10 SETH 280
SETSP 10
ANIMATE 8 14
END

TO BECOME.BIRDS
SETC 46
SETSH 1
WAIT 5
PUTSH 1 :AA
SETH 300
WAIT 3
PUTSH 1 :BB
SETSP 20
WAIT 3
PUTSH 1 :CC
WAIT 5
PUTSH 1 :DD
END

TO FLY
EACH [IF XCOR < -150 [STOP]]
PUTSH 1 :FF
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
PUTSH 1 :EE
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
PUTSH 1 :GG
EACH [TOOT WHO SUM WHO 1 * 1760 10 3]
FLY
END

TO JUMP.OVER.RT
SETSP 0
BK 5
SETSH 3 SETH 20
RAMP 200 3000 200
SETSH 4 SETH 90
REPEAT 7 [FD 2]
SETSH 15 SETH 150
RAMP 3000 400 400
SETH 90
SETSP 10
END

TO ANIMATE :FIRST :LAST
IF XCOR > 90 [SETSP 0 SETH 270 SETX 80 ►
     STOP]
IF YCOR > 25 [STOP]
```

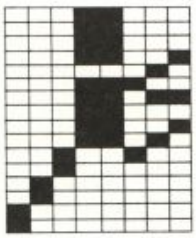
```
IF :FIRST > :LAST [MAKE "FIRST :LAST - ►
     6]
SETSH :FIRST
TOOT 0 440 10 1
ANIMATE :FIRST + 1 :LAST
END

TO RAMP :S :F :R
IF :F < :S [REPEAT ( :S - :F ) / :R ►
     [TOOT 1 :S 15 2 FD 2 MAKE "S :S - ►
     :R]]
IF :S < :F [REPEAT ( :F - :S ) / :R ►
     [TOOT 1 :S 15 2 FD 2 MAKE "S :S + ►
     :R]]
END

TO REFORM
SETH 90
EACH [SETPOS LIST SUM 12 WHO * 30 15]
ASK 2 [SETPOS LIST SUM 8 ASK 0 [XCOR] ►
     15]
ASK 3 [SETPOS LIST SUM -8 ASK 1 [XCOR] ►
     15]
EACH [SETSH SUM WHO 5]
ASK 0 [FD 8]
ASK 1 [BK 8]
ASK [2 3] [WAIT 30 SETSH 4 WAIT 30 ►
     SETSH 3 WAIT 30 HT]
EACH [SETSH SUM 1 WHO]
REPEAT 80 [FD 2 TOOT 0 20 15 2]
END

MAKE "NUM 1
MAKE "DD [0 0 0 129 66 36 153 90 60 0 ►
     0 0 0 0 0 0]
MAKE "CC [0 0 0 129 66 60 24 24 153 90 ►
     36 0 0 0 0 0]
MAKE "BB [0 0 24 152 66 60 24 24 152 ►
     90 36 4 2 2 0 0]
MAKE "AA [24 24 24 152 64 56 216 24 ►
     152 88 36 4 2 2 1 1]
MAKE "GG [0 0 0 0 24 36 60 60 90 165 0 ►
     0 0 0 0 0]
MAKE "EE [0 0 0 129 90 102 60 60 24 0 ►
     0 0 0 0 0 0]
MAKE "FF [0 0 0 0 24 36 60 60 24 0 0 0 ►
     0 0 0 0]
```

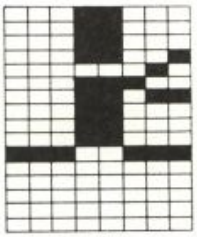
D:JACK2

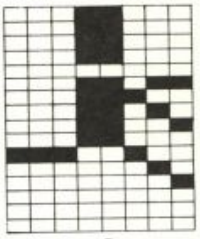
```
MAKE "A [24 24 24 24 0 28 26 25 24 24 ►
     36 36 36 36 36 36]
MAKE "B [24 24 24 24 0 31 24 24 24 24 ►
     39 32 32 32 32 32]
MAKE "C [24 24 24 25 2 28 27 24 25 26 ►
     36 32 64 64 128 128]
MAKE "D [24 24 24 25 2 28 27 24 24 24 ►
     231 0 0 0 0 0]
MAKE "E [24 24 24 24 0 27 28 26 25 24 ►
     228 2 1 0 0 0]
MAKE "F [24 24 24 24 0 24 28 26 25 24 ►
     36 68 130 2 1 1]

MAKE "AA [24 24 24 152 64 56 216 24 ►
     152 88 36 4 2 2 1 1]
MAKE "BB [0 0 24 152 66 60 24 24 152 ►
     90 36 4 2 2 0 0]
MAKE "CC [0 0 0 129 66 60 24 24 153 90 ►
     36 0 0 0 0 0]
MAKE "DD [0 0 0 129 66 36 153 90 60 0 ►
     0 0 0 0 0 0]
MAKE "EE [0 0 0 129 90 102 60 60 24 0 ►
     0 0 0 0 0 0]
MAKE "FF [0 0 0 0 24 36 60 60 24 0 0 0 ►
     0 0 0 0]
MAKE "GG [0 0 0 0 24 36 60 60 90 165 0 ►
     0 0 0 0 0]
MAKE "CARP0 [[SETPOS [30 25] PD SETH ►
     90] [FD 120] [SETY 24 RT 180 FD ►
     120] [SETY 23 RT 180 FD 120]]
MAKE "CARP1 [[SETPOS [150 25] PD SETH ►
     0] [FD 60] [LT 60 FD 46 PU] ►
     [SETPOS [135 60] SETH 270 PD]]
MAKE "CARP2 [[SETPOS [30 25] PD SETH ►
     0] [FD 40] [RT 30 FD 40] [RT 120 ►
     FD 40]]
MAKE "CARP3 [[SETPOS [70 25] PD SETH ►
     0] [FD 60] [RT 60 FD 46 PU] ►
     [SETPOS [85 80] SETH 90 PD]]
MAKE "CARP10 [[SETY 22 RT 180 FD 120 ►
     PU] [SETPOS [70 85] SETH 90 PD FD ►
     80]]
MAKE "CARP11 [[WIN PU] [SETPOS [140 ►
     30] PD DOOR]]
MAKE "CARP12 [[RT 120 FD 40 PU] ►
     [SETPOS [35 25] SETH 0 PD REPEAT ►
     2 [FD 35 RT 90 FD 30 RT 90]]]
MAKE "CARP13 [[WIN PU] [SETPOS [85 55] ►
     PD BAY]]
```

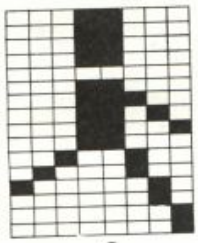

:A

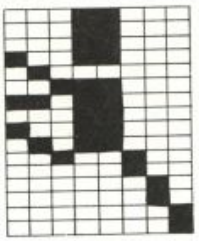

:B

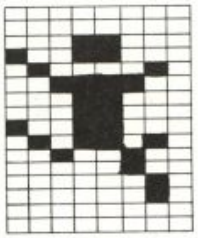

:C

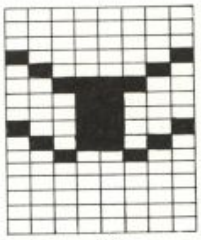

:D

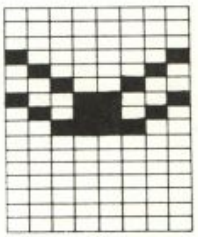

:E

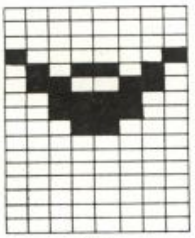

:F

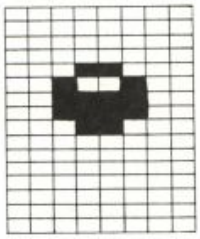

:AA

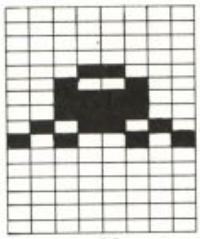

:BB


:CC


:DD


:EE


:FF


:GG

```
D:JACK3

TO RESET
ERALL
WRAP
SETBG 74
ASK [0 1 2 3] [SETC 7 SETSH 0 CS PD]
TELL 0
END

TO RERUN
ER "RERUN
FS
```

```
LOAD "D:JACK1
ACTIVATE [1 2 3 4 5 6] 65
TELL [0 1 2 3]
BUILD.HOUSE
ERNS
TELL 0
ASK [0 1 2 3] [SETSP 0 HT SETC 22 PU ►
     HOME]
RECYCLE
C
END
```

```
MAKE "CARP0 [[SETPOS [30 25] PE SETH ▸
     90] [FD 120] [SETY 24 RT 180 FD ▸
     120] [SETY 23 RT 180 FD 120]]
MAKE "CARP1 [[SETPOS [150 25] PE SETH ▸
     0] [FD 60] [LT 60 FD 46 PU] ▸
     [SETPOS [135 60] SETH 270 PE]]
MAKE "CARP2 [[SETPOS [30 25] PE SETH ▸
     0] [FD 40] [RT 30 FD 40] [RT 120 ▸
     FD 40]]
MAKE "CARP3 [[SETPOS [70 25] PE SETH ▸
     0] [FD 60] [RT 60 FD 46 PU] ▸
     [SETPOS [85 80] SETH 90 PE]]
```

D:JACK4

```
MAKE "CARP10 [[SETY 22 RT 180 FD 120 ▸
     PU] [SETPOS [70 85] SETH 90 PE FD ▸
     80]]
MAKE "CARP11 [[WIN PU] [SETPOS [140 ▸
     30] PE DOOR]]
MAKE "CARP12 [[RT 120 FD 40 PU] ▸
     [SETPOS [35 25] SETH 0 PE REPEAT ▸
     2 [FD 35 RT 90 FD 30 RT 90]]]
MAKE "CARP13 [[WIN PU] [SETPOS [85 55] ▸
     PE BAY]]
MAKE "A [0 0 0 0 0 0 255 255 255 255 ▸
     255 255 255 255 96 96]
MAKE "B [0 0 0 0 0 0 224 224 239 239 ▸
     255 255 255 255 102 102]
MAKE "C [0 0 0 0 32 16 139 135 255 255 ▸
     0 0 0 0 0 0]
MAKE "D [35 19 11 7 6 142 254 12 136 ▸
     240 0 0 0 0 0 0]
MAKE "E [0 0 0 0 0 0 0 0 0 0 0 0 255 255 ▸
     255 96 96]
MAKE "F [0 0 0 0 0 0 0 0 0 0 0 0 239 255 ▸
     255 102 102]
```

The six shapes :A through :F in this file (D:JACK4) are the same as the shapes :SHAPE1 through :SHAPE6 in the file D:JACK.

Rocket

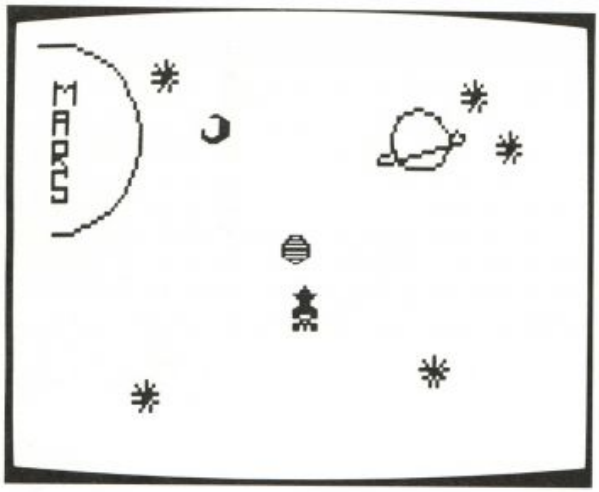
ROCKET creates an outer-space scenario. Imagine that you're standing on Demos, a mythical planet orbiting somewhere in the galaxy. Views of Mars, Saturn, a few stars, and some mountains surround you. A rocketship appears and blasts off with musical fanfare. After traveling halfway up the screen, the ship's crew realizes that it has forgotten the pilot and returns to the original take-off site. The pilot races across the screen and boards the ship. The rocketship takes off again, amid blast-off sounds and color flashes, and heads due north. As the ship travels through space, the view of the mountains disappears and is replaced by a couple of stars. Soon one of the moons orbiting Mars is in view, as well as a mysterious green planet moving west. Eventually the rocketship and green planet collide, and a message appears on the screen that tells you the crew has reached a planet they wish to explore and your adventure is over, and bids you farewell.

Overview

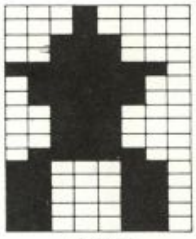
This discussion presents an overview of how the program works and describes the process I went through in designing it. Few of the procedures

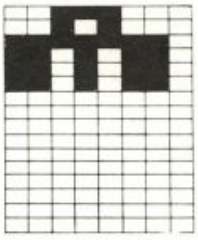
---

By Lauren Young.



are listed within the discussion. They are all listed at the end of this write-up.


:ROCKET


:FIRE

The Rocket

This whole project began with a lone rocketship I had designed, traveling up the screen. Before it took off on a flight through the galaxy, I added fire to it. That is, I designed a shape that I put beneath the rocketship; it looks like fire emerging from the ship. The shapes are in slots 2 and 3.
Turtle 1 carries the shape of the rocketship; turtle 2 carries that of the fire. SET.SHIP sets turtles 1 and 2 near the bottom of the screen, one above the other.

```
TO SET.SHIP
TELL [1 2]
PU
ASK 1 [SETSH 2 SETPOS [0 -90] SETC 7]
ASK 2 [SETSH 3 SETPOS [0 -110] SETC 32]
ST FS
END
```

The Background Scenery

Saturn, Mars, Stars, and Mountains

I wanted something interesting and colorful, but what? I thought about what one might see on a voyage through the galaxy—a planet! That's where I began. When I finished, I had written procedures for two planets, Mars and Saturn, three stars, and some mountains: PLANET.MARS, PLANET.SATURN, STARS, and MOUNTAINS.

I picked Mars because its name is short. This was important because I wanted to draw the planet's name within the outline of the planet. I designed the letters M, A, R, and S and the procedure MARS that puts the letters inside a semicircle on the screen. PLANET.MARS is the procedure that connects these procedures.

I wanted to include a familiar and smaller—seemingly more distant—planet in the upper-right quadrant of the screen. Saturn was an easy choice from among the planets because of its ring.

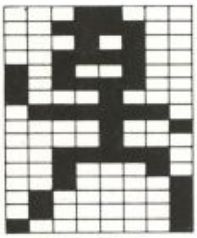
What is a galaxy without stars? The procedure STARS tells turtles 0, 1, and 2 to each draw a star.

I was satisfied with the upper half of the screen, filled with two planets and three stars. The bottom half was still bare and needed a landscape. I thought about mountains and decided to adapt two procedures, MOUNTAINS and SUBMOUNTAIN, from the *Atari Logo Reference Manual.**

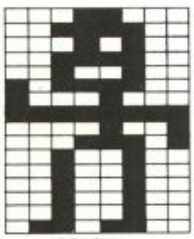
Putting It Together

BACKGROUND.SCENE sets up the entire scene.

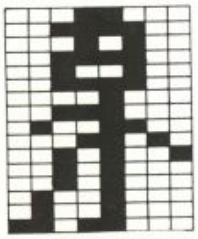
```
TO BACKGROUND.SCENE
TELL 0
SETBG 72
PLANET.MARS
STARS
PLANET.SATURN
MOUNTAINS
END
```



:PERSON.1



:PERSON.2



:PERSON.3

Ready, Set, Action!

Now, for some more action! I wanted to try animating a person walking across the bottom of the screen. Susan Cotten's Exercise project gave me some ideas, and I started experimenting. I designed three shapes (PERSON.1, PERSON.2, and PERSON.3), each with the same head and torso, but with arms and legs in different stances. As in Exercise, I also wanted footprint sounds with each step the person took. The pilot appears to run by having turtle 0 rapidly change its shape to PERSON.2, PERSON.3, and PERSON.1.

Launching the Rocketship

BLAST.OFF sets the rocketship in motion. I put in a few sound effects to jazz up the takeoff. About midscreen, the rocketship stops and a message from the ship's crew is printed:

```
WAIT!!!  WE FORGOT THE PILOT.
```

The rocketship reverses its direction and returns to its blast-off site and stops. The pilot races across the bottom of the screen and boards the rocketship. When turtle 0 (the pilot) and turtle 2 (the fire on the rocketship) touch,

*The original version of these procedures is on p. 124 of the manual.

the pilot disappears (HT).

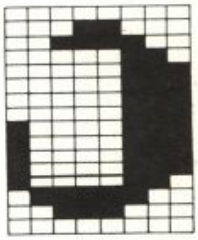
BLAST.OFF2 makes the rocketship take off again. There are take-off sounds and changing background colors; the mountains disappear. The rocketship is on its way. As the rocket continues to travel in space, two more stars are drawn. A message, "GOODBYE ALL!!," is printed on the screen.

Traveling in Space

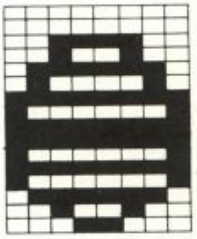
As the ship travels past Saturn and Mars, a moon that belongs to Mars begins to orbit across the screen. I set up a green planet to orbit in the opposite direction.

When the green planet (turtle 0) and the rocketship (turtle 1) touch, HIT.GREEN.PLANET is called. This procedure makes a siren sound (HIT.SOUND) and prints out a few messages:

```
WOW!!!  THIS LOOKS LIKE A GOOD
PLACE TO EXPLORE.  LET'S STOP.

THIS IS THE END OF OUR ADVENTURE.
IT WAS GOOD HAVING YOU ABOARD.
COME BACK AGAIN SOMETIME.

SO LONG.
```



:MOON



:PLANET

TAKE.OFF.ACTION is the procedure that runs most of the action in the story.

```
TO TAKE.OFF.ACTION
SET.SHIP
RECYCLE
BLAST.OFF
GET.PILOT
BLAST.OFF2
MOON.PLANET
FLIGHT.SOUND
END
```

Conversation with the User

To involve you, the user, in a personal way, I added the procedure CONVERSATION. While this is one of the first procedures run in ROCKET, it was one of the last added to the project. At the start of ROCKET, Logo prints out:

```
HI THERE.
WHAT'S YOUR NAME?
```

Logo waits for you to type in something. If you type ROXANNE, for example, it will respond:

```
WELCOME TO DEMOS, ROXANNE
PLEASE WAIT A MINUTE WHILE
I DRAW THE PLANETS AND THE STARS.
```

The name you type in at the beginning of the program is remembered throughout. I edited HIT.GREEN.PLANET, one of the procedures used near the end of the story, so that its last statements refers to you by name.

Rocket

ROCKET is the top-level procedure.

```
TO ROCKET
SET.UP
CONVERSATION
BACKGROUND.SCENE
TAKE.OFF.ACTION
CLEAN.UP
END
```

Setting Up and Cleaning Up

SET.UP puts the shapes into the slots in the shape memory and changes the turtles appropriately. SET.UP also tells Logo to change the background color to black (SETBG 0) and to clear the screen. As I also wanted to leave the turtles in their start-up state, I wrote CLEAN.UP.

PROGRAM LISTING

```
TO PLANET.MARS
HT
SETPC 0 21
DRAW.MARS
MARS
HOME.AGAIN
END

TO DRAW.MARS
PU
SETPOS [-155 120]
SETH 90
PD
REPEAT 18 [FD 10 RT 10]
END

TO MARS
PU
SETPOS [-145 85]
SETH 0
PD
M
PU
BK 18
PD
A
PU
BK 18
PD
R
PU
BK 22
PD
S
HOME.AGAIN
END

TO HOME.AGAIN
PU
HOME
PD
END

TO M
FD 12
RT 135
FD 8
LT 90
FD 8
RT 135
FD 12
```

```
PU
BK 12
LT 135
BK 8
RT 90
BK 8
LT 135
BK 12
END

TO A
FD 12
RT 90
FD 8
RT 90
FD 12
BK 6
RT 90
FD 8
PU
LT 90
FD 6
RT 180
END

TO R
FD 12
RT 90
FD 8
RT 90
FD 6
RT 90
FD 8
LT 135
FD 12
PU
BK 12
RT 45
FD 6
RT 180
END

TO S
FD 2
BK 2
RT 90
FD 8
LT 90
FD 8
LT 90
FD 8
RT 90
FD 8
RT 90
FD 8
END

TO STARS
TELL [0 1 2]
SETPN 0
ASK 0 [SETPLACE [100 90]]
ASK 1 [SETPLACE [120 60]]
ASK 2 [SETPLACE [-80 100]]
STAR
HOME.AGAIN
END

TO STAR
REPEAT 9 [FD 8 BK 8 RT 40]
END

TO SETPLACE :STAR
PU
SETPOS :STAR
PD
END

TO PLANET.SATURN
TELL 0
SETPLACE [50 60]
SATURN
SETRING
RING
HOME.AGAIN
END

TO SATURN
SETPN 2
SETPC 2 52
REPEAT 18 [FD 6 RT 20]
END

TO SETRING
RT 180
REPEAT 6 [FD 1 LT 18]
RT 180
FD 8
END

TO RING
REPEAT 10 [FD 1 LT 18]
FD 45
REPEAT 10 [FD 1 LT 18]
FD 5
END
```

```
TO MOUNTAINS
TELL 0
FS
PU
SETPOS [-158 -60]
PD
RT 45
SETPN 1
SETPC 1 32
SUBMOUNTAIN
END

TO SUBMOUNTAIN
FD 10 + RANDOM 15
IF OR YCOR > 50 YCOR < 0 [SETH 180 - ►
     HEADING]
IF XCOR > 155 [HOME.AGAIN STOP]
SUBMOUNTAIN
END

TO BACKGROUND.SCENE
TELL 0
SETBG 72
PLANET.MARS
STARS
PLANET.SATURN
MOUNTAINS
END

TO GET.PILOT
TELL 0
SETC 20
PU
SETPOS [155 -95]
ST
SETH 270
WHEN TOUCHING 0 2 [HT]
WHEN TOUCHING 0 2 []
REPEAT 10 [WALK]
HT
END

TO WALK
SETSH 5
WALK.SOUND
SETSH 6
WALK.SOUND
SETSH 1
WALK.SOUND
END

TO WALK.SOUND
TOOT 1 200 15 2
WAIT 5
FD 5
END

TO BLAST.OFF
BLAST.OFF.UP
BLAST.OFF.DOWN
END

TO BLAST.OFF.UP
WAIT 60
SETSP 20
SOUND.UP 50
SETSP 0
WAIT 20
CT SS
PR [WAIT!!! WE FORGOT THE PILOT.]
WAIT 100
FS
END

TO SOUND.UP :FREQ
TOOT 1 :FREQ 10 10
IF :FREQ > 1700 [STOP]
SOUND.UP :FREQ + 50
END

TO BLAST.OFF.DOWN
SETSP -20
SOUND.DOWN 1700
SETSP 0
END

TO SOUND.DOWN :FREQ
TOOT 1 :FREQ 10 10
IF :FREQ = 50 [STOP]
SOUND.DOWN :FREQ - 50
END

TO BLAST.OFF2
RECYCLE
TAKE.OFF
BLAST.SOUND
FLASH.COLOR
MOUNTAINS.DISAPPEAR
GOODBYE
MORE.STARS
END

TO TAKE.OFF
TELL [1 2]
ST
WAIT 180
SETSP 21
END
```

```
TO BLAST.SOUND
REPEAT 10 [SOUND.OFF]
END

TO SOUND.OFF
TOOT 0 50 10 3
WAIT 5
TOOT 0 55 10 3
WAIT 5
END

TO FLASH.COLOR
WAIT 60
REPEAT 30 [SETBG RANDOM 128 WAIT 2]
SETBG 0
WAIT 180
END

TO MOUNTAINS.DISAPPEAR
SETPC 1 0
SETSP 39
END

TO GOODBYE
CT SS
PR [GOODBYE ALL!!]
WAIT 60
FS
END

TO MORE.STARS
TELL 3
PU
SETPLACE [-90 -90]
PD
STAR
PU
SETPLACE [75 -75]
PD
STAR
PU
END

TO MOON.PLANET
PU
MOON.MARS
GREEN.PLANET
END

TO MOON.MARS
TELL 3
PU
SETPOS [-155 70]
ST
RT 90
SETSH 4
SETC 7
SETSP 5
END

TO GREEN.PLANET
TELL 0
SETSH 7
WHEN TOUCHING 0 1 [HIT.GREEN.PLANET ▸
     STOP]
SETPOS [107 0]
SETH 270
SETC 99
SETSP 5
ST
END

TO HIT.GREEN.PLANET
HIT.SOUND
CT SS
TELL [0 1 2] SETSP 0
PR [WOW!!!  THIS LOOKS LIKE A GOOD]
PR [PLACE TO EXPLORE.  LET'S STOP.]
WAIT 480 CT
PR [THIS IS THE END OF OUR ADVENTURE.]
PR [IT WAS GOOD HAVING YOU ABOARD]
PR (SE :NAME [.])
PR [COME BACK AGAIN SOMETIME.]
WAIT 540
FS
WAIT 180
CT SS
PR (SE [SO LONG,] :NAME)
END

TO HIT.SOUND
REPEAT 4 [NOISE]
END

TO NOISE
TOOT 0 1000 10 20
WAIT 10
TOOT 1 200 10 20
WAIT 10
END

TO TAKE.OFF.ACTION
SET.SHIP
RECYCLE
BLAST.OFF
GET.PILOT
BLAST.OFF2
MOON.PLANET
FLIGHT.SOUND
END
```

```
TO FLIGHT.SOUND
REPEAT 35 [BEAT]
END

TO BEAT
TOOT 0 100 10 3
WAIT 20
TOOT 1 110 10 3
WAIT 20
END

TO CONVERSATION
CT SS
PR [HI THERE.]
PR [WHAT'S YOUR NAME?]
MAKE "NAME RL
CT
PR (SE [WELCOME TO DEMOS,] :NAME)
PR [PLEASE WAIT A MINUTE WHILE]
PR [I DRAW THE PLANETS AND THE STARS.]
END

TO SET.UP
SETBG 0
PUTSHAPES
TELL 0
SETPN 0
SETPC 0 21
SETSHAPES
CS
END

TO PUTSHAPES
PUTSH 1 :PERSON.1
PUTSH 2 :ROCKET
PUTSH 3 :FIRE
PUTSH 4 :MOON
PUTSH 5 :PERSON.2
PUTSH 6 :PERSON.3
PUTSH 7 :PLANET
END

TO SETSHAPES
ASK [0 1 2 3] [EACH [SETSH WHO + 1 HT ▸
     SETSP 0]]
END

TO SET.SHIP
TELL [1 2]
PU
ASK 1 [SETPOS [0 -90] SETC 7]
ASK 2 [SETPOS [0 -110] SETC 32]
ST FS
END

TO ROCKET
SET.UP
CONVERSATION
BACKGROUND.SCENE
TAKE.OFF.ACTION
CLEAN.UP
END

TO CLEAN.UP
TELL [0 1 2 3]
SETBG 0 SETC 7
CS HT
SETSH 0
TELL 0 ST
SETBG 74
CT SS
END

MAKE "ROCKET [16 16 56 56 254 124 124 ▸
     56 56 56 124 254 254 254 198 198]
MAKE "FIRE   [56 40 254 214 214 214 0 ▸
     0 0 0 0 0 0 0 0 0]
MAKE "PERSON.1 [24 60 20 60 164 188 ▸
     136 126 9 8 62 34 33 65 65 193]
MAKE "PERSON.2 [24 60 20 60 36 188 136 ▸
     255 9 29 36 36 36 36 36 108]
MAKE "PERSON.3 [24 60 20 60 36 60 8 56 ▸
     76 26 41 40 40 72 72 216]
MAKE "MOON   [0 24 60 6 7 7 7 7 135 ▸
     135 135 207 254 124 56 0]
MAKE "PLANET [0 0 60 36 126 66 255 129 ▸
     255 255 129 255 129 126 36 24]
```