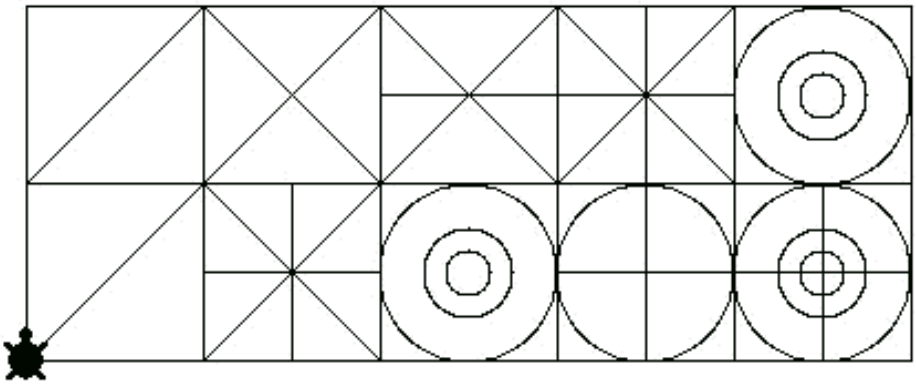
Logo Quilt Project

An adventure in creativity using Lynx



Yours will be much prettier of course!

Objective

You will each contribute to a collaborative quilt, programmed in Logo and drawn by the turtle. This is a classic Logo project modified to use a new web-based dialect of Logo called Lynx. (http:// lynxcoding.club)

 [1]

 [2]

 [3]

Quilting as a craft or art form dates back to ancient Egypt. Quilt making was not only functional as a way of manufacturing blankets, but a collaborative form of expression embraced by Native American, African American, and Amish communities in the United States dating back hundreds of years. There are many styles of quilts, but the combining of different fabric scraps or pieces of uniform size combined to create elaborate

---

[1] "quilt" by Enid H. W. is licensed under CC BY-NC 2.0

[2] "quilt top" by madelinetosh is licensed under CC BY-NC-ND 2.0

[3] "Alphabet quilt" by tirralirra is licensed under CC BY-NC 2.0

geometric patterns lends itself to Logo programming (and constructionism). Quilting traditions may be found in cultures across the globe.

In this project, each of you will be responsible for creating at least one "patch" that will then be shared with your peers. Each of you will then take some of those square patches and assemble a quilt made of them.

Getting started with Logo

The turtle is a metaphor for yourself. When you give it instructions, the turtle does exactly what you tell it to do. If your instructions were inaccurate or wrong, you will either receive an error message or the result of your instruction will be different than what you anticipated. In either case, you need to debug.

The words built into the Logo vocabulary are called *primitives*. Multiple instructions may be run in sequence from the command center of Lynx as long as there are spaces between the words and numbers.

One of the powerful ideas of Logo is that once you figure out how to do something, you can "teach Logo" or "teach the turtle" a new word that remembers that sequence of instructions. These new words are called *procedures*. Procedures behave exactly like primitives except they are unique to a particular project. In other words, user created procedures are available to use as long as they are defined in that project (file).

Procedures are defined in the procedure pane in Lynx. They always begin with the word, *TO,* and end with the word, *END.* Capitalization is never an issue in Logo.

For example, type *foo* in the Command Center and Logo will present the error message, *I don't know how to foo.*

We can define foo by typing the following instructions:

```
to foo
fd 57 rt 144
end
```

Now type `cg pd foo` in the command center and hit enter/return.

CG clears the screen and puts the turtle in the center of the screen. `PD` puts the turtle's pen down. The turtle has a pen stuck in its belly button and when it is down and you command it to move, it leaves a trail. `FD` is the command for forward and it takes a number of turtle steps as its input.

Think of procedure names as infinitive verbs. They produced action when used in Logo.

Procedures and primitives may be combined to create new procedures. Procedures are like building blocks that perform a function and may be combined in infinite variety to produce complexity. Procedures used in other procedures are sometimes called *subprocedures.* There is no limit to the number of procedures you may write. They just all need to be typed in the Lynx procedure pane and follow the rule of beginning with `to` and ending with `end`.

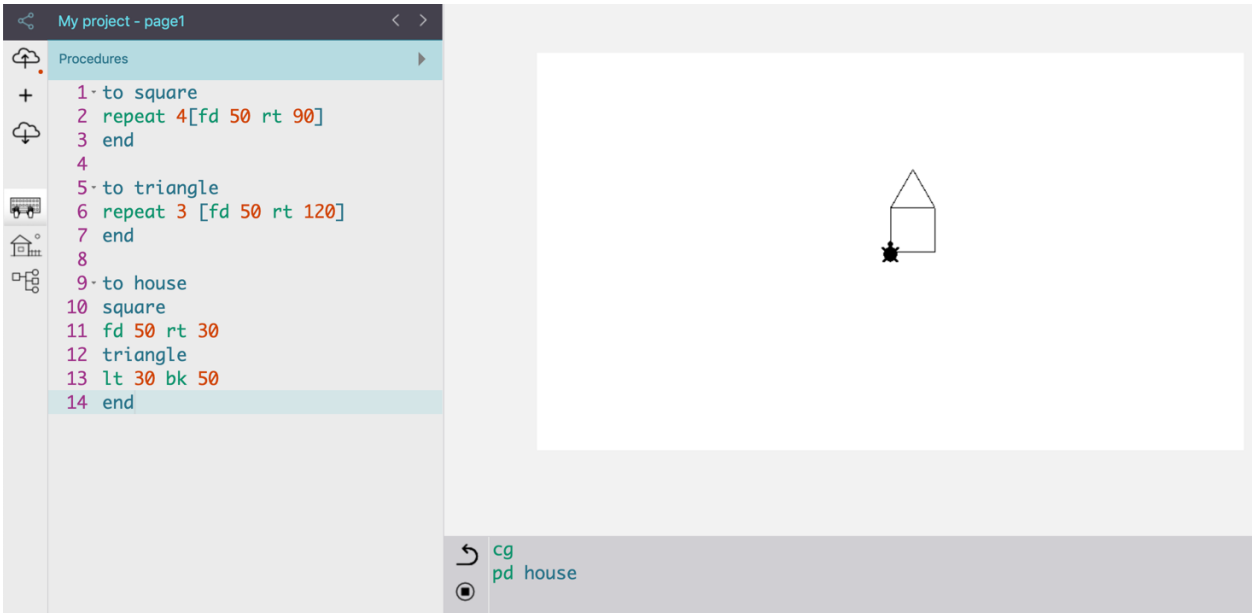
Next, add the following procedure to your procedure pane.

```
to foobar
repeat 5[foo]
end
```

Run `foobar` in the command center. What happened? What does `repeat` do?

Writing and Running Procedures

A procedure is a list of instructions with a name. All procedures begin with `to` and end with `end`.



The order in which procedures are created in the procedure pane does not matter as long as all of the procedures are formatted properly, beginning with `to` and ending with `end`. Putting a blank line between procedures makes them easier to read and debug.

Let's start programming!

1) Each of you must open Lynx, start a new project, name the project, and then type the following procedure into the procedures area.

```
to frame
setcolor "black
pd
repeat 4 [fd 100 rt 90]
end
```

Can you predict what this procedure will do before running it in the command center?

A list of colors the turtle knows may be found here.

2) Next, create a new procedure that is named with your name and perhaps a number (in case you create more than one patch). Each patch will begin with the command, *`frame`*. Then you will tell the turtle what to draw within the constraints of the `patch` (square).

For example:

`to gary1`
`frame`
 	
`end`
	`to jose`
	`frame`
	
	`end`
		`to yumi`
		`frame`
		
		`end`

Important rule!
Everything the turtle draws in your patch must be within the square AND the turtle must return to where it began facing in the same direction. Returning to where you began is called *state transparency* in computer science. It is important for making the patches flexible and portable in this project.

3) Use cg patch and then a series of commands in the command center to design a pattern within the square and return the turtle to where it began. Then copy and paste those instructions into a new procedure, for example:

```
to maria1
frame
rt 45 fd 50 bk 50 lt 45
end
```

4) Create as many quilt patches as you can design. Be sure that each procedure has a unique name.

5) Save your project to the cloud by clicking on the  button in Lynx.

6) Copy and paste your procedures (as text) and share them with your friends via email or posting in a collaborative space.

Make a Quilt!

1) Copy and paste the procedures from your friends into your Lynx procedures. (Make sure that there are no duplicate procedure names. Rename some if necessary. You will only need one `patch` procedure since you are all starting with the same one.

2) Try your friends' procedures and see how they look.

3) Decide which of these patches you wish to assemble into your own quilt.

4) Figure out how to assemble the quilt using the patch procedures and other turtle graphics commands.

5) You should use at least four patches in a quilt.

6) Write a new quilt procedure to automatically draw your new quilt!

7) Save your work to the cloud.

8) Share the project with friends by clicking on the  button and sharing the URL via email or collaborative space.

Here is a sample Quilt project

All of these procedures should be in the Lynx procedure pane if you wish to try our sample `quilt`

```
to frame
setcolor “black
pd
repeat 4 [fd 100 rt 90]
end

to sylvia1
frame
setc “red
pu
rt 90
fd 30
left 90
pd
repeat 4 [fd 40 rt 90]
left 90
pu
fd 30 rt 90
end

to sylvia2
sylvia1
pu setc “blue
rt 90 fd 100
rt 180
sylvia1
pu
fd 100 rt 90
end
```

```
to quilt
repeat 4 [sylvia2 rt 90]
rt 90 pu fd 100 left 90
sylvia3
rt 90 sylvia1 left 90
end

to sylvia3
frame
pu
fd 50 rt 90 fd 50
repeat 360 [pu fd 50 pd fd 0 pu back 50 rt
1]
back 50 rt 90 fd 50 right 180
end
```

`Quilt` is the superprocedure that assembles the quilt you design.

Challenges

- Use one patch procedure as a subprocedure in others.
- What sorts of optical or geometric illusions can you create by just rotating a patch?
- How many patches can you get on the Lynx screen?
- Try the same project with larger or smaller patches.
- Could you program the computer to create random quilts?

Aesthetic tweak

Replace your existing `frame` procedure with this slightly improved version. What does it do differently?

```
to frame
setcolor “black setpensize 3 pd
repeat 4 [fd 100 rt 90]
setpensize 1
end
```

Turtle Cheat Sheet

Here are some turtle graphics primitives to get you started.

Notes:

- # is the sign for inserting a number as the input to a command
- Be sure to use spaces between words and numbers!
- Refrain from using `setpos`. That command makes it hard to move, reorient, or resize quilts.

<table>
<tr><td>Forward #
FD #
For example, <code>fd 50</code></td><td>Back #
BK #</td><td>Right #
RT #</td><td>LEFT #
LT #</td></tr>
<tr><td>CG
clear graphics
Clears the screen and puts the turtle at the center</td><td>Clean
Clears the screen, but leaved the turtle where it is</td><td>PU
Pen up</td><td>PD
Pen down</td></tr>
<tr><td colspan="2">REPEAT # [list of commands]
For example, <code>repeat 4[fd 62 rt 90]</code></td><td>SETC #
set color
SETC 57
<code>SETC “black</code>
<code>SETC “red</code></td><td></td></tr>
<tr><td colspan="2">SETPOS [# #]
For example:
<code>setpos [10 20]</code>
<code>setpos [-25 10]</code>
<code>setpos [-10 -20]</code>
<code>setpos [20 -25]</code></td><td colspan="2">SHOW POS
Displays the current position of the turtle (in coordinates) in the command center</td></tr>
<tr><td colspan="4">SHOW 3 *4
Shows the product of 3 and 4 in the command center. This is the same as asking the turtle to multiple 3 X 4

Show runs a reporter or operation and displays the result in the command center.</td></tr>
</table>

Final Thought

Collaborative expression composed of personal elements created by communicating mathematical ideas to the computer within an extremely open-ended structure makes this project an important “object-to-think-with” for educators.

Resources

- Lynx web site
- Getting Started with Lynx manual